EPFL

Thèse n° 8007

Scalable Multi-agent Coordination and Resource Sharing

Présentée le 28 janvier 2022

Faculté informatique et communications
Laboratoire d'intelligence artificielle
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Panayiotis DANASSIS

Acceptée sur proposition du jury

Prof. M. Jaggi, président du jury
Prof. B. Faltings, directeur de thèse
Prof. F. Oliehoek, rapporteur
Prof. P. Varakantham, rapporteur
Prof. A. Martinoli, rapporteur

École polytechnique fédérale de Lausanne

2022

To my parents, Vassili and Eva, for their love and support.

Στους γονείς μου, Βασίλη και Εύα, για την αγάπη και την υποστήριξή τους.

Acknowledgements

This work would not have been possible without the immense support of my supervisor, Prof. Boi Faltings. My appreciation goes beyond what my words can express! I will always be grateful for his guidance, mentorship, and, most importantly, the freedom he granted me to pursue my research interests.

I would also like to thank my collaborator and friend, Dr. Aris Filos-Ratsikas, to whom I owe a great debt – which I promise to repay in beer and food – for helping me over the years to navigate every aspect of the stressful and competitive world of academia.

I am thankful to Prof. Alcherio Martinoli, Prof. Frans Oliehoek, Prof. Pradeep Varakantham, and Prof. Martin Jaggi for participating in my thesis defence, their feedback, and, most of all, for the insightful discussions that ensued.

To Sylvie Thomet and Patricia Defferrard for their tireless help with all the administrative services.

To current and former lab members for helping with many aspects of my work, be it teaching or research, and for being great colleagues: Aleksei, Naman (who has been an amazing office-mate), Diego (who also provided the French translation of the abstract), Igor, Fei, Zeki, Ljubomir, and Adam.

Life extends beyond research, and I am equally indebted to all the great friends I made in Lausanne for making these past few years an exciting and enjoyable experience. The memories I most cherish over this journey were undoubtedly due to all the amazing people I met. My warm gratitude goes to Praneeth, Periklis, Georgia, Roman, Majed, Kiki, Alessandro, Marco, Lefteris, Karolos, and all the friends from the Greek student association (especially Panayiotis, and Panos).

I am especially grateful to my amazing friend and neighbour Marija for all the fun we had, especially during the stressful years of the pandemic.

I also want to thank my friends in Greece who greatly encouraged me, and for all the fun vacations we had together over the years, especially with Alexis, and Christina. My warm gratitude goes to Artemis, Giorgos, Ilias, Vassiliki, Manos T., Manos K., Alexis K., Eva, Nikos, Irini, Kallia, and Korina.

To the many more people that contributed to this thesis – whether directly, by actively collaborating on research, or indirectly – and are not mentioned above, I sincerely thank you, and I hope I conveyed my appreciation over the years.

Acknowledgements

Above all, I want to thank my family: my parents Vassilis, and Eva, and my brother Costas. Nothing I can say will be enough. I am enormously, and eternally grateful for your everlasting love, encouragement, and support. Μαμά, μπαμπά, σας ευχαριστώ και σας αγαπάω!

Lausanne, January 14, 2022 P. D.

Abstract

A plethora of real world-problems consist of a number of agents that interact, learn, cooperate, coordinate, and compete with others in ever more complex environments. Common examples include autonomous vehicles, robotic agents, intelligent infrastructure, IoT devices, and so on. All these problems can be formulated and studied under the umbrella of multi-agent systems. As more and more autonomous agents are deployed in the real world, it will bring forth the need for novel algorithms, theory, and tools to enable coordination on a massive scale. Moreover, a key question and challenge that societies of the future will have to face is coordination in the use of limited resources. In this thesis, we develop such tools to tackle two central challenges in multi-agent coordination research: solving allocation problems, and resource sharing, focusing on solutions that are *scalable*, *practical*, and applicable to *real-world* problems.

In the first part of the thesis, we tackle one of the fundamental problems in multi-agent systems: allocating resources to agents. This is often captured by the classic problem in computer science of finding a matching (in a weighted graph) of total weight as large as possible, i.e., solving a weighted matching problem. Real-world matching problems may occur in massively large systems (e.g., resource allocation in urban environments), they are distributed and information-restrictive (agents have partial observability and inter-agent communication might not be available), and finally, individuals have to reveal their preferences over the possible matches in order to get a high-quality match, which brings forth significant privacy risks. As such, there are three main challenges: (i) *complexity*, (ii) *communication*, and (iii) *privacy*.
Our proposed approach, ALMA (ALtruistic MAtching heuristic), is a practical heuristic specifically designed for real-world, large-scale (10^3 - 10^6 agents) applications. It is based on a simple altruistic behavioral convention: agents have a higher probability to back-off from contesting a resource if they have good alternatives, potentially freeing the resource for some agent that does not. ALMA tackles all of the aforementioned challenges: it is *decentralized* and runs *on-device*. It requires *no communication* between the participants, and we have proven that it converges in *constant time* – under reasonable assumptions – while providing strong, worst-case, *privacy* guarantees. Moreover, we show that by incorporating learning we can mitigate the loss (due to ALMA being a heuristic) in social welfare (sum of weights), and increase fairness. Finally, we prove that rational agents can use such simple conventions, along with an arbitrary signal from the environment, to

learn a correlated equilibrium (an accessing pattern where no agent has an incentive to deviate) for accessing a set of indivisible resources, under high congestion.
To motivate our research from a practical perspective, we have evaluated ALMA in a variety of real-world applications using *real data*: an online ridesharing and dynamic vehicle relocation scenario with up to 400 thousand requests and 13 thousand taxis, a paper assignment problem, and a meeting scheduling application, which constitutes a more general constraint optimization problem. In all of the cases, ALMA was able to reach *high social welfare* while being *orders of magnitude faster* than the centralized, optimal algorithm.

In the second part of the thesis, we focus on a critical open problem: the question of cooperation in socio-ecological and socio-economical systems, and sustainability in the use of common-pool resources. In recent years, learning agents – especially deep reinforcement learning agents – have become ubiquitous in such systems, which has led to the emergence of machina economica, an approximate counterpart of homo economicus – the perfectly rational agent of neoclassical economics – given computational barriers and the lack of common knowledge. Yet, despite the growing interest in and success of multi-agent deep reinforcement learning, scaling to environments with a large number of learning agents and low observability continues to be a challenge. In our work, we focus on common-pool resources. Individuals face strong incentives to appropriate, which results in overuse and even permanent depletion of the resources. This brings forth the need for techniques that enable the emergence of sustainable cooperation. Our goal is to apply simple interventions to steer the population to desirable states.
To meet the challenge of *cooperation* and *sustainability* in the use of a common-pool resource, we propose a simple technique: allow agents to observe an arbitrary common signal from the environment. The agents learn to couple their policies, which fosters cooperation in large-scale, low observability, and high-stakes environments. Our simple approach proves to be powerful, and robust. It avoids depletion in a wider range of settings while achieving higher social welfare and convergence speed.
Finally, we propose a practical approach to computing market prices and allocations via a deep reinforcement learning policymaker agent, operating in an environment of other learning agents. Compared to the idealized market equilibrium outcome – which can not always be efficiently computed – our policymaker is much more flexible, allowing us to tune the prices with regard to diverse objectives such as *sustainability* and resource *wastefulness*, *fairness*, buyers' and sellers' welfare, etc.

Keywords: ALMA, Altruistic Matching Heuristic, ALMA-Learning, PALMA, CA^{3}NONY, Maximum-weight Matching, Weighted Matching, Assignment Problem, Decentralized, On-device, Multi-agent Systems, Coordination, Cooperation, Resource Allocation, Resource Sharing, Differential Privacy, Piecewise Local Differential Privacy, PLDP, Multi-agent Learning, Deep Reinforcement Learning, Emergent Behaviors, Rationality via Rein-

forcement Learning, Sustainability, Conventions, Social Conventions, Social Dilemmas, Common-pool Resources, Mobility-on-Demand, Ridesharing, Meeting Scheduling, Paper Assignment, Fisheries

Résumé

Nombreux problèmes du monde réel consiste en un certain nombre d'agents qui interagissent, apprennent, coopèrent, se coordonnent et rivalisent avec d'autres dans des environnements de plus en plus complexes. Les exemples les plus courants incluent les véhicules autonomes, les agents robotiques, les infrastructures intelligentes, les appareils IoT, etc. Tous ces problèmes peuvent être formulés et étudiés sous le paradigme des systèmes multi-agents. Au fur et à mesure du déployement d'agents autonomes dans le monde réel, cela entraînera le besoin de nouveaux algorithmes, théories et outils pour permettre la coordination à grande échelle. De plus, une question clé et un défi auquel les sociétés de demain devront faire face est la coordination de l'utilisation de ressources limitées. Dans cette thèse, nous développons de tels outils pour relever deux défis centraux dans la recherche sur la coordination multi-agents : la résolution des problèmes d'allocation et le partage des ressources, en nous concentrant sur des solutions évolutives, pratiques et applicables aux problèmes du monde réel.

Dans la première partie de la thèse, nous abordons l'un des problèmes fondamentaux des systèmes multi-agents : l'allocation de ressources aux agents. Ceci est souvent capturé par le problème classique en informatique de trouver un appariement (dans un graphique pondéré) de poids total aussi grand que possible, c'est-à-dire résoudre un problème d'appariement pondéré. Des problèmes d'appariement du monde réel peuvent survenir dans des systèmes massives (par exemple, l'allocation de ressources dans des environnements urbains) : ils sont distribués et restreints en informations (les agents ont une observabilité partielle et la communication inter-agents peut être indisponible) et les individus doivent révéler leurs préférences afin d'obtenir une correspondance de haute qualité, ce qui entraîne des risques importants pour la vie privée. En tant que tel, il existe trois principaux défis : (i) la complexité, (ii) la communication et (iii) la confidentialité. Notre approche, appelée ALMA (ALtruistic MAtching heuristic), est une heuristique pratique spécialement conçue pour les applications réelles à grande échelle (10^3 - 10^6 agents). Elle est basé sur une convention comportementale altruiste simple : les agents ont une probabilité plus élevée de renoncer à contester une ressource s'ils ont de bonnes alternatives, libérant potentiellement la ressource pour un agent qui n'en a pas. ALMA relève tous les défis susmentionnés : il est décentralisé et fonctionne sur l'appareil utilisateur. Il ne nécessite aucune communication entre les participants, et nous avons prouvé qu'il converge en temps constant - sous des hypothèses raisonnables - tout en

offrant des garanties de confidentialité solides, dans le pire des cas. De plus, nous montrons qu'en intégrant de l'apprentissage, nous pouvons atténuer la perte (due au fait qu'ALMA est une heuristique) de bien-être social (somme des poids) et augmenter l'équité. Enfin, nous prouvons que les agents rationnels peuvent utiliser de telles conventions simples, ainsi qu'un signal arbitraire de l'environnement, pour apprendre un équilibre corrélé (un modèle d'accès où aucun agent n'est incité à dévier) pour accéder à un ensemble de ressources indivisibles, sous haute congestion.
Pour motiver notre recherche d'un point de vue pratique, nous avons évalué ALMA dans une variété d'applications du monde réel en utilisant des données réelles : un scénario de covoiturage en ligne et de relocalisation dynamique de véhicules avec jusqu'à 400 milles demandes et 13 milles taxis, un problème d'affectation d'articles scientifiques, et une application de planification de réunions, qui constitue un problème d'optimisation de contraintes plus général. Dans tous les cas, ALMA a pu atteindre un bien-être social élevé tout en étant significativement plus rapides que l'algorithme centralisé et optimal.

Dans la deuxième partie de la thèse, nous nous concentrons sur un problème ouvert critique : la question de la coopération dans les systèmes socio-écologiques et socio-économiques, et la durabilité dans l'utilisation des ressources communes. Ces dernières années, les agents d'apprentissage - en particulier les agents d'apprentissage par renforcement profond - sont devenus omniprésents dans de tels systèmes, ce qui a conduit à l'émergence de la 'machina economica', une contrepartie approximative de l'homo economicus - l'agent parfaitement rationnel de l'économie néoclassique - étant donné les barrières informatiques et le manque de connaissances communes. Pourtant, malgré l'intérêt croissant et le succès de l'apprentissage par renforcement approfondi multi-agents, l'adaptation à des environnements avec un grand nombre d'agents d'apprentissage et une faible observabilité continue d'être un défi. Dans notre travail, nous nous concentrons sur les ressources communes. Les individus sont fortement incités à s'approprier les ressources, ce qui entraîne une surexploitation voire un épuisement permanent de celles-ci. De ce fait, nous nécessitons des techniques permettant l'émergence d'une coopération durable. Notre objectif est d'appliquer des interventions simples pour orienter la population vers des états souhaitables.
Afin de relever le défi de la coopération et de la durabilité dans l'utilisation d'une ressource commune, nous proposons une technique simple : permettre aux agents d'observer un signal commun arbitraire de l'environnement. Les agents apprennent à coupler leurs politiques, ce qui favorise la coopération dans des environnements à grande échelle, à faible observabilité et à enjeux élevés. Notre approche simple s'avère puissante et robuste. Il évite l'épuisement dans un plus large éventail de contextes, tout en améliorant le bien-être social et la vitesse de convergence.
Enfin, nous proposons une approche pratique du calcul des prix du marché et des allocations via un agent de décision par renforcement en profondeur, opérant dans un environnement d'autres agents. Par rapport au résultat idéalisé de l'équilibre du marché - qui ne peut pas toujours être calculé efficacement - notre décideur est beaucoup plus

flexible, ce qui nous permet d'ajuster les prix en fonction de divers objectifs tels que la durabilité et le gaspillage des ressources, l'équité, le bien-être des acheteurs et des vendeurs, etc.

Mots-clés : ALMA, Heuristique d'Appariement Altruiste, ALMA-Learning, PALMA, CA^{3}NONY, Correspondance de Poids Maximum, Correspondance Pondérée, Problème d'Affectation, Décentralisé, Sur l'Appareil Utilisateur, Systèmes Multi-agents, Coordination, Coopération, Allocation de Ressources, Partage de Ressources, Confidentialité Différentielle, Confidentialité Différentielle Locale par Morceaux, PLDP, Apprentissage Multi-agents, Apprentissage par Renforcement en Profondeur, Comportements Emergents, Rationalité via l'Apprentissage par Renforcement, Durabilité, Conventions, conventions Sociales, Dilemmes Sociaux, Ressources Communes, Mobilité à la Demande, Covoiturage, Planification de Réunions, Assignement d'Articles Scientifiques, Pêche

Contents

1 Introduction

The next technological revolution will be interwoven with the proliferation of intelligent systems (think of autonomous vehicles, robotic agents, intelligent infrastructure, IoT devices, etc.). To truly allow for scalable solutions as we bridge the gap between physical and cyber worlds, we need to shift from traditional centralized approaches to multi-agent solutions, ideally run on-device. Fundamentally, meeting the challenges of coordination on a massive scale will require new theory and algorithms. In this thesis, we develop such tools to tackle two central challenges in multi-agent coordination research: (i) solving allocation problems, and (ii) resource sharing, focusing on solutions that are *scalable*, *practical*, and applicable to *real-world* problems.

1.1 Weighted Matching in Massively Large Systems

In the first part of the thesis, we tackle one of the fundamental problems in multi-agent systems: allocating resources to agents. Consider the following examples: In a mobility-on-demand application (e.g., ridesharing), agents (users of the application) need to be matched with resources (other agents to form a shared ride, and vehicles) with the goal to optimize some performance measure, such as to minimize the total distance driven, the pick-up time, the cost, etc.[1] In an emergency management scenario (e.g., disaster response, search and rescue operations, etc.), agents (rescue robots, or unmanned aerial vehicles) need to be assigned tasks based on their abilities, with the goal to optimize a global objective function (e.g., area coverage). As another example, during the paper assignment phase of a conference, each paper has to be assigned to reviewers based on their preferences and expertise, aiming for an allocation that will result in a high-quality assessment of each paper.

The problem of allocating resources to agents is often captured by the classic problem

[1] Of course, ridesharing is a more general problem, as one can introduce constraints. Yet, as we will demonstrate in Appendix A, it can be viewed as / decomposed into maximum-weight matching problems.

in computer science of finding a high weight matching in a weighted graph, where nodes represent agents and resources (or, more generally, tasks, teammates, etc.), and edge weights represent the utility of matching an agent to a resource. A wide range of applications – spanning from the aforementioned problems of mobility-on-demand, task assignment, and paper assignment, to meeting scheduling, or even kidney exchange – can be formulated and solved as a weighted matching problem. Finding a maximum-weight matching is one of the best-studied combinatorial optimization problems (see (Lovász and Plummer, 2009; Korte and Vygen, 2012; Burkard et al., 2012)). In fact, the first polynomial-time method for maximum-weight matching in bipartite graphs dates back to more than half a century, when Kuhn presented the Hungarian algorithm in his seminal work in 1955.[2] Similarly, the most prominent algorithm for maximum-weight matching in general graphs, the Blossom algorithm of Edmonds, was introduced in 1965. Over the past half a century, the maximum-weight matching problem and its variations have attracted hundreds of researchers, giving birth to a plethora of novel results – including but not limited to tighter bounds on the runtime and memory, novel methods (e.g, using auctions), decentralized implementations, and so on.

Yet, while the problem has been 'solved' from an algorithmic perspective – having both centralized and decentralized polynomial algorithms – it is not so from the perspective of multi-agent systems, for three key reasons: (i) *complexity*, (ii) *communication*, and (iii) *privacy*. Real-world matching problems may occur in massively large settings (e.g., resource allocation in urban environments) and under real-time constraints (i.e., short planning windows), they are distributed and information-restrictive (agents have partial observability and inter-agent communication might not be available), and finally, individuals have to reveal their preferences over the possible matches to get a high-quality match, which brings forth significant privacy risks. State-of-the-art algorithms were not designed to meet the aforementioned challenges risen by the proliferation of intelligent systems and the shift to large-scale multi-agent technologies. In this thesis, we present a novel algorithm (*ALMA*), designed to do so.

1.1.1 ALMA: A Constant Time, Privacy-Preserving Weighted Matching Algorithm

Our proposed approach, *ALMA* (ALtruistic MAtching heuristic), is a practical heuristic specifically designed for real-world, large-scale (10^3 - 10^6 agents) applications. Agents make decisions locally and independently, based on a simple altruistic behavioral convention. More specifically, each agent will independently attempt to acquire a resource, using a selection function that depends on its preferences (utility function). When contesting a resource – i.e., two or more agents attempted to acquire the same resource – each of them

[2]The Hungarian algorithm is the first *widely used* polynomial-time method. The first algorithm for solving the maximum-weight matching problem was introduced by Jacobi in the 19th century (Borchardt and Jacobi, 1865), however it was not discovered until recently (Ollivier, 2009) (see (Su, 2015)).

will back-off independently with a probability that depends on its own utility loss from switching to an alternative resource. Agents have a higher probability to back-off if they have good alternatives, potentially freeing the resource for some agent that does not.

A distinctive characteristic of ALMA is that agents make decisions locally, based on (i) the contest for resources that *they* are interested in, (ii) the agents that are interested in the *same* resources. If each agent is interested in only a *subset* of the total resources – and, thus, each resource is contested by a subset of the total agents – ALMA converges in time polynomial in the maximum size of the subsets; not the total problem size. In particular, if the size of each subset is a constant function of the total number of resources / agents, then the convergence time is *constant*. The same is not true for other algorithms (e.g., the optimal centralized solution, or a greedy algorithm) which require time polynomial in the *total* number of agents/resources – even if the aforementioned condition holds – due to inter-dependencies. The condition holds by default in many real-world applications; agents have only local knowledge of the world, there is typically a cost associated with acquiring a resource, or agents are simply only interested in resources in their vicinity (e.g., resource allocation in urban environments).

ALMA tackles all of the aforementioned challenges: it is *decentralized* and runs *on-device*. It requires *no communication* between the participants, and we have proven that it converges in *constant time* – under reasonable assumptions – while providing strong, worst-case, *privacy* guarantees under Local Differential Privacy. Moreover, we show that by incorporating learning we can mitigate the loss (due to ALMA being a heuristic) in social welfare (i.e., the sum of utilities / weights) and increase fairness. Finally, we have proven that – in some restricted settings – rational agents can use such simple conventions, along with an arbitrary signal from the environment, to learn a correlated equilibrium (an accessing pattern where no agent has an incentive to deviate) for accessing a set of indivisible resources.

1.1.2 Real-world Motivating Applications: Mobility-on-Demand, Meeting Scheduling, and Paper Assignment

As a motivating real-world application, let us expand on the mobility-on-demand example. Consider ridesharing; as mentioned, it has an inherent matching component. Users need to be matched with other users to form shared rides, rides need to be matched with vehicles, and finally, one can also match free vehicles with expected future requests for a better allocation of the available supply to future demand.[3] A natural approach for solving these problems would be to use Integer Linear Programs (ILPs). Yet, as is commonly the case with ILPs, scalability is a major challenge. For example, the highly cited High Capacity algorithm of (Alonso-Mora et al., 2017) for ridesharing can result in $\mathcal{O}(|V||R|^2)$ variables, where $|V|$, and $|R|$ denote the number of vehicles, and requests,

[3] We tackle all of these sub-problems of ridesharing using ALMA in Chapter 3, and Appendix A.

respectively, and $|V|+|R|$ constraints (which results in millions of variables and hundreds of constraints[4]), often making such an approach computationally prohibited. Another natural approach would be to use a classic maximum-weight matching algorithm (e.g., the Blossom algorithm (Edmonds, 1965)), yet it also requires a centralized solution and it is hard to scale, as the run time on a graph (N, E) – where $|E|$ is the number of edges, and $|N|$ is its number of nodes – is of the order of $\mathcal{O}(|E||N|^2)$. A mobility-on-demand company can operate across multiple cities, countries, or even continents, having hundreds of requests per minute (e.g., in our setting – which uses real data from New York City's taxi records – there are 272 new requests per minute on average, totaling to 391479 requests in the broader NYC area in one day), and managing thousands of vehicles (e.g., there are up to 13587 taxis operating at any time in New York City).[4] As a practical solution, one can use a greedy algorithm. Greedy approaches are appealing not only due to their low complexity, but also because real-time constraints dictate short planning windows which diminish the benefit of batch optimization solutions compared to myopic approaches (Widdows et al., 2017). As a matter of fact, (Widdows et al., 2017) reports that GrabShare's scheduling component has used an entirely greedy approach to allocate bookings to vehicles. Lyft also started with a greedy system (Brown, 2016a). ALMA, is of a greedy nature as well, albeit it utilizes a more intelligent backing-off scheme. Most importantly, ALMA was inherently developed for multi-agent applications and can be run *on-device*, independently by each agent. ALMA avoids having to artificially split the problem into sub-problems (e.g, by placing bounds or spatial constraints) and solve those separately, in order to make it tractable, but instead it utilizes a natural domain characteristic – *locality* in the agents' knowledge and inter-agent interactions. This is fundamentally different from a decentralized implementation of the greedy algorithm for example. The number of communication rounds in decentralized algorithms grows with the size of the problem. However, real-time constraints impose a limit on the number of rounds and, thus, on the size of the problem that can be solved within them. Moreover, sharing plans and preferences creates high overhead, and there is often a lack of responsiveness and/or communication between the participants. Most importantly, though, all of these approaches bring about significant *privacy risks* and the potential to reveal highly sensitive information of users. We are the first to develop a practical and scalable framework for weighted matching and resource allocation in massively large systems, that provides *strong worst-case privacy guarantees.* Our work shows that harnessing the potential of intelligent systems does not have to compromise privacy.

To further motivate our research from a practical perspective, we have evaluated ALMA not only using synthetic benchmarks, but also in a variety of real-world applications using *real data*, specifically: (i) an online ridesharing and dynamic vehicle relocation scenario with up to 400 thousand requests and 13 thousand taxis, (ii) a paper assignment problem, and (iii) a meeting scheduling application, which constitutes a more general constraint optimization problem. In all of the cases, ALMA was able to reach *high social*

[4]See Appendix A.

welfare while being *orders of magnitude faster* than the centralized, optimal algorithm.

1.2 Resource Sharing for Deep Reinforcement Learning Agents

In the second part of the thesis, we study the emergent behaviors in a group of independent deep reinforcement learning agents and apply simple interventions to steer the population to desirable states. We focus on the questions of *cooperation* in socio-ecological and socio-economical systems, and *sustainability* in the use of common-pool resources.

Economic and social progress over the last few centuries has brought about environmental degradation and endangered the natural environments and resources that our future development depends on. Sustainability and the preservation of the earth's natural resources constitute one of the most pressing issues and grand challenges in modern societies. For example, in 2015 the United Nations issued a universal call for action to protect the planet and promote sustainable development;[5] an agenda that was adopted by all UN member states. It is becoming increasingly clear that we need to shift our production patterns, decouple the economic growth from environmental degradation, and increase resource efficiency. As autonomous agents proliferate, they will be called upon to interact in ever more complex environments, and as such, will play a key part in *sustainable production.*

In recent years, for example, learning agents have become ubiquitous in socio-economical and socio-ecological systems. This has led to the emergence of machina economica (Parkes and Wellman, 2015), an approximate counterpart of homo economicus – the perfectly rational agent of neoclassical economics – given computational barriers and the lack of common knowledge. The success of multi-agent deep reinforcement learning has led to a growing interest in modeling machinae economicae agents as independent deep reinforcement learning agents that need to interact, learn, cooperate, coordinate, and compete with other learning agents in ever more complex, non-stationary environments. Yet, despite the growing interest in and success of multi-agent deep reinforcement learning, scaling to environments with a large number of learning agents and low observability continues to be a problem. To better understand the impact of self-interested appropriation, and develop sustainable strategies, it would be beneficial to examine the dynamics of *real-world* common-pool renewable resources.

1.2.1 A Real-world Test-bench: The Common Fishery Game

Consider a common-pool resource – i.e., a resource that is challenging and/or costly to exclude individuals from appropriating – of finite yield. Individuals face strong incentives

[5] https://www.un.org/sustainabledevelopment/

to appropriate, which results in overuse and even permanent depletion of the resource. Examples include the degradation of fresh water resources, the over-harvesting of timber, the depletion of grazing pastures, the destruction of fisheries, etc. Many real-world common-pool resource problems are inherently *large-scale* and *partially observable*, which further increases the challenge of sustainability.

We have introduced a realistic common-pool resource appropriation game for multi-agent coordination, based on an abstracted bio-economic model for commercial fisheries. The model describes the dynamics of the stock of a common-pool renewable resource, as a group of appropriators harvest over time. The harvest depends on (i) the effort exerted by the agents and (ii) the ease of harvesting a resource at that point in time, which depends on the stock level. The stock replenishes over time with a rate dependent on the current stock level.

1.2.2 Exploiting Environmental Signals to Enable Policy Correlation

We first focus on the problems of *cooperation* and *sustainability* in the use of a common-pool commercial fishery. We assume the *most information-restrictive setting*: each participant is modeled as an individual agent with its own policy conditioned only on local information, specifically his own history of action/reward pairs (fully decentralized method). Global observations, including the resource stock, the number of participants, and the joint observations and actions, are hidden – as is the case in many real-world applications, like commercial fisheries. Under such a setting, it is impossible to avoid positive probability mass on undesirable actions (i.e., simultaneous appropriation), since there is no correlation between the agents' policies. This leads to either low social welfare, because the agents can not maintain a healthy stock, or, even worse, the *depletion* of the resource. Depletion becomes more likely as the problem size grows due to the non-stationarity of the environment and the global exploration problem.[6] This brings forth the need for techniques that enable the emergence of sustainable cooperation.

We propose a simple technique: allow agents to observe an arbitrary, common signal from the environment. Observing a common signal mitigates the aforementioned problems because it allows for *coupling* between the learned policies, increasing the joint policy space. Agents, for example, can now learn to harvest in turns, and with varying efforts per signal value, or allow for fallow periods. The benefit is twofold: the agents learn to not only avoid depletion, but also to maintain a healthy stock which allows for large harvest and, thus, higher social welfare. It is important to stress that we do not assume any a priori relation between the signal space and the problem at hand. Moreover, we require no communication, no extrinsic incentive mechanism, and we do not change the

[6] The former arises due to simultaneous learning by all agents, which results in a moving-target problem (the best policy changes as the other agents' policies change). The latter refers to the probability that at least one agent explores (i.e., not acting according to the optimal policy) which increases with the number of agents (Busoniu et al., 2008; Matignon et al., 2012; Hernandez-Leal et al., 2019).

underlying architecture or learning algorithm. We simply utilize a means – common environmental signals that are amply available to the agents – to accommodate correlation between policies, and foster cooperation in large-scale (we simulate up to 64 agents), low observability, and high-stakes environments. Our simple approach proves to be powerful and robust: it avoids depletion in a wider range of settings while achieving higher social welfare and convergence speed.

1.2.3 AI-driven Policymaking for Multi-agent Markets

In the last part of the thesis, we extend our common fisheries model to include a complex and realistic market. We propose a *practical* approach to computing market prices and allocations via a deep reinforcement learning policymaker agent, operating in an environment of other learning agents.

The theory of competitive markets, founded in the works of Walras (Walras, 1874), Fisher (Fisher, 1892), and Arrow and Debreu (Arrow and Debreu, 1954) is one of the most prominent economic models of the 20th century. Competitive markets are characterized by the market equilibrium – a stable outcome in which supply equals demand, and all participants are maximally satisfied by the bundles of goods that they buy or sell. It is meant to capture the outcome of a free market, dictated by the market's 'invisible hand' (Smith, 1791). Yet, the convergence of this continuous adjustment of prices is highly dependent on several initial parameters and can therefore be slow, and the centralized computation of market equilibria for certain markets can be computationally hard (Codenotti et al., 2006; Chen et al., 2017, 2009), thus in many cases impractical to compute. Most importantly, the market equilibrium is geared towards very specific goals, namely fairness for the participants and economic efficiency given the set of chosen prices, and can not account for 'exogenous' objectives, like sustainable production.

Compared to the idealized market equilibrium outcome – which we use as a benchmark – our policymaker is much more flexible, allowing us to *tune* the prices with regard to diverse objectives such as *sustainability* and resource *wastefulness*, *fairness*, buyers' and sellers' welfare, etc. We demonstrate that: (a) The introduced policymaker is able to achieve comparable performance to the market equilibrium, showcasing the potential of such approaches in markets where the *equilibrium prices can not be efficiently computed*. (b) Our policymaker can notably outperform the equilibrium solution on certain metrics, while maintaining comparable performance for the remaining ones. (c) As a highlight of our findings, our policymaker is significantly more successful in maintaining resource *sustainability*, compared to the market outcome, in scarce resource environments.

1.3 Contributions and Thesis Organization

This thesis comprises of two major parts. The first part tackles the problem of resource allocation (weighted matching) in massively large multi-agent systems, while the second is centered on the sustainable sharing of common-pool resources amongst deep reinforcement learning agents. Every part begins with a preliminary chapter containing background information, followed by the main content chapters. Each of these chapters maps to one or more papers written by the author (with other collaborators). At the start of each of these chapters there is a preface that includes detailed author contributions using the CRediT taxonomy (Brand et al., 2015), and a summary of the chapter. Finally, at the end of the thesis, there is a third part containing appendices that include modeling details, additional simulation results, etc.

To address the challenges of coordination in large-scale multi-agent systems, we propose a number of novel tools, theory, and algorithms. A list of our main contributions (along with the corresponding chapters) can be found below. We have organized our contributions into two parts, which correspond to the aforedescribed parts of the thesis.

1.3.1 Contributions on the Resource Allocation and Weighted Matching Problems in Large-scale Multi-agent Systems

Chapter 3:

We introduce a novel ALtruistic MAtching (ALMA) heuristic for weighted matching in massively large systems. ALMA is decentralized and runs on-device. It requires no inter-agent communication, and agents observe only their own action/reward pairs. We have proven a polynomial upper bound on the runtime. In the realistic case where each agent is interested in a (fixed size) subset of the total resources – and thus each resource is contested by a (fixed size) subset of the total agents – we have proven that ALMA converges in constant time.

Chapter 4:

We introduce Piecewise Local Differential Privacy (PLDP) – a variant of differential privacy designed for real-world, large-scale settings – and PALMA, a privacy-preserving variant of ALMA that combines ALMA, PLDP, and a privacy accounting method for iterative algorithms. We are the first to develop a practical privacy-preserving algorithm for weighted matching and resource allocation in massively large multi-agent systems.

Chapter 5:

We introduce ALMA-Learning, a variant of ALMA that incorporates learning to mitigate the loss (due to ALMA being a heuristic) in social welfare compared to the optimal solution, and increase the fairness of the allocation. Moreover, we prove that ALMA-Learning converges.

Chapter 6:

We introduce the CA^{3}NONY (Contextual Anti-coordination in Ad-hoc ANONYmous games) framework, a variant of ALMA that utilizes an arbitrary environmental signal coupled with a monitoring authority per resource to learn an accessing pattern for a set of indivisible resources with binary utilities, under high congestion. We prove that the induced strategies constitute an approximate subgame-perfect equilibrium (i.e., the agents do not have an incentive to deviate).

Chapter 3, Chapter 4, Chapter 5:

We evaluate ALMA in a variety of real-world applications using real data such as a paper assignment problem, a meeting scheduling application, and an online ridesharing and dynamic vehicle relocation application with up to 400 thousand requests and 13 thousand taxis. The latter constitutes the largest and most comprehensive evaluation on ridesharing to date, comparing 14 algorithms on 12 different metrics related to global efficiency, complexity, passenger, driver, and platform incentives, in settings designed to closely resemble reality in every aspect.

1.3.2 Contributions on the Problem of Sustainable and Scalable Resource Sharing for Deep Reinforcement Learning Agents

Chapter 8, Chapter 9:

We are the first to introduce a realistic common-pool resource appropriation game for multi-agent coordination. Our model is based on an abstracted bio-economic model of commercial fisheries and incorporates the challenges of resource scarcity and the tragedy of the commons. Moreover, we provide theoretical analysis of the dynamics of the environment. Specifically, we prove the optimal harvesting strategy that maximizes the revenue, and identify two interesting stock values: the 'limit of sustainable harvesting', and the 'limit of immediate depletion'.

Chapter 8:

We propose a simple, yet powerful and robust technique for improving coordination in a set of independent deep reinforcement learning agents: allow agents to observe an arbitrary periodic signal from the environment. Such signals are amply available (e.g., time, date, etc.) and can foster cooperation among agents. Specifically, we show that agents avoid depleting the resources in a wider range of settings while achieving higher social welfare and convergence speed.

Chapter 9:

We propose a practical approach to computing market prices and allocations via a deep reinforcement learning policymaker agent, that allows us to tune the prices with regard to diverse objectives such as sustainability, resource wastefulness, fairness, etc. We demonstrate significant improvements over the market equilibrium for several objectives while maintaining comparable performance for the rest.

Part I

Weighted Matching in Massively Large Settings

2 Preliminaries

2.1 Notation

To avoid introducing unnecessary notation early on, we only introduce the notation related to the problem description. The rest of the notation is chapter specific and will be introduced as needed. Let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of agents, and $\mathcal{R} = \{1, \ldots, R\}$ the set of resources. Finally, let $u_n(r) \in [0, 1]$ denote the utility function.

2.2 The Assignment Problem (Maximum Weight Matching in a Bipartite Graph)

The assignment problem consists of finding a maximum weight matching in a weighted bipartite graph, $\mathcal{G} = \{\mathcal{N} \cup \mathcal{R}, \mathcal{E}\}$. In our context, the nodes are divided in two disjoint sets: agents $\mathcal{N}$ and resources $\mathcal{R}$. The weight of an edge $(n, r) \in \mathcal{E}$ represents the utility $(u_n(r))$ agent n receives by acquiring resource r. Each agent can acquire at most one resource, and each resource can be assigned to at most one agent. The goal is to maximize the social welfare (sum of utilities), i.e., to solve the constraint optimization program of (2.1) – where the variable $x_{n,r}$ is 1 if the edge is contained in the matching and 0 otherwise, and $\mathbf{x} = (x_{1,1}, \ldots, x_{N,R})$.

$$\begin{aligned} \max_{\mathbf{x} \geq 0} \quad & \sum_{(n,r) \in \mathcal{E}} u_n(r) x_{n,r} \\ s.t. \quad & \sum_{r|(n,r) \in \mathcal{E}} x_{n,r} = 1, \quad \forall\, n \in \mathcal{N} \\ & \sum_{n|(n,r) \in \mathcal{E}} x_{n,r} = 1, \quad \forall\, r \in \mathcal{R} \end{aligned} \tag{2.1}$$

For an in-depth description of the assignment problem, theory, and available algorithms, see (Burkard et al., 2012; Korte and Vygen, 2012; Lovász and Plummer, 2009; Su, 2015).

2.3 Maximum Weight Matching in a General Graph

Let $\mathcal{G} = \{\mathcal{N}, \mathcal{E}, w\}$ denote a general graph, where $\mathcal{N}$, $\mathcal{E}$, and w denote the nodes, edges, and edge weight function, respectively. The goal is to maximize the sum of weights, i.e., to solve the constraint optimization program of (2.2) – where the variable x_ϵ is 1 if the edge $\epsilon = (n, n')$ is contained in the matching and 0 otherwise.

$$\max \quad \sum_{\epsilon \in \mathcal{E}} w_\epsilon x_\epsilon \tag{2.2}$$

$$\begin{aligned} s.t. \quad & \sum_{\epsilon=(n,n') \in \mathcal{E}} x_\epsilon \leq 1, \quad \forall\, n \in \mathcal{N} \\ & 0 \leq x_\epsilon \leq 1, \qquad \forall\, \epsilon \in \mathcal{E} \\ & x_\epsilon \text{ is an integer}, \quad \forall\, \epsilon \in \mathcal{E} \end{aligned} \tag{2.3}$$

In the context of multi-agent applications, a general graph could represent matching agents into teams (e.g., matching robots to perform a coordination task, or matching two users of a mobility-on-demand application into a shared ride, see Section 3.6 for such an application), in which case the edge weight function would represent the utility gained by forming the team. For an in-depth description of the maximum-weight matching problem, theory, and available algorithms, see (Burkard et al., 2012; Korte and Vygen, 2012; Lovász and Plummer, 2009).

2.4 Differential Privacy Definition

In this section we provide a short description of the traditional Differential Privacy (DP) definition; (Dwork, 2006; Dwork et al., 2006a,b) we refer the interested reader to (Triastcyn, 2020; Dwork et al., 2014) for a more comprehensive overview of Differential Privacy and Differential Privacy mechanisms.

Differential privacy is often discussed in the context of identifying individuals whose information may be in a database. It relies on an important impossibility result: impossibility of absolute disclosure prevention. The authors of (Dwork, 2006; Dwork et al., 2006a,b) prove that the conventional requirement of statistical database privacy – access to a database should not allow an adversary to learn additional information about an individual than what could be learned without such access – cannot be achieved due to *auxiliary information* available to the adversary (besides the access to the database). As such, the authors argue to switch from absolute privacy guarantees to relative ones: informally, differential privacy captures the increased risk to an individual's privacy incurred by participating in a database. An algorithm is then considered differentially private if an adversary can not infer if a particular individual's information was used in the computation, given the output of said algorithm.

In order to achieve differential privacy, one needs a source of randomness. Let $\mathcal{M} : \mathcal{D} \to \mathcal{A}$ be a random function, mapping sensitive inputs from domain $\mathcal{D}$ to range $\mathcal{A}$ of privatized (or sanitized) outputs. In the context of matching problems in multi-agent systems, $\mathcal{D}$ can be the space of utility functions, and $\mathcal{A}$ the action space. Definition 1 defines a relaxation of differential privacy, called *Approximate Differential Privacy* or (ε, δ)-*Differential Privacy* (Dwork et al., 2014), which is more often used in artificial intelligence (and machine learning).

Definition 1 ((ε, δ)-Differential Privacy)**.** A randomized function (algorithm) $\mathcal{M} : \mathcal{D} \to \mathcal{A}$ with domain $\mathcal{D}$ and range $\mathcal{A}$ satisfies (ε, δ)-differential privacy if for any two adjacent inputs $D, D' \in \mathcal{D}$ and for any set of outcomes $\mathcal{S} \subset \mathcal{A}$ the following holds:

$$\Pr\left[\mathcal{M}(D) \in \mathcal{S}\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}(D') \in \mathcal{S}\right] + \delta.$$

2.4.1 Intuitive Example

In what follows, we provide some intuition on the interpretation of the (ε, δ) values (glossing over some of the technical details).

Imagine a simple, stripped-down example where there is only one agent n, and two resources r_1 and r_2. Suppose that agent n prefers resource r_1, i.e., $u(r_1) > u(r_2)$. Under no regard for privacy, the optimal strategy for n is to acquire resource r_1. However, an outsider observing his action will immediately know agent n's preference. To protect privacy under DP, the agent will randomize its decisions by flipping a coin. Depending on the result (heads or tails), agent n would acquire either resource r_1 or r_2, respectively. Now the observer can not know if the decision was taken based on the agent's actual preference, or due to the coin toss (plausible deniability). If the coin is unbiased it is easy to see that agent n's preference is completely lost in the randomness and privacy is fully protected, but there is no utility benefit compared to a random allocation. *This corresponds to* $\varepsilon = 0$. To increase the utility of the allocation, we will bias the coin towards the preferred resource r_1. Landing on heads is now more probable than landing on tails, and the ratio $Pr[\text{heads}]/Pr[\text{tails}]$ is greater than 1; ε *is the logarithm of this ratio.* The DP literature also refers to ε as *privacy budget.* Finally, imagine that sometimes the agent fails to flip a coin and just goes for the preferred resource. δ refers to this *failure probability* (typically very small). In other words, an (ε, δ)-Differentially Private algorithm provides a privacy guarantee ε with probability $(1 - \delta)$. As such, the pair of these two values fully characterizes the privacy guarantee.

2.5 Fairness Metrics

We have employed several well-established fairness metrics thoughout this thesis, specifically: the Jain index (Jain et al., 1998), the Gini coefficient (Gini, 1912), and the

Atkinson index (Atkinson, 1970). In what follows, we provide the definition and a short description for each one of them.

(a) The Jain index (Jain et al., 1998): Widely used in network engineering to determine whether users or applications receive a fair share of system resources. It exhibits a lot of desirable properties such as population size independence, continuity, scale and metric independence, and boundedness. For an allocation game of N agents, such that the n^{th} agent is allotted x_n, the Jain index is given by Equation 2.4. $\mathbb{J}(\mathbf{x}) \in [0, 1]$. An allocation $\mathbf{x} = (x_1, \ldots, x_N)^\top$ is considered fair, iff $\mathbb{J}(\mathbf{x}) = 1$.

$$\mathbb{J}(\mathbf{x}) = \frac{\left(\sum_{n=1}^{N} x_n\right)^2}{N \sum_{n=1}^{N} x_n^2} \tag{2.4}$$

(b) The Gini coefficient (Gini, 1912): One of the most commonly used measures of inequality by economists intended to represent the wealth distribution of a population of a nation. For an allocation game of N agents, such that the n^{th} agent is allotted x_n, the Gini coefficient is given by Equation 2.5. $\mathbb{G}(\mathbf{x}) \geq 0$. A Gini coefficient of zero expresses perfect equality, i.e., an allocation is fair iff $\mathbb{G}(\mathbf{x}) = 0$.

$$\mathbb{G}(\mathbf{x}) = \frac{\sum_{n=1}^{N} \sum_{n'=1}^{N} |x_n - x_{n'}|}{2N \sum_{n=1}^{N} x_n} \tag{2.5}$$

(c) The Atkinson index (Atkinson, 1970): Is a measure of the amount of social utility to be gained by complete redistribution of a given income distribution, for a given ϵ. In our work, we used $\epsilon = 1$. For an allocation game of N agents, such that the n^{th} agent is allotted x_n, the Atkinson index of $\epsilon = 1$ is given by Equation 2.6. $\mathbb{A}(\mathbf{x}) \in [0, 1]$. An Atkinson index of zero expresses perfect equality, i.e., an allocation is fair iff $\mathbb{A}(\mathbf{x}) = 0$.

$$\mathbb{A}(\mathbf{x}) = 1 - \frac{1}{\frac{\sum_{n=1}^{N} x_n}{N}} \left(\prod_{n=1}^{N} x_n\right)^{1/N} \tag{2.6}$$

3 A Constant Time Algorithm for Weighted Matching

3.1 Preface

3.1.1 Contribution and Sources

This chapter is largely based on (Danassis et al., 2019; Danassis and Faltings, 2020; Danassis et al., 2022b). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable).

Authors' contributions in (Danassis et al., 2019):

PD (author):	Conceptualization, Methodology, Software, Formal analysis, Investigation (lead), Writing – Original Draft, Writing – Review & Editing
Aris Filos-Ratsikas:	Investigation (supporting), Writing – Review & Editing
Boi Faltings:	Writing – Review & Editing, Supervision

Authors' contributions in (Danassis and Faltings, 2020):

PD (author):	Conceptualization, Methodology, Software, Investigation, Writing – Original Draft, Writing – Review & Editing
Boi Faltings:	Supervision

Authors' contributions in (Danassis et al., 2022b):

PD (author):	Conceptualization, Methodology (lead), Software (supporting), Investigation (lead), Writing – Original Draft (lead), Writing – Review & Editing
Marija Sakota:	Software (lead), Investigation (supporting), Writing – Review & Editing
Aris Filos-Ratsikas:	Conceptualization, Methodology (supporting), Investigation (supporting), Writing – Original Draft (supporting), Writing – Review & Editing
Boi Faltings:	Writing – Review & Editing (supporting), Supervision

3.1.2 Chapter Summary

We present a novel anytime heuristic (*ALMA*), inspired by the human principle of altruism, for weighted matching in massively large environments. ALMA is decentralized, completely uncoupled, and requires no communication between the participants. We prove an upper bound on the convergence speed that is polynomial in the desired number of resources and competing agents per resource; crucially, in the realistic case where the aforementioned quantities are bounded independently of the total number of agents/resources, the convergence time remains *constant* as the total problem size increases.

We have evaluated ALMA under three test cases: (i) an anti-coordination scenario where agents with similar preferences compete over the same set of actions, (ii) a resource allocation scenario in an urban environment under a constant-time constraint, and finally, (iii) an on-line ridesharing and dynamic vehicle relocation application using real data. The former two are synthetic benchmarks that simulate environments of massive scale, with up to 131000 agents, and as many resources, while the latter constitutes – to the best of our knowledge – the largest and most comprehensive end-to-end evaluation on ridesharing to date. In particular, we have evaluated 14 different algorithms over 12 metrics related to global efficiency, complexity, passenger, driver, and platform incentives, in settings designed to closely resemble reality in every aspect, using actual data from the NYC's yellow taxi trip records. In all of the cases, ALMA was able to reach high social welfare, while being orders of magnitude faster than the centralized, optimal algorithm. As such, we effectively demonstrate that ALMA is able to scale to realistic scenarios with hundreds of thousands of agents (e.g., vehicle coordination in urban environments).

3.2 Introduction

One of the most relevant problems in multi-agent systems is finding an optimal allocation between agents, i.e., solving a weighted matching problem. This pertains to role allocation (e.g., team formation for autonomous robots (Gunn and Anderson, 2013)), task assignment (e.g., employees of a factory, or taxi-passenger matching (Varakantham et al., 2012)), resource allocation (e.g., parking spaces and/or charging stations for autonomous vehicles (Geng and Cassandras, 2013)), etc. What follows is *applicable to any such scenario*, but for concreteness we will refer to the allocation of a set of resources to a set of agents, a setting known as the *assignment problem* (weighted matching in bipartite graphs, see Section 2.2), one of the most fundamental combinatorial optimization problems (Munkres, 1957).

When designing algorithms for the assignment problem, a significant challenge emerges from the nature of real-world applications, which are often distributed and information-restrictive. Even in decentralized algorithms the number of communication[1] rounds required grows with the size of the problem. However, in practice the real-time constraints impose a limit on the number of rounds, and thus on the size of the problem that can be solved within them. Moreover, sharing plans and preferences creates high overhead, and there is often a lack of responsiveness and/or communication between the participants (Stone et al., 2010). Achieving fast convergence and high efficiency in such information-restrictive settings is extremely challenging. Yet, humans are able to routinely and robustly coordinate in similar everyday scenarios, often with no explicit communication. One driving factor that facilitates human cooperation is behavioral conventions (Lewis, 2008). Inspired by human behavior, the proposed heuristic (*ALMA*: *AL*truistic *MA*tching heuristic) is modeled on an *altruistic* convention. This results to fast convergence to highly efficient allocations, without any communication between the agents.

A distinctive characteristic of ALMA is that agents make decisions locally, based on (i) the contest for resources that *they* are interested in, (ii) the agents that are interested in the *same* resources. If each agent is interested in only a *subset* of the total resources, ALMA converges in time polynomial in the maximum size of the subsets; not the total number of resources. In particular, if the size of each subset is a constant function of the total number of resources, then the convergence time is *constant*, in the sense that it does not grow with the problem size. The same is not true for other algorithms (e.g., the optimal centralized solution) which require time polynomial in the *total* number of agents/resources, even if the aforementioned condition holds. The condition holds by default in many real-world applications; agents have only local knowledge of the world, there is typically a cost associated with acquiring a resource, or agents are simply only interested in resources in their vicinity (e.g., resource allocation in urban environments).

[1] To exchange, for example, cost matrices, pricing information, basis of the LP, etc. – see Section 3.2.2.

This is important, as the proposed approach avoids having to artificially split the problem in sub-problems (e.g, by placing bounds or spatial constraints) and solve those separately, in order to make it tractable. Instead, ALMA utilizes a natural domain characteristic, not an artificial optimization technique (i.e., artificial bounds). Coupled to the convergence time, the decentralized nature of ALMA makes it applicable to large-scale, real-world applications (e.g., IoT devices, intelligent infrastructure, autonomous vehicles, etc.).

3.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We introduce a novel, anytime *AL*truistic *MA*tching heuristic (*ALMA*) for weighted matching.** ALMA is decentralized, completely uncoupled (agents are only aware of own history of action/reward pairs (Talebi, 2013)), and requires no communication between the agents.
2. **We prove that the expected number of steps for any agent to converge is *independent of the total problem size*,** if we bound the maximum number of resources an agent is interested in, and the maximum number of agents competing for a resource. Thus, we do not require to artificially split the problem, or similar techniques, to render it manageable.
3. **We provide a thorough empirical evaluation of ALMA on both synthetic and *real data*.** In particular, we have evaluated ALMA under three test cases: (i) an anti-coordination scenario where agents with similar preferences compete over the same set of actions, (ii) a resource allocation scenario in an urban environment, under a constant-time constraint, and finally, (iii) an on-line ridesharing application using *real* passenger-taxi data. In all of the cases, ALMA achieves high social welfare (total satisfaction of the agents) as compared to the optimal solution, as well as various other algorithms.

3.2.2 Discussion and Related Work

The assignment problem consists of finding a maximum weight matching in a weighted bipartite graph (see Section 2.2) and it is one of the best-studied combinatorial optimization problems in the literature. The first polynomial time algorithm (with respect to the total number of nodes, edges) was introduced by Jacobi in the 19th century (Borchardt and Jacobi, 1865; Ollivier, 2009), and was succeeded by many classical algorithms (Munkres, 1957; Edmonds and Karp, 1972; Bertsekas, 1979) with the *Hungarian* algorithm of Kuhn 1955 being the most prominent one (see (Su, 2015; Burkard et al., 2012) for an overview). The problem can also be solved via linear programming (Dantzig, 1990), as its LP formulation relaxation admits integral optimal solutions (Papadimitriou

and Steiglitz, 1982). ALMA works on non-bipartite graphs too, which corresponds to the more general *maximum weight matching* problem on general graphs (see Section 2.3 for the definition and Section 3.6 for an example application). To compute the optimal solution in this case, we will use the *blossom algorithm* of (Edmonds, 1965) (see (Lovász and Plummer, 2009; Korte and Vygen, 2012)).

In reality, a centralized coordinator is not always available, and if so, it has to know the utilities of all the participants, which is often not feasible. In the literature of the assignment problem, there also exist several decentralized algorithms (e.g., (Giordani et al., 2010; Ismail and Sun, 2017; Zavlanos et al., 2008; Bürger et al., 2012) which are the decentralized versions of the aforementioned well-known centralized algorithms – see also (Kuhn et al., 2016; Elkin, 2004) for general results in distributed approximability under only local information/computation). However, these algorithms require polynomial computational time and polynomial number of messages (such as cost matrices (Ismail and Sun, 2017), pricing information (Zavlanos et al., 2008), or a basis of the LP (Bürger et al., 2012), etc.). Yet, agent interactions often repeat no more than a few hundreds of times. To the best of our knowledge, a decentralized algorithm that requires no message exchange (i.e., no communication network) between the participants, and achieves high efficiency, like ALMA does, has not appeared in the literature before. Let us stress the importance of such a heuristic: as autonomous agents proliferate, and their number and diversity continue to rise, differences between the agents in terms of origin, communication protocols, or the existence of sub-optimal, legacy agents will bring forth the need to collaborate without any form of explicit communication (Stone et al., 2010). Finally, communication between participants creates high overhead as well.

ALMA is inspired by the allocation algorithm of (Cigler and Faltings, 2013) (adapted in (Danassis and Faltings, 2019) to solve resource allocation problems under rationality constraints). Using such a simple learning rule which only requires environmental feedback, allows our approach to scale to hundreds of thousands of agents. Moreover, it does not require global knowledge of utilities; only local knowledge of personal utilities (in fact, we require knowledge of pairwise differences which are far *easier to estimate*).

Finally, the probabilistic response of ALMA is related to threshold-based response algorithms (e.g., see (Agassounon and Martinoli, 2002; Kalra and Martinoli, 2006; Bonabeau et al., 1997)), though these algorithms are designed to respond to stimuli from the environment, while each agent running ALMA acts based on his own (internal) preferences.

3.3 ALMA: Altruistic Matching Heuristic

In this section, we introduce ALMA and prove its convergence properties. We mainly focus on weighted matchings in bipartite graphs (i.e., the assignment problem, see Section

2.2 for the formal definition), but ALMA can also be applied in general graphs (see Section 2.3 for the formal definition), as will be demonstrated in Section 3.6.

Recall that in our setting we have a weighted bipartite graph, $\mathcal{G} = \{\mathcal{N} \cup \mathcal{R}, \mathcal{E}\}$, where $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of agents, $\mathcal{R} = \{1, \ldots, R\}$ the set of resources, and finally, the weight of an edge $(n, r) \in \mathcal{E}$ represents the utility ($u_n(r) \in [0, 1]$) agent n receives by acquiring resource r. Each agent can acquire at most one resource, and each resource can be assigned to at most one agent. Each agent $n \in \mathcal{N}$ can possibly be interested in only a subset of the total resources, i.e., $\mathcal{R}^n \subset \mathcal{R}$. The goal is to maximize the social welfare (i.e., sum of utilities).

For simplicity, and unless stated otherwise, we assume $N = R$. This is *not required* by ALMA. If $R > N$ some resources will remain free, while if $N > R$ some agents will fail to acquire a resource (convergence in the latter case implies that the state of the agent does not change, as we will explain later).

3.3.1 Learning Rule

We make the following two assumptions: First, we assume (possibly noisy) knowledge of personal utilities by each agent. Second, we assume that agents can observe feedback from their environment. This is used to inform collisions and detect free resources. It could be achieved by the use of visual, auditory, olfactory sensors etc., or by any other means of feedback from the resource (e.g., by sending an occupancy message). Note that such messages would be between the requesting agent and the resource, not between the participating agents themselves, and that it suffices to send only a single bit of information (e.g., 0, 1 for occupied / free respectively).

ALMA learns the right action through repeated trials as follows. Each agent sorts his available resources (possibly $\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $(r_1, \ldots, r_i, r_{i+1}, \ldots, r_{R^n})$. The set of available actions is denoted as $\mathcal{A} = \{Y, A_{r_1}, \ldots, A_{r_{R^n}}\}$, where Y refers to yielding, and A_r refers to accessing resource r. Each agent has a strategy (g_n) that points to a resource and it is initialized to the most preferred one. As long as an agent has not acquired a resource yet, at every time-step, there are two possible scenarios. If $g_n = A_r$ (strategy points to resource r), then agent n attempts to acquire that resource (line 5 of Algorithm 1). If there is a collision, the colliding parties back-off (set $g_n \leftarrow Y$) with some probability. Otherwise, if $g_n = Y$, the agent chooses a resource r for monitoring (line 12 of Algorithm 1). If the resource is free, he sets $g_n \leftarrow A_r$. Algorithm 1 presents the pseudo-code of ALMA, which is followed by every agent independently and in parallel. The back-off probability ($P_n(\cdot)$) and the next resource to monitor ($S_n(\cdot)$) are computed individually and locally based on the current resource and each agent's utilities, as will be explained in the following section. Finally, note that if the available resources change over time, the agents simply need to sort again the currently available ones.

Algorithm 1 ALMA: Altruistic Matching Heuristic.

1: Sort resources ($\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $r_1, \ldots, r_i, r_{i+1}, \ldots, r_{R^n}$
2: Initialize $g_n \leftarrow A_{r_1}$, $r_{\text{prev}} \leftarrow r_1$, $converged \leftarrow False$
3: **procedure** ALMA
4: **while** !converged **do**
5: **if** $g_n = A_r$ **then**
6: Agent n attempts to acquire resource r
7: Set $r_{\text{prev}} \leftarrow r$
8: **if** Collision(r) **then**
9: Back-off (set $g_n \leftarrow Y$) with probability $P_n(r, \prec_n)$
10: **else**
11: converged $\leftarrow True$
12: **else** ($g_n = Y$)
13: Agent n monitors resource $r \leftarrow S_n(r_{\text{prev}}, \prec_n)$
14: Set $r_{\text{prev}} \leftarrow r$
15: **if** Free(r) **then**
16: Set $g_n \leftarrow A_r$
17: Output r, such that $g = A_r$

3.3.2 Back-off Probability & Resource Selection

Let $\mathcal{R}$ be totally ordered in decreasing utility under $\prec_n$, $\forall n \in \mathcal{N}$. If more than one agent compete for resource r_i (step 8 of Algorithm 1), each of them will back-off with probability that depends on their utility loss of switching to their respective remaining resources. The loss is given by Equation 3.1.

$$loss_n^i = \frac{\sum_{j=i+1}^{k} (u_n(r_i) - u_n(r_j))}{k - i} \tag{3.1}$$

where $k \in \{i+1, \ldots, R^n\}$ denotes the number of remaining resources to be considered. For $k = i+1$, the formula only takes into account the utility loss of switching to the immediate next best resource, while for $k = R^n$ it takes into account the average utility loss of switching to all of the remaining resources. In the remainder of this chapter we assume $k = i+1$, i.e., $loss_n^i = u_n(r_i) - u_n(r_{i+1})$. The actual back-off probability can be computed with any monotonically decreasing function f on $loss_n$, i.e., $P_n(r_i, \prec_n) = f_n(loss_n^i)$. In the evaluation section, we have used two such functions, a linear function (Equation 3.2), and the logistic function (Equation 3.3). The parameter ϵ places a threshold on the minimum / maximum back-off probability for the linear curve, while γ determines the

steepness of the logistic curve.

$$f(loss) = \begin{cases} 1-\epsilon, & \text{if } loss \leq \epsilon \\ \epsilon, & \text{if } 1-loss \leq \epsilon \\ 1-loss, & \text{otherwise} \end{cases} \tag{3.2}$$

$$f(loss) = \frac{1}{1+e^{-\gamma(0.5-loss)}} \tag{3.3}$$

Using the aforedescribed rule, agents that do not have good alternatives will be less likely to back-off and vice versa. The ones that do back-off select an alternative resource and examine its availability (line 13 of Algorithm 1). The resource selection is performed in sequential order, i.e., $S_n(r_{\text{prev}}, \prec_n) = r_{\text{prev}+1}$, where r_{prev} denotes the resource selected by that agent in the previous round. We also examined using a weighted or uniformly at random selection, but achieved inferior results in terms of social welfare. Yet, such adding randomness to the selection method can be beneficial in other objectives, such as privacy, as we will demonstrate in Chapter 4.

3.3.3 Altruism-Inspired Behavior

Human cooperation is unique in the sense that humans cooperate with strangers, even if there are no prospects of future interactions or reputation gains (Fehr and Rockenbach, 2004). ALMA is inspired by the human principle of altruism (Nowak and Sigmund, 2005; Gintis, 2000). We would expect an altruistic person to give up a resource either to someone who values it more, if that resulted in an improvement of the well-being of society – which does not imply knowledge of the preferences of others, rather than a general expectation – (Charness and Rabin, 2002), or simply to be nice to others (Simon, 2016). Such behavior is especially common in situations where the backing-off subject has equally good alternatives. For example, in human pick-up teams, each player typically attempts to fill his most preferred position. If there is a collision, a colliding player might back-off because his teammate is more competent in that role, or because he has an equally good alternative, or simply to be polite; the player backs-off now and assumes that role at some future game. From an alternative viewpoint, following such an altruistic convention leads to a faster convergence which might outweigh the loss in utility. Such conventions allow humans to routinely and robustly coordinate in large scale, and under dynamic and unpredictable demand. Behavioral conventions are a fundamental part of human societies (Lewis, 2008), yet they have not appeared meaningfully in empirical modeling of multi-agent systems. Inspired by human behavior, ALMA attempts to reproduce these simple rules in an artificial setting.

3.3.4 Convergence

Theorem 1. For N agents and R resources, the expected number of steps until the system of agents following Algorithm 1 converges to a complete matching is bounded by (3.4), where $p^* = f(loss^*)$, and $loss^*$ is given by the Equation 3.5.

$$\mathcal{O}\left(R\frac{2-p^*}{2(1-p^*)}\left(\frac{1}{p^*}\log N + R\right)\right) \tag{3.4}$$

$$loss^* = \arg\min_{loss_n^r}\left(\min_{r\in\mathcal{R},n\in\mathcal{N}}(loss_n^r), 1 - \max_{r\in\mathcal{R},n\in\mathcal{N}}(loss_n^r)\right) \tag{3.5}$$

Proof. (Sketch)

We begin by providing a sketch of the proof; the complete proof follows directly after. The proof is an adaptation of the convergence proof of (Cigler and Faltings, 2013) (and the subsequent adaptation of (Danassis and Faltings, 2019)).

We first assume that every agent, on every collision, backs-off with the same constant probability p. We start with the case of having N agents competing for 1 resource and model our system as a discrete time Markov chain. Intuitively, this Markov chain describes the number of individuals in a decreasing population, but with two caveats: the goal (absorbing state) is to reach a point where only one individual remains, and if we reach zero, we restart. We prove that the expected number of steps until we reach a state where either 1 or 0 agents compete for that resource is $\mathcal{O}\left(\frac{1}{p}\log N\right)$. Moreover, we prove that with high probability, $\Omega\left(\frac{2(1-p)}{2-p}\right)$, only 1 agent will remain (contrary to reaching 0 and restarting the process of claiming the resource), no matter the initial number of agents. Having proven that, we move to the general case of N agents competing for R resources.

At any time, at most N agents can compete for each resource. We call this period a round. During a round, the number of agents competing for a specific resource monotonically decreases, since that resource is perceived as occupied by non-competing agents. Let the round end when either 1 or 0 agents compete for the resource. This will require $\mathcal{O}\left(\frac{1}{p}\log N\right)$ steps. If all agents backed-off, it will take on average R steps until at least one of them finds a free resource. We call this period a break. In the worst case, the system will oscillate between a round and a break. According to the above, one oscillation requires in expectation $\mathcal{O}\left(\frac{1}{p}\log N + R\right)$ steps. If $R = 1$, as mentioned in the previous paragraph, in expectation there will be $\frac{2-p}{2(1-p)}$ oscillations. For $R > 1$ the expected number of oscillations is bounded by $\mathcal{O}\left(R\frac{2-p}{2(1-p)}\right)$. Thus, we conclude that if all the agents back-off with the same constant probability p, the expected number of steps until the system converges to a complete matching is $\mathcal{O}\left(R\frac{2-p}{2(1-p)}\left(\frac{1}{p}\log N + R\right)\right)$.

Next, we drop the constant probability assumption. Intuitively, the worst case scenario corresponds to either all agents having a small back-off probability, thus they keep on competing for the same resource, or all of them having a high back-off probability, thus the process will keep on restarting. These two scenarios correspond to the inner ($\frac{1}{p}$) and outer ($\frac{2-p}{2(1-p)}$) probability terms of bound (3.4) respectively. Let $p^* = f(loss^*)$ be the worst between the smallest or highest back-off probability any agent $n \in \mathcal{N}$ can exhibit, i.e., having $loss^*$ given by Equation 3.5. Using p^* instead of the constant p, we bound the expected convergence time according to bound (3.4). □

Proof. (Complete)

To facilitate the proof, we will initially assume that every agent, on every collision, backs-off with the same constant probability, i.e.,:

$$P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R} \tag{3.6}$$

Case #1: Multiple Agents, Single resource ($R = 1$)

We will describe the execution of ALMA as a discrete time Markov chain (DTMC).[2] In every time-step, each agent performs a Bernoulli trial with probability of 'success' $1 - p$ (remain in the competition), and failure p (back-off). When N agents compete for a single resource, the state of the system is a vector $\{0, 1\}^N$ denoting the individual agents that still compete for that resource. But, since the back-off probability is the same for everyone (Equation 3.6), we are only interested in how many agents are competing and not which ones. Thus, in the single resource case ($R = 1$), we can describe the execution of the proposed algorithm using the following chain:

Definition 2. Let $\{X_t\}_{t \geq 0}$ be a DTMC on state space $S = \{0, 1, \ldots, N\}$ denoting the number of agents still competing for the resource. The transition probabilities are as follows:

$$\begin{aligned}
&Pr(X_{t+1} = N | X_t = 0) = 1 && \text{restart} \\
&Pr(X_{t+1} = 1 | X_t = 1) = 1 && \text{absorbing} \\
&Pr(X_{t+1} = j | X_t = i) = \binom{i}{j} p^{i-j}(1-p)^j && i > 1, j \leq i
\end{aligned}$$

(all the other transition probabilities are zero)

[2] For an introduction on Markov chains see (Norris, 1998)

Intuitively, this Markov chain describes the number of individuals in a decreasing population, but with two caveats: The goal (absorbing state) is to reach a point where only one individual remains, and if we reach zero, we restart.

Before proceeding with Theorem 1's convergence proof, we will restate Mityushin's Theorem (Rego, 1992) for hitting time bounds in Markov chains, define two auxiliary DTMCs, and prove two auxiliary lemmas.

Theorem 2. (Mityushin's Theorem (Rego, 1992)) Let $A = \{0\}$ be the absorbing state of a Markov chain $\{X_t\}_{t\geq 0}$. If $\mathbb{E}(X_{t+1}|X_t = i) < \frac{i}{\beta}$, $\forall i \geq 1$ and some $\beta > 1$, then:

$$\mathbb{E}(T_i^A) < \lceil \log_\beta i \rceil + \frac{\beta}{\beta - 1} \tag{3.7}$$

where T_i^A denotes the hitting time of a state in A, starting from state i.

Definition 3. Let $\{Y_t\}_{t\geq 0}$ be a DTMC on state space $S = \{0, 1, \ldots, N\}$ with the following transition probabilities (two absorbing states, 0 and 1):

$$\begin{aligned} &Pr(Y_{t+1} = 0|Y_t = 0) = 1 && \text{absorbing} \\ &Pr(Y_{t+1} = 1|Y_t = 1) = 1 && \text{absorbing} \\ &Pr(Y_{t+1} = j|Y_t = i) = \binom{i}{j} p^{i-j}(1-p)^j && i > 1, j \leq i \end{aligned}$$

(all the other transition probabilities are zero)

Definition 4. Let $\{Z_t\}_{t\geq 0}$ be a DTMC on state space $S = \{0, 1, \ldots, N\}$ with the following transition probabilities (state 0 the only absorbing state):

$$\begin{aligned} &Pr(Z_{t+1} = 0|Z_t = 0) = 1 && \text{absorbing} \\ &Pr(Z_{t+1} = j|Z_t = i) = \binom{i}{j} p^{i-j}(1-p)^j && i \geq 1, j \leq i \end{aligned}$$

(all the other transition probabilities are zero)

Lemma 1. The expected hitting time of the set of absorbing states $A = \{0\}$, starting from state $Z_0 = N$, of the DTMC $\{Z_t\}$ of Definition 4 is bounded by $\mathcal{O}\left(\frac{1}{p} \log N\right)$.

Proof. If the DTMC $\{Z_t\}$ is in state $Z_t = i$, the next state Z_{t+1} is drawn from a binomial distribution with parameters $(i, 1-p)$. Thus, the expected next state is $\mathbb{E}(Z_{t+1}|Z_t = i) = i(1-p)$. Using Theorem 2 with $\beta = \frac{1}{1-p}$ results in the required bound:

$$\mathbb{E}(T_N^A) = \mathcal{O}\left(\frac{1}{p}\log N\right) \tag{3.8}$$

□

Corollary 1. The expected hitting time of the set of absorbing states $A = \{0, 1\}$, starting from state $Y_0 = N$, of the DTMC $\{Y_t\}$ of Definition 3 is bounded by $\mathcal{O}\left(\frac{1}{p}\log N\right)$.

Proof. The expected hitting time of the absorbing state of $\{Z_t\}$ is an upper bound of the expected hitting time of $\{Y_t\}$. This is because any path that leads into state 0 in $\{Z_t\}$ either does not go through state 1 (thus happens with the same probability as in $\{Y_t\}$), or goes through state 1. But, state 1 in $\{Y_t\}$ is an absorbing state, hence in the latter case the expected hitting time for $\{Y_t\}$ would be one step shorter. □

Let h_i^A denote the hitting probability of a set of states A, starting from state i. We will prove the following lemma.

Lemma 2. The hitting probability of the absorbing state $\{1\}$, starting from any state $i \geq 1$, of the DTMC $\{Y_t\}$ of Definition 3 is given by Equation 3.9. This is a tight lower bound.

$$h_i^{\{1\}} = \Omega\left(\frac{2(1-p)}{2-p}\right), \forall i \geq 1 \tag{3.9}$$

Proof. For simplicity we denote $h_i \stackrel{\Delta}{=} h_i^{\{1\}}$. We will show that for $p \in (0, 1)$, $h_i \geq \lambda = \frac{2(1-p)}{2-p}, \forall i \geq 1$ using induction. First note that since state $\{0\}$ is an absorbing state, $h_0 = 0$, $h_1 = 1 \geq \lambda$ and that $\lambda \in (0, 1)$.

The vector of hitting probabilities $h^A = (h_i^A : i \in S = \{0, 1, \ldots, N\})$ for a set of states A is the minimal non-negative solution to the system of linear equations (3.10):

$$\begin{cases} h_i^A = 1, & \text{if } i \in A \\ h_i^A = \sum\limits_{j \in S} p_{ij} h_j^A, & \text{if } i \notin A \end{cases} \tag{3.10}$$

By replacing p_{ij} with the probabilities of Definition 3, the system of equations (3.10) becomes:

$$\begin{cases} h_i^A = 1, & \text{if } i \in A \\ h_i^A = \sum\limits_{j=0}^{i} \binom{i}{j} p^{i-j}(1-p)^j h_j^A, & \text{if } i \notin A \end{cases} \tag{3.11}$$

Base case:

$$h_2 = (1-p)^2 h_2 + 2p(1-p)h_1 + p^2 h_0 = \frac{2p(1-p)}{1-(1-p)^2} = \frac{2(1-p)}{2-p} \geq \lambda$$

Inductive step: We assume that $\forall j \leq i-1 \Rightarrow h_j \geq \lambda$. We will prove that $h_i \geq \lambda, \forall i > 2$.

$$\begin{aligned}
h_i &= \sum_{j=0}^{i} \binom{i}{j} p^{i-j}(1-p)^j h_j \\
&= p^i h_0 + ip^{i-1}(1-p)h_1 + \sum_{j=2}^{i-1} \binom{i}{j} p^{i-j}(1-p)^j h_j + (1-p)^i h_i \\
&\geq p^i h_0 + ip^{i-1}(1-p)h_1 + \sum_{j=2}^{i-1} \binom{i}{j} p^{i-j}(1-p)^j \lambda + (1-p)^i h_i \\
&= ip^{i-1}(1-p) + [1 - p^i - (1-p)^i - ip^{i-1}(1-p)]\lambda + (1-p)^i h_i \\
\Rightarrow h_i &= \frac{ip^{i-1}(1-p) + [1 - p^i - (1-p)^i - ip^{i-1}(1-p)]\lambda}{1-(1-p)^i} \\
\Rightarrow h_i &= \lambda - \frac{p^i}{1-(1-p)^i}\lambda + \frac{ip^{i-1}(1-p)}{1-(1-p)^i}(1-\lambda)
\end{aligned}$$

We want to prove that:

$$\begin{aligned}
h_i \geq \lambda \Rightarrow \\
\frac{ip^{i-1}(1-p)}{1-(1-p)^i}(1-\lambda) \geq \frac{p^i}{1-(1-p)^i}\lambda \Rightarrow \\
ip^{i-1}(1-p) \geq [p^i + ip^{i-1}(1-p)]\lambda \Rightarrow \\
\frac{ip^{i-1}(1-p) + p^i - p^i}{p^i + ip^{i-1}(1-p)} \geq \lambda \Rightarrow \\
1 - \frac{p^i}{p^i + ip^{i-1}(1-p)} \geq \lambda \Rightarrow \\
1 - \frac{p^i}{p^i + ip^{i-1}(1-p)} \geq \frac{2(1-p)}{2-p} \Rightarrow \\
1 - \frac{p^i}{p^i + ip^{i-1}(1-p)} \geq 1 - \frac{p}{2-p} \Rightarrow \\
\frac{p^i}{p^i + ip^{i-1}(1-p)} \leq \frac{p}{2-p} \Rightarrow \\
p^i(2-p) \leq p[p^i + ip^{i-1}(1-p)] \Rightarrow \\
p^i(2-p) \leq p^i[p + i(1-p)] \Rightarrow \\
2 - 2p - i + ip \leq 0 \Rightarrow \\
2 - i - p(2-i) \leq 0 \Rightarrow \\
(2-i)(1-p) \leq 0 \Rightarrow \\
2 - i \leq 0
\end{aligned}$$

which holds since $i > 2$.

The above bound is also tight since $\exists i \in S : h_i = \lambda$, specifically $h_2 = \lambda$. □

Now we can prove the following theorem that bounds the convergence time of the DTCM of Definition 2, which models the execution of ALMA for the case of a single available resource ($R = 1$) and constant back-off probability.

Theorem 3. The expected hitting time of the set of absorbing states $A = \{1\}$ of the DTMC $\{X_t\}$ of Definition 2, starting from any initial state $X_0 \in \{0, 1, \ldots, N\}$, is bounded by:

$$\mathcal{O}\left(\frac{2-p}{2p(1-p)}\log N\right) \tag{3.12}$$

Proof. Using Lemma 2 we can derive that the DTMC $\{X_t\}$ needs in expectation $\frac{1}{\lambda} = \frac{2-p}{2(1-p)}$ passes until it hits state 1. Each pass requires $\mathcal{O}\left(\frac{1}{p}\log N\right)$ steps (Corollary 1). Thus, the expected hitting time of state $A = \{1\}$ is $\mathcal{O}\left(\frac{2-p}{2p(1-p)}\log N\right)$. □

Case #2: Multiple Agents, Multiple resources ($R > 1$)

Theorem 4. For N agents and R resources, assuming a constant back-off probability for each agent, i.e., $P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R}$, the expected number of steps until the system of agents following Algorithm 1 converges to a complete matching is bounded by (3.13).

$$\mathcal{O}\left(R\frac{2-p}{2(1-p)}\left(\frac{1}{p}\log N + R\right)\right) \tag{3.13}$$

Proof. At most N agents can compete for each resource. We call this period a round. During a round, the number of agents competing for a specific resource monotonically decreases, since that resource is perceived as occupied by non-competing agents. Let the round end when either 1 or 0 agents compete for the resource. Corollary 1 states that in expectation this will require $\mathcal{O}\left(\frac{1}{p}\log N\right)$ steps.

If all agents backed-off, it will take on average R steps until at least one of them finds a free resource. We call this period a break.

In the worst case, the system will oscillate between a round and a break. According to the above, one oscillation requires in expectation $\mathcal{O}\left(\frac{1}{p}\log N + R\right)$ steps. If $R = 1$, Lemma 2 states that in expectation there will be $\frac{1}{\lambda} = \frac{2-p}{2(1-p)}$ oscillations. For $R > 1$ the expected number of oscillations is bounded by $\mathcal{O}\left(R\frac{2-p}{2(1-p)}\right)$. Thus, we derive the required bound (3.13). □

Dynamic Back-off Probability

So far we have assumed a constant back-off probability for each agent, i.e., $P_n(r, \prec_n) = p > 0, \forall n \in \mathcal{N}, \forall r \in \mathcal{R}$. In this section we will drop this assumption. Let $\psi = \max(\log N, R)$. Bound (3.13) becomes:

$$\mathcal{O}\left(\psi^2 \frac{2-p}{2(1-p)}\left(\frac{1}{p}+1\right)\right) \tag{3.14}$$

Intuitively, the worst case scenario corresponds to either all agents having a small back-off probability, thus they keep on competing for the same resource, or all of them having a high back-off probability, thus the process will keep on restarting. These two scenarios correspond to the inner ($\frac{1}{p}$) and outer ($\frac{2-p}{2(1-p)}$) probability terms of bound (3.14), respectively. We can rewrite the right part of bound (3.14) as:

$$\frac{2-p}{2(1-p)}\left(\frac{1}{p}+1\right) = \frac{1}{p} + \frac{1}{1-p} + \frac{1}{2} = \tau \tag{3.15}$$

As seen by Equation 3.15, τ assumes its maximum value on the two extremes, either with a high ($p \to 1^-$), or a low ($p \to 0^+$) back-off probability, i.e., $\lim_{p\to 1^-} \tau = \lim_{p\to 0^+} \tau = \infty$. Let $p^* = f(loss^*)$ be the worst between the smallest or highest back-off probability any agent $n \in \mathcal{N}$ can exhibit, i.e., having $loss^*$ given by Equation 3.17. Using p^* instead of the constant p, we bound the expected convergence time of the system of agents according to bound (3.16).

$$\mathcal{O}\left(R\frac{2-p^*}{2(1-p^*)}\left(\frac{1}{p^*}\log N + R\right)\right) \tag{3.16}$$

$$loss^* = \arg\min_{loss_n^r}\left(\min_{r\in\mathcal{R}, n\in\mathcal{N}}(loss_n^r), 1 - \max_{r\in\mathcal{R}, n\in\mathcal{N}}(loss_n^r)\right) \tag{3.17}$$

This concludes the proof of Theorem 1. □

It is worth noting that the back-off probability p^* in bound (3.4) does not significantly affect the convergence time. For example, using Equation 3.2 with a quite small $\epsilon = 0.01$, the resulting quantities would be at most $100R\log N$, and $50R^2$. Most importantly, though, this is a rather loose bound (e.g., agents would rarely back-off with probabilities as extreme as p^*).

Apart from the convergence of the whole system, we are interested in the expected number of steps any individual agent would require in order to acquire a resource. In real-world scenarios, there is typically a cost associated with acquiring a resource. For example, a taxi driver would not be willing to drive to the other end of the city to pick up a low fare passenger. As a result, each agent is typically interested in a subset of the total resources, i.e., $\mathcal{R}^n \subset \mathcal{R}$, thus at each resource there is a bounded number of competing

agents. Let R^n denote the maximum number of resources agent n is interested in, and N^r denote the maximum number of agents competing for resource r. By bounding these two quantities (i.e., we consider R^n and N^r to be constant functions of N, R), Corollary 2 proves that the expected number of steps any individual agent requires to converge[3] is independent of the total problem size (i.e., N, and R), or, in other words, that the convergence time is *constant* in these quantities.

Corollary 2. Let $R^n = |\mathcal{R}^n|$, such that $\forall r \in \mathcal{R}^n : u_n(r) > 0$, and $N^r = |\mathcal{N}^r|$, such that $\forall n \in \mathcal{N}^r : u_n(r) > 0$. The expected number of steps until an agent $n \in \mathcal{N}$ following Algorithm 1 converges is bounded by (3.18), where $p_n^* = f(loss^\star)$ and $loss^\star$ is given by Equation 3.19, independent of the total problem size N, R.

$$\mathcal{O}\left(\max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} R^{n'} \frac{2 - p_n^*}{2(1 - p_n^*)} \left(\frac{1}{p_n^*} \log(\max_{r \in \mathcal{R}^n} N^r) + \max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} R^{n'}\right)\right) \tag{3.18}$$

$$loss^\star = \arg\min_{loss_n^r} \left(\min_{r \in \mathcal{R}^n, n \in \mathcal{N}^r} (loss_n^r), 1 - \max_{r \in \mathcal{R}^n, n \in \mathcal{N}^r} (loss_n^r)\right) \tag{3.19}$$

Proof. The expected number of steps until an agent $n \in \mathcal{N}$ successfully acquires a resource is upper bounded by the total convergence time of the sub-system he belongs to, i.e., the sub-system consisting of the sets of $\mathcal{R}^n$ resources and $\cup_{r \in \mathcal{R}^n} \mathcal{N}^r$ agents. In such scenario, at most $\max_{r \in \mathcal{R}^n} N^r$ agents can compete for any resource. Using Theorem 1 for $\max_{r \in \mathcal{R}^n} N^r$ agents, $\max_{n' \in \cup_{r \in \mathcal{R}^n} \mathcal{N}^r} R^{n'}$ resources, and worst-case $loss^\star$ given by any agent in $\cup_{r \in \mathcal{R}^n} \mathcal{N}^r$ (i.e., Equation 3.19) results in the desired bound. Note that agents do not compete for already claimed resources (step 13 of Algorithm 1), thus the convergence of an agent does not require the convergence of agents of overlapping sub-systems. □

3.4 Evaluation Results: Uniform, and Noisy Common Preferences

3.4.1 Setting

As a first evaluation test case, we cover the extreme scenarios. The first pertains to an anti-coordination scenario in which agents with similar preferences compete over the same set of actions. For example, autonomous vehicles would prefer the least congested route, bidding agents participating in multiple auctions would prefer the ones with the smallest number of participants, etc. We call this scenario 'noisy common preferences' and model the utilities as follows: $\forall n, n' \in \mathcal{N}, |u_n(r) - u_{n'}(r)| \leq$ noise, where the noise is sampled from a zero-mean Gaussian distribution, i.e., noise $\sim \mathcal{N}(0, \sigma^2)$.[4] In the second scenario the utilities are initialized uniformly at random (UaR) for each agent and resource.

[3] In scenarios where $R < N$, or in case of deadlocks, convergence implies that the state of the agent does not change.

[4] Similar results were achieved using uniform noise, i.e., $\sim \mathcal{U}(-\nu, \nu)$.

3.4.2 Metrics and Setup

In this test-case, we focus on the convergence time and the relative difference in social welfare (SW), i.e., $(achieved - optimal)/optimal$. In every reported metric, except for the social welfare, we report the average value out of 128 runs of the same problem instance. Error bars represent one standard deviation (SD) of uncertainty. For the social welfare, we report the cumulative regret of the aforementioned 128 runs, i.e., for $i \in [1, 128]$ runs, the reported relative difference in social welfare is $(\sum_i achieved - \sum_i optimal)/\sum_i optimal$. This was done to improve visualization of the results in smaller problem sizes, where really small differences result in high SD bars (e.g., if $achieved = 1 \times 10^{-5}$, and $optimal = 2 \times 10^{-5}$, the relative difference would be -50% for practically the same matching). The optimal matchings were computed using the Hungarian algorithm.[5] All the simulations were run on 2x Intel Xeon E5-2680 with 256 GB RAM.

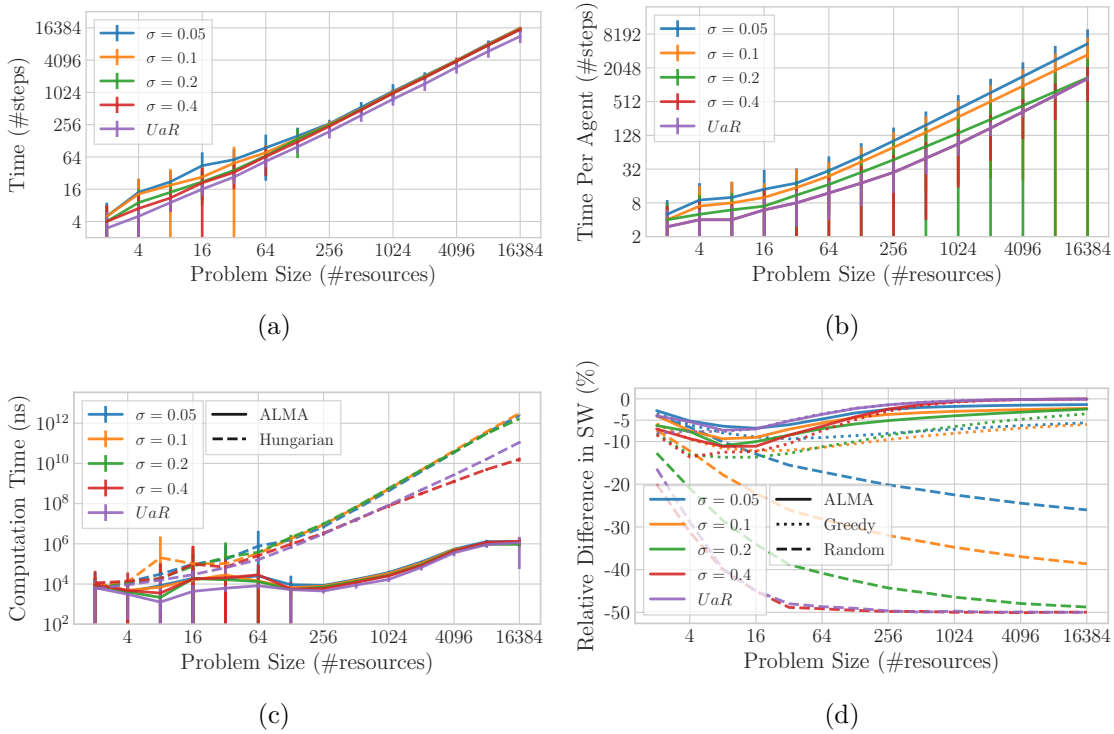
It is important to stress that our goal is not to improve the convergence speed of a centralized, or decentralized algorithm. Rather, the computation time comparisons in this and the following sections are meant to ground the actual speed of ALMA, and argue in favor of its applicability on large-scale, real-world scenarios. Given the nature of the problem, we elected to use a specialized algorithm to compute the optimal solution, rather than a general LP-based technique (e.g., the Simplex method). Specifically, we opted to use the Hungarian algorithm which, first, has proven polynomial worse case bound, and second, as our simulations will demonstrate, can handle sufficiently large problems.

We also compare against two baselines as a reference: a greedy algorithm, and the random solution. The greedy solution goes through the participating agents randomly, and assigns them their most preferred unassigned resource. Finally, for this evaluation, ALMA agents use the logistic function (Equation 3.3) with $\gamma = 2$ to calculate the back-off probability.

3.4.3 Convergence Time

Starting with Figure 3.1a, we can see that the convergence time for the system of agents following Algorithm 1 is linear to the number of resources R. From the perspective of a single agent, Figure 3.1b shows that on average he will successfully acquire a resource significantly ($> 2\times$) faster than the total convergence time. This suggest that there is a small number of agents which take longer in claiming a resource and which in turn delay the system's convergence. We will exploit this property in the next section to present the anytime property of ALMA. Figure 3.1c shows that ALMA requires approximately 4 to 6 orders of magnitude less computation time than the centralized Hungarian algorithm. Furthermore, ALMA seems to scale more gracefully, an important property for real world

[5] We used Kevin L. Stern's $\mathcal{O}(N^3)$ implementation: https://github.com/KevinStern/.



(a) (b) (c) (d)

Figure 3.1: From left to right, top to bottom:
(3.1a) Total convergence time (#steps) for the entire system of agents,
(3.1b) Average time (#steps) for an individual agent to successfully acquire a resource,
(3.1c) Computation time (ns) until the entire system of agents converges, and
(3.1d) Relative difference in social welfare (%),
for increasing number of resources, and $N = R$.
Figures 3.1a, 3.1c, and 3.1b are in double logarithmic scale, while Figure 3.1d is in single logarithmic scale.

applications. Note also that in real-world applications we would have to take into account communication time, communication reliability protocols, etc., which create additional overhead for the Hungarian or any other algorithm for the assignment problem.

3.4.4 Efficiency

The relative difference in social welfare (Figure 3.1d) reaches asymptotically zero as R increases. For a small number of resources, ALMA achieves the worst social welfare, approximately 11% worse than the optimal. Intuitively this is because when we have a small number of choices, a single wrong matching can have a significant impact to the final social welfare, while as the number of resources grow, the impact of an erroneous matching is mitigated. For 16384 resources we lose less than 2.5% of the optimal. As a reference, Figure 3.1d depicts the centralized greedy, and the random solutions as well. In this scenario, the random solution loses up to 50% of the optimal social welfare.

The greedy solution is up to 6.2% worse than ALMA (for $R = 128$ and $\sigma = 0.1$), but we see that this difference decreases as the problem size increases ($R > 256$); achieving at the end similar results to ALMA, especially in high noise situations. This is to be expected, since first, all agents are interested in all the resources, and second, as the noise increases, the agents' preferences become more distinguishable, more diverse. ALMA is of a greedy nature as well, albeit it utilizes a more intelligent backing-off scheme. Contrary to that, the greedy solution does not take into account the utilities between agents, thus there are scenarios where ALMA would significantly outperform the greedy (e.g., for lower noise, and lower number of resources in this test-case, or the test-cases in the following sections). Finally, recall that ALMA operates in a significantly harder domain with no communication, limited feedback, and time constraints. In contrast, the greedy method requires either a central coordinator or message exchange (to communicate users' preferences and resolve collisions).

3.5 Evaluation Results: Resource Allocation in an Urban Environment

Adopting a simple rule allows the applicability of ALMA to large scale multi-agent systems. In this section we will analyze such a scenario. Specifically we are interested in resource allocation in urban environments (e.g., parking spots / charging stations for autonomous vehicles, taxi - passenger matchings, etc.). The aforementioned problems become ever more relevant due to rapid urbanization, and the natural lack of coordination in the usage of resources (Varakantham, 2016). The latter result in the degradation of response (e.g., waiting time) and quality metrics in large cities (Varakantham, 2016).

3.5.1 Setting

Let us consider a Cartesian map representing a city on which are randomly distributed vehicles and charging stations, as depicted in Figure 3.2f. The utility received by a vehicle n for using a charging station r is proportional to the inverse of their distance, i.e., $u_n(r) = 1/d_{n,r}$. Since we are in an urban environment, let $d_{n,r}$ denote the Manhattan distance. Typically, there is a cost each agent is willing to pay to drive to a resource, thus there is a cut-off distance, upon which the utility of acquiring the resource is zero (or possibly negative). This is a typical scenario encountered in resource allocation in urban environments, where there are spatial constraints and local interactions.

The way such problems are typically tackled, is by dividing the map to sub-regions, and solving each individual sub-problem. For example, Singapore is divided into 83 zones based on postal codes (Cheng and Nguyen, 2011), and taxi drivers' policies are optimized according to those (Nguyen et al., 2017; Varakantham et al., 2012). On the other hand, not placing bounds means that the current solutions will not scale. To the

best of our knowledge, we are the first to propose an anytime heuristic for resource allocation in urban environments that can scale in constant time without the need to artificially split the problem. Instead, ALMA exploits the two typical characteristics of an urban environment: the anonymity in interactions and homogeneity in supply and demand (Varakantham, 2016) (e.g., assigning any of two equidistant charging stations to a vehicle, would typically result to the same utility). This results in a simple learning rule which, as we will demonstrate in this section, can scale to hundreds of thousands of agents (we simulated up to 131072 agents and as many resources).

3.5.2 Metrics and Setup

The evaluation metrics and baselines are the same as in Section 3.4.2, with the only difference being that in this section ALMA agents use the linear function (Equation 3.2) with $\epsilon = 0.1$ to calculate the back-off probability.

3.5.3 Convergence Time

In this test-case, we placed a bound on the maximum number of resources each agent is interested in (i.e., the utility is zero for the rest), and on the maximum number of agents competing for a resource, specifically $R^n = N^r \in \{8, 16, 32, 64, 128\}$. According to Corollary 2, bounding these two quantities should result in convergence in constant time, regardless of the total problem size (R, N). The latter is corroborated by Figure 3.2a, which shows that the average number of time-steps until an agent successfully claims a resource remains constant as we increase the total problem size. Same is true for the system's convergence time (Figure 3.2b), which caps as R increases. The small increase is due to outliers, as Figure 3.2b reports the convergence time of the last agent to acquire a resource. This results to approximately 7 orders of magnitude less computation time than the centralized Hungarian algorithm (Figure 3.2c), and this number would grow boundlessly as we increase the total problem size. Moreover, as mentioned, in an actual implementation any algorithm for the assignment problem would face additional overhead due to communication time, reliability protocols, etc.

3.5.4 Efficiency

Along with the constant convergence time, ALMA is able to reach high quality matchings, achieving less than 7.5% worse social welfare (SW) than the optimal (Figure 3.2d). The latter refers to the small bound of $R^n = 8$. As observed in Section 3.4, with a small number of choices, a single wrong matching can have a significant impact to the final social welfare. By increasing the bound to a more realistic number (e.g., $R^n = 32$), we achieve less than 2.5% worse social welfare. In general, for $R > 2$ resources and different values of R^n, ALMA achieves between $1.9 - 12\%$ loss in social welfare, while the greedy

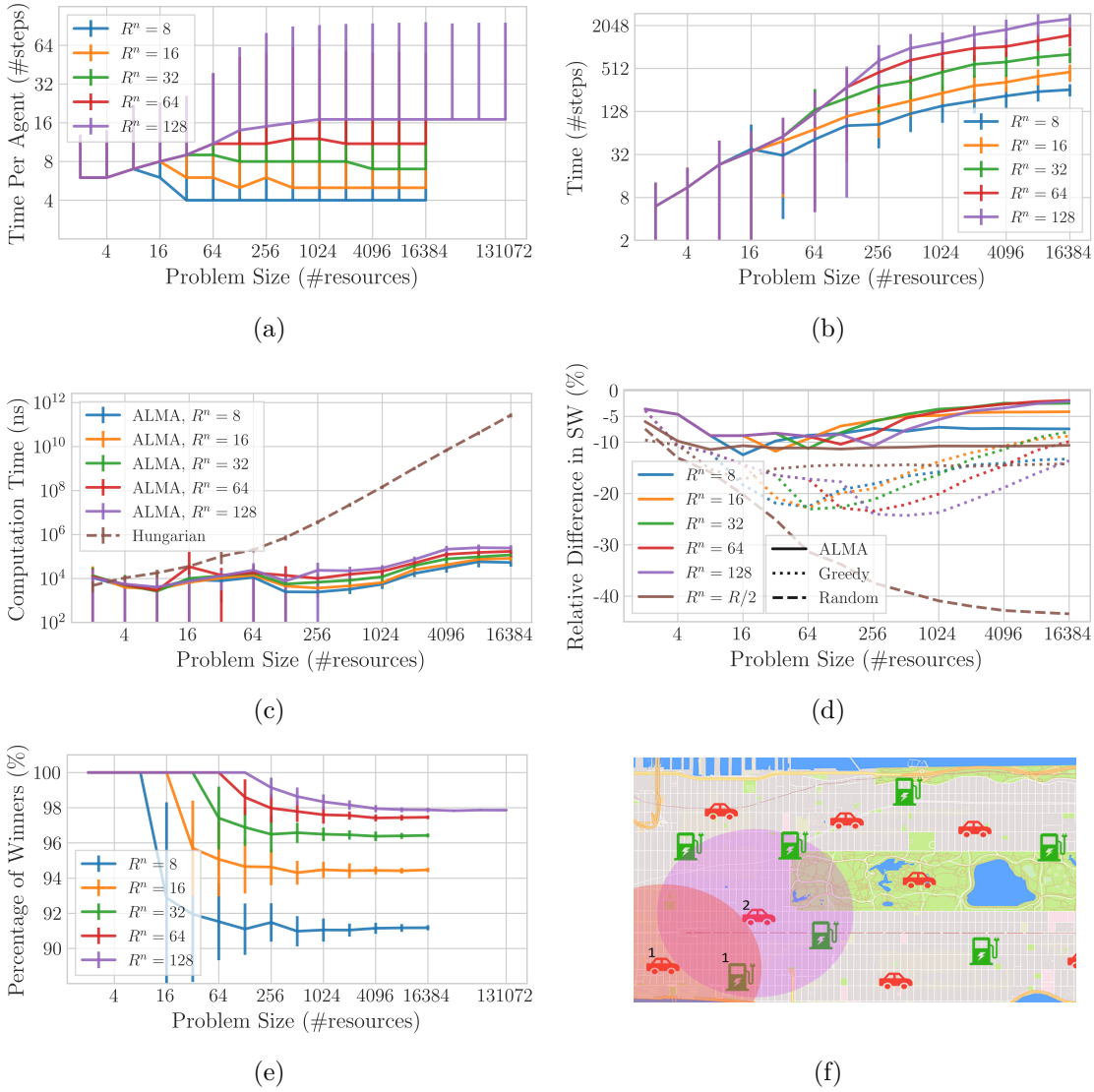


Figure 3.2: Left to right, top to bottom:
(3.2a) Average time (#steps) for an agent to acquire a resource,
(3.2b) Total convergence time (#steps) for the entire system of agents,
(3.2c) Computation time (ns) until the entire system of agents converges,
(3.2d) Relative difference in social welfare (%),
(3.2e) Percentage of 'winners',
for increasing number of resources, and $N = R$.
Figure (3.2a), (3.2b), and (3.2c) are in double logarithmic scale, while the rest are in single logarithmic scale.
Figure (3.2f) presents an example of the studied resource allocation scenario in an urban environment. We assume grid length of $\sqrt{4 \times N}$.

approach achieves $8.0 - 24\%$ and the random $7.3 - 43.4\%$. Specifically, ALMA is up to 18.1% better than greedy (the latter is for $R^n = 128$ and $R = 1024$). The behavior

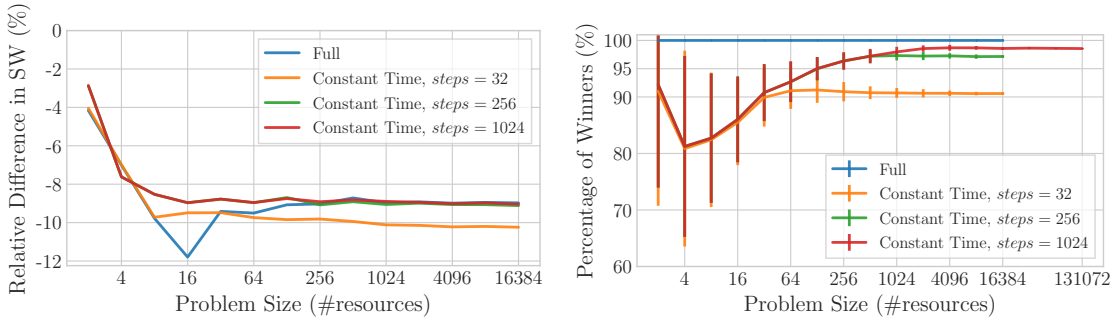
of the graphs depicted in Figure 3.2d for $R^n \in \{8, 16, 32, 64, 128\}$ indicate that, as the problem size (R) increases, the social welfare reaches its lowest value at $R = 2 \times R^n$. To investigate the latter, we have included a graph for increasing R^n (instead of constant to the problem size), specifically $R^n = R/2$. ALMA achieves a constant loss in social welfare (approximately 11%). The greedy approach achieves loss of 14%, while the random solution degrades towards 44% loss.

Compared to the evaluation test case of Section 3.4, this is a significantly harder problem for a decentralized algorithm with no communication and no global knowledge of the resources. The set of resources each agent is interested in is a proper subset of the set of the total resources, i.e., $\mathcal{R}^n \subsetneq \mathcal{R}$ (or could be $R < N$). Furthermore, the lack of communication between the participants, and the stochastic nature of the algorithm can lead to deadlocks, e.g., in Figure 3.2f, if vehicle 2 acquires resource 1, then vehicle 1 does not have an available resource in range. Nonetheless, ALMA results in an almost complete matching. Figure 3.2e, depicts the percentage of 'winners' (i.e., agents that have successfully claimed a resource r such that $u_n(r) > 0$). The aforementioned percentage refers to the total population (N) and not the maximum possible matchings (potentially $< N$). As depicted, the percentage of 'winners' is more than 90%, reaching up to 97.8% for $R^n = 128$. We also employed ALMA in *larger simulations with up to* 131072 *agents, and equal number of resources.*[6] As seen in Figure 3.2e, the percentage of winners remains stable at around 98%. Even though the size of the problem prohibited us from running the Hungarian algorithm (or an out-of-the-box LP solver) and validating the quality of the achieved matching, the fact that the percentage of winners remains the same suggests that the relative difference in social welfare will continue on the same trend as in Figure 3.2d. Moreover, the average steps per agent to claim a resource remains, as proven, constant (Figure 3.2a). The latter validate the applicability of ALMA in large scale applications with hundreds of thousands of agents.

3.5.5 Anytime Property

In the real world, agents are required to run in real time, which imposes time constraints. ALMA can be used as an anytime heuristic as well. To demonstrate the latter, we compare four configurations: the 'full' one, which is allowed to run until the systems converges, and three 'constant time' versions which are given a time budget of 32, 256, and 1024 time-steps. In this scenario, we do not impose a bound on R^n, N^r, but we assume a cut-off distance, upon which the utility is zero. The cut-off distance was set to 0.25 of the maximum possible distance, i.e., as the problem size grows, so do the R^n, N^r. On par with the evaluation test-case of Section 3.4, the full version converges in linear time. As depicted in Figure 3.3a, the achieved social welfare is less than 9% worse than the optimal. The inferior results in terms of social welfare compared to the evaluation test-case of Section 3.4 are because this is a significantly harder problem due

[6] We run for $R^n = 128$ as a representative example.



(a) (b)

Figure 3.3: (3.3a) Relative difference in social welfare (%) in interrupted execution, (3.3b) Percentage of 'winners' in interrupted execution, for increasing number of resources, and $N = R$. Both figures are in single logarithmic scale.

to the aforementioned deadlocks. On the other hand, though, ALMA benefits from the spatial constraints of the problem. The average number of time-steps an individual agent needs to successfully claim a resource is significantly smaller, which suggest that we can enforce computation time constraints. Restricting to only 32, 256, and 1024 time-steps, results in 1.25%, 0.12%, and 0.03% worse social welfare than the unrestricted version, respectively. Even in larger simulations with up to 131072 agents,[7] the percentage of winners (Figure 3.3b) remains stable at 98.6%, which suggests that the relative difference in social welfare will continue on the same trend as in Figure 3.3a (we do not suggest that this is the case in any domain. For example, in the noisy common preferences domain of Section 3.4, the quality of the achieved matching decreases boundlessly as we decrease the allotted time. Nevertheless, the aforedescribed domain is a realistic one, with a variety of real-world applications). Finally, the repeated nature of such problems suggests that even in the case of a deadlock, the agent which failed to win a resource, will do so in some subsequent round.

3.6 Evaluation Results: On-line Ridesharing Using Real Data

This evaluation test-case is part of a larger evaluation on ridesharing[8] and dynamic vehicle relocation, presented in (Danassis et al., 2022b). In this work, we study the optimization of large-scale, real-time ridesharing systems and propose a modular design methodology, Component Algorithms for Ridesharing (CAR). We evaluate a diverse set of CARs (14 in total), focusing on the *key algorithmic components* of ridesharing. First,

[7] We run for 1024 time-steps as a representative example.

[8] Throughout this thesis, we use the more common term 'ridesharing' to refer to passengers (potentially) using the same vehicle at the same time, also referred to as 'ridepooling' (Shaheen and Cohen, 2019).

it is an *online* problem, as the decisions made at some point in time clearly affect the possible decisions in the future, and therefore the the literature of online algorithms and competitive analysis (Borodin and El-Yaniv, 2005; Manasse et al., 1988) offers clear-cut candidates for CARs. Second, all of the components can be seen as some type of *matching* both for bipartite graphs (for matching passengers with taxis, or idle taxis with 'future' requests for vehicle relocation) and for general graphs (for matching passengers to shared rides). In fact, several of the algorithms that have been proposed in the literature for the problem are for different variants of online matching (e.g., (Ashlagi et al., 2017; Dickerson et al., 2018)). Finally, ridesharing displays an inherent connection to the *k-taxi problem* (Coester and Koutsoupias, 2019; Buchbinder et al., 2020; Fiat et al., 1994; Kosoresow, 1996), which, in turn, is a generalization of the well-known *k-server problem* (Koutsoupias and Papadimitriou, 1995; Koutsoupias, 2009). The fundamental idea behind the aforementioned algorithms is a pivotal part of ridesharing, as it aims to serve existing requests efficiently, but at the same time place the vehicles as well as possible to serve future requests. This is also the main principle of the relocation strategies for idle taxis. In terms of the evaluation, we take a multi-objective approach, evaluating 12 metrics related to *global efficiency*, *complexity*, *passenger*, *driver*, and *platform* incentives. Our evaluation setting is specifically designed to *closely resemble reality* in every aspect of the problem. For example, (a) we use actual data from the NYC's yellow taxi trip records, both for modeling customer requests and taxis (b) we follow closely the pricing model employed by ridesharing platforms and (c) we run our simulations to the scale of the actual problem faced by the ridesharing platforms, simulating up to 391479 requests and 12828 taxis. To the best of our knowledge, this is the *largest* and most *comprehensive* evaluation to date. A detailed description of our setting, the problem statement and modeling, the several evaluation metrics, the evaluated algorithms, as well as discussion on related work, scalability challenges, and challenges related to vehicle relocation can be found in Appendix A. In this chapter we will only briefly present a subset of the results that pertain to ALMA. The complete results of our evaluation, as well as high level analysis can also be found in Appendix A.

3.6.1 Problem Statement & Modeling

In the Ridesharing problem there is a (potentially infinite) metric space $\mathcal{X}$ representing the topology of the environment, equipped with a distance function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. Both are known in advance. At any moment, there is a (dynamic) set of available taxi vehicles, ready to serve customer requests (i.e., drive to the pick-up, and subsequently to the destination location). Between serving requests, vehicles can relocate to locations of potentially higher demand, to mitigate spatial search frictions between drivers. Customer requests appear in an online manner at their respective pick-up locations, wait to potentially be matched to a shared ride, and finally are served by a taxi to their respective destination. In order for two requests to be able to share a ride, they must satisfy *spatial*, and *temporal* constraints. The former dictates that requests should be

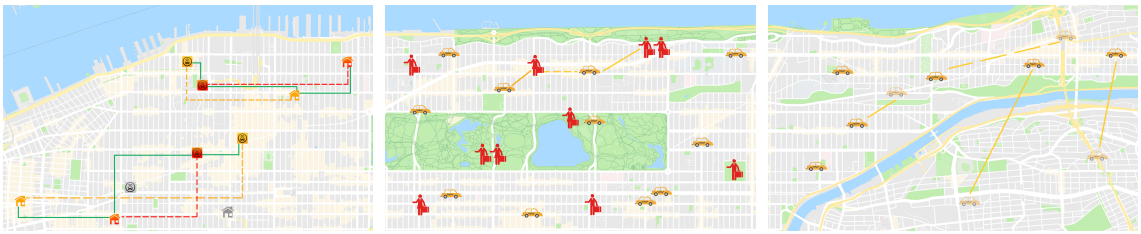
Table 3.1: Evaluated performance metrics (global, passenger (Quality of Service), driver, and platform specific).

Distance Driven	Minimize the cumulative distance driven by all vehicles for serving all the requests. We chose this objective as it directly correlates to passenger, driver, company, and environmental objectives.
Complexity	Real-world time constraints dictate that the employed solution produces results in a reasonable time-frame.
Time to Pair	Expected time to be paired in a shared ride.
Time to Pair with Taxi	Expected time to be paired with a taxi.
Time to Pick-up	Expected time to passenger pickup.
Delay	Additional travel time over the expected direct travel time (when served as a single, instead of a shared ride).
Driver Profit	Total revenue earned minus total travel costs.
Number of Shared Rides	Related to the profit. By carrying more than one passenger at a time, drivers can serve more requests in a day.
Frictions	Waiting time experienced by drivers between serving requests (i.e., time between dropping-off a ride, and getting matched with another). Lower frictions indicate lower regret by the drivers.
Platform Profit	A commission on the driver's fee, and passenger fees.
Quality of Service (QoS)	Refer to the passenger metrics. Improving the QoS to their customers correlates to the growth of the company.
Number of Shared Rides	The matching rate is important especially in the nascent stage of the platform (Dutta and Sholley, 2018).

matched only if there is good spatial overlap among their routes. Yet, due to the latter constraint, requests cannot be matched even if they have perfect spatial overlap, if they are not both 'active' at the same time. Finally, ridesharing is an inherently *online* problem, as we are unaware of the requests that will appear in the future, and need to make decisions before the requests expire, while taking into account the dynamics of the fleet of taxis. For more details, see in Appendix A.

Performance Metrics

The goal is to minimize the cumulative distance driven by the fleet of taxis, while maintaining high Quality of Service (QoS), given that we serve all requests (service guarantee). Serving all requests improves passenger satisfaction, and, most importantly, allows us to ground our evaluation to a common scenario, ensuring a fair comparison. A list and short description of the evaluated metrics can be found in Table 3.1.



(a) Request – Request (b) (Shared) Ride – Taxi (c) Vehicle Relocation

Figure 3.4: The three components of a CAR.

Evaluated Algorithms

Our approach is to initiate the systematic study of Component Algorithms for Ridesharing (CARs), a modular design methodology for ridesharing. Each CAR is composed of three parts (Figure 3.4): (a) request – request matching to create a (shared) ride, (b) ride to taxi matching, and (c) relocation of the idle fleet. Each of these components is a significant problem in its own right. Complexity issues make the simultaneous consideration of all three problems impractical. Instead, a more realistic approach is to tackle each component individually, under minimum consideration of the remaining two. The algorithms that we consider are appropriate modifications of the most significant ones that have been proposed for the key algorithmic primitives of the ridesharing problem, i.e., online and offline matching algorithms, with or without delays for steps (a), (b), and (c), k-taxi/server algorithms for step (b), as well as heuristic approaches that were specifically designed with the ridesharing application in mind. A list of all the CARs that we designed and evaluated (14 in total) can be found in Table 3.2. For more details, see in Appendix A.

Finally, offline algorithms (e.g., ALMA, MWM, Greedy) can be run either in a just-in-time (JiT) manner, or in batches, i.e., every x minutes (given that our dataset has granularity of 1 minute, we run in batches of 1, and 2 minutes).

Evaluating all of the possible combinations of CAR components is infeasible. To make the evaluation tractable, we first consider only the first two steps of the ridesharing problem (i.e., no relocation). When possible, we use the same component for both steps (a) and (b). k-Taxi/Server algorithms, though, can not solve step (a), thus we opted to use the best performing component for step (a) (namely the offline maximum-weight matching (MWM) run in batches). Then, we move to evaluate step (c), testing only the most promising components (namely the MWM and ALMA, plus the Greedy as a baseline). We begin by isolating step (c); we fix the component for (a) and (b) to MWM, to have a common-ground for evaluating relocation. Finally, we present results on *end-to-end* solutions.

Table 3.2: Evaluated CARs.

	Step (a)	Step (b)	Step (c)
Maximum Weight Matching (MWM)	MWM	MWM	MWM/ALMA/Greedy
ALtruistic MAtching Heuristic (ALMA) (Danassis et al., 2019)	ALMA	ALMA	ALMA
Greedy	Greedy	Greedy	Greedy
Approximation (Appr) (Bei and Zhang, 2018)	Appr	Appr	-
Postponed Greedy (PG) (Ashlagi et al., 2018)	PG	MWM	-
Greedy Dual (GD) (Bienkowski et al., 2018)	GD	MWM	-
Balance (Bal) (Manasse et al., 1990)	MWM	Bal	-
Harmonic (Har) (Raghavan and Snir, 1989)	MWM	Har	-
Double Coverage (DC) (Chrobak et al., 1990)	MWM	DC	-
Work Function (WFA) (Koutsoupias and Papadimitriou, 1995)	MWM	WFA	-
***k*-Taxi** (Coester and Koutsoupias, 2019)	MWM	*k*-Taxi	-
High Capacity (HC) (Alonso-Mora et al., 2017)	HC	HC	(HC)
Baseline: Single Ride	-	MWM	-
Baseline: Random	-	Random	-

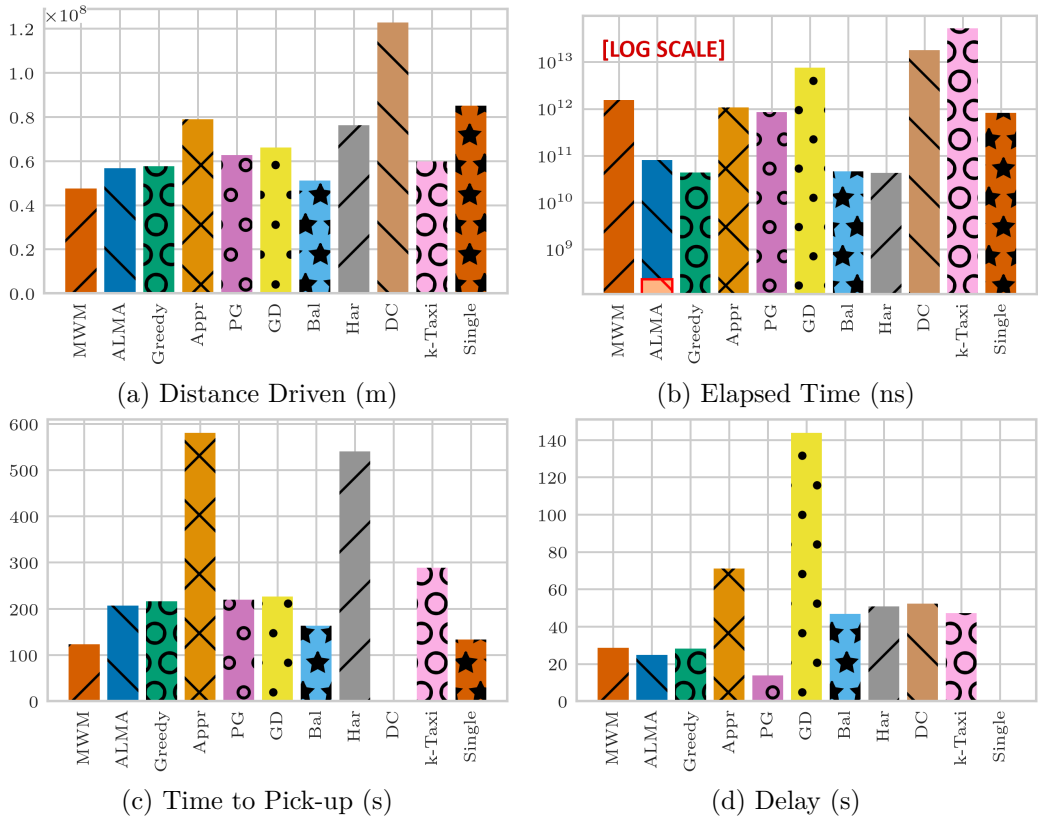
3.6.2 Matching Graphs

One modeling detail that pertains to ALMA is the definition of the matching graphs. In this section, we will provide a short, intuitive explanation. Matching requests to shared rides (part (a) of ridesharing) constitutes a non-bipartite problem. Open requests are matched with other open requests, with the utility of the match representing an approximation (given that it is impossible to know in advance the location of the taxi that will serve the ride) on the travel distance saved by matching the requests. Subsequently, for part (b) of the ridesharing problem, we match (shared or single) rides with taxis. The utility of the match in this part of the problem represents the inverse of the distance between the shared ride and the taxi. Finally, for part (c) – empty vehicle relocation – we use the history to predict a set of *expected future requests*, and then we proceed to match those into shared rides, and rides to idle taxis.

3.6.3 Simulation Results

In this section we present the results of our evaluation. For every metric we report the average value out of 8 runs. Figures 3.5, 3.8, 3.6, and 3.7 present the results without relocation. We first present results on one hour (Figures 3.5 and 3.8) and base number of taxis (minimum number of taxis required to serve all requests as a single ride). Then, we show that the results are robust at a larger time-scale[9] (Figure 3.6), and varying number of vehicles[10] (2138 - 12828) (Figure 3.7). Finally, we present results on the step (c) of the ridesharing problem: dynamic vehicle relocation (Table 3.4, Figure 3.9). As mentioned, the complete results of our evaluation – including, but not limited to, omitted metrics, standard deviation values, algorithms (e.g., WFA, and HC had to be evaluated in smaller test-cases), etc. – can be found in Appendix A.

[9]Missing components were too computationally expensive to simulate for an entire day.
[10]We only present the most promising solutions.



(a) Distance Driven (m)

(b) Elapsed Time (ns)

(c) Time to Pick-up (s)

(d) Delay (s)

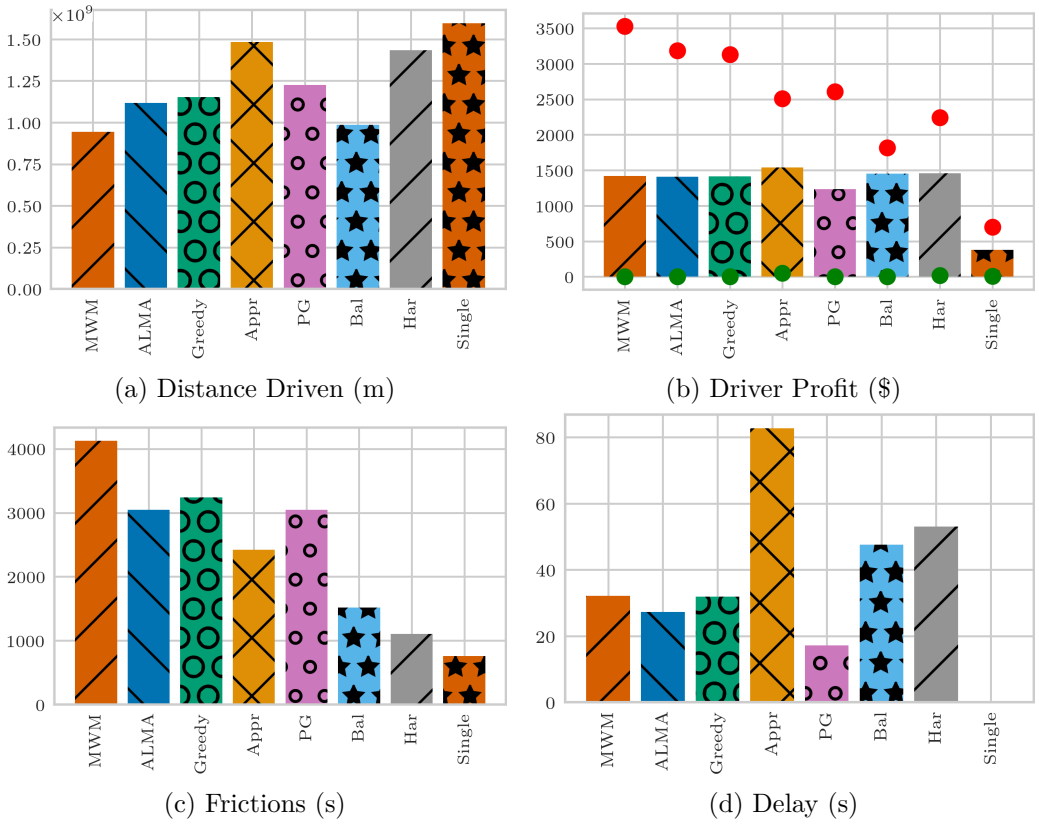
Figure 3.5: 08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

Distance Driven

In the small test-case (Figure 3.5a) MWM performs the best, followed by Bal (+7%). ALMA comes second (+19%), and then Greedy (+21%). The high performance of Bal in this metric is because it uses MWM for step (a), which has a more significant impact on the distance driven. Similar results are observed for the whole day (Figure 3.6a), with Bal, ALMA, and Greedy achieving +4%, +18%, and +22% compared to MWM, respectively. Figure 3.7a shows that as we decrease the number of taxis, Bal loses its advantage, Greedy is pulling away from ALMA (9% worse than ALMA), while ALMA closes the gap to MWM (+17%).

Complexity

To estimate the complexity, we measured the elapsed time of each algorithm. Greedy is the fastest one (Figure 3.5b), closely followed by Har, Bal, and ALMA. ALMA is inherently decentralized. The red overlay denotes the parallel time for ALMA, which is 2.5 orders of magnitude faster than Greedy.



(a) Distance Driven (m)

(b) Driver Profit ($)

(c) Frictions (s)

(d) Delay (s)

Figure 3.6: 00:00 - 23:59 (full day), #Taxis = 5081 (base number). Manhattan, January 15, 2016

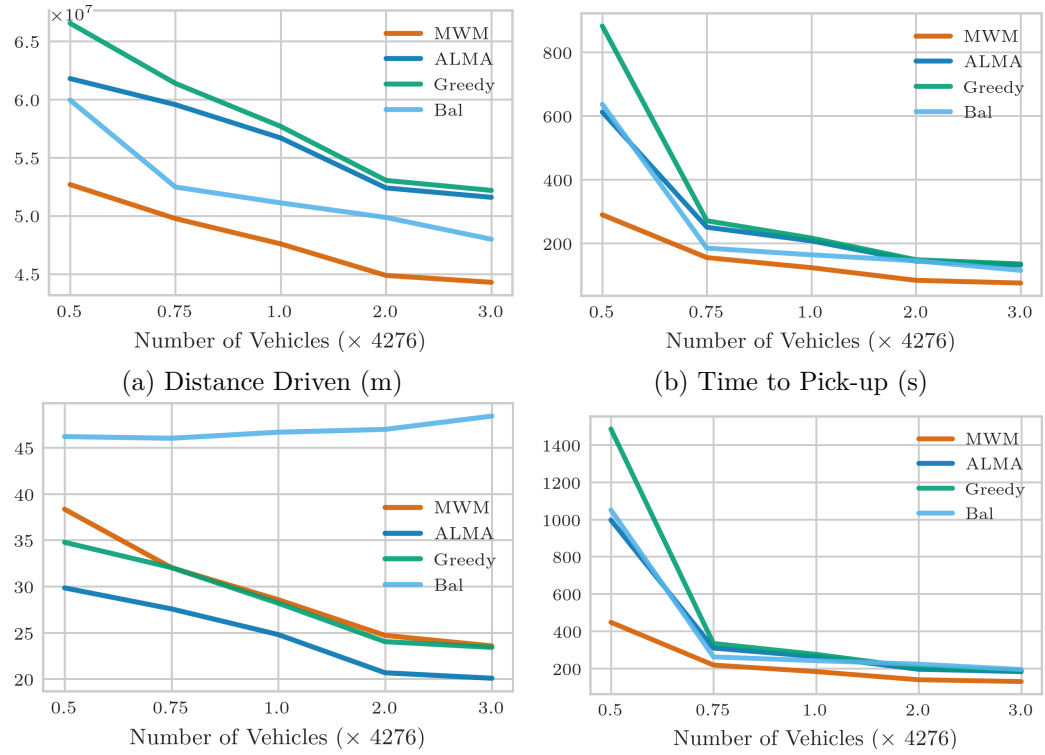
Time to Pick-up

MWM exhibits exceptionally low time to pick-up (Figure 3.5c), lower than the single ride baseline. ALMA, Greedy, and Bal have +69%, +76%, and +33% compared to MWM, respectively. As before, Figure 3.7b shows that as we decrease the number of taxis, Bal loses its advantage, and Greedy is pulling further away from ALMA. Note that to improve visualization, we removed DC's pick-up time as it was one order of magnitude larger than Appr.

Delay

PG exhibits the lowest delay (Figure 3.5d), but this is because it makes 26% fewer shared rides than the rest of the high performing algorithms. ALMA has the smallest delay (−13% compared to MWM), with Greedy following at −1%, while Bal has +63% (both compared to MWM). As the number of taxis decrease (Figure 3.7c), ALMA's gains increase further (−22% compared to MWM).

Figure 3.7d depicts the cumulative delay, which is the sum of all delays, namely the time



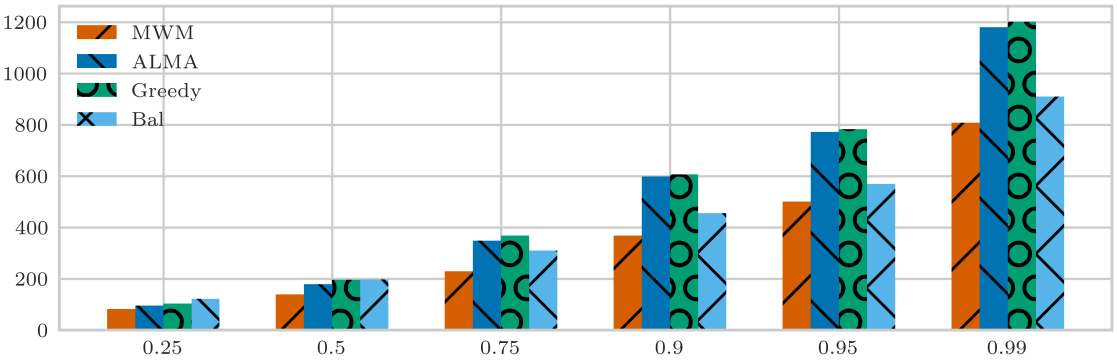
(a) Distance Driven (m)

(b) Time to Pick-up (s)

(c) Delay (s)

(d) Cumulative Delay (s)

Figure 3.7: 08:00 - 09:00, #Taxis = $\{2138, 3207, 4276, 8552, 12828\}$. Manhattan, January 15, 2016



(a) Sequence of Percentiles for Cumulative Delay (s)

Figure 3.8: 08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

to pair, time to pair with taxi, time to pick-up, and delay. An interesting observation is that reducing the fleet size from 12828 (×3.0 of the base number) to just 3207 (×0.75 of the base number) vehicles (75% reduction) results in only approximately 2 minutes of additional delay. This goes to show the great potential for efficiency gains such technologies have to offer, and the ability of ALMA to handle large-scale environments with *scarce* resources.

Table 3.3: Fairness of the Drivers' Profit.
08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

	Jain Index	Relative Diff. to MWM
MWM	0.71	0.0%
ALMA	0.75	6.0%
Greedy	0.75	6.9%
Appr	0.86	21.4%
PG	0.75	7.0%
GD	0.77	9.0%
Bal	0.90	28.2%
Har	0.84	19.1%
DC	0.19	-73.0%
k-Taxi	0.71	0.3%
Single	0.92	30.5%

Finally, we wanted to investigate the distribution of the achieved QoS metrics and, consequently, the reliability/fairness of each CAR. As such, we plotted in Figure 3.8a the sequence of percentiles[11] for the cumulative delay. As shown, the vast majority of the users (75%) experience cumulative delay close to the average value (only 46, 85, 92, 69 additional seconds of cumulative delay than the average value for MWM, ALMA, Greedy, and BAL, respectively). Of course, some of the users experiences high cumulative delay, but this is a small percentage of them. Specifically, less than 5% of requests experience a delay of more than 8.5, 13, 13, and 9.5 minutes for MWM, ALMA, Greedy, and BAL, respectively. Given the size and the average speed of taxi vehicles in Manhattan, such delays could be expected and, thus, acceptable; ultimately, it is up to the ridesharing platform to impose hard constraints and reject requests with potentially high delay.

Profit & Frictions

Contrary to their performance in QoS metrics, GD, and Appr achieve the highest driver profit, 12% and 8% higher than MWM, respectively (although the low QoS and increased distance driven suggest low quality matchings, which can explain the higher revenue, yet deems them undesirable). Bal, and Har follow with $+2-3\%$. ALMA and Greedy achieve the similar profit to MWM. PG exhibits significantly worse results (-13%), due to the lower number of shared rides it matches.

Figure 3.6b also depicts the maximum (red dot), and minimum (green dot) value of a driver's profit. Closer to the mean maximum value suggests a fairer algorithm for the drivers. Moreover, it is worth noting that the minimum value for all the algorithms is

[11] Given a vector V of cumulative delays per request, the q-th percentile of V is the value $q/100$ of the way from the minimum to the maximum in a sorted copy of V.

Table 3.4: Relocation Gains.

	MWM	ALMA	Greedy
Time to Pick-up	-48.95%	-55.18%	-55.03%
Time to Pick-up SD	-52.97%	-58.22%	-58.21%
Delay	-15.95%	-17.79%	-17.73%
Delay SD	-19.25%	-20.96%	-20.98%
Cumulative Delay	-38.37%	-43.23%	-43.11%
Total Distance	5.48%	6.25%	6.24%

zero, meaning that there are taxis which remain unutilized (in spite of the fact that the number of taxis – in this scenario 5081 – is considerably lower than the current fleet size of yellow taxis).

In order to investigate the fairness of the distribution of profits amongst drivers, we calculated the Jain index (Jain et al., 1998), a well-established fairness metric (see Section 2.5). Table 3.3 shows the Jain index, and the relative difference compared to MWM. Bal achieves the most fair allocation (excluding the single ride baseline[12]), with a Jain index of 0.9, closely followed by Appr with 0.86. MWM, ALMA, and Greedy all achieve relatively fair allocations, with the latter two achieving a 6% and 7% improvement over MWM.

Figure 3.6c shows the driver frictions. Just like with the profit, k-server algorithms seem to outperform matching algorithms by far. Compared to MWM, Bal and Har achieve a 63% and 73% decrease, respectively, while ALMA and Greedy achieve a 26%, and 21% decrease, respectively. Given that we have a fixed supply, lower frictions indicate a more even distribution of rides amongst taxis.

Relocation

The aim of any relocation strategy is to improve the spatial allocation of supply. Serving requests redistributes the taxis, resulting in an inefficient allocation. One can assume a 'lazy' approach, relocating vehicles only to serve requests. While this minimizes the cost of serving a request (e.g., distance driven, fuel, etc.), it results in sub-optimal QoS. Improving the QoS (especially the time to pick-up, since it highly correlates to passenger satisfaction) plays a vital role in the growth of a company. Thus, *a crucial trade-off of any relocation scheme is improving the QoS metrics, while minimizing the excess distance driven.*

ALMA successfully balances this trade-off (Table 3.4). In particular, ALMA – *the best*

[12]It is expected that the single ride baseline would result in a fair allocation of the profit, as it constantly utilizes the entire fleet of vehicles.

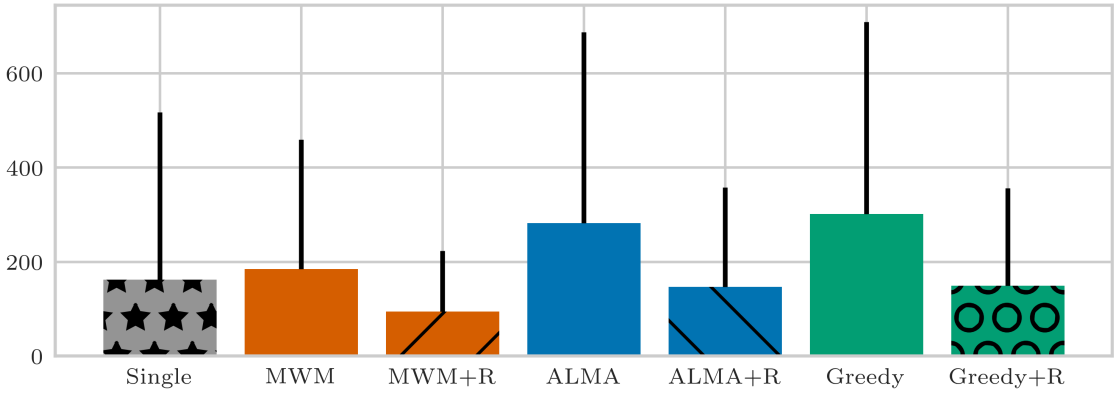


Figure 3.9: Time to Pick-up (s) – End-To-End Solution
January 15, 2016 – 00:00 - 23:59 – Manhattan – #Taxis = 5081

performing overall – radically improves the QoS metrics by more than 50% (e.g., it decreases the pick-up time by 55%, and its standard deviation (SD) by 58%), while increasing the driving distance by only 6%. The cumulative delay is decreased by 43%.

As a final step, we evaluate *end-to-end* solutions, using MWM, ALMA, and Greedy to solve all three steps of the ridesharing problem. Figure 3.9 depicts the time to pick-up (error bars denote one SD of uncertainty), a metric highly correlated to passenger satisfaction level (Tang et al., 2017; Brown, 2016b). We compare against the single ride baseline (no delay due to sharing a ride). Once more, the proposed relocation scheme results in radical improvements, as the time to pick-up drops (compared to the single ride) from +14.09% to −41.76% for MWM, from +74.14% to −9.33% for ALMA, and from +86.10% to −7.97% for Greedy. This comes to show that *simple* relocation schemes, like ALMA, can eliminate the negative effects of ridesharing on the QoS.

ALMA as an End-to-end Algorithm for Ridesharing

While MWM seems to perform the best in the total distance driven, and most QoS metrics – which is reasonable since it makes optimal matches amongst passengers – it hard to scale and requires a centralized solution with a *global* view of the problem. In contrast, greedy approaches are appealing[13] not only due to their low complexity, but also because real-time constraints dictate short planning windows which can diminish the benefit of batch optimization solutions compared to myopic approaches (Widdows et al., 2017).

In fact, ALMA is of a greedy nature as well, albeit it utilizes a more intelligent backing-off scheme. Our simulation results show that ALMA performs on par with the best performing approaches across the board (a high level overview of the results can be

[13](Widdows et al., 2017) reports that GrabShare's scheduling component has used an entirely greedy approach to allocate bookings to drivers. Lyft also started with a greedy system (Brown, 2016a).

Table 3.5: High level (qualitative) ranking of the evaluated CARs.
For each metric, the best performing CAR receives four stars ($\star\star\star\star$). Then, for the rest of the CARs, we compute the relative difference to the best performing one, i.e., $r = (x - OPT)/OPT$, where x is the value for the CAR considered, and OPT is the value achieved by the best performing CAR in the specific metric. If the relative difference is < 0.1, this CAR also receives four stars ($\star\star\star\star$), if it is $0.1 \leq r < 0.5$, the CAR receives three stars ($\star\star\star$), if it is $0.5 \leq r < 1$, the CAR receives two stars ($\star\star$), and finally, if $r \geq 1$, the CAR receives one star ($\star$).
The ranking is primarily based on the one hour test-case and base number of taxis (08:00 - 09:00, Manhattan, #Taxis = 4276).
The same ranking holds in most cases for the full day test-case and base number of taxis (00:00 - 23:59, Manhattan, #Taxis = 5081). The only exceptions are the ones awarded one additional star in the full day test-case, which is denote inside a parenthesis when relevant.
As explained in Section A.8.2, reporting frictions for the one hour test-case can be deceiving, thus we report the ranking based on the full day test-case. To make the distinction clear, the awarded stars are also inside a parenthesis. Algorithms that were too computationally heavy to run for a full day lack a ranking for the frictions.
The ranking for the WFA and HC (which we were able to run only in much smaller test-cases) was extrapolated based on their performance against the baseline CARs that were common in all test-cases (MWM, ALMA, and Greedy).
The Time to Pair with a Taxi is not included as it was zero for most CARs.
It is important to note that since the ranking is based on the relative difference in performance compared to the best performing CAR in each metric, it can be misleading in cases where the change is insignificant in terms of absolute values. Such a case is the Time to Pick-up, where the difference between MWM vs. Bal, or ALMA, or Greedy is only 1-2 minutes.
‡ALMA run in a decentralized manner is orders of magnitude faster than any other CAR. In order to allow for a clear ranking between the rest of the CARs, we performed the ranking based on the second best CAR (i.e., Greedy).
†Again, to allow for a proper ranking between the CARs, PG was not included in the delay metric because its notably low delay is only a result of making significantly fewer shared rides (26% less, see Section 3.6.3).
For more details, see Appendix A.

	Operational Efficiency				Quality of Service			Drivers' Metrics	
	Distance Driven	Computational Complexity	Number of Shared Rides	Time to Pair	Time to Pick-up	Delay	Cumulative Delay	Profit	Frictions
Maximum Weight Matching (MWM)	$\star\star\star$	$\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$(\star)$
ALtruistic MAtching Heuristic (ALMA)	$\star\star\star$	$\star\star\star$ ‡	$\star\star\star$	$\star\star\star$	$\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$(\star)$
Greedy	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star$	$\star\star\star$	$\star\star(\star)$	$\star\star\star$	$(\star)$
Approximation (Appr)	$\star\star$	$\star$	$\star\star\star$	$\star\star\star$	$\star$	$\star$	$\star$	$\star\star\star$	$(\star)$
Postponed Greedy (PG)	$\star\star\star$	$\star$	$\star\star\star$	$\star$	$\star\star$	†	$\star\star(\star)$	$\star\star\star$	$(\star)$
Greedy Dual (GD)	$\star\star\star$	$\star$	$\star\star\star$	$\star\star$	$\star\star$	$\star$	$\star$	$\star\star\star$	-
Balance (Bal)	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star(\star)$	$\star\star$	$\star\star\star$	$\star\star\star$	$(\star\star\star)$
Harmonic (Har)	$\star\star$	$\star\star\star$	$\star\star\star$	$\star\star\star$	$\star$	$\star(\star)$	$\star$	$\star\star\star$	$(\star\star\star)$
Double Coverage (DC)	$\star$	$\star$	$\star\star\star$	$\star\star\star$	$\star$	$\star$	$\star$	$\star\star\star$	-
Work Function (WFA)	$\star\star\star$	$\star$	$\star\star\star$	$\star$	$\star$	$\star\star\star$	$\star$	$\star\star\star$	-
***k*-Taxi**	$\star\star\star$	$\star$	$\star\star\star$	$\star\star\star$	$\star$	$\star\star$	$\star$	$\star\star\star$	-
High Capacity (HC)	$\star\star\star$	$\star$	$\star\star\star$	$\star$	$\star\star\star$	$\star\star$	$\star\star\star$	$\star\star\star$	-

found in Table 3.5). Most importantly, ALMA was inherently developed for multi-agent applications. Agents make decisions locally, using completely uncoupled learning rules, and require only a 1-bit partial feedback, making it an ideal candidate for an *on-device* implementation.

3.7 Chapter Conclusion

Algorithms for solving the assignment problem, whether centralized or distributed, have runtime that increases with the total problem size, even if agents are interested in a small number of resources. Thus, they can only handle problems of some bounded size. Moreover, they require a significant amount of inter-agent communication. Humans on the other hand are routinely called upon to coordinate in large scale, and under dynamic and unpredictable demand. Inspired by human behavior, we have introduced a novel anytime heuristic (ALMA) for weighted matching. ALMA is decentralized, requires only partial feedback, and has *constant* in the total problem size running time, under reasonable assumptions on the preference domain of the agents. The presented results provide an empirical proof of the high quality of the achieved solution in a variety of scenarios (synthetic and *real* data, time constraints, on-line settings). As autonomous agents proliferate (IoT devices, intelligent infrastructure, autonomous vehicles, etc.), having robust algorithms that can scale to hundreds of thousands of agents is of utmost importance.

4 Differentially Private Weighted Matching

4.1 Preface

4.1.1 Contribution and Sources

This chapter is largely based on (Danassis et al., 2022c). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable):

PD (author): Conceptualization, Methodology, Software (lead), Formal analysis, Investigation (lead), Writing – Original Draft (lead), Writing – Review & Editing (lead)

Aleksei Triastcyn: Methodology, Software (supporting), Formal analysis, Investigation (supporting), Writing – Original Draft (supporting), Writing – Review & Editing (supporting)

Boi Faltings: Writing – Review & Editing (supporting), Supervision

4.1.2 Chapter Summary

In this chapter, we focus on protecting the privacy of individuals in large-scale applications. When it comes to large-scale multi-agent systems with a diverse set of agents, traditional differential privacy mechanisms are ill-matched because they consider a very broad class of adversaries, and they protect all users, independent of their characteristics, by the same guarantee. Achieving a meaningful privacy often leads to pronounced reduction in solution quality. We introduce *Piecewise Local Differential Privacy* (PLDP), a privacy model designed to protect the utility function in real-world, unboundedly large multi-agent environments. Moreover, we combine PLDP with the ALMA algorithm presented in Chapter 3 to produce a *practical* and *scalable* algorithm (Privacy-preserving ALtruistic MAtching, or *PALMA*) for solving one of the fundamental problems of multi-agent systems – finding matches and allocations – in unboundedly large settings (e.g., resource allocation in urban environments, mobility-on-demand systems, etc.), while providing *strong worst-case privacy* guarantees. PALMA maintains all the desirable properties of ALMA: it is decentralized, runs on-device, requires no inter-agent communication, and converges in constant time under reasonable assumptions. We evaluate PALMA in a mobility-on-demand and a paper assignment scenario, using real data in both, and demonstrate that it provides a strong level of privacy ($\varepsilon \leq 1$ and median as low as $\varepsilon = 0.5$ across agents) and high quality matchings (up to 86% of the non-private optimal).

4.2 Introduction

Real-world matching problems pose three significant challenges: (i) they may occur in *unboundedly large* settings (e.g., resource allocation in urban environments), (ii) they are *distributed* and *information-restrictive* (agents have partial observability and inter-agent communication might not be available (Stone et al., 2010)), and finally, (iii) individuals have to *reveal their preferences* over the possible matches in order to get a high quality match, which brings forth significant privacy risks. In this chapter, we propose *PALMA* (Privacy-preserving ALtruistic MAtching), a matching algorithm designed to tackle *all* of the aforementioned challenges.

PALMA is a privacy-preserving adaptation of ALMA introduced in Chapter 3, which solves the first two challenges. As such, it is *decentralized*, requires *no communication* between the participants, and converges in *constant* (to the total problem size) time – in the realistic case where each agent is interested in a (fixed size) subset of the total resources (and, thus, there is also a fixed number of agents competing for each resource).

The third challenge requires *protecting the utility functions* of the agents. In recent years, Differential Privacy (DP) (Dwork, 2006) (and its variants) has emerged as the de facto standard for protecting the privacy of individuals. Informally, a DP algorithm ensures indistinguishability on the output distributions for any neighboring inputs. We have designed a defense mechanism for PALMA based on the idea of randomized response (Warner, 1965) – which involves adding controlled randomness – that results in indistinguishability under Local DP (Dwork et al., 2014).

One final challenge arises when it comes to *large-scale* multi-agent systems with a *diverse* set of agents, as it is hard to achieve a meaningful privacy guarantee – in a practical way – using standard (L)DP if the problem has a large output space (e.g., matches, and allocations) (Tong et al., 2017; Hsu et al., 2014). Conventional (L)DP mechanisms often require adding a lot of random noise to achieve a meaningful privacy guarantee, which in turn leads to a pronounced drop in the solution quality. More often than not, this is not due to the inherent difficulty of the problem at hand, but rather due to the generality of the DP definition. Not only does DP consider a very broad class of adversaries, it also protects all users – independent of their characteristics – by the same guarantee. While this property is being praised as one of the strongest arguments in favor of DP, it can be completely redundant in many real-world applications for three key reasons: (i) users might be willing to disclose less-sensitive information (e.g., city of residence, but not exact location), (ii) the attacker might already know coarser-grained information because it is likely public or easily available and, thus, does not need to be hidden (e.g., city of residence in a mobility-on-demand system, or reviewer expertise in a paper assignment problem), and (iii) domain characteristics might exclude a subset of solutions (e.g., a taxi in Manhattan will not be assigned to serve a request in Brooklyn, and an expert on auctions would not be assigned to review a robotics paper, thus, there is no need for

indistinguishably between taxis in different boroughs or reviewers on different fields).

To solve this challenge, we motivate and develop a 'context-aware' privacy definition (*Piecewise Local Differential Privacy* – PLDP), which takes into account the 'distance' between the images of two utility functions. The level of protection depends on that distance; agents with utility functions that have images close in distance to each other would be indistinguishable from the attacker's point of view. The definition is inspired by existing work on 'data-aware' privacy notions (Triastcyn and Faltings, 2020; Triastcyn, 2020) and distance-based generalisations of DP (Chatzikokolakis et al., 2013; Andrés et al., 2013).

As a matter of fact, there are works that utilize such distance-based notions to solve a weighted matching problem in specific domains (e.g., (Prorok and Kumar, 2017; Fioretto et al., 2018)). Yet, these are centralised approaches and thus face a (computation and communication) complexity barrier (refer back to the aforementioned challenges (i) and (ii) of real-world matching problems). ALMA can also be combined with other existing notion of privacy (e.g., LDP or geo-indistinguishability (Andrés et al., 2013)), yet the solution quality is inferior compared to the proposed, carefully crafted (with ALMA in mind) noise, as we demonstrate in our evaluation. *Ultimately, we are the first to develop a practical and scalable framework for weighted matching and resource allocation in general, unboundedly large, multi-agent systems.*

4.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We propose *PALMA*, a practical and scalable privacy preserving algorithm** for weighted matching in *unboundedly large* settings with thousands of agents (e.g., resource allocation in urban environments, intelligent infrastructure, IoT devices, etc.).

2. **We introduce *Piecewise Local Differential Privacy* (PLDP)**, a variant of differential privacy designed to protect the utility function in multi-agent applications. PLDP enables significant improvements in solutions quality and strong theoretical privacy guarantees, while being applicable in *real-world, unboundedly large settings.*

3. **We evaluate PALMA in a mobility-on-demand and a paper assignment scenario, using *real data*.** PALMA is able to provide a high degree of privacy, $\varepsilon \leq 1$ and a median value as low as 0.5 across agents for $\delta = 10^{-5}$, and matchings of high quality (up to 86% of the non-private optimal).

4.2.2 Discussion and Related Work

Finding a maximum-weight matching is one of the best-studied combinatorial optimization problems (Su, 2015; Lovász and Plummer, 2009). Yet, while the problem has been 'solved' from an algorithmic perspective – having both centralized and decentralized polynomial algorithms – it is not so from the perspective of multi-agent systems, for three key reasons: (i) *complexity*, (ii) *communication*, and (iii) *privacy*.

The proliferation of intelligent systems will give rise to large-scale, multi-agent based technologies. Algorithms for maximum-weight matching, whether centralized or distributed, have runtime that increases with the total problem size, even in the realistic case where agents are interested in a small number of resources. Thus, they can only handle problems of bounded size. Moreover, they require a significant amount of inter-agent communication. Yet, communication might not always be an option (Stone et al., 2010), and sharing utilities, plans, and preferences creates high overhead. ALMA on the other hand achieves *constant* in the total problem size running time – under reasonable assumptions – while requiring no message exchange (i.e., no communication network) between the participating agents (see Chapter 3). The proposed approach, PALMA, *preserves* the aforementioned two properties of ALMA, thus, dealing with the first two of the posed challenges.

Differential Privacy (DP) (Dwork, 2006; Dwork et al., 2006a,b) has emerged as the de facto standard for protecting the privacy of individuals.[1] Informally, DP captures the increased risk to an individual's privacy incurred by his participation. A variation of differential privacy, especially useful in our context, given the decentralized nature of PALMA, is Local Differential Privacy (LDP) (Dwork et al., 2014). LDP is a generalization of DP that provides a bound on the outcome probabilities for any pair of individual agents rather than populations differing on a single agent. Intuitively, it means that one cannot hide in the crowd. Another strength of LDP is that it does not use a centralized model to add noise—individuals sanitize their data themselves—providing privacy protection against a malicious data curator. As a result, LDP requires adding even more random noise to achieve a meaningful bound, which would result in the decline of the solution quality. In fact, it is impossible to have both meaningful social welfare and privacy guarantees in matching problems under (L)DP (Hsu et al., 2014). (L)DP ignores specifics of AI applications, such as a focus on a given task or a particular data distribution.

Inspired by the notions of Bayesian DP (Triastcyn and Faltings, 2019) – which is based on the observation that machine learning models are designed and tuned for a particular data distribution which is also often available to the attacker – and metric-based DP (Chatzikokolakis et al., 2013) and geo-indistinguishability (Andrés et al., 2013) – where indistinguishability depends on an arbitrary notion of distance – we propose a

[1] See Section 2.4 for the definition of DP. For a more comprehensive overview of DP and DP mechanisms, we refer the reader to (Triastcyn, 2020; Dwork et al., 2014).

new privacy model, namely Piecewise Local Differential Privacy (PLDP). PLDP takes into account the 'distance' between the images of two utility functions, and the level of protection depends on that distance. The rationale is that instead of guaranteeing local privacy in the entire domain of agents, which can be quite difficult and would result in low quality solutions due to excessive noise, we focus on indistinguishability of agents with similar preferences.

4.3 Piecewise Local Differential Privacy (PLDP)

Let $\mathcal{M} : \mathcal{D} \to \mathcal{A}$ be a randomized function with domain $\mathcal{D}$ and range $\mathcal{A}$. In the context of matching problems in multi-agent systems, $\mathcal{D}$ is the space of utility functions and $\mathcal{A}$ is the action space. In the context of PALMA specifically, an action is either an attempt to acquire a certain resource, or a back-off from a previously contested resource, as will explained in the following section.

Definition 5. Let $\varphi(\cdot)$ be a set function that fragments $\mathcal{D}$ into a collection of subsets $\{\mathcal{D}_i\}$. Then, a randomized algorithm $\mathcal{M} : \mathcal{D} \to \mathcal{A}$ satisfies $(\varepsilon, \delta, \varphi)$-piecewise local privacy if for any two inputs $x, x' \in \mathcal{D}_i,\ \forall i$, and for any set of outcomes $\mathcal{S} \subset \mathcal{A}$ it holds:

$$\Pr\left[\mathcal{M}(x) \in \mathcal{S} \mid x \in \mathcal{D}_i\right] \leq e^{\varepsilon} \Pr\left[\mathcal{M}(x') \in \mathcal{S} \mid x' \in \mathcal{D}_i\right] + \delta.$$

4.3.1 Motivation

Consider a mobility-on-demand (MoD) application (e.g., ridesharing). A MoD company can operate across multiple cities, countries, or even continents. If a MoD provider employs traditional DP (e.g., LDP) to protect all users (independently of their characteristics) with the same guarantee, the achieved social welfare will be as good as a *random solution*[2] in large-scale environments. This is because the *support* of any agent has to include *every resource* (otherwise an adversary could distinguish between agents), i.e., a request in Manhattan might be paired with a taxi in Brooklyn. Moreover, it is reasonable to assume an informed attacker (e.g., one that knows the city of residence), and users may be willing to reveal approximate location information (it is most likely acceptable to disclose the fact that an individual is in Manhattan, however disclosing the exact location is undesirable). Similarly, in a paper assignment problem (reviewers to manuscripts), ensuring indistinguishably between an expert on Markets & Auctions, and one on Robotics might be futile, especially if the attacker possesses additional information (e.g., the tracks of the papers) that would exclude infeasible matches.

The rationale behind PLDP is the following. Instead of guaranteeing local privacy in the entire domain of agents, which may be quite difficult, we focus on indistinguishability

[2]The solution that results of picking edges randomly in a fully connected bipartite graph containing all agents and resources.

of agents with *similar preferences*. We fragment the space of utilities into regions and guarantee privacy within these regions but not between them.

A useful real-world analogy is ZIP codes. Assume we would like to release some location statistic with PLDP and we choose φ such that the initial location space is mapped into ZIP codes. Then, $(\varepsilon, \delta, \varphi)$-PLDP guarantee would certify that the reported statistic is (ε, δ)-locally private within every ZIP code. However, it would not tell us anything about privacy of the reported statistic outside the given ZIP code. In other words, while an agent can be distinguished from agents outside his zip code, he is still indistinguishable from *all* agents inside its ZIP code.

4.3.2 Privacy Properties

Note that PLDP is a straightforward relaxation of local privacy and all the properties of LDP are satisfied within sub-domains $\mathcal{D}_i$. In order to see that this is true, it is sufficient to consider the following. Once the space $\mathcal{D}$ has been partitioned, the PLDP definition is equivalent to the LDP definition within each sub-space $\mathcal{D}_i$. Hence, basic properties of (L)DP, such as *composition*, *post-processing*, and *group privacy*, as well as several instances of *advanced composition* (Dwork et al., 2014; Abadi et al., 2016), will also hold for any pair x, x' from a given $\mathcal{D}_i$, as long as these points do not dynamically change sub-domains between applications of the privacy mechanism. The latter condition is satisfied in all considered scenarios: every new matching routine starts with a fresh set of agents with random identifiers, and agents do not change their utilities during the matching process.

4.3.3 Advantages of PLDP (vs. Distance-based Generalisations of DP)

PLDP closely resembles another well-known privacy notion, geo-indistinguishability (Andrés et al., 2013), which is based on a generalization of DP (Chatzikokolakis et al., 2013). Nonetheless, there is a notable distinction. To put it in terms of the definition above, in geo-indistinguishability, the region within which privacy is protected is centered at x. In our definition, these regions are predefined by φ. As a downside, our privacy guarantee is limited to the given region rather than fading gradually with increasing region radius. However, there is also a crucial upside to this subtle difference in real-world applications due to composition properties. To the best of our knowledge, in spite of conveniently adopting the use of distances between inputs to adjust levels of privacy guarantees, geo-indistinguishability has only been proven to satisfy basic composition. As a result, ε grows linearly with the number of privacy mechanism invocations. It is not sufficiently tight for iterative AI and ML applications, which typically require a lot of repetitive applications of privacy mechanisms (Abadi et al., 2016). On the other hand, PLDP allows to use *tighter composition theorems* developed for the conventional DP,

reducing the growth of ε from linear w.r.t. the total number of algorithm iterations T to $\mathcal{O}(\sqrt{T})$ (Abadi et al., 2016).

A second advantage of PLDP is that, contrary to geo-indistinguishability, it does not require a metric space (i.e., a natural ordering). As an example, this makes PLDP easier to apply in settings like our paper assignment application where each agent/resource is represented by a 25-dimensional binary label (see Section 4.8.1). In this example, there is ordering in each dimension, but not across them.

4.4 PALMA: A Privacy-Preserving Weighted Matching Algorithm

In this section, we introduce PALMA (Privacy-preserving ALtruistic MAtching), a privacy-preserving adaptation of the ALMA algorithm presented in Chapter 3. We mainly focus on weighted matchings in bipartite graphs (i.e., the assignment problem, see Section 2.2 for the formal definition), but PALMA – just like ALMA – can also be applied in general graphs (see Section 2.3 for the formal definition).

The assignment problem refers to finding a maximum-weight matching in a weighted bipartite graph, $\mathcal{G} = \{\mathcal{N} \cup \mathcal{R}, \mathcal{E}\}$. In the studied scenario, $\mathcal{N} = \{1, \ldots, N\}$ agents compete to acquire $\mathcal{R} = \{1, \ldots, R\}$ resources. The weight of an edge $(n, r) \in \mathcal{E}$ represents the utility ($u_n(r) \in [0, 1]$) agent n receives by acquiring resource r. Each agent can acquire at most one resource, and each resource can be assigned to at most one agent. The goal is to maximize the sum of utilities. For a more detailed description, see Section 2.2. Finally, for simplicity, in the rest of the chapter we assume $N = R$. This is *not required* by PALMA (or ALMA). If $R > N$ some resources will remain free, while if $N > R$ some agents will fail to acquire a resource (convergence in the latter case implies that the state of the agent does not change).

4.4.1 Learning Rule

The learning process is similar to ALMA's (presented in Section 3.3.1); the main differences lie in the resource selection and back-off functions. For completeness, we repeat part of the algorithm's description of Section 3.3.1.

We assume each agent is interested in (potentially) a subset of the total resources $\mathcal{Q}^n \subseteq \mathcal{R}$. Let $\mathcal{A} = \{Y, A_{r_1}, \ldots, A_{r_{Q^n}}\}$ denote the set of actions, where Y refers to yielding, and A_r refers to accessing resource r. Let g denote the agent's strategy. PALMA is run *independently and in parallel by all the agents*. Each agent converges to a resource through repeated trials, specifically:

As long as an agent has not acquired a resource yet, at every time-step, there are two

possible scenarios: If $g = A_r$ (strategy points to resource r), then agent n attempts to acquire that resource. If there is a collision,[3] the colliding parties back-off with some probability, $P_B^n(\cdot)$. Otherwise, if $g = Y$, the agent chooses a resource r for monitoring according to probability , $P_S^n(\cdot)$. If the resource is free, he sets $g \leftarrow A_r$. The pseudo-code can be found in Algorithm 2.

Resource Selection Distribution

In the original implementation of ALMA, each agent sorts the resources in decreasing order of utility $(r_1, \ldots, r_R)$. Then, he moves in a sequential manner, starting from the most preferred resource (r_1), and moving down the list until he acquires one. This method of resource selection results in the highest social welfare, but it is impossible to guarantee privacy due to the deterministic nature of the selection process. On the other end of the spectrum, we can select a resource in a weighted at random fashion, where resource r_i is selected with probability $\frac{u_n(r_i)}{\sum_{r\in\mathcal{R}} u_n(r)}$. This method provides high degree of privacy, but can result in low social welfare. To elaborate the latter, consider the following adversarial scenario: in a large-scale urban domain ($|\mathcal{R}| \to \infty$) where agents are interested only in resources that are physically close to them, the majority of resources would have utility ≈ 0. If we select a resource in a weighted at random fashion, the probability of selecting a low utility resource would be high – due to the large number of resources – resulting in low social welfare.

In this work, we combine the aforedescribed two approaches. Let $\mathcal{N}^n$ denote the set of every possible agent that belongs to the same region of utility space as n, i.e., $\mathcal{N}^n = \{n' : u_{n'}(\cdot) \in \mathcal{D}_i \wedge u_n(\cdot) \in \mathcal{D}_j \Rightarrow i = j\}$. We refer to $\mathcal{N}^n$ as the set of neighbors of n. Note that the neighbors of an agent do not need to be in $\mathcal{N}$, we account for every potential agent (i.e., $\cup_{n\in\mathcal{N}}\mathcal{N}^n \supset \mathcal{N}$). The *neighbors* are the *set of agents that PLDP guarantees indistinguishability.* Then, each agent n independently generates the sets $(\mathcal{R}_1^n, \ldots, \mathcal{R}_i^n, \ldots, \mathcal{R}_R^n)$, where the set $\mathcal{R}_i^n$ contains the i^{th} most preferred resource of each neighbor, i.e., $\mathcal{R}_i^n = \cup_{\forall n'\in\mathcal{N}^n}\{r_i^{n'}\}$.

Agent n moves in a *sequential* manner from set to set (starting from the set of the most preferred resources, $\mathcal{R}_1^n$, and looping back to it after $\mathcal{R}_R^n$). The resource selection is performed in a *weighted at random* fashion in the sets $\mathcal{R}_i^n$. Specifically, at step $s = t \mod R$, where t is the current time-step, agent n will select resource $r_i \in \mathcal{R}_s^n$ with probability[4] $P_S^n(\cdot) : \mathcal{R} \to \Delta^R$ given by (line 18 of Algorithm 2):

$$P_S^n(i, s, \zeta_S) = \zeta_S \times P_{\text{WaR}}(i, s, n) + (1 - \zeta_S) \times S_{\text{Noise}}(i, s, n^*) \tag{4.1}$$

[3] We assume that agents can observe feedback from their environment to inform collisions and detect free resources (e.g., by the use of sensors, or by a single bit feedback from the resource).

[4] Where Δ^R denotes an R-simplex, i.e., a set of R values that are in $[0, 1]$ and sum to 1.

$$P_{\text{WaR}}(i, s, n) = \frac{u_n(r_i)}{\sum_{r \in \mathcal{R}_s^n} u_n(r)} \tag{4.2}$$

Equation 4.1 defines a mixture distribution, composed of (a) selecting in a weighted at random fashion using the utilities of agent n ($P_{\text{WaR}}(i, s, n)$, given by Equation 4.2), and (b) a distribution that introduces noise ($S_{\text{Noise}}(\cdot)$) to the selection process. ζ_S tunes the magnitude of the introduced randomness.

The introduced noise can be any distribution that is known and common for all agents (can be domain specific). For example, it could simply be a uniformly at random selection in the set of resources $\mathcal{R}_s^n$. In this work, we take advantage of domain knowledge. Specifically, let n^* denote a 'representative' agent of the Neighborhood of agent n. This can be for example a(n) (potential) agent located in the center of the neighborhood in a mobility-on-demand application. Then, the common distribution (i.e., noise) can be to play in a uniformly at random manner according to the utility function of the representative agent, i.e., $S_{\text{Noise}}(i, s, n^*) = P_{\text{WaR}}(i, s, n^*)$. In section 4.4.1, we provide a concrete example on the fragmentation of the utility space into neighborhoods, the representative agent, and the selection and back-off probabilities.

Back-off Distribution

The back-off probability, $P_B^n(\cdot) : \mathcal{R} \rightarrow [0, 1]$ (line 9 of Algorithm 2), is computed individually and locally based on each agent's expected utility loss that he will incur if he switches:

$$loss(i, s, n) = u_n(r_i) - \sum_{r_j \in \mathcal{R}_{s+1}^n} \frac{u_n(r_j)}{\sum_{r \in \mathcal{R}_{s+1}^n} u_n(r)} u_n(r_j) \tag{4.3}$$

The actual back-off probability can be computed with any monotonically decreasing function f on $loss(\cdot)$, e.g.:

$$f(loss) = \begin{cases} 1 - \gamma, & \text{if } loss \leq \gamma \\ \gamma, & \text{if } 1 - loss \leq \gamma \\ 1 - loss, & \text{otherwise} \end{cases} \tag{4.4}$$

where γ places a threshold on the minimum / maximum back-off probability. According to the above distribution, agents that do not have good alternatives will be less likely to back-off and vice versa. The ones that do back-off select an alternative resource, according to the resource selection probability $P_S^n(\cdot)$, and examine its availability (line

18 of Algorithm 2). Finally, $P_B^n(\cdot)$ is given by Equation 4.5:

$$P_B^n(i, s, \zeta_B) = \zeta_B \times f(loss(i, s, n)) + (1 - \zeta_B) \times B_{\text{Noise}}(i, s, n^*) \tag{4.5}$$

The back-off distribution is mixture between acting according to an agent's own utility function ($f(loss(i, s, n))$), and a distribution that introduces noise ($B_{\text{Noise}}(\cdot)$) to the back-off process. ζ_B tunes the magnitude of the introduced randomness. As was the case with the selection distribution, the introduced noise for the back-off distribution can be any distribution that is known and common for all agents. In this work, we set $B_{\text{Noise}}(i, s, n^*) = f(loss(i, s, n^*))$, i.e., the 'noise' distribution refers to backing-off according to the utility function of the 'representative' agent (described in Section 4.4.1).

Elaborative Example on Neighborhoods

In what follows, along with Section 4.5.1, we will provide an elaborative, practical example of the key notions of PALMA.

PLDP is used to protect the utility function of agents. Consider the space of all possible utility functions, and then consider the space of the images of those utility functions. We fragment the former into sub-spaces D_i, such that for two utility functions that belong to the same D_i, their image is 'close' in distance. In simple terms this means that the actual utility value of a resource would be similar for agents with utility functions in the same sub-space D_i. The fragmentation is performed by $\phi(\cdot)$.

Each agents selects his own $\phi(\cdot)$ based on his privacy needs. The choice of $\phi(\cdot)$ is public information. For simplicity, in this work, we assume that every agent has the same $\phi(\cdot)$. The choice of $\phi(\cdot)$ fragments the space of agents into regions; the image of the utility function of every agent in a region is close in distance to every other agent in the same region. The definition of the region ($\phi(\cdot)$) as well as the distance metric are domain specific.

As a concrete example, consider a mobility-on-demand (MoD) application (e.g., ridesharing). Let the utility of each agent (ridesharing user) be inversely proportional to the distance (in meters) from the resource (taxi vehicle). In this case, we can split the are of operation of a ridesharing company into rectangular regions, as shown in Figure 4.1; agents in the same region would have similar utilities for each resource.[5]

To compute his neighbors, an agent considers *every possible agent* that could belong in his region, regardless if this agent exists. Expanding on our MoD example, we can consider having an agent ridesharing user) every, e.g., 10m on the map. In a 10^6m^2

[5]Note that we protect the privacy of the agents, not the resources; thus, the resources (taxi vehicles) do not need to belong to any region, and can be matched with any agent regardless of his region.

Table 4.1: Nomenclature, Algorithm 2

s	Current step (indicates a specific set $\mathcal{R}_s^n$)
g	Specifies which resource to access
$\{Y, A_{r_1}, \ldots, A_{r_R}\}$	Y refers to yielding, and A_r refers to accessing resource r
$P_S^n(\cdot) : \mathcal{R} \to [0, 1]$	Resource selection probability distribution
$P_B^n(\cdot) : \mathcal{R} \to [0, 1]$	Back-off probability distribution
c	Accumulated privacy cost
c_{max}	Highest possible privacy cost for selection or back-off
B_n	Privacy budget

Algorithm 2 PALMA:
Privacy-preserving ALtruistic MAtching Algorithm.

Initialize $s \leftarrow 1$, $g \sim P_S^n(\cdot)$, $c \leftarrow 0$, $converged \leftarrow False$
Calculate c_{max} according to Equation 4.9
procedure PALMA
    while !converged **do**
        if $g = A_r$ **then**
            Agent n attempts to acquire r
            if Collision(r) **then**
                if $c + c_{max} \leq B_n$ **then**
                    Back-off (set $g \leftarrow Y$) with probability $P_B^n(\cdot)$
                    $c \leftarrow c + c_{max}$
                else
                    Back-off (set $g \leftarrow Y$) with prob. $B_{\text{Noise}}(\cdot)$
            else
                converged $\leftarrow True$
        else ($g = Y$)
            $s \leftarrow (s + 1) \mod R$
            if $c + c_{max} \leq B_n$ **then**
                Agent n monitors $r \sim P_S^n(\cdot)$
                $c \leftarrow c + c_{max}$
            else
                Agent n monitors $r \sim S_{\text{Noise}}(\cdot)$
            if Free(r) **then** set $g \leftarrow A_r$
Output r, such that $g = A_r$, and $(\varepsilon, \delta) \leftarrow$ getPrivacy(c) (Equation 4.10)

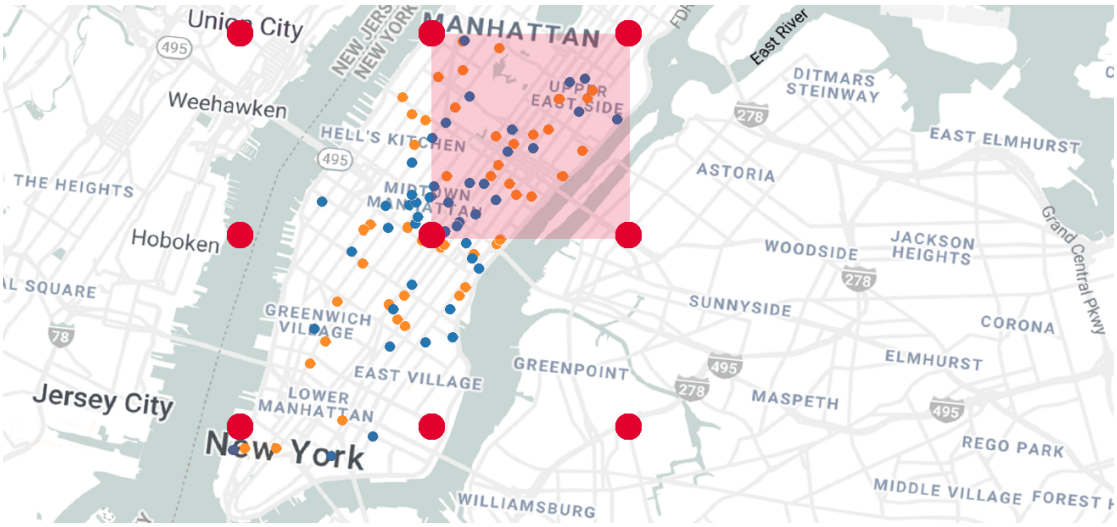


Figure 4.1: A visual representation of the regions ($\{\mathcal{D}_i\}$) of PLDP for the mobility-on-demand application. Red dots denote the edge points of each region ($\ell = 4000$). Orange dots represent the agents (requests), and blue dots represent the resources (vehicles) in our dataset. As an example, an agent in the overlaid rectangle could be located *anywhere* in the rectangle from the attacker's point of view.

region, the neighborhood will include 10^4 agents. Each of these agents has his own preference (ordering) of resources. Using these preferences, we can construct the sets $\mathcal{R}_1^n, \ldots, \mathcal{R}_R^n$, where the set $\mathcal{R}_i^n$ contains the i^{th} most preferred resource of each neighbor. The construction of the neighborhoods needs to be performed once, offline. PLDP guarantees that each agent is *indistinguishable from all his neighbors* (i.e., every potential agent that could exist in his region) from the attacker's point of view.

Finally, the 'representative' agent of each region can be a 'virtual' agent located at the center of the region. Given that $\phi(\cdot)$ is public – and thus the fragmentation into regions as well – the selection and back-off distribution of the representative agent is also public and common for all agents.

4.4.2 Complexity

Bounding the Set of Desirable Resources

An important characteristic of many real-world applications is that there is typically a cost associated with acquiring a resource. As a result, each agent is typically interested in a subset of the total resources, i.e., $\mathcal{Q}^n \subset \mathcal{R}$, and thus at each resource there is a bounded number of competing agents. For example, a taxi would not be willing to drive to the other end of the city to pick up a low fare passenger, a driver would not be willing to charge his vehicle at a station in a different part of the city, and a reviewer would not be willing to review a paper outside his scope of expertise. This results in faster

convergence (*constant* time, see Section 3.3.4), but can also potentially lead to higher social welfare.[6] The sets $(\mathcal{R}_1^n, \ldots, \mathcal{R}_i^n, \ldots, \mathcal{R}_R^n)$ can be contracted in the same manner as before.

Communication and Computation Complexity

PALMA (just like ALMA) does not require any inter-agent communication.[3] The initialization is linear to the size of the region, $\mathcal{O}(\max_i |\mathcal{D}_i|)$, but this can be done once off-line. The accounting of the privacy loss is $\mathcal{O}(1)$. Finally, the convergence bounds proven for ALMA (see Section 3.3.4) hold for PALMA as well.

4.4.3 Privacy Mechanism

PALMA's defense mechanism is based on the idea of randomized response (Warner, 1965), and involves adding controlled randomness in (i) the resource selection and (ii) back-offs, parametrized by ζ_S and ζ_B, respectively (see Equation 4.1 and 4.5). The idea is that the agent first flips a coin to decide whether to act truthfully. Then, with probability ζ_S (or ζ_B), the agent plays according to its true selection (or back-off) function; with probability $1 - \zeta_S$ (or $1 - \zeta_B$), the agent plays according to a public, common distribution.

Moreover, each agent has a privacy budget of $\varepsilon = B_n$. Upon depletion in the course of using the above mechanisms (see lines 8 & 17 of Algorithm 2), the agent will play *noisy* actions (see lines 12 & 21 of Algorithm 2). Note also that each agent can select the fragmentation function $\varphi(\cdot)$ of PLDP and adjust the size of the neighborhood $\mathcal{N}^n$ according to his privacy needs.

4.5 Privacy Accounting

Since PALMA is an iterative algorithm, we need to compute (ε, δ) guarantees over multiple applications of the privacy mechanism. This can be done via *privacy accounting* methods (e.g., (Dwork et al., 2014)). We employ the accounting framework introduced in (Triastcyn and Faltings, 2020) and extend it to generic subsampled mechanisms. While developed for the notion of Bayesian DP, this framework is applicable to the traditional DP as well, and in such a case, is equivalent to the moments accountant (Abadi et al., 2016) for the subsampled Gaussian mechanism and Rényi accountant (Mironov, 2017). Let us briefly outline the method.

Let σ_t and σ_t' denote signals sent by agents x and x' in time-step t, and ξ_t any auxiliary information. A set of signals (auxiliary information) sent in time-steps 1 through T

[6] The agent will loop back to $\mathcal{R}_1^n$, increases his chances of winning a high utility resources, instead of moving through a large number of undesirable resources.

is denoted by $\sigma_{1:T}$ ($\xi_{1:T}$). In the context of PALMA, these signals represent either an attempt to acquire a resource, or a back-off from a previously contested resource,[7] while the auxiliary information corresponds to s (which determines the set of resources $\mathcal{R}_s$, see Equation 4.1, 4.5). Following (Triastcyn and Faltings, 2020), we also introduce the notion of *privacy cost*:

$$c_t(\sigma_t, \xi_t, x, x', \lambda) \triangleq \max \begin{cases} \lambda \mathcal{D}_{\lambda+1}[p(\sigma_t|\xi_t, x) \| p(\sigma_t|\xi_t, x')] \\ \lambda \mathcal{D}_{\lambda+1}[p(\sigma_t|\xi_t, x') \| p(\sigma_t|\xi_t, x)] \end{cases} \tag{4.6}$$

where $\mathcal{D}_\lambda(\cdot\|\cdot)$ is the Rényi divergence[8] of order[9] λ, defined as (Triastcyn, 2020):

$$\mathcal{D}_\lambda(P\|Q) = \frac{1}{\lambda - 1} \log \mathbb{E}_p \left[\left(\frac{p(x)}{q(x)} \right)^{\lambda - 1} \right] dx \tag{4.7}$$

$$= \frac{1}{\lambda - 1} \log \mathbb{E}_q \left[\left(\frac{p(x)}{q(x)} \right)^{\lambda} \right] dx, \tag{4.8}$$

Note that since our selection and back-off distributions are mixtures of two categorical distributions (see Equations 4.1 and 4.5), it is simple to compute the Rényi divergence.

4.5.1 PALMA's Privacy Cost

Every matching game starts with a fresh set of agents with random identifiers. Each agent computes (*once*, and *off-line*) the highest possible privacy cost at any round (c_{max}), i.e., the maximum value between the worst possible privacy cost during resource selection and back-off:

$$c_{max} = \max \begin{cases} \max\limits_{\xi_t \in \{1,\dots,R\}} \max\limits_{x' \in \mathcal{N}^x} \max\limits_{\sigma_t \in \mathcal{R}^x_{\xi_t} \sim P^n_S(\cdot)} c_t(\cdot) \\ \max\limits_{\xi_t \in \{1,\dots,R\}} \max\limits_{x' \in \mathcal{N}^x} \max\limits_{\sigma_t \in \mathcal{R}^x_{\xi_t} \sim P^n_B(\cdot)} c_t(\cdot) \end{cases} \tag{4.9}$$

The agents do not change their utilities during the matching process (i.e., the distributions $P^n_S(\cdot)$ and $P^n_B(\cdot)$ stay fixed), thus each agent can *compute a priori* the total privacy cost (*worst case privacy guarantees*) and the maximum number of rounds until the budget B_n is exhausted and he has to play according to the noise distributions. Agents can then adjust their privacy parameters accordingly. The actual privacy loss is accounted on the fly during execution (see lines 10 and 19 of Algorithm 2).

[7] In an arbitrary multi-agent domain, the signal would correspond to an action of an agent.

[8] Analytic expressions for Rényi divergence exist for many common distributions and can be found in (Gil et al., 2013). (Van Erven and Harremos, 2014) provides a good survey of Rényi divergence properties in general.

[9] λ is a hyper-parameter – assume for simplicity $\lambda \in \mathbb{N}$.

To bound the total privacy loss over multiple rounds and compute ε from δ or vice versa, we can use an advanced composition theorem. As stated, the advanced compositions theorem for the Bayesian accountant (Triastcyn and Faltings, 2020), the moments accountant (Abadi et al., 2016) and the Rényi accountant (Mironov, 2017) are equivalent in this case, resulting in:

$$\log \delta \leq \sum_{t=1}^{T} c_{max}(\cdot) - \lambda\varepsilon \qquad\qquad \varepsilon \leq \frac{1}{\lambda}\sum_{t=1}^{T} c_{max}(\cdot) - \frac{1}{\lambda}\log \delta \tag{4.10}$$

It is important to note that the above ε and δ should not be published, since the agent uses his own utility function to calculate the cost (in Equation 4.9).

Elaborative Example on the Privacy Cost Calculation

In this section we expand on our practical example on MoD systems introduced in Section 4.4.1.

Recall that PLDP provides *Local* DP guarantee, meaning a bound on the outcome probabilities for *any* pair of individual agents, inside the region. As such, to compute the privacy cost per round, each agent n has to identify the neighbors that would result to the maximum privacy loss (i.e., their selection (back-off) distributions result in the largest Rényi divergence, see Equation 4.9). Thus, each agent n independently identifies two agents n', and n'' *from his neighborhood* that result in the worst privacy loss given the agent's selection and back-off distributions (Equation 4.1 and 4.5, respectively). Then, he can compute the *worst case* privacy loss in any round by taking the maximum of the two values (Equation 4.9). Using this information, each agent is able to (i) compute his total privacy cost *a priori* and adjust his privacy parameters accordingly, (ii) keep track of his privacy budget at every time-step, and (iii) calculate his total ε after convergence. This process needs to happen *once*, *offline*. As mentioned, each agent can adjust the size of the neighborhood $\mathcal{N}^n$ (e.g., length ℓ, see Section 4.7.2) according to his privacy needs.

4.6 Evaluation Setup

We evaluate PALMA in a *mobility-on-demand* and a *paper assignment* application, using *real-data* for both. We focus on the social welfare (sum of utilities, $\sum_{n\in\mathcal{N}} u_n(\cdot)$) and level of privacy ($\varepsilon$ given $\delta = 10^{-5}$). Each problem instance is run 32 times. We report the average value for the social welfare, the average value for the median of ε, and the maximum value of ε. Error bars represent one standard deviation. We set $\zeta_S = 0.2$, $\zeta_B = \gamma = 0.05$, $B_n = 1$, $\lambda = 32$ for the Mobility-on-Demand test-case and $\zeta_S = 0.1$, $\zeta_B = \gamma = 0.05$, $B_n = 1$, $\lambda = 32$ for the paper assignment.

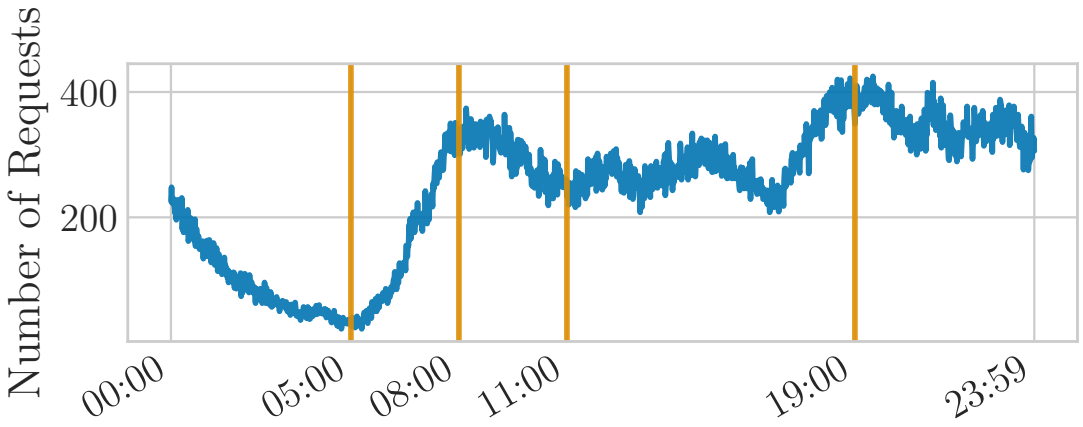


Figure 4.2: Request per minute in Manhattan on Jan. 15, 2016. Vertical lines denote the selected evaluation instances.

4.7 Evaluation Results: Mobility-on-Demand

4.7.1 Motivation

The emergence and widespread use of mobility-on-demand (MoD) services (e.g., ridesharing platforms like Uber or Lyft) in recent years has had a profound impact on urban transportation. Normally the process is facilitated by a centralized operator, that requires accurate location information of passengers and vehicles, which raises privacy concerns. Such a problem is ideal to showcase PALMA, as explained in Section 4.3.1. Moreover, contrary to other approaches (e.g., (Prorok and Kumar, 2017; Fioretto et al., 2018)) PALMA is *decentralized* and employs *Local DP*, providing privacy against a malicious data curator.

The Ridesharing and Fleet Relocation problems can be decomposed into three weighted matching sub-problems, all of which can be solved efficiently by ALMA (as demonstrated in Section 3.6, and in Appendix A), and thus by PALMA as well. In this test-case we will focus on passenger to vehicle matching, using PLDP and PALMA to provide a *scalable*, *on-device*, decentralized solution that *protects user preferences* (user location in this context).

4.7.2 Setting

Our evaluation setting is specifically designed to *resemble reality as closely as possible*, following the modeling described in Appendix A. We have used the NYC yellow taxi trip records (Taxi and Commission, 2016). For every request, the dataset provides amongst others the geo-location coordinates.

We report results on four 30s instances on a typical day (Jan 15th). These instances were selected to represent various distributions of demand (see Figure 4.2): the two highest peaks, the lowest peak, and a mid-day low.[10] We selected 30s periods because in practice the granularity of in-batches approaches for MoD services is between[11] 10s to 30s (Alonso-Mora et al., 2017; Riley et al., 2020; Prorok and Kumar, 2017; Danassis et al., 2022b). It is important to stress this *does not affect the scalability* of the proposed approach. Running PALMA for a day, for example, would simply result in running $24 \times 60 \times 2$ batches (as was done in Section 3.6, and Appendix A). Assuming similar distributions for requests and vehicles,[12] the social welfare and privacy cost of each agent will remain approximately the *same*, since the privacy cost (Equation 4.9) *only depends on the size of the region* $\mathcal{D}_i$.

The set of agents $\mathcal{N}$ is composed by the requests in Manhattan (17, 154, 116, and 174 requests in total on each of the evaluated batches). The set of resources $\mathcal{R}$ includes an equal number of vehicles scattered across the map. To avoid cold start, the position of each of the vehicles was set to the drop-off geo-location of the last (prior to the start time of the simulation) x requests (where x is the number of vehicles in each case). We used the Manhattan distance as a distance function (using the Haversine formula[13] to calculate the distance in each coordinate), as we found it to be a close approximation of the actual driving distance in Manhattan (see Appendix A). The utility function is $u_n(r) = e^{-\frac{d(n,r)}{\alpha}}$, where $\alpha = 4000$ controls the steepness and $d(n, r)$ denotes the distance between agent n and resource r (in m). We opted to use an exponential function to enable short pick-up times, as research conducted by ridesharing companies shows that a short pick-up time is important for passengers' satisfaction (Tang et al., 2017; Brown, 2016b).

The map is divided into *fixed* square regions of edge length ℓ (which correspond to the $\mathcal{D}_i$). PLDP demands that a user is indistinguishable,[14] from the attacker's point of view, from any potential user that could exist in the same region[15] (i.e., all his neighbors, see Sections 4.4.1 and 4.4.1). We have evaluated $\ell \in \{1000, 2000, 3000, 4000\}$ m, which

[10] Specifically, 05:00:00 - 05:00:30 represents the lowest demand, 08:00:00 - 08:00:30 and 19:00:00 - 19:00:30 represent the two rush hours (in the morning and evening, respectively), and finally, 11:00:00 - 11:00:30 represents a mid-day low.

[11] We also ran the same instances in batches of 10s and obtained *better results* (in terms of social welfare), but opted to present the worst case.

[12] A reasonable assumption given that our choice of evaluated distributions covers all the extremes, and a typical mid-day demand.

[13] https://en.wikipedia.org/wiki/Haversine_formula

[14] In the ridesharing scenario, we face repeated weighted matching problems; after a driver drops off a passenger, he is matched with a new one. Usually the matching process is performed in batches (e.g., every 10s). Assuming there is no vehicle relocation between the last drop off and the next match, we might have information leakage on the drop off location of the last passenger. To avoid this problem, we can use one-time ids for both the taxis and the passengers in every match, since both sets change dynamically anyway. Note that this problem is *only relevant in this domain*; other applications, like the paper assignment problem, are not susceptible to this vulnerability.

[15] We assume that potential neighbors are 100m apart in every direction.

Algorithm 3 Method for obfuscating the geo-location coordinates of agents and resources (based on (Andrés et al., 2013)).

Obfuscating geo-location (lat, lon) by drawing a point (r, θ) from a polar Laplacian

1. Draw θ uniformly in $[0, 2\pi)$
2. Draw p uniformly in $[0, 1)$ and set $r = C_\epsilon^{-1}(p)$, where $C_\epsilon^{-1}(p) = -\frac{1}{\epsilon}\left(W_{-1}(\frac{p-1}{e}) + 1\right)$, $\epsilon = \frac{\varepsilon}{l/2}$, l is the privacy region's diameter, and $W_{-1}(\cdot)$ is the Lambert W function (the -1 brunch).
3. Set $dx = r\cos(\theta)$ and $dy = r\sin(\theta)$
4. Set $lat = lat + (dy \times 0.00000899)$ and $lon = lon + (dx \times 0.00000899)/\cos(lat \times \pi/180)$, where 0.00000899 is one meter in degrees, calculated as 1 over the earth's radius in meters.

roughly correspond to an area of $\{45.6, 182.5, 410.5, 730\}$ city blocks.[16] Figure 4.1 offers a visual representation of the setting.

4.7.3 Baselines

We employ the centralized Hungarian algorithm (Kuhn, 1955) to compute the non-private optimal – in terms of social welfare – solution, which we use to compare the loss in social welfare of all of the evaluated algorithms. We compare PALMA against three privacy-preserving baselines:

1. The Hungarian algorithm (Kuhn, 1955) – which is an optimal assignment centralized algorithm – made private by obfuscating (adding noise) the geo-location coordinates according to geo-indistinguishability (Andrés et al., 2013) (similarly to (Prorok and Kumar, 2017)).

2. The original ALMA algorithm under similarly obfuscated (noisy) geo-location coordinates.[17]

3. The maximally private solution (i.e., the centralized random).

For the geo-indistinguishability-based baselines, we calculated a noisy geo-location for each agent and resource, according to Algorithm 3.

4.7.4 Social Welfare

For $\varepsilon = B_n = 1$ given $\delta = 10^{-5}$ (Figure 4.3), PALMA loses between $13.9 \pm 4.1\%$ ($\ell = 1000$) to $31.7 \pm 3.6\%$ ($\ell = 4000$) in social welfare compared to the non-private, optimal solution. The dotted lines represent the upper and lower bound; the upper bound assumes infinite

[16] The standard city block in Manhattan is about 80 m × 274 m (https://en.wikipedia.org/wiki/City_block).

[17] Note that we also attempted to use the original ALMA with Local Differential Privacy, yet, due to the large problem size, the privacy budget only sufficed for one round.

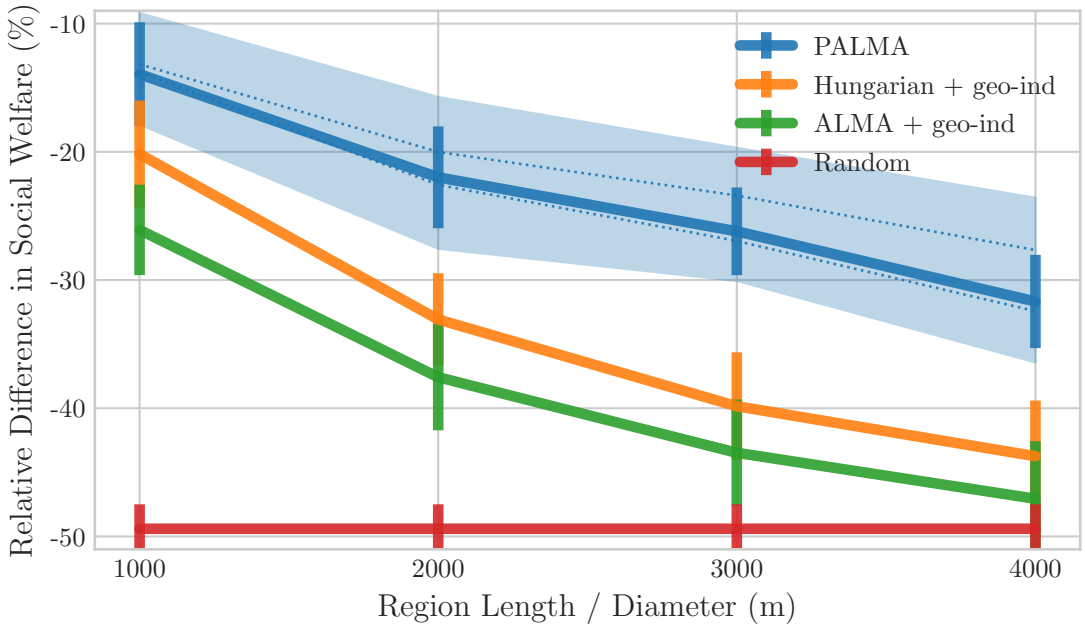


Figure 4.3: Loss in SW compared to the non-private, optimal solution for increasing region edge length (ℓ) and $\varepsilon = 1$. The dotted lines represent the upper ($\varepsilon \to \infty$) and lower ($\varepsilon = 0$) bound for PALMA, while the shaded area adds one standard deviation to the aforementioned bounds (see Section 4.7.4).

budget ($B_n \to \infty$) thus the agents play according to their own utilities ($\zeta_S = \zeta_B = 1$), while the lower bound assumes zero budget ($B_n = 0$) thus the agents play according to the noise distribution ($\zeta_S = \zeta_B = 0$), i.e., according to the utilities of the representative agent. The shaded area adds one standard deviation to the aforementioned bounds.

For the same ε guarantee and the same length as the privacy diameter, Hungarian + geo-ind loses between $20.2 \pm 4.2\%$ to $43.7 \pm 4.3\%$, while ALMA + geo-ind loses between $26.1 \pm 3.5\%$ to $47.1 \pm 4.5\%$. Finally, the maximally private solution (i.e., the centralized random), losses $49.4 \pm 2\%$.

PLDP and the carefully crafted noise of PALMA, allows PALMA to *outperform even the centralized optimal* solution (Hungarian + geo-ind) by 27.6% ($\ell = 4000$) to 30.9% ($\ell = 1000$). In fact, if we increase the privacy requirement to $\varepsilon = 0.75$, the improvement increases to 31.3% ($\ell = 4000$) to 45.9% ($\ell = 1000$). Note that, besides the higher social welfare for the same privacy guarantee, PALMA is inherently decentralized and orders of magnitude faster than the Hungarian.

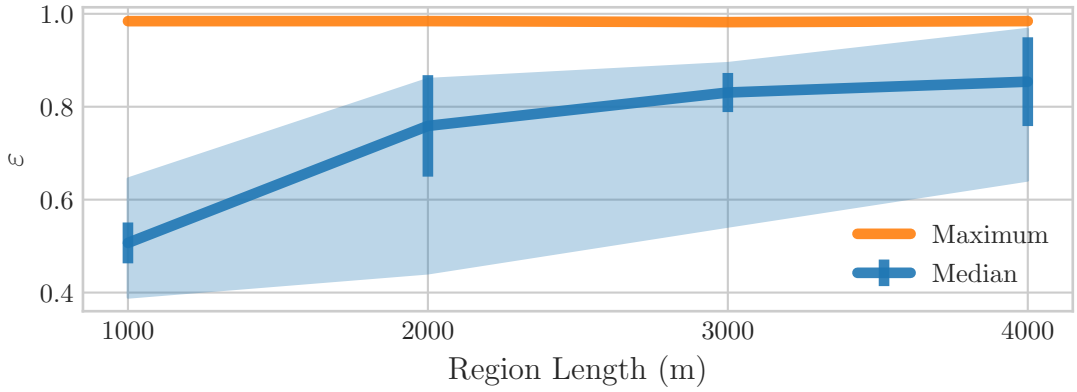
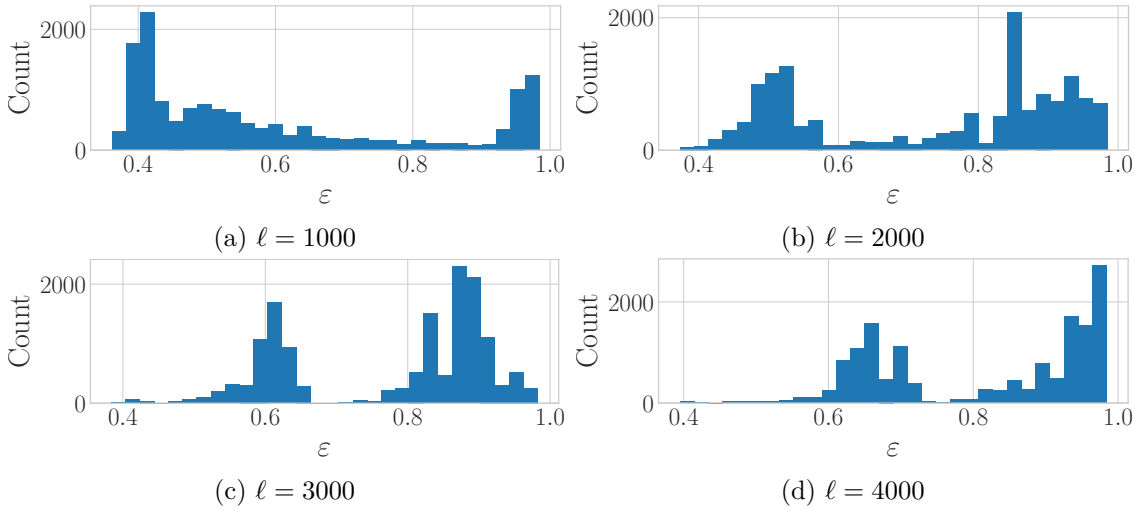


Figure 4.4: Maximum (orange) and median (blue) per-agent ε for increasing region length (ℓ). The shaded area represents the range between the max and min value of the median.



(a) $\ell = 1000$ (b) $\ell = 2000$

(c) $\ell = 3000$ (d) $\ell = 4000$

Figure 4.5: Histogram of per-agent ε for varying privacy region edge length. We include all 32 runs (32 (runs) $\times$ (17 + 154 + 116 + 174) (agents) = 14752 data points).

4.7.5 Privacy

While the worst-case guarantee is the same across the evaluated methods, PALMA yields a stronger result on a per-agent basis. In PALMA, every agent has a budget $\varepsilon = B_n$ and can compute a priori the maximum number of rounds until the budget is exhausted and he has to play according to the noise distributions (see Section 4.5.1). During runtime, though, most agents converge in a few rounds (i.e, few privacy mechanism invocations), thus accumulating smaller privacy loss compared to geo-ind based methods.

To demonstrate the latter, Figure 4.4 depicts the maximum (out of all the 32 runs) and median (average median value over the 32 runs) per-agent ε for increasing values of privacy region length ℓ. PALMA is able to achieve a *strong level of privacy* even in

large-scale simulations. The average value of the median for $\ell = 1000$ is only 0.5. Of course, the maximum per-agent ε is bounded by the privacy budget (i.e., $\varepsilon = 1$). Recall that $\ell = 1000$ m corresponds to an area of 45.6 city blocks, and $\ell = 4000$ m is larger than the width of Manhattan (which is 3700 m wide at its widest).

Figure 4.5 plots the histogram of the per-agent ε for varying privacy region edge length (ℓ). For $\ell = 1000$ (Figure 4.5a), only 3572 out of 14752 agents (24.2%) have $\varepsilon > 0.75$. This is because the majority of the agents converge fast (see Sections 3.4, and 3.5), thus only a small percentage of them exhaust their budget. In fact, almost half of the total agents (6759 / 14752, or 45.8%) have $\varepsilon \leq 0.5$. It is clear that the vast majority of agents benefit from really high degree of privacy.

4.7.6 Regions, Representative Agents, and Noise

In addition to the advantages of PLDP described in Section 4.3.3, there is another, more practical advantage that stems from the use of domain knowledge. The fragmentation function $\phi(\cdot)$ and the choice of the representative agent per region are domain specific. If the problem at hand (and by extension the utility function of the participating agents) is such that the representative agent has similar utilities to other agents in the region (and if we properly select the correct representative agent so that he is indicative of the agents in the region), then the social welfare will not degrade much, even under really strict budgets. Acting according to the representative agent, in such cases, allows for more informed allocations. This is a fundamental difference compared to, e.g., geo-indistinguishability, where the social welfare degrades in a significantly higher rate (as demonstrated in Figure 4.3). The latter can also be important for outlier agents, whose privacy cost per round might be high and thus lack the budget to play according to their own utilities for many rounds.

Regarding the choice of the fragmentation function $\phi(\cdot)$, there is a clear trade-off between the region size and the privacy cost per round, which in turn informs the amount of noise (ζ_S and ζ_B). Restricting our privacy guarantees to a region helps reduce the required noise, since all the agents in a region have similar preferences (less noise is needed to become indistinguishable). If the privacy cost per round is small, an agent can afford lower noise (larger ζ_S and ζ_B). Alternatively, acting according to the utilities of a properly chosen representative agent will still result in high quality allocations (especially in smaller regions, e.g., $\ell = 1000$), thus an agent might choose to accept higher noise in order to end up with much lower privacy cost at the end.

Finally, while in this work ζ_S and ζ_B are the same for all agents (see Section 4.6), one can potentially achieve better results using adaptive noise. For example, agents can assume lower noise for the first few time-steps, and gradually increase it over time. Note, that the noise selection scheme must not depend on the agents' preferences. We leave this

open for future work.

4.8 Evaluation Results: Paper Assignment

4.8.1 Setting

In this test-case, we protect the reviewers' preferences during the paper assignment phase of a conference. We used the multi-aspect review assignment evaluation dataset (Karimzadehgan and Zhai, 2008). It contains 73 papers (which corresponds to the set of resources $\mathcal{R}$ in our setting) from the ACM SIGIR conference of 2007, and 189 prospective reviewers (which corresponds to the set of agents $\mathcal{N}$) composed by authors of published papers in the top information retrieval conferences between 1971-2006. Each paper and each reviewer is represented by a 25-dimensional binary label, representing one of the 25 major areas of ACM SIGIR (Karimzadehgan et al., 2008).

We used the 25 major areas to define the privacy regions. Specifically, for each reviewer and paper, we selected uniformly at random one of the subject areas that they belong to, and set it as the *primary* subject area. The primary subject area is unique, and identifies the region. The proposed PLDP demands that users belonging to the same region be indistinguishable from the attacker's point of view. This would correspond to reviewers with the same primary subject area. We refer to the remaining subject areas as *secondary*. The maximum number of secondary subject areas of any adversary in a region defines the range of that region (reviewers are indistinguishable in that range). In this test-case, we consider adversaries with at most 2, 3, and 4 additional subject areas.[18] In layman's terms, a reviewer would be indistinguishable from any other reviewer that has the same primary subject area, and is an expert in at most 3, 4, and 5 areas in total.

Finally, for each paper and reviewer, we convert the 25-dimensional binary label to a continuous-valued vector. Specifically, the primary subject area is assigned the value 1, all the secondary subject areas are assigned the value 0.5, and the rest of the areas are assigned the value 0.1. The latter reflects the fact that conferences trust the expertise of reviewers to asses the quality of papers in a broader area. Following the literature (Ahmed et al., 2017), we used the cosine similarity (Equation 4.11) of their label vectors to compute the utility of a paper to a reviewer.

$$u_n(r) = \frac{\vec{n} \cdot \vec{r}}{\|\vec{n}\| \|\vec{r}\|} \tag{4.11}$$

where $\vec{n}$ ($\vec{r}$) denotes the 25-dimensional label of agent n (resource r).

[18] This would correspond to cosine distance of ≤ 0.2, ≤ 0.25, and ≤ 0.3, respectively, from an agent that has a single subject area (the primary subject area of the corresponding region).

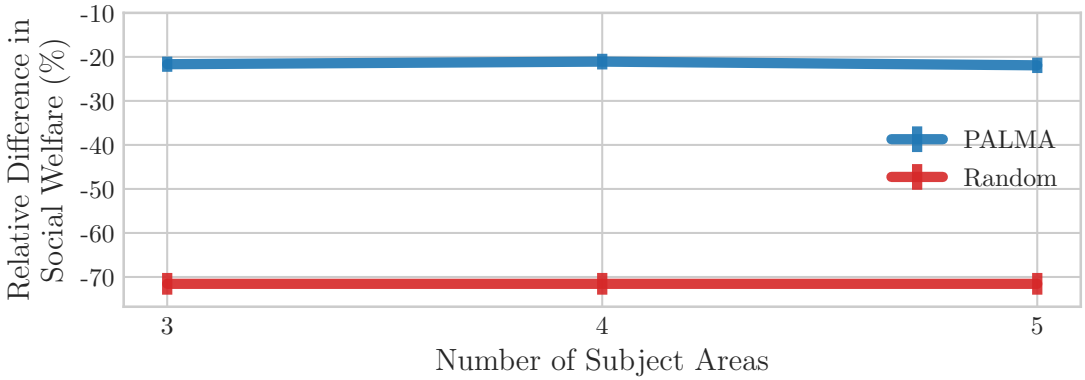


Figure 4.6: Loss in social welfare compared to the non-private, optimal solution for increasing size of the privacy region (i.e., number of subject areas).

Note that in a real-world paper assignment scenario, each reviewer would be required to review more than one paper (i.e., our matching graph would be a bipartite hypergraph). This can be easily handled by PALMA. Specifically, each reviewer will be represented by x 'copies', where x is the number of papers each reviewer should review. Then, a resource (paper) would only signal agent n that it is free (line 22 of Algorithm 2) if (i) it has been assigned to less than y agents – where y represents the number of reviews per paper – and (ii) a 'copy' of agent n has not acquired the resource. Nevertheless, this is out of the scope of this thesis; the goal of this test-case is to provide additional evidence on the performance of PALMA on *real data*. Thus, we opted to assign each reviewer to only one paper.

4.8.2 Baselines

As before, we employ the centralized Hungarian algorithm (Kuhn, 1955) to compute the non-private optimal – in terms of social welfare – solution, which we use to compute the loss in social welfare of PALMA. Note that in this test-case we do not compare to any geo-indistinguishability (Andrés et al., 2013) baselines because geo-indistinguishability is not directly applicable in this domain, and modifying it to fit the domain is out of the scope of this thesis. To the best of our knowledge, there is *no other privacy-preserving weighted matching algorithm to compare to.*

4.8.3 Social Welfare

For $\varepsilon = 1$ given $\delta = 10^{-5}$ (Figure 4.6), PALMA loses between $21.1 \pm 1.8\%$ to $21.9 \pm 1.8\%$ in social welfare compared to the non-private, optimal solution. The maximally private solution (i.e., the centralized random), losses $71.6 \pm 2.5\%$.

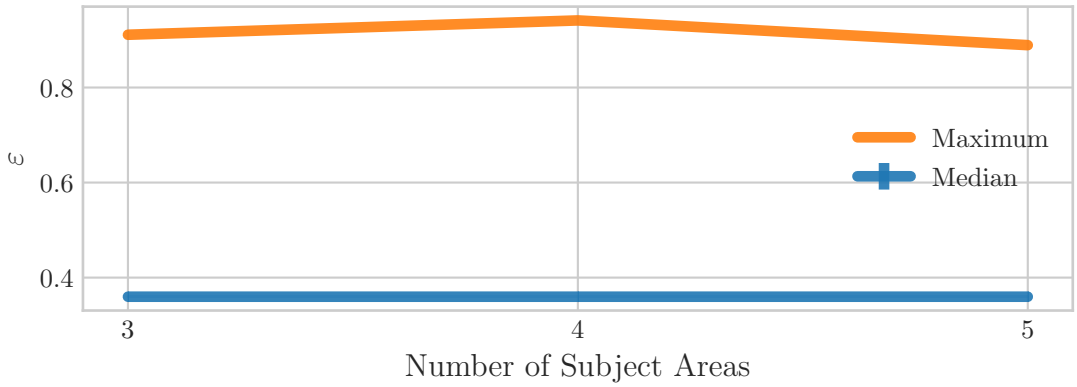


Figure 4.7: Maximum (orange line) and median (blue line) per-agent ε for increasing values of the size of the privacy region (i.e., number of subject areas).

Contrary to the Mobility-on-Demand test-case of Section 4.7, we observe a drop in social welfare. This is because in this test-case the number of agents is 2.6 times bigger than the number of resources (189 reviewers vs. 73 papers in the dataset). As a result, the majority of the reviewers remain un-matched. This does not constitute a problem for the centralized Hungarian, since it can compute a maximum-weight matching. Yet, in a randomized algorithm like PALMA, having an agent probabilistically backing-off can lead to a drop in solution quality, as the majority of the agents will end up without a resource (i.e., zero reward). This is also reflected in the dramatic drop in social welfare of the random solution, which now losses 71.6% compared to the 49.4% loss in the Mobility-on-Demand test-case. This also suggests that in a real-world setting, where the number of papers is actually larger than the number of reviewers, PALMA will be able to close the gap in social welfare compared to the optimal solution.

4.8.4 Privacy

Figure 4.7 depicts the maximum (out of all the 32 runs) and median (average median value over the 32 runs) per-agent ε for increasing values of the size of the privacy region (i.e., number of additional subject areas). The average value of the median is 0.36. Only between $0.9 - 2.1\%$ of the agents have $\varepsilon > 0.75$ (for the three privacy regions cases). The maximum per-agent ε is bounded by the privacy budget (i.e., $\varepsilon = 1$).

4.9 Societal Impact

The rapid proliferation of intelligent systems and autonomous agents has the potential to positively impact many facets of our daily lives. However, harnessing their power requires massive amounts of personal data to be collected, stored, processed, and analyzed – often by resource-constrained devices. The latter has raised serious privacy concerns and has

resulted in an accelerated growth of privacy advocacy movements. Our work shows that harnessing the potential of intelligent systems does not have to compromise privacy.

4.10 Chapter Conclusion

Bridging the gap between physical and cyber worlds will bring about significant privacy risks and the potential to reveal highly sensitive information of users. In this chapter, we consider the problem of hiding the utility function in multi-agent coordination problems. We propose *PALMA*, a practical and scalable privacy-preserving algorithm for weighted matching along with PLDP, a 'context-aware' privacy model that takes into account the 'distance' between two utility functions. This ensures indistinguishability between agents with similar preferences. PALMA is decentralized, runs on-device, requires no inter-agent communication, converges in constant time under reasonable assumptions, and provides a *strong level of privacy* ($\varepsilon \leq 1$ and median as low as $= 0.5$), while achieving high quality matchings (up to 86% of the non-private optimal). To the best of our knowledge, we are the first to develop a practical and scalable framework for weighted matching and resource allocation in general, unboundedly large, multi-agent systems.

5 Learning to Estimate Resource Contention

5.1 Preface

5.1.1 Contribution and Sources

This chapter is largely based on (Danassis et al., 2021c). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable):

PD (author):	Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – Original Draft, Writing – Review & Editing
Florian Wiedemair:	Software, Investigation, Writing – Review & Editing (supporting)
Boi Faltings:	Investigation (supporting), Writing – Review & Editing (supporting), Supervision

5.1.2 Chapter Summary

We present a multi-agent learning algorithm, *ALMA-Learning*, for efficient and fair allocations in large-scale systems. We circumvent the traditional pitfalls of multi-agent learning (e.g., the moving target problem, the curse of dimensionality, or the need for mutually consistent actions) by relying on the ALMA heuristic as a coordination mechanism for each stage game. ALMA-Learning is decentralized, observes only own action/reward pairs, requires no inter-agent communication, and achieves near-optimal ($< 5\%$ loss) and fair coordination in a variety of synthetic scenarios and a real-world meeting scheduling problem. The lightweight nature and fast learning constitute ALMA-Learning ideal for on-device deployment.

5.2 Introduction

A significant challenge for any algorithm for the assignment problem (and weighted matching in general) emerges from the nature of real-world applications, which are often of *massive* scale, *distributed* and *information-restrictive*. Our proposed heuristic, ALMA (see Chapter 3), was specifically designed to address the aforementioned challenges. ALMA is decentralized, completely uncoupled (agents are only aware of their own history), and requires no communication between the agents. Instead, agents make decisions locally, based on the contest for resources that they are interested in, and the agents that are interested in the same resources. This lightweight nature of ALMA coupled with the highly efficient allocations (as demonstrated in Chapters 3, and 4), make it ideal for an on-device solution for large-scale intelligent systems (e.g., IoT devices, smart cities and intelligent infrastructure, industry 4.0, autonomous vehicles, etc.).

Despite ALMA's high performance in a variety of domains, it remains a heuristic; i.e., sub-optimal by nature. In this chapter, we introduce a learning element (ALMA-Learning) that allows to quickly close the gap in social welfare compared to the optimal solution, while simultaneously increasing the fairness of the allocation. Specifically, in ALMA, while contesting for a resource, each agent will back-off with probability that depends on their *own utility loss* of switching to some alternative. ALMA-Learning improves upon ALMA by allowing agents to learn the chances that they will actually obtain the alternative option they consider when backing-off, which helps guide their search.

ALMA-Learning is applicable in *repeated allocation* games (e.g., self organization of intelligent infrastructure, autonomous mobility systems, etc.), but can be also applied as a *negotiation protocol* in one-shot interactions, where agents can simulate the learning process offline, before making their final decision. A motivating *real-world* application is presented in Section 5.5, where ALMA-Learning is applied to solve a large-scale meeting scheduling problem.

5.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We introduce ALMA-Learning, a distributed learning algorithm for large-scale multi-agent coordination**, focusing on *scalability* and *on-device* deployment in real-world applications.
2. **We prove that ALMA-Learning converges.**
3. **We provide a thorough evaluation in a variety of synthetic benchmarks and a real-world meeting scheduling problem.** In all of them ALMA-Learning is able to quickly (as little as 64 training steps) reach allocations of high social

welfare (less than 5% loss) and fairness (up to almost 10% lower inequality compared to the best performing baseline).

5.2.2 Discussion and Related Work

Multi-agent coordination can usually be formulated as a matching problem. Finding a maximum weight matching is one of the best-studied combinatorial optimization problems (see (Su, 2015; Lovász and Plummer, 2009)). There is a plethora of polynomial time algorithms, with the *Hungarian* algorithm (Kuhn, 1955) being the most prominent centralized one for the bipartite variant (i.e., the assignment problem). In real-world problems, a centralized coordinator is not always available, and if so, it has to know the utilities of all the participants, which is often not feasible. Decentralized algorithms (e.g., (Giordani et al., 2010)) solve this problem, yet they require polynomial computational time and polynomial number of messages – such as cost matrices (Ismail and Sun, 2017), pricing information (Zavlanos et al., 2008), or a basis of the LP (Bürger et al., 2012), etc. (see also (Kuhn et al., 2016; Elkin, 2004) for general results in distributed approximability under only local information/computation).

While the problem has been 'solved' from an algorithmic perspective – having both centralized and decentralized polynomial algorithms – it is not so from the perspective of multi-agent systems; two of the key reasons for that are: (1) complexity, and (2) communication. The proliferation of intelligent systems will give rise to *large-scale*, *multi-agent* based technologies. Algorithms for maximum-weight matching, whether centralized or distributed, have runtime that increases with the total problem size, even in the realistic case where agents are interested in a small number of resources. Thus, they can only handle problems of some bounded size. Moreover, they require a significant amount of inter-agent communication. As the number and diversity of autonomous agents continue to rise, differences in origin, communication protocols, or the existence of legacy agents will bring forth the need to collaborate without any form of explicit communication (Stone et al., 2010). Most importantly though, communication between participants (sharing utility tables, plans, and preferences) creates high overhead. On the other hand, under reasonable assumptions about the preferences of the agents, ALMA's runtime is *constant* in the total problem size, while requiring no message exchange (i.e., no communication network) between the participating agents. The proposed approach, ALMA-Learning, *preserves* the aforementioned two properties of ALMA.

From the perspective of Multi-Agent Learning (MAL), the problem at hand falls under the paradigm of multi-agent reinforcement learning, where for example it can be modeled as a Multi-Armed Bandit (MAB) problem (Auer et al., 2002b), or as a Markov Decision Process (MDP) and solved using a variant of Q-Learning (Busoniu et al., 2008). In MAB problems an agent is given a number of arms (resources) and at each time-step has to decide which arm to pull to get the maximum expected reward. In Q-learning agents solve Bellman's

optimality equation (Bellman, 2013) using an iterative approximation procedure so as to maximize some notion of expected cumulative reward. Both approaches have arguably been designed to operate in a more challenging setting, thus making them susceptible to many pitfalls inherent in MAL. For example, there is no stationary distribution, in fact, rewards depend on the joint action of the agents and since all agents learn simultaneously, this results in a moving-target problem. Thus, there is an inherent need for coordination in MAL algorithms, stemming from the fact that the effect of an agent's action depends on the actions of the other agents, i.e. actions must be mutually consistent to achieve the desired result. Moreover, the curse of dimensionality makes it difficult to apply such algorithms to large scale problems. ALMA-Learning solves both of the above challenges *by relying on ALMA as a coordination mechanism for each stage of the repeated game.* Another fundamental difference is that the aforementioned algorithms are designed to tackle the exploration/exploitation dilemma. A bandit algorithm for example will constantly explore, even if an agent has acquired his most preferred alternative. In matching problems, though, agents know (or have an estimate of) their own utilities. ALMA-Learning in particular, requires the knowledge of personal preference ordering and pairwise differences of utility (which are far easier to estimate than the exact utility table). The latter gives a great advantage to ALMA-Learning, since agents do not need to continue exploring after successfully claiming a resource, which stabilizes the learning process.

5.3 Proposed Approach: ALMA-Learning

In this section, we present ALMA-Learning – a novel learning algorithm for large-scale multi-agent coordination, built on top of the ALMA algorithm presented in Chapter 3 – and prove its convergence properties. We focus on weighted matchings in bipartite graphs (i.e., the assignment problem, see Section 2.2 for the formal definition).

The assignment problem refers to finding a maximum-weight matching in a weighted bipartite graph, $\mathcal{G} = \{\mathcal{N} \cup \mathcal{R}, \mathcal{E}\}$. In the studied scenario, $\mathcal{N} = \{1, \ldots, N\}$ agents compete to acquire $\mathcal{R} = \{1, \ldots, R\}$ resources. The weight of an edge $(n, r) \in \mathcal{E}$ represents the utility ($u_n(r) \in [0, 1]$) agent n receives by acquiring resource r. Each agent can acquire at most one resource, and each resource can be assigned to at most one agent. The goal is to maximize the sum of utilities.

5.3.1 Learning Rule

We begin by describing (a slightly modified version of) the ALMA heuristic, which is used as a subroutine by ALMA-Learning. The pseudo-codes for ALMA and ALMA-Learning are presented in Algorithms 4 and 5, respectively. Both ALMA and ALMA-Learning are run *independently and in parallel by all the agents* (to improve readability, we have

omitted the subscript $_n$).

Similar to the original ALMA algorithm, we make the following two assumptions: First, we assume (possibly noisy) knowledge of personal utilities by each agent. Second, we assume that agents can observe feedback from their environment to inform collisions and detect free resources. It could be achieved by the use of *sensors*, or by a *single bit* (0 / 1) feedback from the resource (note that these messages would be between the requesting agent and the resource, not between the participating agents themselves).

For both ALMA, and ALMA-Learning, each agent sorts his available resources (possibly $\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing utility $(r_0, \ldots, r_i, \ldots, r_{R^n-1})$ under his preference ordering $\prec_n$.

ALMA: ALtruistic MAtching Heuristic

ALMA converges to a resource through repeated trials. Let $\mathcal{A} = \{Y, A_{r_1}, \ldots, A_{r_{R^n}}\}$ denote the set of actions, where Y refers to yielding, and A_r refers to accessing resource r, and let g denote the agent's strategy. As long as an agent has not acquired a resource yet, at every time-step, there are two possible scenarios: If $g = A_r$ (strategy points to resource r), then agent n attempts to acquire that resource. If there is a collision, the colliding parties back-off (set $g \leftarrow Y$) with some probability. Otherwise, if $g = Y$, the agent chooses another resource r for monitoring. If the resource is free, he sets $g \leftarrow A_r$.

The back-off probability ($P(\cdot)$) is computed individually and locally based on each agent's expected loss. If more than one agent compete for resource r_i (line 8 of Algorithm 4), each of them will back-off with probability that depends on their expected utility loss. The expected loss array is computed by ALMA-Learning and provided as input to ALMA. The actual back-off probability can be computed with any monotonically decreasing function on $loss$. In this work we use $P(loss) = f(loss)^\beta$, where β controls the aggressiveness (willingness to back-off), and $f(\cdot)$ is given by:

$$f(loss) = \begin{cases} 1 - \epsilon, & \text{if } loss \leq \epsilon \\ \epsilon, & \text{if } 1 - loss \leq \epsilon \\ 1 - loss, & \text{otherwise} \end{cases} \tag{5.1}$$

Agents that do not have good alternatives will be less likely to back-off and vice versa. The ones that do back-off select an alternative resource and examine its availability. The resource selection is performed in sequential order, starting from the most preferred resource (see line 3 of Algorithm 4).

Algorithm 4 ALMA: Altruistic Matching Heuristic.

Require: Sort resources ($\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $r_0, \ldots, r_{R^n-1}$ under $\prec_n$

procedure ALMA(r_{start}, $loss[R]$)
  Initialize $g \leftarrow A_{r_{start}}$
  Initialize $current \leftarrow -1$
  Initialize $converged \leftarrow False$
  while !$converged$ **do**
    if $g = A_r$ **then**
      Agent n attempts to acquire resource r
      if Collision(r) **then**
        Back-off (set $g \leftarrow Y$) with probability $P(loss[r])$
      else
        $converged \leftarrow True$
    else ($g = Y$)
      $current \leftarrow (current + 1) \bmod R$
      Agent n monitors resource $r \leftarrow r_{current}$.
      if Free(r) **then**
        Set $g \leftarrow A_r$
  return r, such that $g = A_r$

Sources of Inefficiency

ALMA is a heuristic, i.e., sub-optimal by nature. It is worth understanding the sources of inefficiency, which in turn motivated ALMA-Learning. To do so, we provide a couple of adversarial examples.

In the original ALMA algorithm, all agents start attempting to claim their most preferred resource, and back-off with probability that depends on their loss of switching to the immediate next best resource. Specifically, in the simplest case, the probability to back-off when contesting resource r_i would be given by $P(loss(i)) = 1 - loss(i)$, where $loss(i) = u_n(r_i) - u_n(r_{i+1})$ and r_{i+1} is the next best resource according to agent n's preferences $\prec_n$.

The first example is given in Table 5.1a. Agent n_3 backs-off with high probability (higher than agent n_1) when contesting for resource r_1 assuming a good alternative, only to find resource r_2 occupied. Thus, n_3 ends up matched with resource r_3. The social welfare of the final allocation is 2, which is 20% worse than the optimal (where agents n_1, n_2, n_3 are matched with resources r_3, r_2, r_1, respectively, achieving a social welfare of 2.5). ALMA-Learning solves this problem by learning an empirical estimate of the loss an agent will incur if he backs-off from a resource. In this case, agent n_3 will learn that his loss is not $1 - 0.9 = 0.1$, but actually $1 - 0 = 1$, and thus will not back-off in subsequent stage games, resulting in an optimal allocation.

In another example (Table 5.1b), agents n_1 and n_3 always start by attempting to acquire resource r_1, reasoning that it is the most preferred one. Yet, in a repeated game, each of them only wins r_1 half of the time (for a social welfare 2, which is 28.5% worse than the

Agents \ Resources	r_1	r_2	r_3
n_1	1	0	0.5
n_2	0	1	0
n_3	1	0.9	0

(a)

Agents \ Resources	r_1	r_2	r_3
n_1	1	0.9	0
n_2	0	1	0.9
n_3	1	0.9	0

(b)

Table 5.1: Adversarial examples:
(5.1a) *Inaccurate loss estimate.* Agent n_3 backs-off with high probability when contesting for resource r_1 assuming a good alternative, only to find resource r_2 occupied.
(5.1b) *Inaccurate reward expectation.* Agents n_1 and n_3 always start by attempting to acquire resource r_1, reasoning that it is the most preferred one, yet each of them only wins r_1 half of the time.

optimal 2.8), thus, in expectation, resource r_1 has utility 0.5. ALMA-Learning solves this by learning an empirical estimate of the reward of each resource. In this case, after learning, either agent n_1 or n_3 (or both), will start from resource r_2. Agent n_2 will back-off since he has a good alternative, and the result will be the optimal allocation where agents n_1, n_2, n_3 are matched with resources r_2, r_3, r_1 (or r_1, r_3, r_2), respectively.

ALMA-Learning

ALMA-Learning uses ALMA as a sub-routine, specifically as a coordination mechanism for each stage of the repeated game. Over time, ALMA-Learning learns which resource to select first (r_{start}) when running ALMA, and an accurate empirical estimate on the loss the agent will incur by backing-off ($loss[]$). By learning these two values agents take more informed decisions, specifically: (1) If an agent often loses the contest of his starting resource, the expected reward of that resource will decrease, thus in the future the agent will switch to an alternative starting resource, and (2) if an agent backs-off from contesting resource r expecting low loss, only to find that all his high utility alternatives are already occupied, then his expected loss of resource r ($loss[r]$) will increase, making him more reluctant to back-off in some future stage game. In more detail, ALMA-Learning learns and maintains the following information:[1]

(i) $rewardHistory[R][L]$: A 2D array. For each $r \in \mathcal{R}$ it maintains the L most recent reward values received by agent n, i.e., the L most recent $u_n(r_{won})$, where $r_{won} \leftarrow ALMA(r, loss[])$. See line 11 of Algorithm 5. The array is initialized to the utility of each resource (line 3 of Algorithm 5).

(ii) $reward[R]$: A 1D array. For each $r \in \mathcal{R}$ it maintains an empirical estimate on the expected reward received by starting at resource r and continue playing according to Algorithm 4. It is computed by averaging the reward history of the resource, i.e.,

[1] We have omitted the subscript $_n$ from all the variables and arrays, but every agent maintains their own estimates.

Algorithm 5 ALMA-Learning

Require: Sort resources ($\mathcal{R}^n \subseteq \mathcal{R}$) in decreasing order of utility $r_0, \ldots, r_{R^n-1}$ under $\prec_n$
Require: $rewardHistory[R][L]$, $reward[R]$, $loss[R]$
1: **procedure** ALMA-Learning
2: **for all** $r \in \mathcal{R}$ **do** ▷ Initialization
3: $rewardHistory[r]$.add($u(r)$)
4: $reward[r] \leftarrow rewardHistory[r]$.getMean()
5: $loss[r] \leftarrow u(r) - u(r_{next})$
6: $r_{start} \leftarrow \arg\max_r reward[r]$
7:
8: **for** $t \in [1, \ldots, T]$ **do** ▷ T: Time horizon
9: $r_{won} \leftarrow ALMA(r_{start}, loss[])$ ▷ Run ALMA
10:
11: $rewardHistory[r_{start}]$.add($u(r_{won})$)
12: $reward[r_{start}] \leftarrow rewardHistory[r_{start}]$.getMean()
13: **if** $u(r_{start}) - u(r_{won}) > 0$ **then**
14: $loss[r_{start}] \leftarrow$
15: $(1-\alpha)loss[r_{start}] + \alpha\,(u(r_{start}) - u(r_{won}))$
16:
17: **if** $r_{start} \neq r_{won}$ **then**
18: $r_{start} \leftarrow \arg\max_r reward[r]$

$\forall r \in \mathcal{R} : reward[r] \leftarrow rewardHistory[r].getMean()$. See line 12 of Algorithm 5.

(iii) $loss[R]$: A 1D array. For each $r \in \mathcal{R}$ it maintains an empirical estimate on the loss in utility agent n incurs if he backs-off from the contest of resource r. The loss of each resource r is initialized to $loss[r] \leftarrow u_n(r) - u_n(r_{next})$, where r_{next} is the next most preferred resource to r, according to agent n's preferences $\prec_n$ (see line 5 of Algorithm 5). Subsequently, for every stage game, agent n starts by selecting resource r_{start}, and ends up winning resource r_{won}. The loss of r_{start} is then updated according to the following averaging process, where α is the learning rate:

$$loss[r_{start}] \leftarrow (1-\alpha)loss[r_{start}] + \alpha\,(u(r_{start}) - u(r_{won}))$$

Finally, the last condition (lines 17-18 of Algorithm 5) ensures that agents who have acquired resources of high preference stop exploring, thus stabilizing the learning process.

5.3.2 Convergence

Convergence of ALMA-Learning does not translate to a fixed allocation at each stage game. The system has converged when agents no longer switch their starting resource, r_{start}. The final allocation of each stage game is controlled by ALMA, which means that even after convergence there can be contest for a resource, i.e., having more than one agent selecting the same starting resource. As we will demonstrate later, this translates

to fairer allocations, since agents with similar preferences can alternate between acquiring their most preferred resource.

Theorem 5. There exists time-step t_{conv} such that $\forall t > t_{conv} : r^n_{start}(t) = r^n_{start}(t_{conv})$, where $r^n_{start}(t)$ denotes the starting resource r_{start} of agent n at the stage game of time-step t.

Proof. (Sketch) Theorem 1 proves that ALMA (called at line 9 of Algorithm 5) converges in polynomial time (in fact, under some assumptions, it converges in constant time, i.e., *each stage game converges in constant time*). In ALMA-Learning agents switch their starting resource only when the expected reward for the current starting resource drops below the best alternative one, i.e., for an agent to switch from r_{start} to r'_{start}, it has to be that $reward[r_{start}] < reward[r'_{start}]$. Given that utilities are bounded in $[0, 1]$, there is a maximum, finite number of switches until $reward_n[r] = 0, \forall r \in \mathcal{R}, \forall n \in \mathcal{N}$. In that case, the problem is equivalent to having N balls thrown randomly and independently into N bins (since $R = N$). Since both R, N are finite, the process will result in a distinct allocation in finite steps with probability 1. □

Proof. (Complete) Theorem 1 of Chapter 3 proves that ALMA (called at line 9 of Algorithm 2) converges in polynomial time.

In fact, under the assumption that each agent is interested in a subset of the total resources (i.e., $\mathcal{R}^n \subset \mathcal{R}$) and thus at each resource there is a bounded number of competing agents ($\mathcal{N}^r \subset \mathcal{N}$) Corollary 2 proves that the expected number of steps any individual agent requires to converge is independent of the total problem size (i.e., N and R). In other words, by bounding these two quantities (i.e., we consider R^n and N^r to be constant functions of N, R), the convergence time of ALMA is *constant* in the total problem size N, R. Thus, under the aforementioned assumptions: *each stage game converges in constant time.*

Now that we have established that the call to the ALMA procedure will return, the key observation to prove convergence for ALMA-Learning is that agents switch their starting resource only when the expected reward for the current starting resource drops below the best alternative one, i.e., for an agent to switch from r_{start} to r'_{start}, it has to be that $reward[r_{start}] < reward[r'_{start}]$. Given that utilities are bounded in $[0, 1]$, there is a maximum, finite number of switches until $reward_n[r] = 0, \forall r \in \mathcal{R}, \forall n \in \mathcal{N}$. In that case, the problem is equivalent to having N balls thrown randomly and independently into N bins (since $R = N$). Since both R, N are finite, the process will result in a distinct allocation in finite steps with probability 1. In more detail, we can make the following arguments:

(i) Let r_{start} be the starting resource for agent n, and $r'_{start} \leftarrow \arg\max_{r \in \mathcal{R}/\{r_{start}\}} reward[r]$. There are two possibilities. Either $reward[r_{start}] > reward[r'_{start}]$ for all time-steps

$t > t_{converged}$ – i.e., $reward[r_{start}]$ can oscillate but always stays larger than $reward[r'_{start}]$ – or there exists time-step t when $reward[r_{start}] < reward[r'_{start}]$, and then agent n switches to the starting resource r'_{start}.

(ii) Only the reward of the starting resource r_{start} changes at each stage game. Thus, for the reward of a resource to increase, it has to be the r_{start}. In other words, at each stage game that we select r_{start} as the starting resource, the reward of every other resource remains (1) unchanged and (2) $reward[r] < reward[r_{start}], \forall r \in \mathcal{R}\ \{r_{start}\}$ (except when an agent switches starting resources).

(iii) There is a finite number of times each agent can switch his starting resource r_{start}. This is because $u_n(r) \in [0, 1]$ and $|u_n(r) - u_n(r')| > \delta, \forall n \in \mathcal{N}, r \in \mathcal{R}$, where δ is a small, strictly positive minimum increment value. This means that either the agents will perform the maximum number of switches until $reward_n[r] = 0, \forall r \in \mathcal{R} \forall n \in \mathcal{N}$ (which will happen in finite number of steps), or the process will have converged before that.

(iv) If $reward_n[r] = 0, \forall r \in \mathcal{R}, \forall n \in \mathcal{N}$, the question of convergence is equivalent to having N balls thrown randomly and independently into R bins and asking whether you can have exactly one ball in each bin – or in our case, where $N = R$, have no empty bins. The probability of bin r being empty is $\left(\frac{R-1}{R}\right)^N$, i.e., being occupied is $1 - \left(\frac{R-1}{R}\right)^N$. The probability of all the bins to be occupied is $\left(1 - \left(\frac{R-1}{R}\right)^N\right)^R$. The expected number of trials until this event occurs is $1/\left(1 - \left(\frac{R-1}{R}\right)^N\right)^R$, which is finite, for finite N, R. □

5.3.3 Complexity

ALMA-Learning is an anytime algorithm. At each training time-step, we run ALMA once. Thus, the computational complexity is bounded by T times the bound for ALMA, where T denotes the number of training time-steps (see Equation 5.2, where N and R denote the number of agents and resources, respectively, $p^* = f(loss^*)$, and $loss^*$ is given by the Equation 5.3).

$$\mathcal{O}\left(TR\frac{2-p^*}{2(1-p^*)}\left(\frac{1}{p^*}\log N + R\right)\right) \tag{5.2}$$

$$loss^* = \arg\min_{loss_n^r}\left(\min_{r\in\mathcal{R},n\in\mathcal{N}}(loss_n^r), 1 - \max_{r\in\mathcal{R},n\in\mathcal{N}}(loss_n^r)\right) \tag{5.3}$$

5.3.4 Fairness

The usual predicament of efficient allocations is that they assign the resources only to a fixed subset of agents, which leads to an unfair result. Consider the simple example of

		Resources		
		r_1	r_2	r_3
Agents	n_1	1	0.5	0
	n_2	0	1	0
	n_3	1	0.75	$\epsilon \to 0$

Table 5.2: Adversarial example: Unfair allocation. Both ALMA (with higher probability) and any optimal allocation algorithm will assign the coveted resource r_1 to agent n_1, while n_3 will receive utility 0.

Table 5.2. Both ALMA (with higher probability) and any optimal allocation algorithm will assign the coveted resource r_1 to agent n_1, while n_3 will receive utility 0. But, using ALMA-Learning, agents n_1 and n_3 will update their expected loss for resource r_1 to 1, and randomly acquire it between stage games, increasing fairness. Recall that convergence for ALMA-Learning does not translate to a fixed allocation at each stage game. To capture the fairness of this 'mixed' allocation, we report the average fairness on 32 evaluation time-steps that follow the training period.

In this chapter, we have evaluated two different fairness metrics (see Section 2.5): the *The Jain index* (Jain et al., 1998) and *The Gini coefficient* (Gini, 1912).

5.4 Evaluation Results: Synthetic Benchmarks

5.4.1 Setting

We evaluate ALMA-Learning in a variety of synthetic benchmarks and a meeting scheduling problem (Section 5.5) based on *real* data from (Romano and Nunamaker, 2001). Error bars representing one standard deviation (SD) of uncertainty. In this section, we present results on three synthetic benchmarks:

(a) *Map*: Consider a Cartesian map on which the agents and resources are randomly distributed. The utility of agent n for acquiring resource r is proportional to the inverse of their distance, i.e., $u_n(r) = 1/d_{n,r}$. Let $d_{n,r}$ denote the Manhattan distance. We assume a grid length of size $\sqrt{4 \times N}$.

(b) *Noisy Common Utilities*: This pertains to an anti-coordination scenario, i.e., competition between agents with similar preferences. We model the utilities as: $\forall n, n' \in \mathcal{N}, |u_n(r) - u_{n'}(r)| \leq$ noise, where the noise is sampled from a zero-mean Gaussian distribution, i.e., noise $\sim \mathcal{N}(0, \sigma^2)$.

(c) *Binary Utilities*: This corresponds to each agent being indifferent to acquiring any resource amongst his set of desired resources, i.e., $u_n(r)$ is randomly assigned to 0 or 1.

5.4.2 Baselines and Evaluation Setup

We compare against: (i) the *Hungarian algorithm* (Kuhn, 1955), which computes a maximum-weight matching in a bipartite graphs, (ii) *ALMA*, and (iii) the *Greedy* algorithm, which goes through the agents randomly, and assigns them their most preferred, unassigned resource.

We run each configuration 16 times and report the average values. Since we have randomized algorithms, we also run each problem instance of each configuration 16 times. ALMA, and ALMA-Learning's parameters were set to: $\alpha = 0.1, \beta = 2, \epsilon = 0.01, L = 20$.

5.4.3 Social Welfare

We begin with the loss in social welfare compared to the optimal solution. Figures 5.1a, 5.2a, and 5.3a present the results for the three test-cases. Table 5.3 aggregates the results.

In all of the test-cases, ALMA-Leaning is able to quickly reach *near-optimal* allocations. On par with previous results from Chapter 3, ALMA loses around 10% to 15%.[2] Is is worth noting that test-cases that are 'harder' for ALMA – specifically test-case (a) Map, where ALMA maintains the same gap on the optimal solution as the number of resources grow, and test-case (c) Binary Utilities, where ALMA exhibits the highest loss for 16 resources[3] – are 'easier' to learn for ALMA-Learning. In the aforementioned two test cases, ALMA-Learning was able to learn near-optimal to optimal allocations in just $64 - 512$ training time-steps (in fact in certain cases it learns near-optimal allocations in as little as 32 time-steps). Contrary to that, in test-case (b) Noisy Common Utilities, ALMA-Learning requires significantly more time to learn (we trained for 8192 time-steps), especially for larger games ($R > 256$). Intuitively we believe that this is because ALMA already starts with a near optimal allocation, and given the high similarity on the agent's utility tables (especially for $\sigma = 0.1$), it requires a lot of fine-tuning to improve the result.

5.4.4 Fairness

Moving on to fairness, ALMA-Learning achieves the *most fair allocations in all of the test-cases*, fairer than the optimal (in terms of social welfare) solution. As an example, ALMA-Learning's Gini coefficient is -18% to -90% lower on average (across problem sizes) than ALMA's, -24% to -93% lower than Greedy's, and -0.2% to -10% lower than Hungarian's.

Figures 5.1b, 5.2b, and 5.3b depict the Gini coefficient (lower is better) for the three

[2]Note that both ALMA and ALMA-Learning use the same function $P(loss) = f(loss)^{\beta}$ (see Section 5.3.1) to compute the back-off probability, in order to provide a fair common ground for the evaluation.

[3]Binary utilities represent a somewhat adversarial test-case for ALMA, since the agents can not utilize the more sophisticated back-off mechanism based on the loss (loss is either 1, or 0 in this case).

Table 5.3: Range of the average loss (%) in social welfare compared the the (centralized) optimal for the three different benchmarks.

	Greedy	ALMA	**ALMA-Learning**
(a) Map	1.51% – 18.71%	0.00% – 9.57%	0.00% – 0.89% (512 training time-steps) 0.00% – 1.68% (64 training time-steps)
(b) Noisy Common Utilities, $\sigma = 0.1$	8.13% – 12.86%	2.96% – 10.58%	1.34% – 2.26% (8192 training time-steps)
(b) Noisy Common Utilities, $\sigma = 0.2$	5.40% – 14.11%	1.37% – 12.33%	0.35% – 1.97% (8192 training time-steps)
(b) Noisy Common Utilities, $\sigma = 0.4$	0.79% – 12.64%	0.75% – 10.74%	0.05% – 2.26% (8192 training time-steps)
(c) Binary Utilities	0.10% – 14.70%	0.00% – 16.88%	0.00% – 0.39% (64 training time-steps)

Table 5.4: Fairness – Jain index (the higher, the better) for the three different benchmarks. In parenthesis we include the average improvement in fairness of ALMA-Learning compared to ALMA, Greedy, and Hungarian, respectively.

	Hungarian	Greedy	ALMA	**ALMA-Learning**
(a) Map	0.79 – 0.86	0.70 – 0.73	0.75 – 0.80	0.86 – 0.89 (11.95%, 22.44%, 5.03%)
(b) Noisy Common Utilities, $\sigma = 0.1$	0.81 – 0.92	0.77 – 0.88	0.79 – 0.89	0.85 – 0.93 (5.58%, 7.58%, 1.81%)
(c) Binary Utilities	0.84 – 1.00	0.72 – 1.00	0.82 – 0.98	0.88 – 1.00 (10.18%, 5.36%, 0.58%)

Table 5.5: Fairness – Gini Coefficient (the lower, the better) for the three different benchmarks. In parenthesis we include the average improvement (decrease in inequality) of ALMA-Learning compared to ALMA, Greedy, and Hungarian, respectively.

	Hungarian	Greedy	ALMA	**ALMA-Learning**
(a) Map	0.19 – 0.23	0.30 – 0.34	0.25 – 0.28	0.17 – 0.20 (−29.04%, −42.91%, −9.63%)
(b) Noisy Common Utilities, $\sigma = 0.1$	0.17 – 0.24	0.21 – 0.29	0.19 – 0.28	0.16 – 0.23 (−18.29%, −23.66%, −6.52%)
(c) Binary Utilities	0.00 – 0.16	0.00 – 0.28	0.02 – 0.18	0.00 – 0.13 (−90.43%, −92.61%, −0.18%)

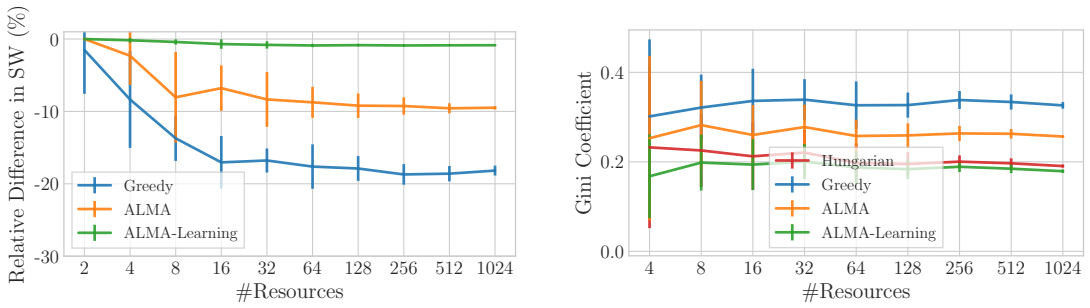
test-cases, while Tables 5.4 and 5.5 aggregate the results for both the Gini coefficient and the Jain index.[4]

5.5 Evaluation Results: Meeting Scheduling

5.5.1 Motivation

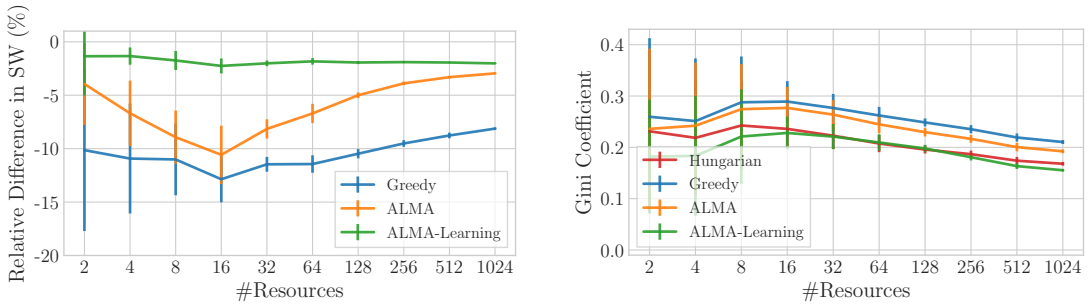
The problem of scheduling a large number of meetings between multiple participants is ubiquitous in everyday life (Nigam and Srivastava, 2020; Ottens et al., 2017; Zunino and Campo, 2009; Maheswaran et al., 2004; BenHassine and Ho, 2007; Hassine et al., 2004; Crawford and Veloso, 2005; Franzin et al., 2002). The advent of social media brought forth the need to schedule large-scale events, while the era of globalization and the shift to working-from-home require business meetings to account for participants with diverse preferences (e.g., different timezones).

[4]To improve readability, we have omitted the results for test-case (b) Noisy Common Utilities for $\sigma = 0.2$ and $\sigma = 0.4$. In both cases ALMA-Learning performed better than the reported results for $\sigma = 0.1$.



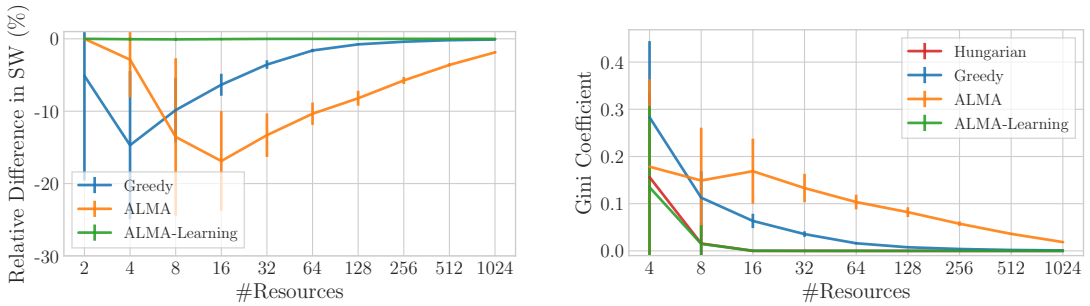
(a) Relative Difference in Social Welfare (%) (b) Gini Coefficient (lower is better)

Figure 5.1: Test-case (a): Map, we report the relative difference in social welfare, and the Gini coefficient, for increasing number of resources ($[2, 1024]$, x-axis in log scale), and $N = R$. For each problem instance, we trained ALMA-Learning for 512 time-steps.



(a) Relative Difference in Social Welfare (%) (b) Gini Coefficient (lower is better)

Figure 5.2: Test-case (b): Noisy Common Utilities, $\sigma = 0.1$, we report the relative difference in social welfare, and the Gini coefficient, for increasing number of resources ($[2, 1024]$, x-axis in log scale), and $N = R$. For each problem instance, we trained ALMA-Learning for 8192 time-steps.



(a) Relative Difference in Social Welfare (%) (b) Gini Coefficient (lower is better)

Figure 5.3: Test-case (c): Binary Utilities, we report the relative difference in social welfare, and the Gini coefficient, for increasing number of resources ($[2, 1024]$, x-axis in log scale), and $N = R$. For each problem instance, we trained ALMA-Learning for 64 time-steps.

Meeting scheduling is an inherently decentralized problem. Traditional approaches (e.g., distributed constraint optimization (Ottens et al., 2017; Maheswaran et al., 2004)) can only handle a bounded, small number of meetings. Interdependences between meetings' participants can drastically increase the complexity. While there are many commercially available electronic calendars (e.g., Doodle, Google calendar, Microsoft Outlook, Apple's Calendar, etc.), none of these products is capable of autonomously scheduling meetings, taking into consideration user preferences and availability.

While the problem is inherently online, meetings can be aggregated and scheduled in batches, similarly to the approach for tackling matching for mobility-on-demand applications (see Appendix A). In this test-case, we map meeting scheduling to an allocation problem and solve it using ALMA and ALMA-Learning. This showcases an application were ALMA-Learning can be used as a negotiation protocol.

5.5.2 Modeling

Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ denote the set of events and $\mathcal{P} = \{P_1, \ldots, P_m\}$ the set of participants. To formulate the participation, let part : $\mathcal{E} \rightarrow 2^{\mathcal{P}}$, where $2^{\mathcal{P}}$ denotes the power set of $\mathcal{P}$. We further define the variables $days$ and $slots$ to denote the number of days and time slots per day of our calendar (e.g., $days = 7, slots = 24$). In order to add length to each event, we define an additional function len : $\mathcal{E} \rightarrow \mathbb{N}$. Participants' utilities are given by:

$$\text{pref} : \mathcal{E} \times \text{part}(\mathcal{E}) \times \{1, \ldots, days\} \times \{1, \ldots, slots\} \rightarrow [0, 1] \tag{5.4}$$

Mapping the above to the assignment problem of Section 2.2, we would have the set of (day, $slot$) tuples to correspond to $\mathcal{R}$, while each event is represented by one event agent (that aggregates the participant preferences), the set of which would correspond to $\mathcal{N}$.

Detailed modeling of the meeting scheduling problem can be found in Appendix B.

5.5.3 Baselines and Evaluation Setup

We compare against four baselines: (i) We used the IBM ILOG CP optimizer (Laborie et al., 2018) to formulate and solve the problem as a CSP.[5] An additional benefit of this solver is that it provides an upper bound for the optimal solution (which is infeasible to compute). (ii) A modified version of the MSRAC algorithm (BenHassine and Ho, 2007), and finally, (iii) the greedy and (iv) ALMA, as before.

We run each configuration 10 times and report the average values. The time limit for the CPLEX optimizer was set to 20 minutes.[6]

[5] Computation time limit 20 minutes.

[6] CPLEX ran on an Intel i7-7700HQ CPU (4 cores, 2.8 Ghz base clock) and 16 GB of ram.

We used the SMAC[7] tool (Hutter et al., 2011) to choose the hyper-parameters. This resulted in selecting a logistic curve to compute the back-off probability, specifically Equation 5.5, where $\gamma = 15.72$. To compute the loss, we used Equation 5.6, where $k = 13$. ALMA-Learning's parameters were set to: $\alpha = 0.1, L = 20$.

$$f(loss) = \frac{1}{1 + e^{-\gamma(0.5 - loss)}} \tag{5.5}$$

$$loss_n^i = \frac{\sum_{j=i+1}^{k} (u_n(r_i) - u_n(r_j))}{k - i} \tag{5.6}$$

5.5.4 Designing Large Test-Cases

As the problem size grows, CPLEX's estimate on the upper bound of the optimal solution becomes too loose (see Figure 5.4a). To get a more accurate estimate on the loss in social welfare for larger test-cases, we designed a large-instance by combining smaller problem instances, making it easier for CPLEX to solve which in turn allowed for tighter upper bounds as well (see Figure 5.4b).

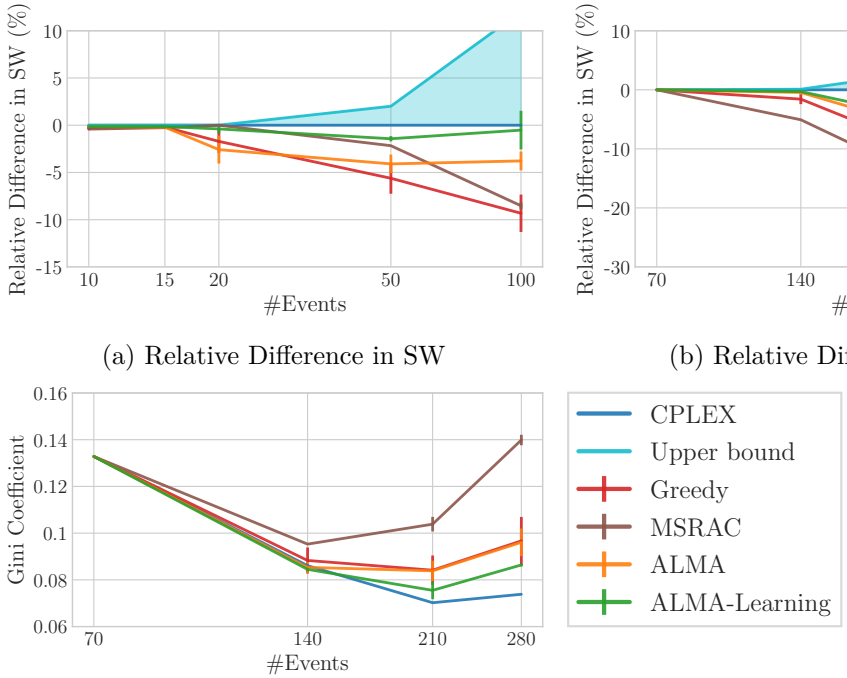
We begin by solving two smaller problem instances with a low number of events. We then combine the two in a calendar of twice the length by duplicating the preferences, resulting in an instance of twice the number of events (agents) and calendar slots (resources). Specifically, in this case we generated seven one-day long sub-instances (with 10, 20, 30 and 40 events each), and combined then into a one-week long instance with 70, 140, 210 and 280 events, respectively. The fact that preferences repeat periodically, corresponds to participants being indifferent on the day (yet still have a preference on time).

These instances are depicted in Figure 5.4b and in the last line of Table 5.6.

5.5.5 Social Welfare

Figures 5.4a and 5.4b depict the relative difference in social welfare compared to CPLEX for 100 participants ($|\mathcal{P}| = 100$) and increasing number of events for the regular ($|\mathcal{E}| \in [10, 100]$) and larger test-cases ($|\mathcal{E}|$ up to 280), respectively. Table 5.6 aggregates the results for various values of $\mathcal{P}$. ALMA-Learning is able to achieve less than 5% loss compared to CPLEX, and this difference diminishes as the problem instance increases (less than 1.5% loss for $|\mathcal{P}| = 100$). Finally, for the largest hand-crafted instance ($|\mathcal{P}| = 100, |\mathcal{E}| = 280$, last line of Table 5.6 and Figure 5.4b), ALMA-Learning loses less than 9% compared to the *possible upper bound* of the optimal solution.

[7]SMAC (sequential model-based algorithm configuration) is an automated algorithm configuration tool that uses Bayesian optimization (https://www.automl.org/automated-algorithm-design/algorithm-configuration/smac/).



(a) Relative Difference in SW

(b) Relative Difference in SW

(c) Gini Coefficient (lower is better)

Figure 5.4: Meeting Scheduling. Results for 100 participants ($\mathcal{P}$) and increasing number of events (x-axis in log scale). ALMA-Learning was trained for 512 time-steps.

Simulation Results in more detail can be found in Section B.6 of Appendix B.

5.5.6 Fairness

Moving on to fairness, Figure 5.4c depicts the Gini coefficient for the large, hand-crafted instances ($|\mathcal{P}| = 100, |\mathcal{E}|$ up to 280). ALMA-Learning exhibits low inequality, up to -9.5% lower than ALMA in certain cases. It is worth noting, though, that the fairness improvement is not as pronounced as in Section 5.4. In the meeting scheduling problem, all of the employed algorithms exhibit high fairness, due to the nature of the problem. Every participant has multiple meetings to schedule (contrary to only being matched to a single resource), all of which are drawn from the same distribution. Thus, as you increase the number of meetings to be scheduled, the fairness naturally improves.

Simulation Results in more detail can be found in Section B.6 of Appendix B.

Table 5.6: Range of the average loss (%) in social welfare compared to the IBM ILOG CP optimizer for increasing number of participants, $\mathcal{P}$ ($|\mathcal{E}| \in [10, 100]$). The final line corresponds to the loss compared to the upper bound for the optimal solution for the large test-case with $|\mathcal{P}| = 100, |\mathcal{E}| = 280$ (Figure 5.4b).

	Greedy	MSRAC	ALMA	**ALMA-Learning**
$\|\mathcal{P}\| = 20$	$6.16\% - 18.35\%$	$0.00\% - 8.12\%$	$0.59\% - 8.69\%$	$0.16\% - 4.84\%$
$\|\mathcal{P}\| = 30$	$1.72\% - 14.92\%$	$1.47\% - 10.81\%$	$0.50\% - 8.40\%$	$0.47\% - 1.94\%$
$\|\mathcal{P}\| = 50$	$3.29\% - 12.52\%$	$0.00\% - 15.74\%$	$0.07\% - 7.34\%$	$0.05\% - 1.68\%$
$\|\mathcal{P}\| = 100$	$0.19\% - 9.32\%$	$0.00\% - 8.52\%$	$0.15\% - 4.10\%$	$0.14\% - 1.43\%$
$\|\mathcal{E}\| = 280$	$0.00\% - 15.31\%$	$0.00\% - 22.07\%$	$0.00\% - 10.81\%$	$0.00\% - 8.84\%$

5.6 Chapter Conclusion

The next technological revolution will be interwoven with the proliferation of intelligent systems. To truly allow for scalable solutions, we need to shift from traditional approaches to multi-agent solutions, ideally run *on-device*. In this chapter, we present a novel learning algorithm (ALMA-Learning), which exhibits such properties, to tackle a central challenge in multi-agent systems: finding an optimal allocation between agents, i.e., computing a maximum-weight matching. We prove that ALMA-Learning converges, and provide a thorough empirical evaluation in a variety of synthetic scenarios and a real-world meeting scheduling problem. ALMA-Learning is able to quickly (in as little as 64 training steps) reach allocations of high social welfare (less than 5% loss) and fairness.

6 A Correlated Equilibrium for Accessing Indivisible Resources

6.1 Preface

6.1.1 Contribution and Sources

This chapter is largely based on (Danassis and Faltings, 2019). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable):

PD (author): Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – Original Draft, Writing – Review & Editing

Boi Faltings: Conceptualization (supporting), Methodology (supporting), Writing – Review & Editing, Supervision

6.1.2 Chapter Summary

We investigate the problem of multi-agent coordination under *rationality constraints*. Specifically, role allocation, task assignment, resource allocation, etc. Inspired by human behavior, we propose a framework (CA^3NONY) that enables fast convergence to efficient and fair allocations based on a simple convention of courtesy. We prove that following such convention induces a strategy which constitutes an ϵ-subgame-perfect equilibrium of the repeated allocation game with discounting, for a setting with indivisible resources with binary utilities. Simulation results highlight the effectiveness of CA^{3}NONY as compared to state-of-the-art bandit algorithms, since it achieves more than two orders of magnitude faster convergence, higher efficiency, fairness, average payoff, and can gracefully handle increasing number of resources, and *high congestion*.

6.2 Introduction

In multi-agent systems, agents are often called upon to implement a joint plan in order to maximize their rewards. Typically, coordination in a joint plan incorporates (possibly a combination of) two distinct elements: agents may be required to take the same action (Schelling, 1960; Cooper, 1999), or agents may be required to take distinct actions (Grenager et al., 2002; Cigler and Faltings, 2013). The latter is ubiquitous in everyday life, e.g., managing common-pool resources, or deciding on car/ship/airplane routes in traffic management, etc. In this chapter, we study coordination in repeated allocation games. Consider for example the problem of channel allocation in wireless networks (Wang et al., 2013). In such a problem, N wireless devices contend for R transmission slots (i.e., a particular frequency band in a certain period of time), where $N \gg R$. If more than one agent access a slot simultaneously, a collision occurs and the colliding parties incur a cost $\zeta < 0$. The goal then for the agents is to transmit on different slots to minimize collisions over time. Other scenarios include role allocation (e.g., teammates during a game), task assignment (e.g., employees of a factory), resource allocation (e.g., parking spaces and/or charging stations for autonomous vehicles), etc. What follows is applicable to *any such scenario*. For simplicity hereafter we will refer only to the allocation of indivisible resources (i.e., resources which can not be shared). Beyond the scope of repeated games, the proposed framework can also be applied as a negotiation protocol in one-shot interactions. For example, self-driving vehicles attempting to get a parking space can utilize such a protocol in a simulated environment with message exchange. Conforming to the relevant literature, in what follows we refer to the coordination problem of selecting distinct actions as the *anti-coordination problem* (Grenager et al., 2002; Cigler and Faltings, 2013; Bramoullé et al., 2004; Kun et al., 2013).

The most straightforward solution would be to have a central coordinator, with complete information, recommend an action to each agent. This, though, would require the agents to communicate their preferences/plans, which creates high overhead and raises incentive compatibility and truthfulness issues. Moreover, in real-world applications with partial observability, agents might not be willing to trust such recommendations. On the other hand, agents can learn to anti-coordinate their actions in a completely decentralized manner. However, in a fully decentralized scheme, agents have an incentive to 'bully' others (stick to your action until you drive out the competition) (Littman and Stone, 2002), which makes it impossible to converge to solutions that are both efficient and fair.[1] As such, a mechanism is needed that employs some sort of external authority. The proposed framework employs simple monitoring authorities (MA), with the primary goal to keep track of successful accesses. We do not require any planning capabilities by the MAs, nor any knowledge of the plans and preferences of the participants; only the ability to monitor

[1] In a setting where agents use ALMA, for example, a rational agent might not back-off from his most preferred resource.

successful accesses on their respective resource. Such MAs already exist naturally in many domains (e.g., the port authority in maritime traffic management (Agussurja et al., 2018), or the access point in wireless networks, etc.), and can be easily introduced in the rest (e.g., an agent monitoring each charging / parking station, etc.). At the end, agents learn to anti-coordinate their actions in a decentralized manner, while the introduced MAs allow us to impose quotas to the use of resources and punishments upon violating them to align individual incentives with an efficient and fair correlated equilibrium.

A second aspect of anti-coordination problems involves convergence speed. Inter-agent interactions often need to take place in an ad-hoc fashion. Typical approaches (e.g., Monte Carlo algorithms, Bayesian learning, bandit algorithms, etc.) tend to require too many rounds to converge to be feasible in dynamic environments, and real-life applications. Humans on the other hand are able to routinely coordinate in such an ad-hoc fashion. One key concept that facilitates human ad-hoc (anti-)coordination is the use of *conventions* (Lewis, 2008). Behavioral conventions are a fundamental part of human societies, yet they have not appeared meaningfully in empirical modeling of multi-agent systems. Inspired by human behavior, we propose the adoption of a simple convention of *courtesy*. Courtesy arises by the social nature of humans. Society demands that an individual should conduct himself in consideration of others. This allows for fast convergence, albeit it is not game theoretically sound; people adhere to it due to social pressure. Such problems become even more severe in situations with scarcity of resources. Under such conditions, courtesy breaks down and in the name of self-preservation people exhibit urgency and competitive behavior (Gupta and Gentry, 2016). Thus, to satisfy our rationality constraint (no incentive to deviate for self-interested agents) in an artificial system, we need a deterrent mechanism.

In this chapter we present a framework (CA^{3}NONY: **C**ontextual **A**nti-coordination in **A**d-hoc **Anony**mous games) for repeated allocation games with discounting. CA^{3}NONY reproduces courtesy based on ALMA and the allocation algorithm of (Cigler and Faltings, 2013), and uses monitoring and punishments as a deterrent mechanism. It exhibits *fast convergence*, and high *efficiency* and *fairness*, while relying only on occupancy feedback which facilitates scalability and does not require any inter-agent communication.

6.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We introduction an anti-coordination framework (CA^{3}NONY)** for repeated allocation games of indivisible resources, which consists of a *courteous convention* and a *monitoring scheme*.
2. **We prove that the use of the courteous convention induces strategies that constitute an approximate *subgame-perfect equilibrium***, under the

proposed CA^{3}NONY framework.

3. **We provide a thorough comparison to state-of-the-art bandit algorithms**, and demonstrate faster convergence speed, efficiency, and fairness.

6.2.2 Discussion and Related Work

In ad-hoc multi-agent (anti-)coordination the goal is to design autonomous agents that achieve high flexibility and efficiency in a setting that admits no prior coordination between the participants (Stone et al., 2010). Typical scenarios include the use of Monte Carlo algorithms (Barrett et al., 2017), Bayesian learning (Albrecht et al., 2016), or bandit algorithms (Chakraborty et al., 2017; Barrett and Stone, 2011). Traditionally, pure ad-hoc approaches suffer from slow learning (Crandall, 2014), which makes pure ad-hoc coordination a very ambitious goal for real-life applications. In this chapter we propose a middle-ground approach. Inspired by human ad-hoc coordination, we incorporate prior knowledge in the form of simple *conventions*. The coordination can still be considered ad-hoc as it is not pre-programmed, rather it involves learning. This allows for faster convergence compared to pure ad-hoc approaches.

A convention is defined as a customary, expected and self-enforcing behavioral pattern (Young, 1996; Lewis, 2008).[2] In multi-agent systems, there are two scopes through which we study conventions. First, a convention can be considered as a behavioral rule, designed and agreed upon ahead of time or decided by a central authority (Shoham and Tennenholtz, 1995; Walker and Wooldridge, 1995). Second, a convention may emerge from within the system itself through repeated interactions (Mihaylov et al., 2014; Walker and Wooldridge, 1995). The proposed courteous convention falls on the first category. It is incorporated as prior knowledge, and it is self-enforcing since the induced strategies constitute an approximate subgame-perfect equilibrium of the repeated allocation game.

An alternative way to model the anti-coordination problem is as a multi-armed bandit problem (Auer et al., 2002a). In multi-armed bandit problems an agent is given a number of arms and at each time-step has to decide which arm to pull to get the maximum expected reward. Bandit (or no-regret) algorithms typically minimize the total regret of each agent, which is the difference between the expected received payoff and the payoff of the best strategy in hindsight. As such, they satisfy our rationality constraint since they constitute an approximate correlated or coarse correlated equilibrium (Nisan et al., 2007; Roughgarden, 2016; Hart and Mas-Colell, 2000). However, the studied problem presents many challenges: there is no stationary distribution (adversarial rewards), all agents are able to learn (similar to recursive modeling) which results to a moving-target problem, and yielding gives a reward of 0 (desirable option for minimizing regret, but not in respect to fairness). Moreover, regret minimization does not necessarily lead to payoff

[2] See also (Wasik et al., 2018, 2017; Roelofsen, 2020) for related work on institutional robotics (using conventions/norms inspired by institutional economics).

maximization (Crandall, 2014). Nevertheless, due to their ability to learn from partial feedback, bandit algorithms constitute the natural choice for a pure ad-hoc approach. The latter motivates our choice to use them as a baseline, since our agents only receive binary feedback of success or failure upon taking their action.

Game theoretic equilibria are desirable – since they satisfy the rationality constraint – but hard to obtain. Deciding whether an anti-coordination (anonymous) game has a pure Nash equilibrium (NE) is NP-complete (Brandt et al., 2009). Furthermore, allocation games often admit undesirable equilibria: pure NE which are efficient but not fair, or mixed-strategy NE which are fair but not efficient (Cigler and Faltings, 2013). Hence, iterative best-response algorithms are not satisfactory. On the other hand, correlated equilibria (CE) (Aumann, 1974) can be both efficient & fair, while from a practical perspective they constitute perhaps the most relevant non-cooperative solution concept (Hart and Mas-Colell, 2000). An optimal CE of an anonymous game may be found in polynomial time (Papadimitriou and Roughgarden, 2008). Moreover, it is possible to achieve a CE without a correlation device (central agent) (Foster and Vohra, 1997; Hart and Mas-Colell, 2000), using the history of the actions taken by the opponents. However, we are interested in information-restrictive learning rules (i.e., completely uncoupled (Talebi, 2013)), where each agent is only aware of his own history of action/reward pairs. Such an approach was applied in (Cigler and Faltings, 2013) to design a decentralized algorithm for reaching efficient & fair CE in wireless channel allocation games. Yet, while the algorithm reaches an equilibrium in a polynomial number of steps, cooperation to achieve this state is not rational. A self-interested agent could keep accessing a resource forever, until everyone else backs-off (also known as 'bully' strategy (Littman and Stone, 2002)). ALMA (Chapter 3) faces a similar problem, and thus is only applicable in cooperative scenarios, and not in the presence of strategic agents. In this chapter, we build upon the ideas of Cigler and Faltings and develop an anti-coordination strategy that constitutes an approximate subgame-perfect equilibrium, i.e., cooperation with the algorithm is a best-response strategy at each sub-game of the original stage game, given any history of the play.

A generalization of anti-coordination games, called dispersion games, was described in (Grenager et al., 2002). In a dispersion game, agents are able to choose from several actions, favoring the one that was chosen by the smallest number of agents (analogous to minority games (Challet et al., 2013)). The authors in (Grenager et al., 2002) define a maximal dispersion outcome as an outcome where no agent can switch to an action chosen by fewer agents. The agents themselves do not have any particular preference for the attained equilibrium. Contrary to that, we are interested in achieving an efficient and fair outcome. Besides, in many real world applications, the agents are indifferent to which role/task/resource they attain, as long as they receive one (e.g., wireless frequencies). Tackling dispersion games, and therefore non-binary utilities, remains open for future research.

6.3 The CA^{3}NONY framework

6.3.1 The Repeated Allocation Game

Let a 'resource' be any element that can be successfully assigned to only one agent at a time. At each time-step, $\mathcal{N} = \{1, \ldots, N\}$ agents try to access $\mathcal{R} = \{1, \ldots, R\}$ identical and indivisible resources, where possibly $N \gg R$. The set of available actions is denoted as $\mathcal{A} = \{Y, A_1, \ldots, A_R\}$, where Y refers to yielding and A_r refers to accessing resource r. We assume that access to a resource is slotted and of equal duration.[3] A successful access yields a positive payoff, while no access has a payoff of 0. If more than one agent access a resource simultaneously, a collision occurs and the colliding parties incur a cost $\zeta < 0$. Thus, the agents only receive a binary feedback of success or failure. Let a_n denote agent n's action, and $a_{-n} = \times_{\forall n' \in \mathcal{N} \setminus \{n\}} a_{n'}$ the joint action for the rest of the agents. The payoff function is defined as:

$$u_n(a_n, a_{-n}) = \begin{cases} 0, & \text{if } a_n = Y \\ 1, & \text{if } a_n \neq Y \wedge a_i \neq a_n, \forall i \neq n \\ \zeta, & \text{otherwise} \end{cases} \tag{6.1}$$

We assume that rewards are discounted by $\delta \in (0, 1)$, and, conforming to real-world scenarios, that each agent n is only aware of his own history of action/reward pairs.

Finally, we assume that the agents can observe side information from their environment at each time-step t. We call this side information context (e.g., time, date etc.). The agents utilize this context as a common signal in their decision-making process, a means to learn, and anti-coordinate their actions. Let $\mathcal{K} = \{1, \ldots, K\}$ denote the context space. The rationale behind the introduction of the common context is that, under completely uncoupled learning rules, having positive probability mass on undesirable actions (e.g., collisions) is unavoidable. Moreover, from a practical perspective, common environmental signals are amply available to the agents (Hart and Mas-Colell, 2000). We do not assume any a priori relation between the context space and the problem. The only constraints are that the values should repeat periodically, and satisfy $K = \lceil N/R \rceil$.

The context signals could be produced either on the resource side (e.g., by the port authority in maritime traffic management, or the access point in wireless networks), or in a decentralized manner (e.g., in distributed networks with no authorities the senders can attach identifier signals to data traffic (Wang et al., 2013)). Finally, in situations where communication is possible, the agents can agree upon the signal themselves by solving the distributed consensus problem.

[3] This is done to facilitate the proofs. Real world problems have this property anyway (e.g., access to radio channels, role allocation in a plan, etc.), and the algorithms we compare to all require slotted resources too.

Algorithm 6 Learning rule.

Initialize g_n uniformly at random in $\mathcal{A}$. Set $accessed_n \leftarrow False$.
for $k_t \in \mathcal{K}$ **do**
    Agents observe context k_t
    if $g_n(k_t) = A_r$ & $accessed_n = False$ **then**
        Agent n accesses resource r
        if Collision(r) **then**
            Set $g_n(k_t) \leftarrow Y$ with probability $p_{backoff} > 0$
        else
            Set $accessed_n \leftarrow True$
    else if $g_n(k_t) = Y$ **then**
        Agent n monitors random resource $r \in \mathcal{R}$
        if Free(r) **then**
            Set $g_n(k_t) \leftarrow A_r$
Set $accessed_n \leftarrow False$

6.3.2 Adopted Convention

The adopted convention is based on ALMA, and the cooperative allocation algorithm of (Cigler and Faltings, 2013). Each agent n has a strategy $g_n : \mathcal{K} \rightarrow \mathcal{A}$ that determines a resource to access at time-step t after having observed context k_t. The strategy is initialized uniformly at random in $\mathcal{A}$. If $g_n(k_t) = A_r$, then agent n accesses resource r. Otherwise, if $g_n(k_t) = Y$, the agent does not access a resource but instead chooses uniformly at random a resource r to monitor for activity. If it is free, then the agent updates $g_n(k_t) \leftarrow A_r$. The pseudo-code of the learning rule can be found in Algorithm 6.

In (Cigler and Faltings, 2013), agents back-off probabilistically in case of a collision (set $g_n(k_t) \leftarrow Y$ with probability $p_{n_{backoff}}$). In such a setting, it is possible to reach a symmetric subgame-perfect equilibrium. But in order to actually play it, the agents need to be able to calculate it. It is not always possible to obtain the closed form of the back-off probability distribution of each resource. Furthermore, a self-interested agent could stubbornly keep accessing a resource forever, until everyone else backs off ('bully' strategy (Littman and Stone, 2002)).

Instead, we adopted a simple convention where agents are being *courteous*, i.e., if there is a collision, the colliding agents will back-off with some constant positive probability: $p_{n_{backoff}} = p > 0, \forall n \in \mathcal{N}$. Being courteous, though, does not satisfy the rationality constraint. However, a uniform distribution of resources is socially optimal (i.e., fair allocations maximize the social welfare). Hence, if we introduce quotas to the resources and punishments upon violating them, courtesy induces rational strategies. In the following sections we introduce a monitoring scheme and prove that the resulting strategy constitutes an ϵ-subgame-perfect equilibrium.

6.3.3 Rationality

In order to ensure the proposed convention's rationality, the agents must be assured that they will eventually be successful, i.e., we must provide safeguards against the monopolization of resources. The proposed framework employs simple Monitoring Authorities (MA). The MAs do not require any planning capabilities, nor any knowledge of the agents' plans and preferences. Their primary goal is to keep track of successful accesses. Depending on the domain, MAs may already exist naturally, or can easily be introduced. Examples include centralized MAs like the port authority in maritime traffic management, or a set of decentralized MAs (one per resource) like the access points in wireless networks, or agents monitoring a charging / parking station. Their function is twofold. First, they could provide occupancy signals. Agents (e.g., CA^{3}NONY, bandit, or Q-learning agents) must be able to receive some form of feedback from their environment to inform collisions and detect free resources. This could be achieved (if possible) by the use of various sensors, or by receiving occupancy signals (e.g., 0, 1) from the MAs. Second, which is their principal purpose, MAs deter agents from monopolizing resources to the point that each agent can access a resource only for one context value out of K. To achieve the latter, MAs must be able to keep track of successful accesses. Upon the violation of the imposed quotas, the framework is responsible to enforce the necessary punishments.[4] Punishments are application specific. They can be individual (e.g., in a wireless scenario, if an agent transmits to some other than the designated channel, then his packets will no longer be relayed), or group punishments (e.g., if quotas are exceeded, access is denied for everyone). The imposed punishments make exceeding the quotas an irrational strategy (simulated in Algorithm 6 by the $accessed_n$ flag), which in turn aligns the individual incentives with an efficient and fair correlated equilibrium.

Access Monitoring

The primary function of the employed MAs is the tracking of successful accesses by the agents. The latter can be achieved in various ways, depending on the domain. The simplest one would be to maintain a log with successful accesses per episode (period of context values), whether we are dealing with a centralized MA or set of decentralized MAs that are able to communicate to ensure coherence. For more involved scenarios (e.g., if we require agent anonymity per access) we may employ more complex schemes, e.g., using tokens, or artificial currency (solely as an internal mechanism). Note that the latter does not fence the resources (i.e., punishments are still required). In what follows, we present such a self-regulated monitoring scheme.

Initially all the agents that 'buy-in' are issued the same amount $m \in \mathbb{R}$ of artificial cash (AC). This amount also corresponds to the initial fee for every resource $f_r \leftarrow m$.

[4]The punishments can potentially be enforced retrospectively, thus the MAs – depending on the domain – can potentially operate asynchronously to the agents.

To allow access to resource r, the MA of r charges f_r units of AC, and monitors the event. If there was a successful access, the MA reimburses the amount of $(1-\xi)f_r$ AC to the accessing agent, where $\xi \to 0 \in \mathbb{R}$ is a commission fee. Otherwise, the MA reimburses the full amount of f_r AC to the colliding agents, so that they are able to try again for a different context value. Finally, after each episode, the MAs lower the fee to $f'_r \leftarrow (1-\xi)f_r, \forall r \in \mathcal{R}$. After every successful access, the amount of AC that an agent possesses drops below the access fee of a resource. Waiting for the fee to drop to the point that $f_r = m/2$ is not rational since, assuming $\xi \to 0$, the number of iterations required to allow accessing two resources at the same time will reach ∞. At that point the rest of the agents will have reached a correlated equilibrium and the adversarial agent will not have an incentive to access an additional resource, besides the one that corresponds to him, since it would result in a collision. If, due to implementation constraints, we can not select a small ξ, the MAs can change the artificial currency every I episodes, invalidating the old one and again making such strategy irrational.

6.3.4 Rate of Convergence

Theorem 6. In a repeated allocation game with N agents and R resources, the expected number of steps before Algorithm 6 converges to a correlated equilibrium is bounded by:

$$\mathcal{O}\left(N\left(\log\left\lceil\frac{N}{R}\right\rceil+1\right)(\log N+R)\right) \tag{6.2}$$

Proof. The adopted learning rule is based on the allocation algorithm of (Cigler and Faltings, 2013). We provide a slightly tighter bound on the convergence speed based on Theorems 12 and 13 of (Cigler and Faltings, 2013), and Theorem 1.

Theorem 7. For N agents, $R \geq 1$, $K \geq 1$, and back-off probability $0 < p < 1$, the expected number of steps before the learning algorithm converges to a correlated equilibrium of the allocation game for every $k \in \mathcal{K}$ is bounded by:

$$\mathcal{O}\left((K\log K+2K)\,R\frac{2-p}{2(1-p)}\left(\frac{1}{p}\log N+R\right)+1\right) \tag{6.3}$$

Proof. Plugging the hitting probability bound of Lemma 2 to the convergence proof of (Cigler and Faltings, 2013), results on the required bound. □

Finally, for a constant back-off probability and $K = \lceil N/R \rceil$, (6.3) results in the required bound. □

Corollary 3. Under a common, constant back-off probability assumption, $p^* = 2-\sqrt{2}$ minimizes the convergence time of Algorithm 6 in high congestion scenarios, i.e., $N \gg R$.

Proof. According to bound (6.3), in high congestion scenarios (i.e., $\frac{N}{R} = K \to \infty$), the common, constant back-off probability that minimizes the convergence time is:

$$p^* = \arg\min \left(\frac{2-p}{2(1-p)} \frac{1}{p} \right) = 2 - \sqrt{2} \tag{6.4}$$

□

6.3.5 Courtesy Pays Off

In this section we prove that if the agents back-off with a constant positive probability $p_{backoff} > 0$, then Algorithm 6 induces a strategy that is an ϵ-equilibrium.

Suppose that in a repeated allocation game with discounting ($\delta \in (0,1)$) the agents who collide back-off with a constant probability $p_{backoff} > 0$. Let σ^p denote the aforementioned strategy (courteous strategy), and σ^* denote the optimal (best-response) strategy under the monitoring authorities (possibly better than a stage game NE). Moreover, let $U_n(\sigma, \sigma_{-n}, \delta) = \sum_{t=0}^{\infty} \delta^t u_n(a_n^t, a_{-n}^t)$ denote the cumulative payoff of agent n following strategy σ, assuming the rest of the agents follow the strategy σ_{-n}. The following theorem proves that starting at any time-step, agent n does not gain more than ϵ by deviating to the optimal strategy σ^*.

Theorem 8. Under a high enough discount factor, the courteous strategy σ^p constitutes an approximate subgame-perfect equilibrium, i.e., $\forall \epsilon > 0$, $\exists \delta_0 \in (0,1)$ such that $\forall \delta, \delta_0 \leq \delta < 1$:

$$\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)] > (1-\epsilon)\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^p, \delta)]$$

Proof. Note that $\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^*, \delta)] \geq \mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^p, \delta)]$, since by playing σ_{-n}^p the rest of the agents can potentially introduce additional collisions. Thus, it suffices to prove that $\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)] > (1-\epsilon)\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^*, \delta)]$.

The introduced monitoring scheme prohibits the monopolization of resources, i.e., each agent can only access a resource for his corresponding context value. Thus, the best-response strategy's (σ^*) payoff for some δ is bounded by:

$$\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^*, \delta)] \leq 1 + \delta^K + \delta^{2K} + \cdots = \sum_{i=0}^{\infty} \delta^{iK} = \frac{1}{1-\delta^K} \tag{6.5}$$

When agents adopt the courteous convention, in each round until the system converges to a correlated equilibrium, the agents receive a payoff between $\zeta < 0$ (collision cost) and 1. Thus, until convergence, the expected payoff is lower bounded by $\zeta \sum_{i=0}^{\tau-1} \delta^{iK} = \zeta \frac{1-\delta^{\tau K}}{1-\delta^K}$, where τ is the number of steps to converge. After convergence, their expected payoff is

$\sum_{i=0}^{\infty} \delta^{\tau+iK} = \frac{\delta^\tau}{1-\delta^K}$. Hence, the convention induced strategy's payoff is at least:

$$\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)] \geq \sum_{\tau=1}^{\infty} Pr[\text{converge in } \tau \text{ steps}] \cdot \left(\frac{\zeta(1-\delta^{\tau K}) + \delta^\tau}{1-\delta^K} \right)$$

We can define a random variable X such that $X = \tau$ if the algorithm converges after exactly τ steps. Since δ^x is a convex function we have that $\mathbb{E}(\delta^x) \geq \delta^{\mathbb{E}(x)}$, therefore:

$$\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)] \geq \frac{\zeta(1-\delta^{\mathbb{E}(X)K}) + \delta^{\mathbb{E}(X)}}{1-\delta^K} \tag{6.6}$$

By dividing (6.6) by (6.5) we get:

$$\frac{\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)]}{\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^*, \delta)]} \geq \zeta(1-\delta^{\mathbb{E}(X)K}) + \delta^{\mathbb{E}(X)} \tag{6.7}$$

$\mathbb{E}(X)$ does not depend on δ. Moreover, $\delta^{\mathbb{E}(X)}$ is continuous in δ, monotonous, and $\lim_{\delta \to 1^-} \delta^{\mathbb{E}(X)} = 1$. Thus, we can take the limit of (6.7) as $\delta \to 1^-$, which equals to $\lim_{\delta \to 1^-} \frac{\mathbb{E}[U_n(\sigma_n^p, \sigma_{-n}^p, \delta)]}{\mathbb{E}[U_n(\sigma_n^*, \sigma_{-n}^*, \delta)]} = 1$ □

In order to guarantee rationality, the discount factor δ must be close to 1 since, as δ gets closer to 1, the agents do not care whether they access now or in some future round. Since the proposed monitoring scheme guarantees that every agent will access a resource for his corresponding context value, when $\delta \to 1$, the expected payoff for agents who are accessing a resource and for those who have not accessed a resource yet will be the same. In other words, the cost (overhead) of learning the correlated equilibrium decreases.

6.3.6 Indifference Period

In many real world applications, agents are indifferent in claiming a resource in a period of T_{ind} rounds, i.e., $\delta_t = 1, \forall t \leq T_{ind}$. For example, data of wireless transmitting devices might remain relevant for a specific time-window, during which the agent is indifferent of transmitting. In such cases, we can use the Markov bound to prove that with high probability the proposed algorithm will converge in under T_{ind} time-steps, thus satisfying the rationality constraint. We assume the agents are willing to accept linear 'delay' with regard to the number of resources R, the number of agents N, and the size of the context space K, specifically:

$$T_{ind} = \mathcal{O}(RNK) \tag{6.8}$$

Theorem 9. Under a linear indifference period T_{ind} (i.e., $\delta_t = 1, \forall t \leq T_{ind} = \mathcal{O}(RNK)$)

the probability of the system of agents following Algorithm 6 not having converged during T_{ind} diminishes as the congestion increases ($\frac{N}{R} \to \infty$).

Proof. Using the Markov bound, it follows that the probability that the system takes more than the accepted number of steps (T_{ind}) to converge is:

$$Pr[\neg\text{converge after } T_{ind}] = \mathcal{O}\left(\frac{\left(\log\left\lceil\frac{N}{R}\right\rceil + 1\right)(\log N + R)}{N}\right)$$

Taking the limit: $\lim_{\frac{N}{R}\to\infty} Pr[\neg\text{converge after } T_{ind}] = 0.$ □

Even though the Markov's inequality generally does not give very good bounds when used directly, Theorem 9 proves that our algorithm converges in the required time with high probability. The latter holds under high congestion, which constitutes the more interesting scenario since for small number of agents the required time to converge is only a few hundred time-steps. Moreover, the higher the indifference period, the higher the probability to converge under such time-constraint. For quasilinear indifference period for example, the system converges in the required time-window with high probability even for a small number of agents. We can further strengthen our rationality hypothesis by using a tighter bound (e.g., Chebyshev's inequality), albeit computing the theoretical variance of the convergence time is an arduous task, thus it remains open for future work.

6.4 Evaluation

In this section we model the resource allocation problem as a multi-armed bandit problem and provide simulation results of CA^{3}NONY's performance in comparison to state-of-the-art, well established bandit algorithms, namely the EXP4 (Auer et al., 2002b), EXP4.P (Beygelzimer et al., 2011), and EXP3 (Auer et al., 2002b). In every case we report the average value over 128 runs of the same simulation. For the EXP family of algorithms, the input parameters are set to their optimal values, as prescribed in the aforementioned publications. We assume a reward of 1 for a successful access, -1 if there is a collision, and 0 if the agent yielded.

6.4.1 Level of Courtesy

We evaluated different back-off probabilities ($p_{backoff} \in \{0.1, 0.25, 2-\sqrt{2}, 0.75, 0.9\}$) for $R = K \in \{2, 4, 8, 16\}$ and $N = R \times K$. There was no significant difference in convergence time, but since $p_{backoff}$ is directly correlated with the number of collisions, it can have significant impact on the average payoff (see Table 6.1). It is important to note, though, that agents do not have global nor local knowledge of the level of congestion of the

Table 6.1: Average payoff depending on the level of courtesy (back-off probability), $K = R, N = R \times K$.

$p_{backoff}$	$R = 2$	$R = 4$	$R = 8$	$R = 16$
0.1	37.6	-4.7	-39.5	-63.1
0.25	43.6	10.0	-16.9	-40.0
$2 - \sqrt{2}$	45.7	15.6	-5.2	-23.0
0.75	45.5	15.7	-3.7	-20.6
0.9	44.2	12.5	-5.8	-19.4

system (they only receive binary occupancy feedback for one resource per time-step). Thus, agents can not select the optimal back-off probability depending on congestion. In what follows, the back-off probability of CA^{3}NONY is set to $p_{backoff} = 2 - \sqrt{2}$ of Equation 6.4, which is only optimal under high congestion settings (i.e., $\frac{N}{R} = K \to \infty$), yet constitutes a safe option. Note that the provided theoretical analysis holds for any constant back-off probability.

6.4.2 Bandits and Monitoring

CA^{3}NONY has three meta actions (Access, Yield, and Monitor $\{A, Y, M\}$), while bandit algorithms have only two ($\{A, Y\}$), and CA^{3}NONY assumes the existence of monitoring authorities (MAs). For fairness' sake in the reported results, we made the following two modifications. First, we include a variation of CA^{3}NONY, denoted as CA^{3}NONY*, where we assume agents incur a cost (equal to collision cost ζ) every time they monitor a resource (reflected in Table 6.3). Second, all the employed bandit algorithms make use of the MAs, which indirectly grants them the the ability to monitor resources as well. More specifically, the accumulated payoff is updated only if they are allowed to access a resource. If not, we consider it a monitoring action, which means the agents still receive occupancy feedback. The latter is important, otherwise the bandit algorithms would require significantly longer time to converge. Note that, contrary to CA^{3}NONY*, monitoring is free for bandit algorithms with respect to the accumulated payoff, i.e., they do not incur a collision cost.

6.4.3 Employed Bandit Algorithms

In our setting, the reward of each arm does not follow a fixed probability distribution (adversarial setting). Moreover, the agents are able to observe side-information (context) at each time-step t. The arm that yields the highest expected reward can be different depending on the context. Hence we focus on adversarial contextual bandit algorithms (see (Zhou, 2015) for a survey). A typical approach is to use expert advice. In this method we assume a set of experts $\mathcal{M} = \{1, \ldots, M\}$ who generate a probability distribu-

tion on which arm to pull depending on the context. A no-regret algorithm performs asymptotically as well as the best expert. Such algorithms are the EXP4 and EXP4.P, the difference being that EXP4 can exhibit high variance (Zhou, 2015), while EXP4.P achieves the same regret with high probability by combining the confidence bounds of UCB1 (Auer et al., 2002a) and EXP4. The computational complexity and memory requirements of the above algorithms are linear in M, making them intractable for large number of experts. In order to deal with the increased complexity in larger simulations, we gave an edge to these algorithms by restricting the set of experts $\mathcal{M}$ to the same uniform correlated equilibria (CE) that CA^{3}NONY converges to. The latter enabled us to perform larger simulations (the unrestricted versions could not handle $N \geq 64$), while resulting in the same order of magnitude convergence time, slightly higher average payoff, and significantly higher fairness ($> 35\%$ since by design the experts proposed fair CE). Alternatively, we can use non-contextual adversarial bandit algorithms, such as the EXP3. Moreover, we can convert EXP3 to a contextual algorithm by setting up a separate instance for all $k \in \mathcal{K}$. We call this 'CEXP3'. This results in a contextual bandit algorithm which has the edge over EXP4 from an implementation viewpoint since its running time at each time-step is $\mathcal{O}(R)$ and its memory requirement is $\mathcal{O}(KR)$ (for reference, CA^{3}NONY's is $\mathcal{O}(1)$ and $\mathcal{O}(K)$, respectively).

6.4.4 Convergence Speed and Efficiency

We know that CA^{3}NONY converges to a CE which is efficient. If all agents follow the courteous convention of Algorithm 6, the system converges to a state where no resources remain un-utilized and there are no collisions (Theorem 13 of (Cigler and Faltings, 2013)). Furthermore, Theorem 6 argues for fast convergence. The former are both corroborated by Figure 6.1 (similar results ($> \times 10^2$ faster convergence) were acquired for $R \in \{2, 8, 16\}$ as well.). Figure 6.1 depicts the total utilization of resources for a simulation period of $T = 10^6$ time-steps. Note that the x-axis is in logarithmic scale. CA^{3}NONY converges significantly ($> \times 10^2$) faster than the bandit algorithms to a state of 100% efficiency. On the other hand, the bandit algorithms exhibit high variance, never achieve 100% efficiency, and are not able to handle efficiently the increase in context space size and number of resources.

6.4.5 Fairness

The usual predicament of efficient equilibria for allocation games is that they assign the resources only to a fixed subset of agents, which leads to an unfair result (e.g., an efficient pure NE (PNE) is for R agents to access and $N - R$ agents to yield). This is not the case for CA^{3}NONY, which converges to an equilibrium that is not just efficient but fair as well. Due to the enforced monitoring scheme, all agents acquire the same amount of resources. As a measure of fairness, we use the Jain index (Jain et al., 1998). For an

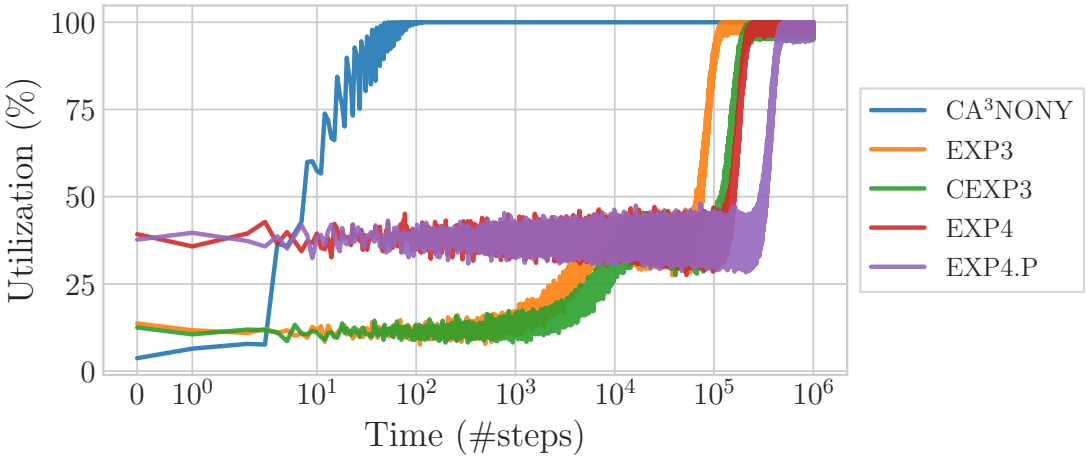


Figure 6.1: Resource utilization for $R = 4, N = 16$ (x-axis is in logarithmic scale).

allocation game of N agents, such that the n^{th} agent receives an allocation of x_n, the Jain index is given by $\mathbb{J}(\mathbf{x}) = |\sum_{n\in\mathcal{N}} x_n|^2 \,/\, N \sum_{n\in\mathcal{N}} x_n^2$ (see Section 2.5). An allocation is considered fair, iff $\mathbb{J}(\mathbf{x}) = 1$, $\mathbf{x} = (x_1, \ldots, x_N)^\top$.

Table 6.2 presents the expected Jain Index of the evaluated algorithms at the end of the time horizon T. CA^{3}NONY converges to a fair equilibrium, achieving a Jain index of 1. The EXP4 and EXP4.P were the fairest amongst the bandit algorithms, achieving a Jain index of close to 1. This is to be expected since the set of experts $\mathcal{M}$ is limited to the same set of equilibria that CA^{3}NONY converges to. On the other hand, EXP3 and CEXP3 performed considerably worse, with the EXP3 exhibiting the worst performance in terms of fairness, equal to a PNE: $\mathbb{J}_{PNE}(\mathbf{x}) = \frac{R^2}{NR} = \frac{1}{K} = \mathbb{J}_{EXP3}(\mathbf{x})$.

6.4.6 Average Payoff

The average payoff corresponds to the total discounted payoff an agent would receive in the time horizon T. Table 6.3 presents the average payoff for the studied algorithms. The clustered pairs of bandit algorithms exhibited $< 1\%$ difference, hence we included the average value of the pair. The discount factor was set to $\delta = 0.99$. At the top half we do not assume any indifference period, while at the bottom half we assume linear indifference period ($\delta_t = 1, \forall t \leq T_{ind}$, where T_{ind} is given by Equation 6.8). Recall that CA^{3}NONY and the bandit algorithms do not incur any cost for monitoring resources, while CA^{3}NONY* incurs a cost equal to the collision cost ζ.

Once more, CA^{3}NONY (and even CA^{3}NONY*) significantly outperforms all the bandit algorithms. The latter have relatively similar performance, with EXP4 and EXP4.P being the best amongst them. It is worth noting that adding an indifference period has a dramatic effect on the results. Comparing CA^{3}NONY to bandit algorithms we observe

Table 6.2: Fairness (Jain Index), $K = R, N = R \times K$.

	$R = 2$	$R = 4$	$R = 8$	$R = 16$
CA^{3}NONY	1.0000	1.0000	1.0000	1.0000
EXP3	0.5000	0.2500	0.1250	0.0625
CEXP3	0.7018	0.5865	0.5341	0.9638
EXP4	1.0000	0.9999	0.9959	0.9198
EXP4.P	1.0000	0.9999	0.9798	0.7503

Table 6.3: Average Payoff, $K = R, N = R \times K$.

	$R = 2$	$R = 4$	$R = 8$	$R = 16$
CA^{3}NONY	45.4	15.7	-5.3	-23.0
CA^{3}NONY*	20.0	-22.4	-50.6	-72.6
(C)EXP3	-72.4	-93.1	-99.8	-100.0
EXP4(.P)	-59.0	-81.0	-90.7	-95.4
CA^{3}NONY	53.3	79.0	502.5	4057.5
CA^{3}NONY*	23.8	-55.5	-1337.1	-26723.9
(C)EXP3	-84.0	-331.0	-4175.7	-60595.1
EXP4(.P)	-68.4	-288.4	-3807.1	-62631.5

that CA^{3}NONY achieves a large increase on average payoff, while the opposite happens for the bandit algorithms. This is because the learning rule of Algorithm 6 prohibits from accessing an already claimed resource, thus minimizing collisions. On the contrary, bandit algorithms constantly explore (they assign a positive probability mass to every arm) which leads to collisions. In a multi-agent system where every agent learns this can have a cascading effect. The latter becomes apparent when fixing $\delta = 1$ for T_{ind} steps. The collision cost remains high for longer which, as seen by Table 6.3, has a significant impact on the bandit algorithms' performance.

6.4.7 Large Scale Systems

The innovation of CA^{3}NONY stems from the adoption of a simple convention, which allows its applicability to large scale multi-agent systems. To evaluate the latter, Figures 6.2 and 6.3 depict the convergence time for increasing number of resources R, and increasing system congestion ($\frac{N}{R} = K$) respectively. Both graphs are in a double logarithmic scale, and the error bars represent one standard deviation of uncertainty. The total number of agents is given by $N = R \times K$. The largest simulations involve 16384 agents. Along with CA^{3}NONY, we depict the fastest (based on the previous simulations) of the bandit algorithms, namely the EXP3. In both cases we acquire $\times 10^3 - \times 10^5$ faster convergence. The above validate CA^{3}NONY's performance in both scenarios with abundance ($N \approx R$ or small K), and scarcity of resources ($N \gg R$ or large K). As depicted, CA^{3}NONY

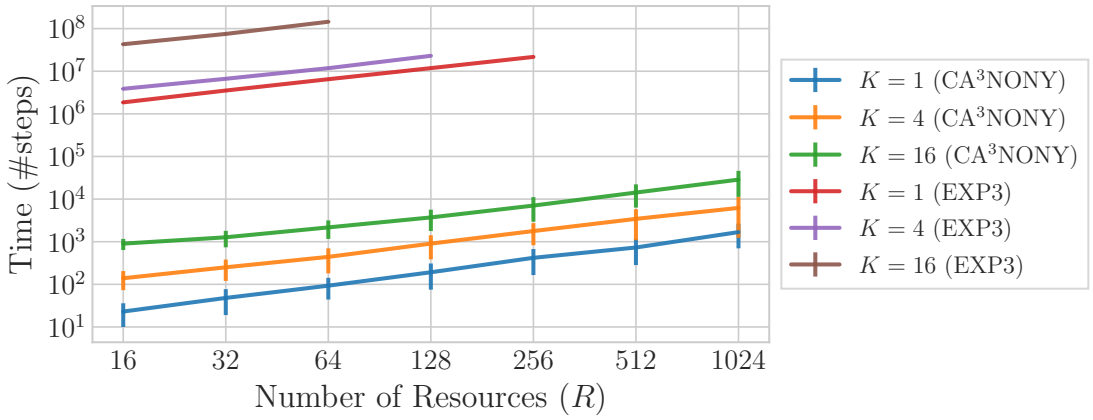


Figure 6.2: Convergence time for increasing number of resources R, varying context space size K, and $N = R \times K$ (double logarithmic scale).

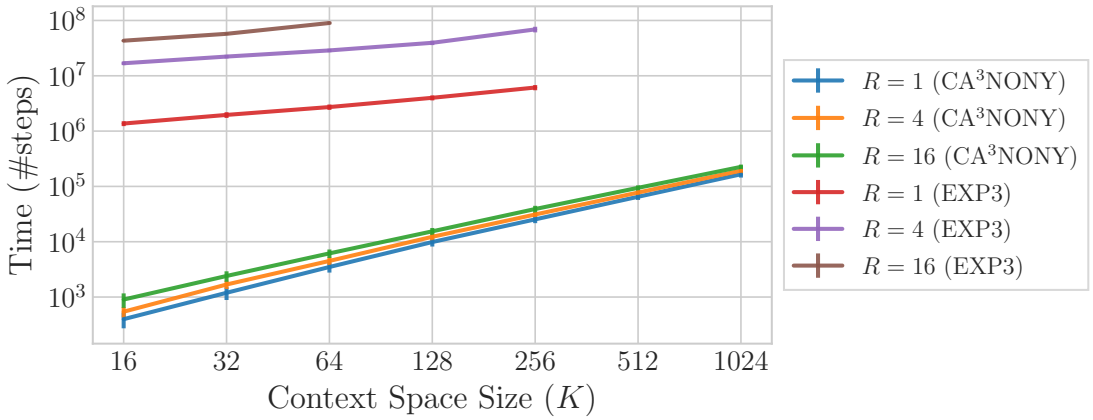


Figure 6.3: Convergence time for increasing congestion ($\frac{N}{R} = K$), varying number of resources R, and $N = R \times K$ (double logarithmic scale).

is significantly faster than the EXP3 and can gracefully handle increasing number of resources, and high congestion. Finally note that, in several of the simulations, EXP3 was unable to reach its convergence goal of 90% efficiency (utilization of resources) in a reasonable amount of computation time (1.5×10^8 time-steps), hence the resulting gaps in EXP3's lines in Figures 6.2 and 6.3. Especially in situations with scarcity of resources the utilization was significantly lower.

6.5 Chapter Conclusion

In this chapter we propose CA^{3}NONY, an anti-coordination framework under rationality constraints. It is based on a simple, human-inspired convention of courtesy which prescribes a positive back-off probability in case of a collision. Coupled with a monitoring

scheme which deters the monopolization of resources, we proved that the induced strategy constitutes an ϵ-subgame-perfect equilibrium. We compared CA^{3}NONY to state-of-the-art bandit algorithms, namely EXP3, EXP4, and EXP4.P. Simulation results demonstrated that CA^{3}NONY outperforms these algorithms by achieving more than two orders of magnitude faster convergence, a fair allocation, higher average payoff, and can gracefully handle increasing number of resources, and high congestion. The aforementioned gains suggest that human-inspired conventions may prove beneficial in other ad-hoc coordination scenarios as well.

Part II

Resource Sharing for Deep Reinforcement Learning Agents

7 Preliminaries

7.1 Multi-Agent Deep Reinforcement Learning

In the following chapters, we consider a *decentralized* multi-agent reinforcement learning scenario in a partially observable general-sum Markov game (e.g., (Shapley, 1953; Littman, 1994)). At each time-step, agents take actions based on a partial observation of the state space, and receive an individual reward. Each agent learns a policy independently. More formally, let $\mathcal{N} = \{1, \ldots, N\}$ denote the set of agents as before, and $\mathcal{M}$ be an N-player, partially observable Markov game defined on a set of states $\mathcal{S}$. An observation function $\mathcal{O}^n : \mathcal{S} \rightarrow \mathbb{R}^d$ specifies agent n's d-dimensional view of the state space. Let $\mathcal{A}^n$ denote the set of actions for agent $n \in \mathcal{N}$, and $\boldsymbol{a} = \times_{\forall n \in \mathcal{N}} a^n$, where $a^n \in \mathcal{A}^n$, the joint action. The states change according to a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the set of discrete probability distributions over $\mathcal{S}$. Every agent n receives an individual reward based on the current state $\sigma_t \in \mathcal{S}$ and joint action $\boldsymbol{a}_t$. The latter is given by the reward function $r^n : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$. Finally, each agent learns a policy $\pi^n : \mathcal{O}^n \rightarrow \Delta(\mathcal{A}^n)$ independently through their own experience of the environment (observations and rewards). Let $\boldsymbol{\pi} = \times_{\forall n \in \mathcal{N}} \pi^n$ denote the joint policy. The goal for each agent is to maximize the long term discounted payoff, as given by $V^n_{\boldsymbol{\pi}}(\sigma_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^n(\sigma_t, \boldsymbol{a}_t) | \boldsymbol{a}_t \sim \boldsymbol{\pi}_t, \sigma_{t+1} \sim \mathcal{T}(\sigma_t, \boldsymbol{a}_t)\right]$, where γ is the discount factor and σ_0 is the initial state.

8 Exploiting Environmental Signals to Enable Policy Correlation

8.1 Preface

8.1.1 Contribution and Sources

This chapter is largely based on (Danassis et al., 2021a, 2022a). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions (for both the aforementioned papers) are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable):

PD (author):	Conceptualization, Methodology (lead), Formal analysis, Investigation, Writing – Original Draft, Writing – Review & Editing
Zeki Doruk Erden:	Methodology (supporting), Software, Formal Analysis, Investigation, Writing – Review & Editing
Boi Faltings:	Writing – Review & Editing (supporting), Supervision

8.1.2 Chapter Summary

Human societies manage to successfully self-organize and resolve the tragedy of the commons in common-pool resources, in spite of the bleak prediction of non-cooperative game theory. Adding to the challenge, real-world problems are inherently large-scale and of low observability. One key concept that facilitates human coordination in such settings is the use of conventions. Inspired by human behavior, we investigate the learning dynamics and emergence of temporal conventions, focusing on common-pool resources.

Uncoupled policies and low observability make cooperation hard to achieve; as the number of agents grow, the probability of taking a correct gradient direction decreases exponentially. By introducing an *arbitrary common signal* (e.g., date, time, or any periodic set of numbers) as a means to couple the learning process, we show that temporal conventions can emerge and agents reach *sustainable* harvesting strategies. The introduction of the signal consistently improves the social welfare (by 258% on average, up to 3306%), the range of environmental parameters where sustainability can be achieved (by 46% on average, up to 300%), and the convergence speed in low abundance settings (by 13% on average, up to 53%).

Extra emphasis was given in designing a *realistic evaluation setting*: (a) environment dynamics are modeled on real-world fisheries, (b) we assume decentralized learning, where agents can observe only their own history, and (c) we run large-scale simulations (up to 64 agents).

8.2 Introduction

The question of *cooperation* in socio-ecological systems and *sustainability* in the use of common-pool resources constitutes a critical open problem. Classical non-cooperative game theory suggests that rational individuals will exhaust a common resource, rather than sustain it for the benefit of the group, resulting in the 'the tragedy of the commons' (Hardin, 1968). The tragedy of the commons arises when it is challenging and/or costly to exclude individuals from appropriating common-pool resources (CPR) of finite yield (Ostrom et al., 1994). Individuals face strong incentives to appropriate, which results in *overuse* and even *permanent depletion* of the resource. Examples include the degradation of fresh water resources, the over-harvesting of timber, the depletion of grazing pastures, the destruction of fisheries, etc.

In spite of the bleak prediction of non-cooperative game theory, the tragedy of the commons is not inevitable, though conditions under which cooperation and sustainability can be achieved may be more demanding, the higher the stakes. Nevertheless, humans have been systematically shown to successfully self-organize and resolve the tragedy of the commons in CPR appropriation problems, even without the imposition of an extrinsic incentive structure (Ostrom, 1999). E.g., by enabling the capacity to communicate, individuals have been shown to maintain the harvest to an optimal level (Ostrom et al., 1994; Casari and Plott, 2003). Though, communication creates overhead, and might not always be possible (due to for example partial observability, different communication protocols, existence of legacy agents, etc.) (Stone et al., 2010). One of the key findings of empirical field research on sustainable CPR regimes around the world is the employment of *boundary rules*, which prescribe who is authorized to appropriate from a resource (Ostrom, 1999). Such boundary rules can be of temporal nature, prescribing the *temporal order* in which people harvest from a common-pool resource (e.g., 'protocol of play' (Budescu et al., 1997)). The aforementioned rules can be enforced by an authority, or emerge in a self-organized manner (e.g., by utilizing environmental signals such as the time, date, season, etc.) in the form of a *social convention*.

Many real-world CPR problems are inherently *large-scale* and *partially observable*, which further increases the challenge of sustainability. In this work we deal with the *most information-restrictive setting*: each participant is modeled as an individual agent with its own policy conditioned only on *local information*, specifically its own history of action/reward pairs (*fully decentralized* method). Global observations, including the resource stock, the number of participants, and the joint observations and actions, are hidden – as is the case in many real-world applications, like commercial fisheries. Under such a setting, it is *impossible to avoid positive probability mass on undesirable actions* (i.e., simultaneous appropriation), since there is no correlation between the agents' policies. This leads to either low social welfare or, even worse, the *depletion* of the resource. Depletion becomes more likely as the problem size grows due to the *non-stationarity* of

the environment and the *global exploration problem.*[1]

We propose a simple technique: allow agents to observe an *arbitrary, common signal* from the environment. Observing a common signal mitigates the aforementioned problems because it allows for *coupling* between the learned policies, increasing the joint policy space. Agents, for example, can now learn to harvest in turns, and with varying efforts per signal value, or allow for fallow periods. The benefit is twofold: the agents learn to not only avoid depletion, but also to maintain a healthy stock which allows for large harvest and, thus, higher social welfare. It is important to stress that *we do not assume any a priori relation between the signal space and the problem at hand.* Moreover, we require no communication, no extrinsic incentive mechanism, and we do not change the underlying architecture, or learning algorithm. We simply utilize a means – common environmental signals that are *amply available to the agents* (Hart and Mas-Colell, 2000) – to accommodate correlation between policies. This in turn enables the emergence of *ordering conventions* of temporal nature (henceforth referred to as temporal conventions) and *sustainable harvesting* strategies.

8.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We are the first to introduce a realistic common-pool resource appropriation game for multi-agent coordination**, based on bio-economic models of commercial fisheries, and provide theoretical analysis on the dynamics of the environment. Specifically, we prove the optimal harvesting strategy that maximizes the revenue, and identify two interesting stock values: the 'limit of sustainable harvesting', and the 'limit of immediate depletion'.

2. **We propose a simple and novel technique: allow agents to observe an arbitrary periodic environmental signal.** Such signals are *amply available* in the environment (e.g., time, date etc.) and can *foster cooperation* among agents.

3. **We provide a thorough (quantitative & qualitative) analysis** on the learned policies and demonstrate significant improvements on sustainability, social welfare, and convergence speed, and the emergence of temporal harvesting conventions.

[1]The former arises due to simultaneous learning by all agents, which results to a moving-target problem (the best policy changes as the other agents' policies change). The latter refers to the probability that at least one agent explores (i.e., not acting according to the optimal policy) which increases with the number of agents (Busoniu et al., 2008; Matignon et al., 2012; Hernandez-Leal et al., 2019).

8.2.2 Discussion and Related Work

As autonomous agents proliferate, they will be called upon to interact in ever more complex environments. This will bring forth the need for techniques that enable the emergence of sustainable cooperation. Despite the growing interest in and success of multi-agent deep reinforcement learning (MADRL), scaling to environments with a large number of learning agents continues to be a problem (Ganapathi Subramanian et al., 2020). A multi-agent setting is inherently susceptible to many pitfalls: non-stationarity (moving-target problem), curse of dimensionality, credit assignment, global exploration, relative overgeneralization (Hernandez-Leal et al., 2019; Matignon et al., 2012; Wiegand and Jong, 2004).[2] Recent advances in the field of MADRL deal with only a limited number of agents. It is shown that as the number of agents increase, the probability of taking a correct gradient direction decreases exponentially (Hernandez-Leal et al., 2019), thus the proposed methods cannot be easily generalized to complex scenarios with many agents.

Our approach aims to mitigate the aforementioned problems of MADRL by introducing *coupling* between the learned policies. It is important to note that the proposed approach does not change the underlying architecture of the network (the capacity of the network stays the same), nor the learning algorithm or the reward structure. We simply augment the input space by allowing the observation of an arbitrary common signal. The signal has no a priori relation to the problem, i.e., *we do not need to design an additional feature*; in fact *we use a periodic sequence of arbitrary integers*. It is still possible for the original network (without the signal) to learn a sustainable strategy. Nevertheless, we show that the simple act of augmenting the input space drastically increases the social welfare, speed of convergence, and the range of environmental parameters in which sustainability can be achieved. Most importantly, the proposed approach requires no communication, creates no additional overhead, it is simple to implement, and scalable.

The proposed technique was inspired by temporal conventions in resource allocation games of non-cooperative game theory. The closest analogue is the courtesy convention of the CA^{3}NONY framework (Chapter 6), where rational agents learn to coordinate their actions to access a set of indivisible resources by observing a signal from the environment. Closely related is the concept of the correlated equilibrium (CE) (Aumann, 1974; Nisan et al., 2007), which, from a practical perspective, constitutes perhaps the most relevant non-cooperative solution concept (Hart and Mas-Colell, 2000).[3] Most importantly, it

[2]Some of these adversities can be mitigated by the centralized training, decentralized execution paradigm. Yet, centralized methods likewise suffer from a plethora of other problems: they are computationally heavy, assume unlimited communication (which is impractical in many real-world applications), the exact same team has to be deployed (in the real-world we cooperate with strangers), and, most importantly, the size of the joint action space grows exponentially with the number of agents.

[3]Correlated equilibria also relate to boundary rules and temporal conventions in human societies; the most prominent example of a correlated equilibrium in real life is the traffic lights, which can also be viewed as a temporal convention for the use of the road.

is possible to achieve a correlated equilibrium without a central authority, simply by utilizing meaningless environmental signals (Danassis and Faltings, 2019; Cigler and Faltings, 2013; Borowski et al., 2014). Such common environmental signals are *amply available to the agents* (Hart and Mas-Colell, 2000). The aforementioned line of research studies pre-determined strategies of rational agents. Instead, we study the emergent behaviors of a group of independent learning agents aiming to maximize the long term discounted reward.

The importance of correlation has also been recognized in the Dec-POMDP community (e.g., see (Bernstein et al., 2005), which is also closely related to and inspired by the concept of correlated equilibria in game theory). Contrary to that work, we do not require a correlation device that learns, we have a non-cooperative setting, where each agent has a separate reward function that it optimizes independently, and we use deep reinforcement learning agents for the policy approximation, which makes our approach more widely applicable.

A second source of inspiration is behavioral conventions; one of the key concepts that facilitates human coordination.[4] A convention is defined as a customary, expected, and self-enforcing behavioral pattern (Young, 1996; Lewis, 2008). It can be considered as a behavioral rule, designed and agreed upon ahead of time (Shoham and Tennenholtz, 1995; Walker and Wooldridge, 1995), or it may emerge from within the system itself (Mihaylov et al., 2014; Walker and Wooldridge, 1995). The examined temporal convention in this work falls in the latter category.

Moving on to the application domain, there has been great interest recently in CPR problems (and more generally, social dilemmas (Kollock, 1998)) as an application domain for MADRL (Perolat et al., 2017; Leibo et al., 2017; Peysakhovich and Lerer, 2018a; Hughes et al., 2018; Peysakhovich and Lerer, 2018b; Jaques et al., 2019; Wang et al., 2019; Lupu and Precup, 2020; Koster et al., 2020). CPR problems offer complex environment dynamics and relate to real-world socio-ecological systems. There are a few distinct differences between the CPR models presented in the aforementioned works and the model introduced in this work: First and foremost, we designed our model to *resemble reality as closely as possible using bio-economic models of commercial fisheries* (Clark, 2006; Diekert, 2012), resulting in complex environment dynamics. Second, we have a *continuous action space* which further complicates the learning process. Finally, we opted not to learn from visual input (raw pixels). The problem of direct policy approximation from visual input does not add complexity to the social dilemma itself; it only adds complexity in the feature extraction of the state. It requires large networks because of the additional complexity of extracting features from pixels, while only a small part of what is learned is the actual policy (Cuccu et al., 2019). Most importantly, it makes it

[4]Humans are able to routinely and robustly cooperate in their every day lives in large-scale and under dynamic and unpredictable demand. They also have access to auxiliary information that help correlated their actions (e.g., time, date etc.).

harder to study the policy in isolation, as we do in this work. Moreover, from a practical perspective, learning from a visual input would be meaningless, given that we are dealing with a low observability scenario where the resource stock and the number and actions of the participants are hidden.

In terms of the methodology for dealing with the tragedy of the commons, the majority of the aforementioned literature falls broadly into two categories: Reward shaping (Hughes et al., 2018; Jaques et al., 2019; Peysakhovich and Lerer, 2018b), which refers to adding a term to the extrinsic reward an agent receives from the environment, and opponent shaping (Koster et al., 2020; Lupu and Precup, 2020; Perolat et al., 2017), which refers to manipulating the opponent (by e.g., sharing rewards, punishments, or adapting your own actions). Contrary to that, we only allow agents to observe an *existing* environmental signal. *We do not modify the intrinsic or extrinsic rewards, design new features, or require a communication network.* Finally, boundary rules emerged in (Perolat et al., 2017) as well in the form of spatial territories. Such territories can increase inequality, while we maintain high levels of fairness.

8.3 The Common Fishery Model

In order to better understand the impact of self-interested appropriation, it would be beneficial to examine the dynamics of *real-world* common-pool renewable resources. To that end, we present an abstracted bio-economic model for commercial fisheries (Clark, 2006; Diekert, 2012). The model describes the dynamics of the stock of a common-pool renewable resource, as a group of appropriators harvest over time. The harvest depends on (i) the effort exerted by the agents and (ii) the ease of harvesting a resource at that point of time, which depends on the stock level. The stock replenishes over time with a rate dependent on the current stock level.[5]

More formally, let $\mathcal{N}$ denote the set of appropriators, $\epsilon_{n,t} \in [0, \mathcal{E}_{max}]$ the effort exerted by agent n at time-step t, and $E_t = \sum_{n \in \mathcal{N}} \epsilon_{n,t}$ the total effort at time-step t. The total harvest is given by Equation 8.1, where $s_t \in [0, \infty)$ denotes the stock level (i.e., amount of resources) at time-step t, $q(\cdot)$ denotes the catchability coefficient (Equation 8.2), and S_{eq} is the equilibrium stock of the resource.

$$H(E_t, s_t) = \begin{cases} q(s_t)E_t & \text{, if } q(s_t)E_t \leq s_t \\ s_t & \text{, otherwise} \end{cases} \tag{8.1}$$

$$q(x) = \begin{cases} \frac{x}{2S_{eq}} & \text{, if } x \leq 2S_{eq} \\ 1 & \text{, otherwise} \end{cases} \tag{8.2}$$

[5] In Chapter 9, we extended our model to account for multiple resources, and harvesters with varying skill levels.

Each environment can only sustain a finite amount of stock. If left unharvested, the stock will stabilize at S_{eq}. Note also that $q(\cdot)$, and therefore $H(\cdot)$, are proportional to the current stock, i.e., the higher the stock, the larger the harvest for the same total effort. The stock dynamics are governed by Equation 8.3, where $F(\cdot)$ is the spawner-recruit function (Equation 8.4) which governs the natural growth of the resource, and r is the growth rate. To avoid highly skewed growth models and unstable environments ('behavioral sink' (Calhoun, 1962, 1973)), the growth rate should be $r \in [-W(-1/(2e)), -W_{-1}(-1/(2e))] \approx [0.232, 2.678]$, where $W_k(\cdot)$ is the Lambert W function (see Section 8.3.1).

$$s_{t+1} = F(s_t - H(E_t, s_t)) \tag{8.3}$$

$$F(x) = xe^{r(1-\frac{x}{S_{eq}})} \tag{8.4}$$

We assume that the individual harvest is proportional to the exerted effort (Equation 8.5), and the revenue of each appropriator is given by Equation 8.6, where p_t is the price (\$ per unit of resource), and c_t is the cost (\$) of harvesting (e.g., operational cost, taxes, etc.). Here lies the 'tragedy': the benefits from harvesting are private ($p_t h_{n,t}(\epsilon_{n,t}, s_t)$), but the loss is borne by all (in terms of a reduced stock, see Equation 8.3).

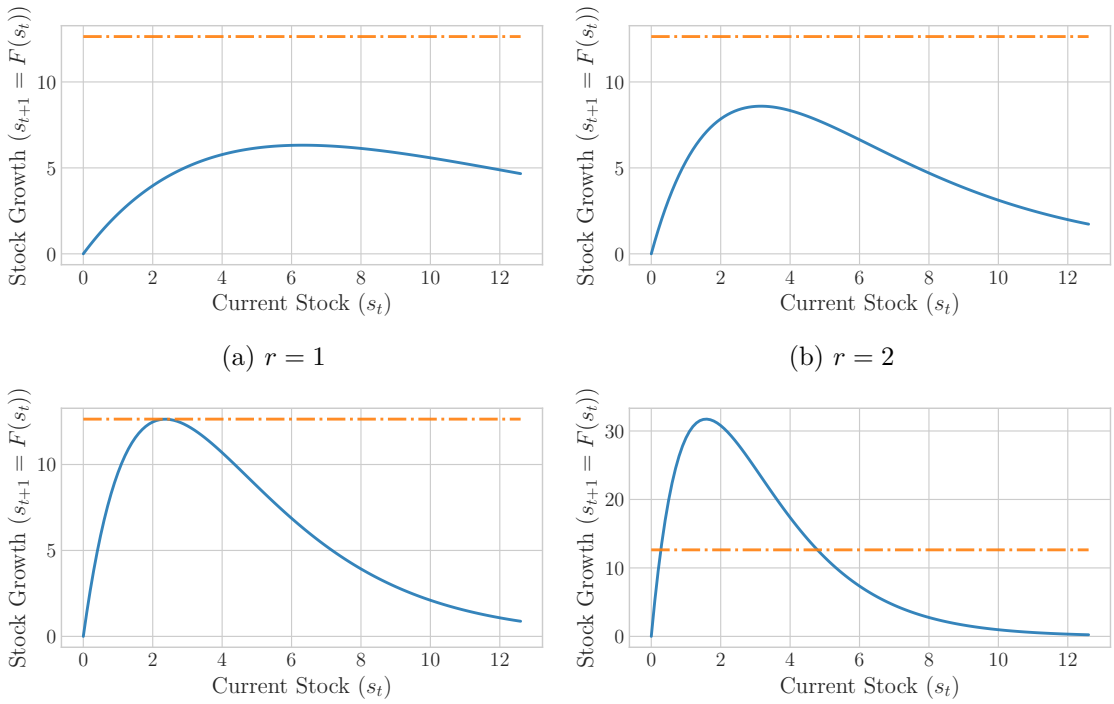
$$h_{n,t}(\epsilon_{n,t}, s_t) = \frac{\epsilon_{n,t}}{E_t} H(E_t, s_t) \tag{8.5}$$

$$u_{n,t}(\epsilon_{n,t}, s_t) = p_t h_{n,t}(\epsilon_{n,t}, s_t) - c_t \tag{8.6}$$

8.3.1 Growth Rate

The growth rate, r, plays a significant role in the stability of the stock dynamics. A high growth rate – and subsequently high population density – can even lead to the extinction of the population due to the collapse in behavior from overcrowding (a phenomenon known as 'behavioral sink' (Calhoun, 1962, 1973)). This is reflected by the spawner-recruit function (Equation 8.4) in our model. As depicted in Figure 8.1, the higher the growth rate, r, the more skewed the stock dynamics. Specifically, Figure 8.1 plots the spawner-recruit function $F(\cdot)$ for various growth rates: $r = 1$, 2, $-W_{-1}(-\frac{1}{2e}) \approx 2.678$, and 4 (Figure 8.1a, 8.1b, 8.1c, and 8.1d, respectively). The x-axis denotes the current stock level (s_t), while the y-axis depicts the stock level on the next time-step, assuming no harvest (i.e., $s_{t+1} = F(s_t)$). The dashed line indicates a stock level equal to $2S_{eq}$. For a growth rate of $r = 4$ for example (Figure 8.1d), we have a highly skewed growth model that can lead to the depletion of the resource (high stock values (s_t) result to[6]

[6]In practice, δ represents the minimum resource 'unit', and is enforced by the granularity of the resource.



(a) $r = 1$

(b) $r = 2$

(c) $r = -W_{-1}(-\frac{1}{2e}) \approx 2.678$

(d) $r = 4$

Figure 8.1: Plot of the spawner-recruit function $F(\cdot)$ for various growth rates: $r = 1$, 2, $-W_{-1}(-\frac{1}{2e}) \approx 2.678$, and 4 (Figure 8.1a, 8.1b, 8.1c, and 8.1d, respectively). The x-axis denotes the current stock level (s_t), while the y-axis depicts the stock level on the next time-step, assuming no harvest (i.e., $s_{t+1} = F(s_t)$). The dashed line indicates a stock level equal to $2S_{eq}$.

$s_{t+1} < \delta \to 0$, i.e., permanent depletion of the resource). For this reason, we want an unskewed growth model, specifically we want the stock to remain below two times the equilibrium stock point, i.e., $s_{t+1} \leq 2S_{eq}$.[7] For this reason, we need to bound the growth rate according to the following theorem:

Theorem 10. For a continuous resource governed by the dynamics of Section 8.3, the stock value does not exceed the limit of $2S_{eq}$, if $r \in [-W(-1/(2e)), -W_{-1}(-1/(2e))] \approx [0.232, 2.678]$, where $W_k(\cdot)$ is the Lambert W function.

Proof. Let $x \triangleq s_t \leq 2S_{eq}$ for a time-step t. We want $s_{t+1} \leq 2S_{eq}$, thus we need to bound the maximum value of the spawner-recruit function, $F(x)$ (Equation 8.4). Taking the derivative:

$$\frac{\partial}{\partial x} F(x) = e^{r\left(1-\frac{x}{S_{eq}}\right)} \left(1 - \frac{rx}{S_{eq}}\right)$$

[7] This limit is imposed by the chosen parameters of the model equations.

We have that:

$$\frac{\partial}{\partial x}F(x) = 0 \Rightarrow x = \frac{S_{eq}}{r}, r \neq 0 \text{ and } S_{eq} \neq 0$$

Thus the maximum value is:

$$F\left(\frac{S_{eq}}{r}\right) = \frac{S_{eq}}{r}e^{(r-1)}$$

We want to bound the maximum value:

$$F\left(\frac{S_{eq}}{r}\right) \leq 2S_{eq} \Rightarrow \frac{e^{(r-1)}}{r} \leq 2 \Rightarrow e^r - 2er \leq 0$$
$$\Rightarrow -W\left(-\frac{1}{2e}\right) \leq r \leq -W_{-1}\left(-\frac{1}{2e}\right)$$

where $W_k(\cdot)$ is the Lambert W function. $-W(-1/(2e)) \approx 0.232$ and $-W_{-1}(-1/(2e)) \approx 2.678$. □

8.3.2 Optimal Harvesting

The question that naturally arises is: what is the 'optimal' effort in order to harvest a yield that maximizes the revenue (Equation 8.6). We make two assumptions: First, we assume that the entire resource is owned by a single entity (e.g., a firm or the government), which possesses complete knowledge of and control over the resource. Thus, we only have a single control variable, E_t. This does not change the underlying problem since the total harvested resources are linear in the proportion of efforts put by individual agents (Equation 8.5). Second, we consider the case of zero discounting, i.e., future revenues are weighted equally with current ones. Of course firms (and individuals) do discount the future and bio-economic models should take that into account, but this complicates the analysis and it is out of the scope of this work. We argue we can still draw useful insight into the problem.

Our control problem consists of finding a piecewise continuous control E_t, so as to maximize the total revenue for a given episode duration T (optimization problem (8.7), where $U_t(E_t)$ is the cumulative revenue at time-step t, given by Equation 8.8). The maximization problem can be solved using Optimal Control Theory (Ding and Lenhart, 2010; Lenhart and Workman, 2007).

$$\max_{E_t} \quad \sum_{t=0}^{T} U_t(E_t) \tag{8.7}$$

$$s.t. \quad s_{t+1} = F(s_t - H(E_t, s_t))$$

$$U_t(E_t) = \sum_{n \in \mathcal{N}} u_{n,t}(\epsilon_{n,t}, s_t) = p_t H(E_t, s_t) - N c_t = p_t H(E_t, s_t) - C_t \tag{8.8}$$

The optimal[8] control is given by the following theorem:

Theorem 11. The optimal control variable E_t^* that solves the maximization problem of (8.7) given the model dynamics described in Section 8.3 is given by Equation 8.9, where λ_t are the adjoint variables of the Hamiltonians:

$$E_{t+1}^* = \begin{cases} E_{max}, & \text{if } (p_{t+1} - \lambda_{t+1})q(F(s_t - H(E_t, s_t))) \geq 0 \\ 0, & \text{if } (p_{t+1} - \lambda_{t+1})q(F(s_t - H(E_t, s_t))) < 0 \end{cases} \tag{8.9}$$

Proof. (Sketch) We formulate the Hamiltonians (Lenhart and Workman, 2007; Ding and Lenhart, 2010), which turn out to be linear in the control variables E_{t+1} with coefficients $(p_{t+1} - \lambda_{t+1})q(F(s_t - H(E_t, s_t)))$. Thus, the optimal sequence of E_{t+1} that maximizes the Hamiltonians is given according to the sign of those coefficients. □

Proof. (Complete) In order to decouple the state (s_t, the current resource stock) and the control[9] (E_t) and simplify the calculations, we resort to a change of variables. We define the new state w_t as the remaining stock after harvest at time-step t:

$$w_t \triangleq s_t - H(E_t, s_t)$$

Therefore,

$$s_{t+1} = w_{t+1} + H(E_{t+1}, s_{t+1}) \tag{8.10}$$

and

$$s_{t+1} = F(s_t - H(E_t, s_t)) = F(w_t) \tag{8.11}$$

Using Equation 8.10 and 8.11, we can write the new state equation as:

$$w_{t+1} = F(w_t) - H(E_{t+1}, F(w_t)) \tag{8.12}$$

In the current form of the state equation (Equation 8.12), the harvested resources appear outside the nonlinear growth function $F(\cdot)$, making the following analysis significantly simpler.

Under optimal control, the resource will not get depleted before the end of the horizon

[8] 'Optimal' is used in a technical sense, as the strategy that maximizes the revenue subject to the model equations, and it does not carry any moralistic implications.

[9] In accordance to the literature on Optimal Control Theory (Lenhart and Workman, 2007), 'state' in the context of the proof refers to the variable describing the the behavior of the underlying dynamical system, and 'control' refers to the input function used to steer the state of the system.

T, thus $q(s_t)E_t \leq s_t$, $\forall t < T$.[10] We can rewrite the total harvest as:

$$H(E_{t+1}, F(w_t)) = q(F(w_t))E_{t+1} \tag{8.13}$$

The state equation (Equation 8.12) becomes:

$$w_{t+1} = F(w_t) - q(F(w_t))E_{t+1}$$

and the optimization problem:

$$\begin{aligned} \max \quad & \sum_{t=-1}^{T-1} (p_{t+1}q(F(w_t))E_{t+1} - C_{t+1}) \\ s.t. \quad & w_{t+1} = F(w_t) - q(F(w_t))E_{t+1} \end{aligned} \tag{8.14}$$

Let $w_{-1} = s_0 = S_{eq}$. Solving the optimization problem is equivalent to finding the control that optimizes the Hamiltonians (Lenhart and Workman, 2007; Ding and Lenhart, 2010). Let $\lambda = (\lambda_{-1}, \lambda_0, \ldots, \lambda_{T-1})$ denote the adjoint function. The Hamiltonian at time-step t is given by:

$$\begin{aligned} \mathbf{H}_t &= p_{t+1}q(F(w_t))E_{t+1} - C_{t+1} + \lambda_{t+1}(F(w_t) - q(F(w_t))E_{t+1}) \\ &= (p_{t+1} - \lambda_{t+1})q(F(w_t))E_{t+1} - C_{t+1} + \lambda_{t+1}F(w_t) \end{aligned} \tag{8.15}$$

The adjoint equations are given by (Ding and Lenhart, 2010):

$$\begin{aligned} \lambda_t &= \frac{\partial \mathbf{H}_t}{\partial w_t} \\ \lambda_T &= 0 \\ \frac{\partial \mathbf{H}_t}{\partial u_t} &= 0 \text{ at } u_t = u_t^* \end{aligned}$$

where u_t is the control input, which corresponds to E_{t+1} in our formulation. The last condition corresponds to the maximization of the Hamiltonian $\mathbf{H}_t$ in Equation 8.15 for all time-steps t (Lenhart and Workman, 2007, Chapter 23). In our case, Equation 8.15 is linear in E_{t+1} with coefficient $(p_{t+1} - \lambda_{t+1})q(F(w_t))$. Therefore, the optimal sequence of

[10] Let us assume this is not the case and the optimal strategy would deplete the stock at certain time-step T_{dep}. That means that rewards are 0 and the optimal E_t is arbitrary for $t > T_{dep}$. Using the modified equation for the total harvest (Equation 8.13), we allow $H(E_t, s_t) > s_t$ or $s_t - H(E_t, s_t) < 0$. This would lead to $s_{t+1} = F(s_t - H(E_t, s_t)) < 0$ $\forall t > T_{dep}$, i.e., the stock would become negative. In such a case, any positive effort would decrease the revenue (since it would result in a negative harvest), thus the optimal strategy would be to set $E_t = 0$. Thus, using the modified equation for the total harvest (Equation 8.13), does not change the optimal solution.

E_{t+1} that maximizes Equation 8.15 is given based on the sign of the coefficient:

$$E^*_{t+1} = \begin{cases} E_{max}, & \text{if } (p_{t+1} - \lambda_{t+1})q(F(w_t)) \geq 0 \\ 0, & \text{if } (p_{t+1} - \lambda_{t+1})q(F(w_t)) < 0 \end{cases}$$

□

The optimal strategy is a bang–bang controller, which switches based on the adjoint variable values, stock level, and price. The values for λ_t do not have a closed form expression (because of the discontinuity of the control), but can be found iteratively for a given set of environment parameters (r, S_{eq}) and the adjoint equations (Lenhart and Workman, 2007; Ding and Lenhart, 2010). However, the discontinuity in the control input makes solving the adjoint equations quite cumbersome. We can utilize iterative forward/backward methods as in (Ding and Lenhart, 2010), but this is out of the scope of this work.

There are a few interesting key points. First, to compute the optimal level of effort we require observability of the resource stock, which is not always a realistic assumption (in fact in this work we do not make this assumption). Second, we require complete knowledge of the strategies of the other appropriators. Third, even if both the aforementioned conditions are met, the bang-bang controller of Equation 8.9 does not have a constant transition limit; the limit changes at each time time-step, determined by the adjoint variable λ_{t+1}, thus finding the switch times remains quite challenging.

8.3.3 Harvesting at Maximum Effort

To gain a deeper understanding of the dynamics of the environment, we will now consider a baseline strategy where every agent harvests with the maximum effort at every time-step, i.e., $\epsilon_{n,t} = \mathcal{E}_{max}, \forall n \in \mathcal{N}, \forall t$. This corresponds to the Nash Equilibrium of a stage game (myopic agents). For a constant growth rate r and a given number of agents N, we can identify two interesting stock equilibrium points (S_{eq}): the 'limit of sustainable harvesting', and the 'limit of immediate depletion'.

The limit of sustainable harvesting ($S^{N,r}_{LSH}$) is the stock equilibrium point where the goal of sustainable harvesting becomes trivial: for any $S_{eq} > S^{N,r}_{LSH}$, the resource will not get depleted, even if all agents harvest at maximum effort. Note that the coordination problem *remains far from trivial* even for $S_{eq} > S^{N,r}_{LSH}$, especially for increasing population sizes. Exerting maximum effort in environments with S_{eq} close to $S^{N,r}_{LSH}$ will yield low returns because the stock remains low, resulting in a small catchability coefficient. In fact, this can be seen in Figure 8.2 which depicts the social welfare (SW), i.e., sum of utilities, against increasing S_{eq} values ($N \in [2, 64]$, $\mathcal{E}_{max} = 1$, $r = 1$). Red dots[11] denote

[11] Slight deviations from the predicted theoretical values of Equation 8.16 due to the finite episode

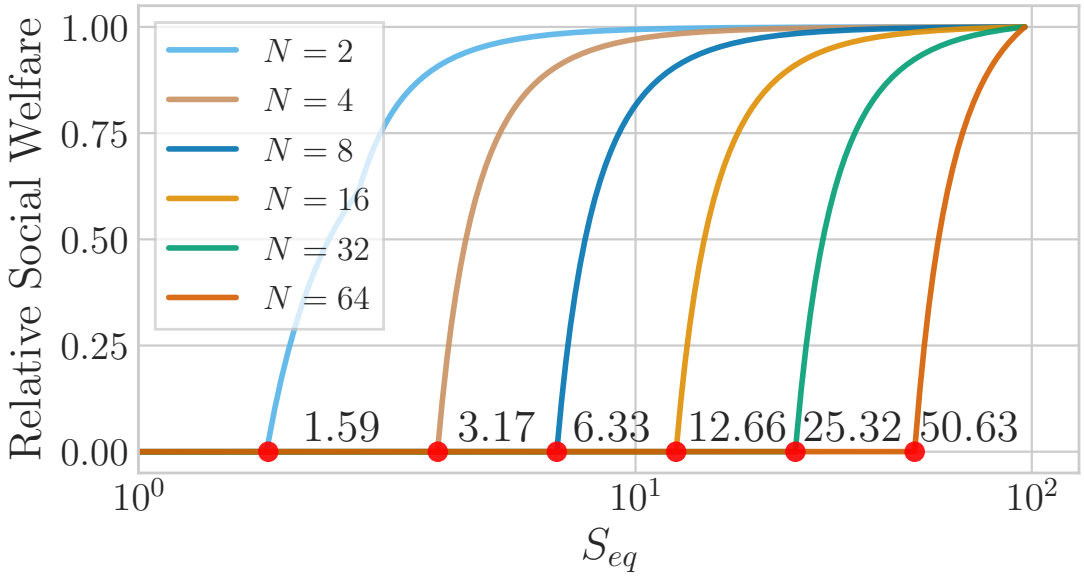


Figure 8.2: Social welfare (SW) – normalized by the maximum SW obtained in each setting – against increasing S_{eq} values. $N \in [2, 64]$, $\mathcal{E}_{max} = 1$, and $r = 1$. x-axis is in logarithmic scale.

the $S^{N,r}_{LSH}$. Thus, *the challenge is not only to keep the strategy sustainable, but to keep the resource stock high, so that the returns can be high as well.*

On the other end of the spectrum, the limit of immediate depletion ($S^{N,r}_{LID}$) is the stock equilibrium point where the resource is depleted in one time-step (under maximum harvest effort by all the agents). The problem does not become impossible for $S_{eq} \leq S^{N,r}_{LID}$, yet, *exploration can have catastrophic effects* (amplifying the problem of global exploration in MARL). The following two theorems prove the formulas for $S^{N,r}_{LSH}$ and $S^{N,r}_{LID}$.

Theorem 12. The limit of sustainable harvesting $S^{N,r}_{LSH}$ for a continuous resource governed by the dynamics of Section 8.3, assuming that all appropriators harvest with the maximum effort $\mathcal{E}_{max}$, is:

$$S^{N,r}_{LSH} = \frac{e^r N \mathcal{E}_{max}}{2(e^r - 1)} \tag{8.16}$$

Proof. Note that for $S_{eq} > S^{N,r}_{LSH}$, $q(s_t)E_t < s_t, \forall t$, otherwise the resource would be depleted. Moreover, if $s_0 = S_{eq}$ – which is a natural assumption, since prior to any intervention the stock will have stabilized on the fixed point – then[12] $s_t < 2S_{eq}, \forall t$. Thus, we can re-write Equation 8.1 and 8.2 as:

$$H(E_t, s_t) = \frac{s_t}{2S_{eq}} E_t = \frac{s_t N \mathcal{E}_{max}}{2S_{eq}}$$

length and non-zero threshold.

[12]Given that $r \in [-W(-1/(2e)), -W_{-1}(-1/(2e))]$.

Let $\alpha \triangleq \frac{N\mathcal{E}_{max}}{2S_{eq}}$, and $\beta = 1 - \alpha$. The state transition becomes:

$$s_{t+1} = F(s_t - \alpha s_t) = \beta s_t e^{r(1-\frac{\beta}{S_{eq}}s_t)}$$

We write it as a difference equation:

$$\Delta_t(s_t) \triangleq s_{t+1} - s_t = (\beta e^{r(1-\frac{\beta}{S_{eq}}s_t)} - 1)s_t$$

At the limit of sustainable harvesting, as the stock diminishes to[13] $s_t = \delta \to 0$, to remain sustainable it must be that $\Delta_t(s_t) > 0$. Thus, it must be that:

$$\lim_{s_t \to 0^+} sgn(\Delta_t(s_t)) > 0 \overset{s_t \to 0^+ > 0}{\Rightarrow} \beta e^r - 1 > 0 \Rightarrow S_{eq} > \frac{e^r N\mathcal{E}_{max}}{2(e^r - 1)}$$

□

Theorem 13. The limit of immediate depletion $S_{LID}^{N,r}$ for a continuous resource governed by the dynamics of Section 8.3, assuming that all appropriators harvest with the maximum effort $\mathcal{E}_{max}$, is given by:

$$S_{LID}^{N,r} = \frac{N\mathcal{E}_{max}}{2} \tag{8.17}$$

Proof. The resource is depleted if:

$$H(E_t, s_t) = s_t \Rightarrow q(s_t)E_t \geq s_t \Rightarrow \frac{s_t}{2S_{eq}}E_t \geq s_t \Rightarrow S_{eq} \leq \frac{N\mathcal{E}_{max}}{2}$$

□

8.3.4 Environmental Signal

We introduce an auxiliary signal; side information from the environment (e.g., time, date etc.) that agents can potentially use in order to facilitate coordination and reach more sustainable strategies. Real-world examples include shepherds that graze on particular days of the week or fishermen that fish on particular months. In our case, the signal can be thought as a mechanism to increase the set of possible (individual and joint) policies. Such signals are *amply available to the agents* (Hart and Mas-Colell, 2000; Danassis and Faltings, 2019). *We do not assume any a priori relation between the signal and the problem at hand.* In fact, in this work we use a set of arbitrary integers, that repeat periodically. We use $\mathcal{G} = \{1, \ldots, G\}$ to denote the set of signal values.

[13] In practice, δ is enforced by the granularity of the resource.

8.4 Simulation Results

8.4.1 Setup

Environment Settings

Let $p_t = 1$, and $c_t = 0$, $\forall t$. We set the growth rate[14] at $r = 1$, the initial population at $s_0 = S_{eq}$, and the maximum effort at $\mathcal{E}_{max} = 1$. The findings of Section 8.3.3 provide a guide on the selection of the S_{eq} values. Specifically we simulated environments with S_{eq} given by Equation 8.18, where $K = \frac{S^{N,r}_{LSH}}{N} = \frac{e^r \mathcal{E}_{max}}{2(e^r-1)} \approx 0.79$ is a constant and $M_s \in \mathbb{R}^+$ is a multiplier that adjusts the scarcity (difficulty). $M_s = 1$ corresponds to $S_{eq} = S^{N,r}_{LSH}$.

$$S_{eq} = M_s K N \tag{8.18}$$

Agent Architecture

We consider a decentralized multi-agent learning scenario in a partially observable general-sum Markov game (Shapley, 1953), where each agent is modeled as an independent deep reinforcement learning agent. At each time-step, agents take actions based on a partial observation of the state space, and receive an individual reward. See Section 7.1 for a detailed description.

Each agent uses a two-layer (64 neurons each) feed-forward neural network for the policy approximation. The input (observation $o^n = \mathcal{O}^n(S)$) is a tuple $\langle \epsilon_{n,t-1}, u_{n,t-1}(\epsilon_{n,t-1}, s_{t-1}), g_t \rangle$ consisting of the individual effort exerted and reward obtained in the previous time-step and the current signal value. The output is a continuous action value $a_t = \epsilon_{n,t} \in [0, \mathcal{E}_{max}]$ specifying the current effort level. The policies are trained using the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). PPO was chosen because it avoids large policy updates, ensuring a smoother training, and avoiding catastrophic failures. The reward received from the environment corresponds to the individual revenue, i.e., $r^n(\sigma_t, \boldsymbol{a}_t) = u_{n,t}(\epsilon_{n,t}, s_t)$, and the discount factor was set to $\gamma = 0.99$.

Signal Implementation

The introduced signal is represented as a G-dimensional one-hot encoded vector, where the high bit is shifted periodically. In particular, its value at index i at time-step t is given by:

$$\begin{cases} 1 & \text{if } mod(t - t_{init},\ G) = i \\ 0 & \text{otherwise} \end{cases} \tag{8.19}$$

[14] We also run simulations for $r = 2$. See Appendix C, Tables C.17, and C.18.

where t_{init} is a random offset chosen at the beginning of each episode to avoid bias towards particular values. Throughout this chapter, the term *no signal* will be used interchangeably to a unit signal size $G = 1$, since a signal of size 1 in one-hot encoding is just a constant input that yields no information. We evaluated signals of varying cardinality (see Section 8.4.6).

Termination Condition

An episode terminates when either (a) the resource stock falls below a threshold $\delta = 10^{-4}$ (which represents the granularity of the resource[15]), or (b) a fixed number of time-steps $T_{max} = 500$ is reached. We trained our agents for a maximum of 5000 episodes, with the possibility of early stopping if both of the following conditions are satisfied: (i) a minimum of 95% of the maximum episode duration (i.e., 475 time-steps) is reached for 200 episodes in a row, and, (ii) the average total reward obtained by agents in each episode of the aforementioned 200 episodes does not change by more than 5%. In case of early stopping, the metric values for the remainder of the episodes are extrapolated as the average of the last 200 episodes, in order to properly average across trials.

We implemented early stopping for two reasons: (i) to speed up the training process, and most importantly (ii) as a practical way to determine the convergence time and evaluate the effects of the introduced signal on the training speed (see Section 8.4.4). We consider the system to be converged when the global state does not change significantly, i.e., when it reaches our termination condition.

Measuring The Influence of the Signal

It is important to have a quantitative measure of the influence of the introduced signal. As such, we adapted the Causal Influence of Communication (CIC) (Lowe et al., 2019) metric, initially designed to measure positive listening in emergent inter-agent communication.

The CIC estimates the mutual information between the signal and the agent's action. The mutual information between two random variables $\tilde{X}$ and $\tilde{Y}$ is defined as the reduction of uncertainty (measured in terms of entropy $H_S(\cdot)$) in the value of $\tilde{X}$ with the observation of $\tilde{Y}$:

$$I(\tilde{X}, \tilde{Y}) = I(\tilde{Y}, \tilde{X}) = H_S(\tilde{X}) - H_S(\tilde{X}|\tilde{Y}) = E\left\{log\left(\frac{P_{\tilde{X},\tilde{Y}}(\tilde{x}, \tilde{y})}{P_{\tilde{X}}(\tilde{x})P_{\tilde{Y}}(\tilde{y})}\right)\right\}$$

[15] We set $\delta = 10^{-4}$ as it provides reasonable granularity given the S_{eq} values (which are in the order of 10 to 10^2). Preliminary results suggest that using a smaller value does not affect the results significantly. This is because once the system goes below such a small stock value, replenishment is practically impossible in the duration of an episode.

Algorithm 7 CIC Implementation (based on (Lowe et al., 2019))

1: **input:** Agent policy $\pi(\cdot)$
2: $p(g_j) = \frac{1}{G}$ for all possible signals
3: Discretize the action space $[0, \mathcal{E}_{max}]$ into N_{bins} intervals
4: **for** i=1 to N_{states} **do**
5: Generate a state without a signal σ_{-g} randomly
6: **for** all possible signals g_j **do**
7: Generate agent observation $\sigma = [g_j, \sigma_{-g}]$
8: Estimate $p(a_i|g_j)$ by sampling $N_{samples}$ actions from $\pi(\sigma)$
9: $p(a_i, g_j) = p(a_i|g_j)p(g_j)$
10: $p(a_i) = \sum_j p(a, g_j)$
11: $CIC \mathrel{+}= \frac{1}{N_{states}} \sum_{a_i|p(a_i)\neq 0,\ g_j} p(a, g_j) log\left(\frac{p(a,g_j)}{p(a)p(g_j)}\right)$

The pseudo-code for calculating the CIC for a single agent is presented in Algorithm 7. Note that the CIC implementation in (Lowe et al., 2019) considers a multi-dimensional, one-hot, discrete action space with accessible probabilities for every action, while in our case we have a single, continuous action (specifically, the effort $\epsilon_{n,t}$). To solve this problem, we discretize our action space into N_{bins} intervals between minimum ($\mathcal{E}_{min} = 0$) and maximum ($\mathcal{E}_{max} = 1$) effort values, and each interval is assumed to correspond to a single discrete action. Let a_i denote the event of an action $\epsilon_{n,t}$ belonging to interval i. To calculate the CIC value, we start by generating N_{states} random 'partial' states (i.e., without signal), σ_{-g}, which are then concatenated with each possible signal value to obtain a 'complete' state, $\sigma = [g_j, \sigma_{-g}]$ (Lines 5 - 7 of Algorithm 7). Then, we estimate the probability of an action given a signal value, $p(a_i|g_j)$, by generating $N_{samples}$ actions from our policy given the 'complete' state ($\pi(\sigma)$), and normalizing the number of instances in which the action belongs to a particular bin with the total number of samples. The remaining aspects of the calculation are the same as in the original implementation. In our calculations we used $N_{states} = N_{samples} = 100$.

Fairness Metrics

We also evaluated the fairness of the final allocation, to ensure that agents are not being exploited by the introduction of the signal. We used two of most established fairness metrics: the Jain index (Jain et al., 1998) and the Gini coefficient (Gini, 1912) (see Section 2.5).

Reproducibility, Reporting of Results, Limitations

Reproducibility is a major challenge in (MA)DRL due to different sources of stochasticity, e.g., hyper-parameters, model architecture, implementation details, etc. (Henderson et al., 2018; Hernandez-Leal et al., 2019; Engstrom et al., 2020). Moreover, recent work has

Table 8.1: List of hyper-parameters.

Parameter	Value
Learning Rate (α)	0.0001
Clipping Parameter	0.3
Value Function Clipping Parameter	10.0
KL Target	0.01
Discount Factor (γ)	0.99
GAE Parameter Lambda	1.0
Value Function Loss Coefficient	1.0
Entropy Coefficient	0.0

demonstrated that code-level optimizations play an important role in performance, both in terms of achieved reward and underlying algorithmic behavior (Engstrom et al., 2020). To minimize those sources of stochasticity, the implementation[16] was done using RLlib[17], an open-source library for DRL (Liang et al., 2017). All the hyper-parameters were left at the default values specified in Ray and RLlib.[18] We opted to do so since the focus of this this work is in the performance of the introduced signal and not of the training algorithm. For completeness, Table 8.1 presents a list of the most relevant of them.

All simulations were *repeated* 8 *times* and the reported results are the average values of the last 10 episodes over those trials (excluding Figure 8.8 which depicts a representative trial). (MA)DRL also lacks common practices for statistical testing (Henderson et al., 2018; Hernandez-Leal et al., 2019). In this work, we opted to use the Student's T-test (Student, 1908) due to it's robustness (Colas et al., 2019). Nearly all of the reported results have p-values < 0.05.

Finally, we strongly believe that the community would benefit from reporting negative results. As such, we want to make clear that the proposed solution is not a panacea for all multi-agent coordination problems, not even for the proposed domain. For example, we failed to find sustainable policies using DDPG (Lillicrap et al., 2015) – with or without the signal – for any set of environment parameters. This also comes to show the difficulty of the problem at hand. We suspect that the clipping in PPO's policy changes plays an important role in averting catastrophic failures in high-stakes environments.

[16]The source code can be found here: https://github.com/panayiotisd/Improved-Cooperation-by-Exploiting-a-Common-Signal.

[17]https://docs.ray.io/en/latest/rllib.html

[18]See https://docs.ray.io/en/latest/rllib-algorithms.html#ppo and https://docs.ray.io/en/latest/rllib-training.html#common-parameters.

8.4.2 Results

We present the result from a systematic evaluation of the proposed approach on a wide variety of environmental settings ($M_s \in [0.2, 1.2]$, i.e., ranging from way below the limit of immediate depletion, $M_s^{LID} \approx 0.63$, to above the limit of sustainable harvesting, $M_s^{LSH} = 1$) and population size ($N \in [2, 64]$).

In the majority of the results, we study the influence of a signal of cardinality $G = N$ compared to no signal ($G = 1$). Thus, unless stated otherwise, the term 'with signal' will refer to $G = N$.

We run simulations with a growth rate of $r = 1$ and $r = 2$. In all of the reported results in the following sections the growth rate is set to $r = 1$, except in Section 8.4.7 for $N = 64$, where we set[19] $r = 2$.

All reported results are the average values over 8 trials.

Some results were omitted from the main text for brevity and to improve readability. The numerical values of all the results – the ones presented in the following sections and those that are omitted (e.g., larger growth rate, smaller population sizes, fairness, etc.) – can be found in detail in Appendix C.

8.4.3 Sustainability & Social Welfare

Sustainability

We declare a strategy 'sustainable', iff the agents reach the maximum episode duration (500 steps), i.e., they do not deplete the resource. Figure 8.3 depicts the achieved episode length – with and without the presence of a signal ($G = N$) – for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$). The introduction of the signal significantly increases the range of environments (M_s) where sustainability can be achieved. Assuming that $M_s \in [0, 1]$ – since for $M_s \geq 1$ sustainability is guaranteed by definition – we have an increase of $17\% - 300\%$ (46% on average) in the range of sustainable M_s values. Moreover, as the number of agents increases ($N = 32$ & 64), depletion is avoided in non-trivial M_s values *only with the introduction of the signal*. Finally, note that the M_s value where a sustainable strategy is found increases with N, which demonstrates that the difficulty of the problem increases superlinearly to N (given that $S_{eq} \propto M_s N$).

The numerical values can be found in Appendix C, Tables C.3 and C.4.

[19] Note that, as was specified in Section8.4.1, $K = \frac{S_{LSH}^{N,r}}{N} = \frac{e^r \mathcal{E}_{max}}{2(e^r - 1)}$. Therefore, in the settings where the growth rate is $r = 1$, $K \approx 0.79$, while in the settings where $r = 2$, $K \approx 0.58$.

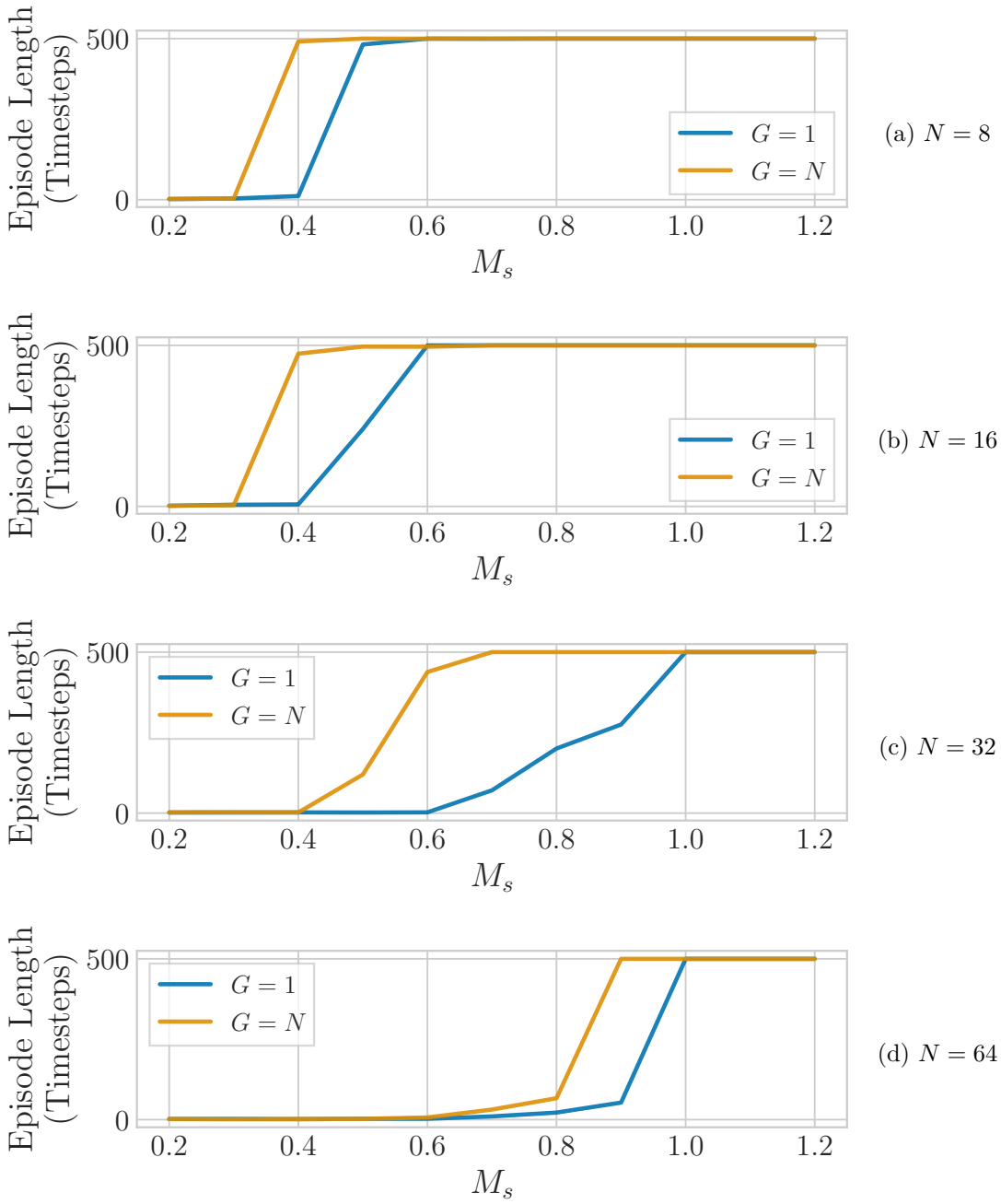


Figure 8.3: Episode length, with and without the signal ($G = N$), for environments of decreasing difficulty (increasing equilibrium stock multiplier M_s).

Social welfare

Reaching a sustainable strategy – i.e., avoiding resource depletion – is only one piece of the puzzle; an agent's revenue depends on the harvest (Equation 8.1), which in turns depends on the catchability coefficient (Equation 8.2). Thus, in order to achieve a high

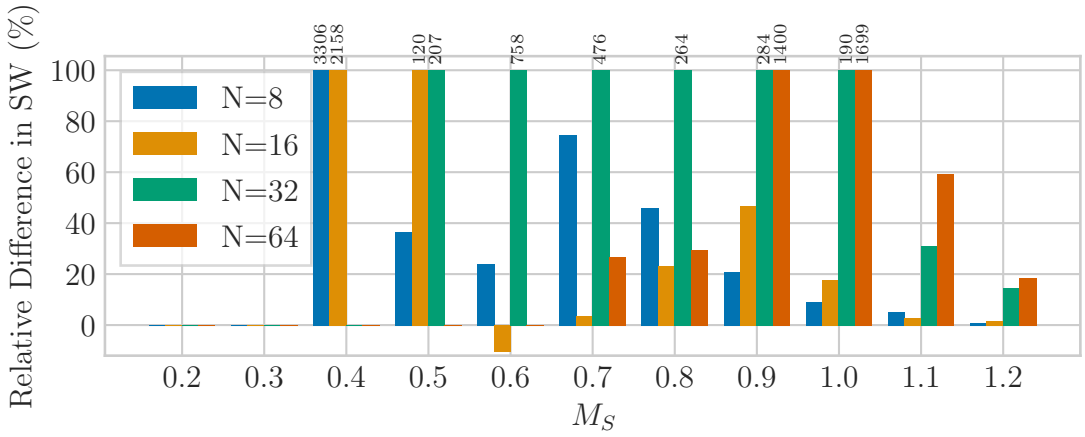


Figure 8.4: Relative difference in social welfare (SW) when signal of cardinality $G = N$ is introduced ($(SW_{G=N} - SW_{G=1})/SW_{G=1}$, where $SW_{G=X}$ denotes the SW achieved using a signal of cardinality X), for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$) and varying population size ($N \in [4, 64]$). To improve readability, changes greater than 100% are shown with numbers on the top of the bars.

social welfare (sum of utilities, i.e., $\sum_{n \in \mathcal{N}} r^n(\cdot)$), the agents need to learn policies that balance the trade-off between maintaining a high stock (which ensues a high catchability coefficient), and yielding a large harvest (which results to a higher reward). This problem becomes even more apparent as resources become more abundant (i.e., for $M_s = 1 \pm x$, i.e., close to the limit of sustainable harvesting (below or, especially, *above*), see Section 8.3.3). In these settings, it is easy to find a sustainable strategy; a myopic best-response strategy (harvesting at maximum effort) by all agents will not deplete the resource. Yet, it will result in low social welfare (SW).

Figure 8.4 depicts the relative difference in SW, in a setting with and without the signal ($(SW_{G=N} - SW_{G=1})/SW_{G=1}$, where $SW_{G=X}$ denotes the SW achieved using a signal of cardinality X), for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$) and varying population size ($N \in [8, 64]$). To improve readability, changes greater than 100% are shown with numbers on the top of the bars. Given the various sources of stochasticity, we opted to omit settings in which agents were not able to reach an episode duration of more than 10 time-steps (either with or without the signal).

The presence of the signal results in a significant improvement in SW.[20] Specifically, we have an average of 258% improvement *across all the depicted settings*[21] in Figure

[20] One exception is the case with $N = 16$ and $M_s = 0.6$. Even though it seems that the introduction of the signal has a negative effect, this is *not* actually the case. At this point both methods manage to find sustainable strategies, and the small difference is only due to noise (a fact that is corroborated by the p-value, see Table C.2).

[21] The averaging is performed across the entire range of the depicted $M_s \in [0.2, 1.2]$, including the really scarce environments of $M_s = 0.2$ and 0.3 where there is no sustainable strategy with or without

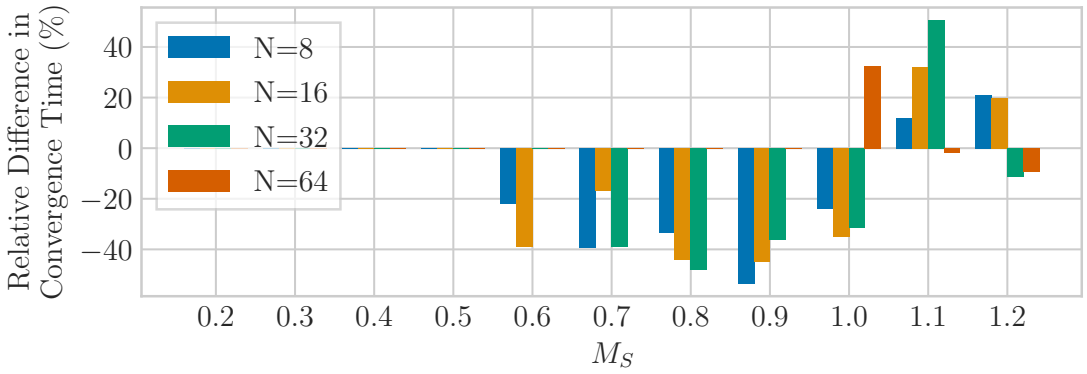


Figure 8.5: Relative difference in convergence time with the introduction of a signal ($(CT_{G=N} - CT_{G=1})/CT_{G=1}$, where $CT_{G=X}$ denotes the time until convergence when using a signal of cardinality X), for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$) and varying population size ($N \in [8, 64]$)

8.4, while the maximum improvement is 3306%. These improvements stem from (i) achieving more sustainable strategies, and (ii) improved cooperation. The former results in higher rewards due to longer episodes in settings where the strategies without the signal deplete the resource. The latter allows to avoid over-harvesting, which results in higher catchability coefficient, in settings where both strategies (with, or without the signal) are sustainable. The contribution of the signal is much more pronounced under scarcity: the difference in achieved SW decreases as M_s increases, eventually becoming less than 10% ($M_s > 1$ for $N = 8$ & 16, and $M_s > 1.2$ for $N = 32$ & 64). This suggests that the proposed approach is of high practical value in environments where resources are *scarce* (like most real-world applications), a claim that we further corroborate in Sections 8.4.5 and 8.4.7.

The numerical values can be found in Appendix C, Tables C.1 and C.2.

8.4.4 Convergence Speed

The second major influence of the introduction of the proposed signal – besides the sustainability and efficiency of the learned strategies – is on the convergence time. Let the system be considered converged when the global state does not change significantly. As a practical way to pinpoint the time of convergence, we used the 'Termination Criterion' of Section 8.4.1. Figure 8.5 depicts the relative difference in convergence time with the introduction of a signal ($(CT_{G=N} - CT_{G=1})/CT_{G=1}$, where $CT_{G=X}$ denotes the time until convergence, in #episodes, when using a signal of cardinality X), for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$) and varying population size ($N \in [8, 64]$). We have omitted the settings in which agents were not able to reach an episode duration

the signal and, thus, the change is zero.

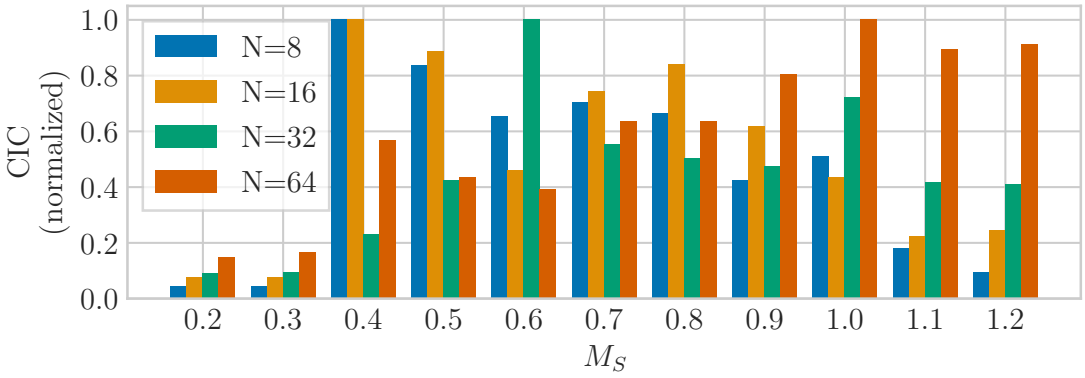


Figure 8.6: Average (over agents and trials) CIC values (normalized) vs. the equilibrium stock multiplier M_s, for population/signal size $N = G \in \{8, 16, 32, 64\}$.

of more than 10 time-steps (either with or without the signal).

There is a disjoint effect of the signal on the convergence speed. Up to the limit of sustainable harvesting ($M_s \leq 1$), the signal significantly improves the convergence speed (13% improvement on average, *across all the depicted settings* including the ones with no improvement, and up to 53%). This is vital, as the majority of *real-world problems involve managing scarce resources*. On the other hand, for $M_s > 1$, i.e., settings with abundant resources, the system converges faster without the signal (14% slower with the signal on average, across all the depicted settings). One possible explanation is that as resources become more abundant, it is harder (impossible for $M_s > 1$) for agents to deplete them. Therefore the learning is more efficient – and the convergence is faster – since the episodes tend to last longer (without needing the signal). Moreover, having an abundance of resources decouples the effects of the agents' actions to each other, reducing the variance, and again making easing the learning process without the signal.

The numerical values can be found in Appendix C, Tables C.5 and C.6.

8.4.5 Influence of Signal on Agent Strategies

The results presented so far provide an indirect measure of the influence of the introduced signal through the improvement on sustainability, social welfare, and convergence speed. They also indicate a decrease on the influence of the signal as resources become abundant. The question that naturally arises is: how much do agents actually take the signal into account in their policies? To answer this question, Figure 8.6 depicts the CIC values – a quantitative measure of the influence of the introduced signal (see Section 8.4.1) – versus increasing values of M_s (i.e, increasing $S_{eq} \propto M_s$, or more abundant resources), for population/signal size $N = G \in \{8, 16, 32, 64\}$. The values are averaged across the 8 trials and the agents, and are normalized with respect to the maximum value for each

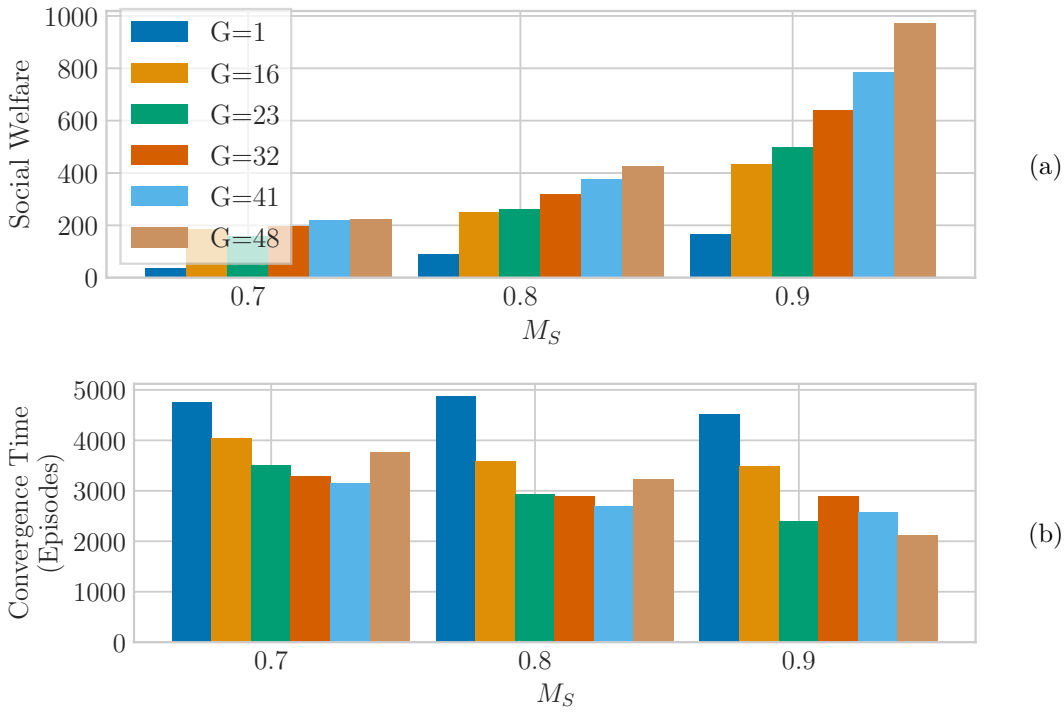


Figure 8.7: Achieved social welfare (Figure 8.7a) and convergence time (Figure 8.7b) for different signals of cardinality (G) 1, $\frac{N}{2} = 16$, 23, $N = 32$, 41, and $\frac{3N}{2} = 48$ (for varying resource levels M_s).

population.[22] Higher CIC values indicate a higher causal influence of the signal.

CIC is low for the trials in which a sustainable strategy could not be found ($M_s = 0.2-0.3$ for $N = 8$, 16, $M_s = 0.2 - 0.5$ for $N = 32$, and $M_s = 0.2 - 0.8$ for $N = 64$, see Figure 8.3). In cases where a sustainable strategy was reached (e.g., $M_s \geq 0.4$ for $N = 8$), we see significantly higher CIC values on scarce resource environments, and then the CIC decreases as M_s increases. *The harder the coordination problem, the more the agents rely on the environmental signal.*

The numerical values can be found in Appendix C, Table C.11.

8.4.6 Robustness to Signal Size

Up until now we have evaluated the presence (or lack thereof) of an environmental signal of cardinality equal to the population size ($G = N$). This requires exact knowledge of N, thus it is interesting to test the robustness of the proposed approach under varying signal sizes. As a representative test-case, we evaluated different signals of cardinality $G = 1$, $\frac{N}{2}$, 23, N, 41, and $\frac{3N}{2}$ for $N = 32$ and moderate scarcity for the resource (M_s

[22] Figure 8.6 shows trends across M_s values – *not* between populations sizes (due to the normalization).

values of 0.7, 0.8 and 0.9). The values 23 and 41 were chosen as they are prime numbers (i.e., *not multiples* of N). Figure 8.7 depicts the achieved social welfare and convergence time under the aforementioned settings.

Starting with Figure 8.7a we can see that the SW increases with the signal cardinality. Specifically, we have 263%, 255%, 341%, 416%, and 474% improvement on average across the three M_s values for $G = \frac{N}{2}$, 23, N, 41, and $\frac{3N}{2}$, respectively. We hypothesize that the improvement stems from an increased joint strategy space that the larger signal size allows. A signal size larger than N can also allow the emergence of 'rest' (fallow) periods – signal values where the majority of agents harvests at really low efforts. This would allow the resource to recuperate, and increase the SW through a higher catchability coefficient. See Section 8.4.7 / Figure 8.8b for an example of such a joint strategy.

Regarding the convergence speed (Figure 8.7b), we have 22%, 38%, 36%, 41%, and 36% improvement on average (across M_s values) for the aforementioned signal values, respectively.

These results showcase that the introduction of the signal itself – regardless of its cardinality or, more generally, its temporal representative power – provides a clear benefit to the agents in terms of SW and convergence speed. This greatly improves the real-world applicability of the proposed technique, as the the *knowledge of the exact population size is not required*; instead the agents can opt to select any signal available in their environment.[23] Moreover, the signal cardinality can also be considered as a design choice, decided depending on the requirements and limitations of the system in consideration.

The numerical values can be found in Appendix C, Tables C.12, C.13, C.14, C.15, and C.16.

8.4.7 Emergence of Temporal Conventions

Qualitative Analysis

We have seen that the introduction of an arbitrary signal facilitates cooperation and the sustainable harvesting. But do temporal conventions actually emerge?

Figure 8.8a presents an example of the evolution of the agents' strategies for each signal value for a population of $N = 4$, signal size $G = N = 4$, and equilibrium stock multiplier $M_s = 0.5$ (smoothed over 50 episodes). Each row represents an agent (agent n_i), while each column represents a signal value (value g_j). Each line represents the average effort the agent exerts on that specific signal value – calculated by averaging the actions of the

[23] The signal is represented as a one-hot vector, i.e., Figure 8.7 provides strong evidence that a network with 32 inputs can work for population sizes $N \in [16, 48]$, or equivalently, that agents in a population of size $N = 32$ can use networks with $16 - 48$ inputs for the signal.

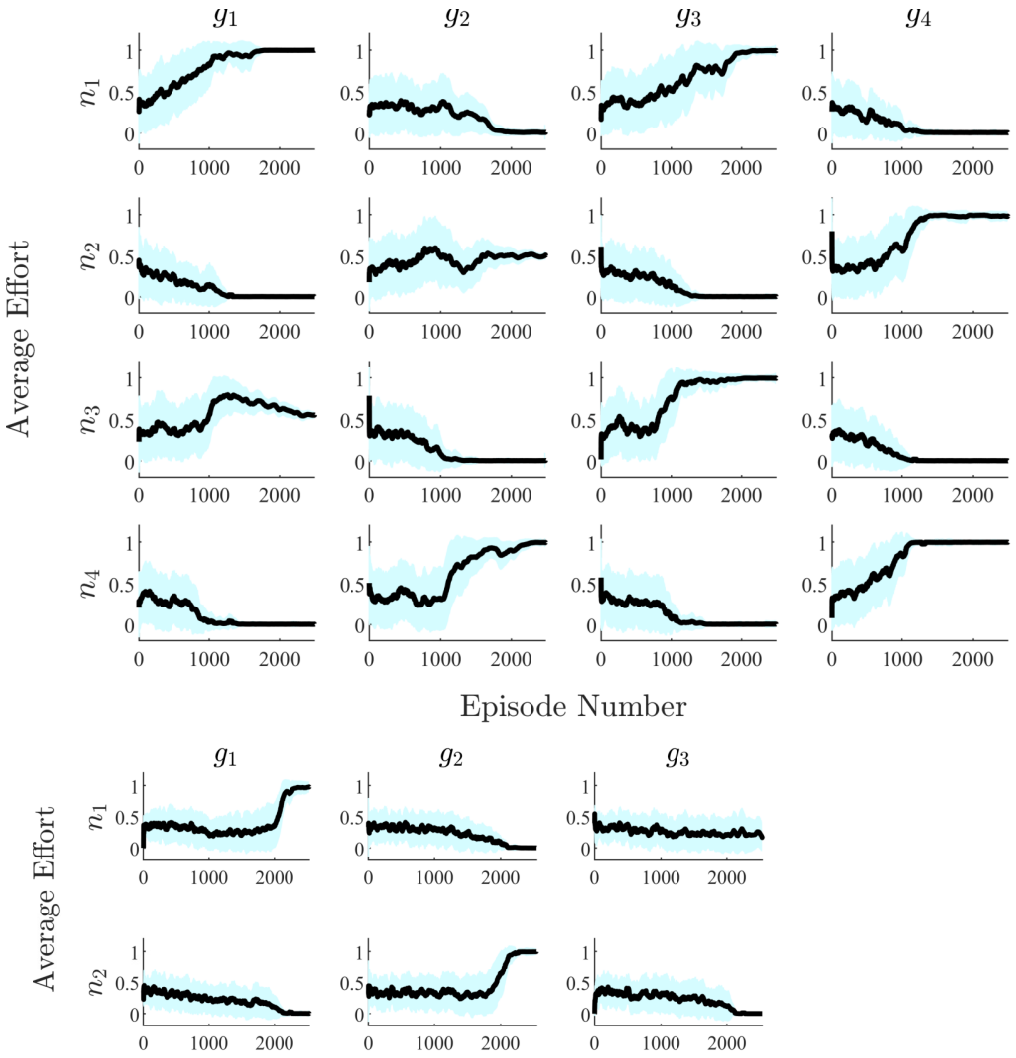


Figure 8.8: Evolution of the agents' strategies for each signal value, smoothed over 50 episodes. Figure 8.8a pertains to a population of $N = 4$ and signal size $G = N = 4$, while Figure 8.8b to a population of $N = 2$ and signal size $G = \frac{3N}{2} = 3$. In both cases the equilibrium stock multiplier is $M_s = 0.5$. Each row represents an agent (n_i), while each column a signal value (g_j). Each line depicts the average effort the agent exerts on that specific signal value – calculated by averaging the actions of the agent in each corresponding signal value across the episode. Shaded areas represent one standard deviation.

agent in each corresponding signal value across the episode.

We can see a clear temporal convention emerging: at signal value g_1 (first column), only agents n_1 and n_3 harvest (first and third row), at g_2, n_2 and n_4 harvest, at g_3, n_1 and n_3 harvest, and, finally, at g_4, n_2 and n_4. Contrary to that, in a sustainable joint strategy without the use of the signal, every agent harvests at every time-step with an average

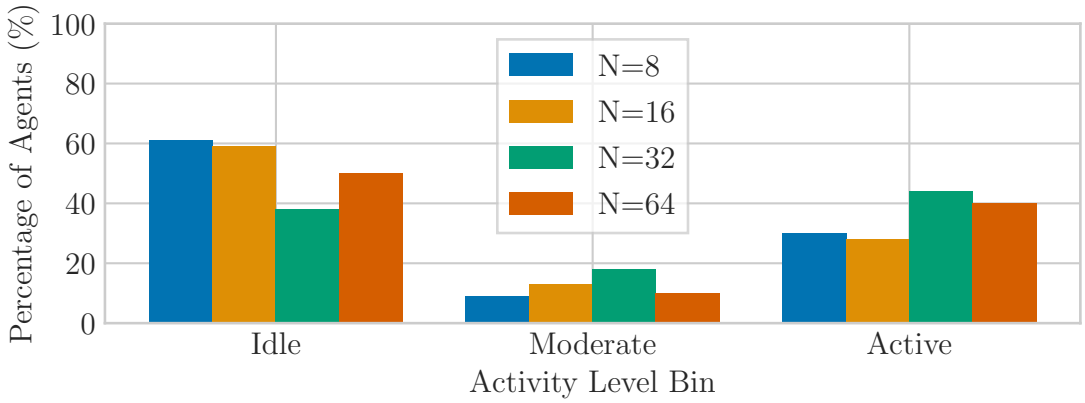


Figure 8.9: Percentage of agents in each 'activity' level bin, for population/signal size $N = G \in \{8, 16, 32, 64\}$.

(across all agents) effort of $\approx 40\%$ (for the same setting of $N = 4$ and $M_s = 0.5$). *Having all agents harvesting at every time-step makes coordination increasingly harder as we increase the population size*, mainly due to the non-stationarity of the environment (high variance) and the global exploration problem.

Access Rate

In order to facilitate a systematic analysis of the accessing patterns, we discretized the agents into three bins: agents harvesting with effort $\epsilon \in [0 - 0.33)$ ('idle'), $[0.33 - 0.66)$ ('moderate'), and $[0.66 - 1]$ ('active'). Then we counted the average number of agents in each bin at the first equilibrium stock multiplier (M_s) where a non-depleting strategy was achieved in each setting. Without a signal, either the majority of the agents are 'moderate' harvesters (specifically 84% for $N = 8$ and 16), or *all* of them are 'active' harvesters (100% for $N = 32$ and 64). With the signal, we have a clear separation into 'idle' and 'active': ('idle', 'active') $= (61\%, 30\%)$, $(59\%, 28\%)$, $(38\%, 44\%)$, $(50\%, 40\%)$, for $N = 8$, 16, 32, and 64, respectively (see Figure 8.9).[24] It is apparent that with the signal the agents learn a temporal convention; *only a minority is 'active' per time-step*,

[24]The setting with $N = 64$ was run with $r = 2$ in both cases (with and without the signal). Every resource has a natural upper limit on the size of the population it can sustain. Figure 8.3 shows that as the number of agents grows, we reach sustainable strategies at higher equilibrium stock multipliers (M_s). Thus, we expect that as we increase the growth rate, the effect of the signal will be even more pronounced in larger populations (N) – which is corroborated by the results for $r = 2$ (see Tables C.17, and C.18). The environment's ability to sustain a population also affects the counts of 'active' agents. As we can see in Figure 8.3, for a growth rate of $r = 1$ and even with the addition of the signal, the first sustainable strategy is reached at $M_s = 0.9$ (i.e., the resource is too scarce). This is a high equilibrium stock multiplier, close to the limit of sustainable harvesting. As a result, the number of 'active' agents is naturally really high because they do not need to harvest in turns. By increasing the growth rate to $r = 2$, we have an environment that can sustain larger populations. Therefore, the first strategy that does not result to an immediate depletion is reached much earlier, and we can observe the emergence of a temporal convention (see Table C.19).

allowing to maintaining a healthy stock and reach *sustainable strategies of high social welfare*.

The numerical values can be found in Appendix C, Table C.19.

Fallowing

A more interesting joint strategy can be seen in Figure 8.8b ($N = 2$, $M_s = 0.5$). In this setting, we have an increased number of available signals, specifically $G = \frac{3N}{2} = 3$. We can see that agents harvest alternatingly in the first two signal values, and rest on the third (*fallow period*), potentially to allow resources to replenish and consequently obtain higher rewards in the future due to a higher catchability coefficient. This also resembles the optimal (bang-bang) harvesting strategy of Theorem 11.

8.4.8 Fairness

Finally, we evaluated the fairness of the final allocation. Naturally, if the resource is depleted, the final allocation is fair (all agents receive zero reward from a depleted resource). As the social welfare increases, it is important to maintain fairness and ensure that the introduction of the signal does not result in an exploiter-exploitee situation. Both fairness metrics showed that learning both with and without the signal results in fair allocations, with no significant change with the introduction of the signal.[25] Specifically, both methods achieved a Jain index $\mathbb{J}(\mathbf{x}) > 0.97$ ($\mathbb{J}(\mathbf{x}) \in [0, 1]$, higher being fairer), and Gini coefficient $\mathbb{G}(\mathbf{x}) < 0.08$ ($\mathbb{G}(\mathbf{x}) \geq 0$, lower being fairer).

The numerical values can be found in Appendix C, Tables C.7 and C.8 for the Jain index, and Tables C.9 and C.10 for the Gini coefficient.

8.5 Chapter Conclusion

The challenge to cooperatively solve 'the tragedy of the commons' remains as relevant now as when it was first introduced by Hardin in 1968. Sustainable development and avoidance of catastrophic scenarios in socio-ecological systems – like the permanent depletion of resources, or the extinction of endangered species – constitute critical open problems. To add to the challenge, real-world problems are inherently large in scale and of low observability. This amplifies traditional problems in multi-agent learning, such as the global exploration and the moving-target problem. Earlier work in common-pool resource appropriation utilized intrinsic or extrinsic incentives (e.g., reward or opponent shaping). Yet, such techniques need to be designed for the problem at hand

[25] The relative values show a consistent improvement with the introduction of the signal (especially in the Gini coefficient) but, in absolute terms, both methods actually result in fair allocations.

and/or require communication or observability of states/actions, which is not always feasible (e.g., in commercial fisheries, the stock or harvesting efforts can not be directly observed). Humans on the other hand show a remarkable ability to self-organize and resolve common-pool resource dilemmas, often *without any extrinsic incentive mechanism or communication*. Social conventions and the use of auxiliary environmental information constitute key mechanisms for the emergence of cooperation under low observability. In this chapter, we demonstrate that utilizing such environmental signals – which are amply available – is a simple, yet powerful and robust technique, to foster cooperation in large-scale, low observability, and high-stakes environments. We are the first to tackle a realistic common-pool resource appropriation scenario modeled on real-world commercial fisheries and under low observability. Our approach avoids permanent depletion in a wider (up to 300%) range of settings, while achieving higher social welfare (up to 3306%) and convergence speed (up to 53%).

9 Achieving Diverse Objectives with AI-driven Prices

9.1 Preface

9.1.1 Contribution and Sources

This chapter is largely based on (Danassis et al., 2021b). Ideation, theory, experiment design, and most of the writing was done by the author. The detailed individual contributions are listed below using the CRediT taxonomy (Brand et al., 2015) (terms are selected as applicable):

PD (author):	Conceptualization (lead), Methodology, Software, Investigation (lead), Writing – Original Draft, Writing – Review & Editing
Aris Filos-Ratsikas:	Conceptualization (supporting), Investigation (supporting), Writing – Original Draft, Writing – Review & Editing
Boi Faltings:	Supervision

9.1.2 Chapter Summary

We propose a practical approach to computing market prices and allocations via a *deep reinforcement learning policymaker* agent, operating in an environment of other learning agents. Compared to the idealized market equilibrium outcome – which we use as a benchmark – our policymaker is much more flexible, allowing us to *tune* the prices with regard to diverse objectives such as sustainability and resource wastefulness, fairness, buyers' and sellers' welfare, etc. To evaluate our approach, we design a realistic market with multiple and diverse buyers and sellers. Additionally, the sellers, which are deep learning agents themselves, compete for resources in a common-pool appropriation environment based on bio-economic models of commercial fisheries.

We demonstrate that: (a) The introduced policymaker is able to achieve comparable performance to the market equilibrium, showcasing the potential of such approaches in markets where the equilibrium prices can not be efficiently computed. (b) Our policymaker can notably outperform the equilibrium solution on certain metrics, while at the same time maintaining comparable performance for the remaining ones. (c) As a highlight of our findings, our policymaker is significantly more successful in maintaining resource *sustainability*, compared to the market outcome, in scarce resource environments.

9.2 Introduction

The theory of competitive markets, founded in the works of Walras (Walras, 1874), Fisher (Fisher, 1892) and Arrow and Debreu (Arrow and Debreu, 1954) is one of the most prominent economic models of the 20th century. The *market equilibrium* – a stable outcome in which supply equals demand, and all participants are maximally satisfied by the bundles of goods that they buy or sell – is meant to capture the outcome of a free market, dictated by the market's 'invisible hand' (Smith, 1791), which adjusts the prices until market clearance is achieved via the so-called *tâtonnment process* (Walras, 1874) (i.e., continuous adjustment of supply and demand). However, the fundamental principles of this theory are based on often idealized assumptions, namely that the participants are price-takers – and thus they do not influence the prices of the market – and that this continuous adjustment of prices will indeed lead to the desired outcome. In reality, there have been several examples, especially in markets with a limited number of sellers, where instances of collusion and price manipulation have been observed;[1] practices which are known to deter the market from its intended equilibrium outcome. Furthermore, it has been shown (e.g., see (Birnbaum et al., 2011; Cheung et al., 2018)) that in fundamental market models, the convergence of the tâtonnment process is highly dependent on several initial parameters and can therefore be slow, and even the centralized computation of market equilibria for certain markets can be computationally hard (Codenotti et al., 2006; Chen et al., 2017, 2009), thus in many cases *impractical* to compute.

Additionally, even under these idealized assumptions, the market equilibrium is geared towards very specific goals, namely fairness for the participants and economic efficiency given the set of chosen prices. However, from a more global perspective, there are several other important objectives which are not immediately captured by the market ecosystem, since it is based heavily on the economic principles of supply and demand. For example, one of the most pressing issues in modern societies is *sustainability* and the preservation of the Earth's natural resources.[2] Clearly, the extent to which natural resources are harvested is correlated with the potential monetary gains that the sellers of those resources can achieve in the market. With these 'exogenous' objectives being of paramount importance, it is only natural to assume that some form of intervention to the reign of free markets is not only inevitable, but also fully justified.

A clear-cut way of imposing some institutional or governmental control on the outcomes of the markets is via taxation on products and sales. Taxes can indeed be an effective measure, but they also sometimes have adverse effects, such as driving businesses away from the local markets, and often come in contrast with the principles of free markets. In many scenarios, the ideal means of intervention would be less intrusive, simultaneously

[1] E.g., see http://news.bbc.co.uk/1/hi/business/7132108.stm or https://fortune.com/2015/06/30/apple-conspired-with-book-publishers-appeals-court-confirms/.

[2] E.g., see UN's sustainable development goals https://www.un.org/sustainabledevelopment/, or OECD's 25 Climate Actions https://www.oecd.org/environment/25-climate-actions.pdf.

catering to the exogenous objectives and salvaging some of the attractive properties of the market equilibrium. Additionally, we would like these means to be fully effective in a world where the market participants are *learning agents*.

Learning agents have become ubiquitous in socio-economical and socio-ecological systems in recent years (e.g., (Danassis et al., 2021a; Zheng et al., 2020; Yang et al., 2020)). This has led to the emergence of machina economica (Parkes and Wellman, 2015), an approximate counterpart of homo economicus – the perfectly rational agent of neoclassical economics – given computational barriers and the lack of common knowledge. For example, with the emergence of machine learning, it has been observed (e.g., see (Tardos, 2019)) that enterprises use learning agents as forms of bounded rationality (Rubinstein and Dalgaard, 1998) (see also (Abel, 2019)). The success of multi-agent deep reinforcement learning has led to a growing interest in modeling machinae economicae agents as *independent* deep reinforcement learning agents that need to interact, learn, cooperate, coordinate, and compete with other learning agents in ever more complex, non-stationary environments (e.g., (Danassis et al., 2021a; Yang et al., 2020; Perolat et al., 2017; Leibo et al., 2017; Peysakhovich and Lerer, 2018a; Hughes et al., 2018; Peysakhovich and Lerer, 2018b; Jaques et al., 2019; Wang et al., 2019; Lupu and Precup, 2020; Koster et al., 2020)). Reinforcement learning is also considered a candidate theory of animal habit learning (Leibo et al., 2017; Niv, 2009). Moreover, it does not make use of any economic modeling assumptions, rather it learns from observational data alone, making it ideal for multi-objective optimization in complex domains.

In this new regime, it is unclear whether the market's 'laissez-faire' will lead to desirable outcomes, and how robust the notion of the market equilibrium is. Instead, we take an alternative approach, using deep reinforcement learning for policy making. In particular, we study the *emergent behaviors* as a group of deep learners interact in a complex and realistic market. Our market model consists of all the established market parameters (i.e., a dynamic set of buyers and sellers, endowments, and utilities), as well as a realistic exogenous common-pool resource appropriation component, which exhibits properties related to the tragedy of the commons (Hardin, 1968). In our model, both the pricing policy and the harvesting behaviors are learned *simultaneously*. Neither the policy maker nor the harvesters have prior knowledge / assumptions of domain dynamics or economic theory, and every agent only makes use of information that it can individually observe. Our policymaker can be used to optimize any desired social outcome allowing us to *tune* the prices with regard to diverse objectives such as sustainability and resource wastefulness, fairness, buyers' and sellers' welfare, etc.

9.2.1 Chapter Contributions

The main contributions of this chapter are:

1. **We propose a practical approach to computing market prices and allocations via a *deep reinforcement learning policymaker* agent**, that allows us to *tune* the prices with regard to diverse objectives such as sustainability and resource wastefulness, fairness and buyers' and sellers' welfare.

2. **We introduced a novel multi-agent socio-economic environment** combining established principles of competitive markets with the challenges of resource scarcity and the tragedy of the commons.

3. **We provide a thorough (quantitative & qualitative) analysis** on the learned policies and demonstrate that they can achieve significant improvements over the market equilibrium benchmark for several objectives, while maintaining comparable performance for the rest. As a highlight of our results, we show that our policymaker fares notably better in terms of sustainability of resources, essentially without compromising any of the remaining objectives.

Given that it is often quite hard to experiment with real-world pricing policies, traditional work in economics often results to simplifying assumptions which are hard to validate. Our approach provides an alternative route, enabling experimentation (via tuning of the parameters and simulating the multi-agent environment) to find the best possible policies.

9.2.2 Discussion and Related Work

The origins of competitive market theory date back to the late 1800s and the pioneering ideas of Walras (Walras, 1874) and Fisher (Fisher, 1892; Brainard and Scarf, 2005). Most instrumental in assembling these ideas into a cohesive theory was the work in economics in the mid-1900s, most notably by Arrow and Debreu (Arrow and Debreu, 1954), who defined and studied what we today know as the standard, most general model of competitive markets, and proved the existence of a market equilibrium under mild assumptions. The market that we consider in this work is a special case of the Arrow-Debreu model, due to Fisher (Fisher, 1892; Brainard and Scarf, 2005), where the market participants are divided into buyers and sellers, and buyers do not have intrinsic value for money, but rather use money as a means of facilitating the trade. This so-called 'Fisher market' model has been extensively studied in economics but also in computer science, in terms of computation and convergence to equilibria (Jain and Vazirani, 2010; Bei et al., 2016; Chakrabarty et al., 2006; Zhang, 2011; Dvijotham et al., 2020; Cheung et al., 2018). We chose the (linear) Fisher market as our benchmark because, contrary to the case of general Arrow-Debreu markets, computing a market equilibrium for this market can be done in polynomial time via convex programming (see Section 9.3.2 for the details). We also remark that while similar in spirit, the market equilibrium is a different notion from the well-known notion of the Nash equilibrium (Nash et al., 1950);

the former is a stable point of the market supply and demand adjustment, whereas the latter is a stable point of the participants' strategic play. In particular, the classic market equilibrium results assume that agents are *not strategic* and therefore do not attempt to influence the prices of the markets (i.e., they are price-takers). It has been shown that in the presence of (perfectly rational) strategic agents, the outcome of the market can be fundamentally different from the intended outcome of the market equilibrium (Brânzei et al., 2014; Adsul et al., 2010; Chen et al., 2016).

As discussed in the introduction, the last few years have witnessed a shift towards bounded rationality models; most prominently viewing rational decisions via the lens of machine learning (Nekipelov et al., 2015; Cai et al., 2018) and in particular reinforcement learning. Our work is one of the first to design a policy maker via deep reinforcement learning in economic environments. There is some recent work that has adopted a similar agenda, but on markedly different domains and using different approaches. Duetting et al. (2019) and Shen et al. (2019) consider the problem of finding optimal prices in revenue-maximizing auctions via deep reinforcement learning, and Cai et al. (2018) design a policy maker for impression allocation in e-commerce platforms against learning agents. Very recently, Zheng et al. (2020) consider an abstract socio-economic domain via the lens of deep reinforcement learning for policy making, focusing on taxation as a means of institutional intervention, rather than the adjustment of the prices.

Moving on to the common-pool resource appropriation component of our work, there has been great interest recently in Common-Pool Resource (CPR) appropriation problems (and more generally, social dilemmas (Kollock, 1998)) as an application domain for Multi-agent Deep Reinforcement Learning (MADRL) (Danassis et al., 2021a; Perolat et al., 2017; Leibo et al., 2017; Peysakhovich and Lerer, 2018a; Hughes et al., 2018; Peysakhovich and Lerer, 2018b; Jaques et al., 2019; Wang et al., 2019; Lupu and Precup, 2020; Koster et al., 2020). CPR problems offer complex environment dynamics and relate to real-world socio-ecological systems. Using deep reinforcement learning to address sustainability problems was done recently by Danassis et al. (2021a), who were also the first to introduce the realistic common fishery appropriation environment – based on bio-economic models of commercial fisheries – that we also employ in this chapter (see also Chapter 8). We have extended this model to deal with multiple resources, and harvesters with diverse skill levels, as we explain in Section 9.3.1.

In terms of the methodology, our work falls broadly into the following three categories: Reward shaping (Hughes et al., 2018; Jaques et al., 2019; Peysakhovich and Lerer, 2018b), which refers to adding a term to the extrinsic reward an agent receives from the environment, opponent shaping (Koster et al., 2020; Lupu and Precup, 2020; Perolat et al., 2017), which refers to manipulating the opponent (by e.g., sharing rewards, punishments, or adapting your own actions), and automated mechanism design (Baumann et al., 2020; Cai et al., 2018; Duetting et al., 2019), where an an external agent distributes additional rewards and punishments to promote desirable objectives on a population of artificial

learners.

9.3 Environment Models

In this section we provide a detailed description of our complex economic model. It consists of (i) a common-pool resource appropriation game – where a group of appropriators compete over the harvesting of a set of common resources and which exhibits properties related to the tragedy of the commons (Hardin, 1968) and the challenge of *sustainability* – and (ii) a complex and realistic market (with a dynamic set of buyers and sellers, endowments, and utilities), where the appropriators sell their harvest.

9.3.1 The Common Fishery Model

In this chapter, we adopt the common fishery model introduced in Chapter 8, which is based on an abstracted bio-economic model for *real-world* commercial fisheries (Clark, 2006; Diekert, 2012). It is important to note that we chose this environment due to its *complex dynamics*, but the proposed approach can be employed in any market and for any resources, not just fisheries. We have extended the model presented in Chapter 8 to account for multiple resources, and harvesters with varying skill levels. The model describes the dynamics of the stock of a set of common-pool renewable resources, as a group of appropriators harvest over time. The harvest depends on (i) the effort exerted by the agents, and (ii) the ease of harvesting that particular resource at that point of time, which depends on its stock level. The stock replenishes over time with a rate dependent on the current stock level.

More formally, let $\mathcal{N} = \{1, \dots, N\}$ denote the set of appropriators (harvesters) and $\mathcal{R} = \{1, \dots, R\}$ the set of resources. Let $\boldsymbol{\eta}_n = [\eta_{n,1}, \dots, \eta_{n,r}, \dots, \eta_{n,R}]$, where $\eta_{n,r} \in [0, 1]$ denotes the skill[3] (competence) of harvester n for harvesting resource r. At each time-step t, every agent exerts a vector of efforts $\boldsymbol{\phi}_{n,t} = [\phi_{n,1,t}, \dots, \phi_{n,r,t}, \dots, \phi_{n,R,t}]$, where $\phi_{n,r,t} \in [0, \Phi_{max}]$ is the effort exerted to harvest resource r.

Let $\boldsymbol{\varepsilon}_{n,t} = \boldsymbol{\phi}_{n,t} \cdot \boldsymbol{\eta}_n = [\varepsilon_{n,1,t}, \dots, \varepsilon_{n,r,t}, \dots, \varepsilon_{n,R,t}]$ denote the 'effective effort', and $E_{r,t} = \sum_{n \in \mathcal{N}} \epsilon_{n,r,t}$ the total effort exerted by all the harvesters at resource r at time-step t. Then, the total harvest of resource r is given by Equation 9.1, where $s_{r,t} \in [0, \infty)$ denotes the stock level at time-step t, $q_r(\cdot)$ denotes the catchability coefficient (Equation 9.2), and S_r^{eq} is the equilibrium stock of the resource.

$$H_r(E_{r,t}, s_{r,t}) = \begin{cases} q_r(s_{r,t})E_{r,t} & \text{, if } q_r(s_{r,t})E_{r,t} \leq s_{r,t} \\ s_{r,t} & \text{, otherwise} \end{cases} \tag{9.1}$$

[3] In our model $\eta_{n,r}$ does not depend on time, but one can consider agents that increase their skill level and become more efficient as they harvest a particular resource. Moreover one can introduce social castes and consider the problem of social mobility.

$$q_r(x) = \begin{cases} \frac{x}{2S_r^{eq}} & \text{, if } x \leq 2S_r^{eq} \\ 1 & \text{, otherwise} \end{cases} \tag{9.2}$$

Each environment can only sustain a finite amount of stock. If left unharvested, the stock will stabilize at S_r^{eq}. Note also that $q_r(\cdot)$, and therefore $H_r(\cdot)$, are proportional to the current stock, i.e., the higher the stock, the larger the harvest for the same total effort. The stock dynamics of each resource are governed by Equation 9.3, where $F(\cdot)$ is the spawner-recruit function (Equation 9.4) which governs the natural growth of the resource, and g_r is the growth rate.[4]

$$s_{r,t+1} = F(s_{r,t} - H_r(E_{r,t}, s_{r,t})) \tag{9.3}$$

$$F(x) = xe^{g_r(1-\frac{x}{S_r^{eq}})} \tag{9.4}$$

We assume that the individual harvest is proportional to the exerted effective effort (Equation 9.5), and the revenue of each appropriator is given by Equation 9.6, where $p_{r,t}$ is the price (\$ per unit of resource), and $c_{n,t}$ is the cost (\$) of harvesting (e.g., operational cost, taxes, etc.). Here lies the 'tragedy': the benefits from harvesting are private ($p_{r,t}h_{n,r,t}(\cdot)$), but the loss is borne by all (in terms of a reduced stock, see Equation 9.3).

$$h_{n,r,t}(\varepsilon_{n,r,t}, s_{r,t}) = \frac{\varepsilon_{n,r,t}}{E_{r,t}} H_r(E_{r,t}, s_{r,t}) \tag{9.5}$$

$$u_{n,r,t}(\varepsilon_{n,r,t}, s_{r,t}) = p_{r,t}h_{n,r,t}(\varepsilon_{n,r,t}, s_{r,t}) - c_{n,t} \tag{9.6}$$

Having a realistic environment that exhibits 'the tragedy of the commons' and the challenge of sustainability is not only important for the sake of realism, but it can potentially drastically impede the learning process. The benefits of harvesting can lead to greedy agents, which in turn deplete the resources *early* in the episode. This will result in short episodes, limiting the learning per episode.[5]

9.3.2 The Fisher Market Model

In a Fisher market there is a set of buyers $\mathcal{B} = \{1, \ldots, B\}$ and a set of divisible goods (resources) $\mathcal{R} = \{1, \ldots, R\}$, sold by one or multiple sellers. Every seller brings to the market a quantity of each good, with e_r denoting the total quantity of good $r \in \mathcal{R}$

[4] According to Section 8.3.1, to avoid highly skewed growth models and unstable environments, $g_r \in [-W(-1/(2e)), -W_{-1}(-1/(2e))] \approx [0.232, 2.678]$, where $W_k(\cdot)$ is the Lambert W function.

[5] In contrast to alternative common-pool resource appropriation games in the literature (e.g., (Zheng et al., 2020)) where resources re-spawn after being depleted.

brought collectively by the sellers. Every buyer brings a monetary endowment, or simply a *budget* of β_b, for $b \in \mathcal{B}$. Additionally, every buyer b has a *valuation* $v_{b,r}$ for each unit of good r. An allocation $\mathbf{x}$ is an $|\mathcal{B}| \times |\mathcal{R}|$ matrix, where $x_{b,r}$ denotes the amount of good r that is allocated to buyer b. In a feasible allocation, it holds that $\sum_{b \in \mathcal{B}} x_{b,r} \leq e_r$, for any good r. We will consider linear Fisher markets, where the *utility* of a buyer given allocation $\mathbf{x}$ is defined as:

$$u_b(\mathbf{x}) = \sum_{r \in \mathcal{R}} x_{b,r} v_{b,r} \tag{9.7}$$

These markets are also often called *Eisenberg-Gale Markets*[6] (Eisenberg and Gale, 1959). In such markets a *(competitive) market equilibrium* is a pair $(\mathbf{x}, \boldsymbol{p})$ of an allocation and a vector of prices, one for each good, such that at these prices each buyer is allocated a utility-maximizing bundle of goods, the budgets are entirely spent, and the goods are entirely sold; the latter condition is typically referred to as *market clearance*. For Eisenberg-Gale markets in particular, a market equilibrium can be found as the solution to the following convex optimization program:

$$\begin{aligned} \max \quad & \sum_{b \in \mathcal{B}} \beta_b \cdot \log(u_b) \\ s.t. \quad & u_b = \sum_{r \in \mathcal{R}} v_{b,r} \cdot x_{b,r}, \quad \forall\, b \in \mathcal{B} \\ & \sum_{b \in \mathcal{B}} x_{b,r} \leq e_r, \quad \forall\, r \in \mathcal{R} \\ & x_{b,r} \geq 0, \quad \forall\, b \in \mathcal{B}, \quad r \in \mathcal{R} \end{aligned} \tag{9.8}$$

While the prices do not strictly appear in this formulation, they can be recovered as the Lagrangian multipliers for the second set of constraints (the feasibility constraints of the good supply). Given the above formulation, a market equilibrium in a Fisher market always exists and it can also be computed in polynomial time. In fact, there are also combinatorial algorithms for equilibrium computation in Fisher markets, e.g., see (Jain and Vazirani, 2010; Bei et al., 2016; Chakrabarty et al., 2006).

9.4 Simulation Setup

9.4.1 Common Fishery

We simulated an environment with 8 harvesters, 8 buyers, and 4 resources ($N = 8, R = 4, B = 8$). We set the maximum effort at $\Phi_{max} = 1$, the growth rate at $g_r = 1$, and the initial population at $s_0 = S_r^{eq}$ (i.e., the stock starts from the equilibrium population), for every resource $r \in \mathcal{R}$. The findings of Chapter 8 provide a guide on the selection of the

[6] Strictly speaking, the term 'Eisenberg-Gale Market' is often used to refer to Fisher markets with CES utility functions, which are a superclass of the linear utility functions that we consider here.

S_r^{eq} values. Specifically, we set $S_r^{eq} = M_s K N$, where $K = (e^{g_r}\Phi_{max})/(2(e^{g_r}-1)) \approx 0.79$ is a constant, and $M_s \in \mathbb{R}^+$ is a multiplier that adjusts the scarcity of the resource (difficulty of the problem). For $M_s = 1$ the resource will not get depleted, even if all agents harvest at maximum effort.[7] We simulated two scenarios, one with $M_s = 0.8$, and a *scarce resources* scenario with $M_s = 0.45$.[8]

9.4.2 Harvesters

We set the skill level $\eta_{n,r} = 0.5$ for all agents and resources, except for one resource for each agent, specifically $\eta_{n,r} = 1$ if $n = r$. Finally, we assume no cost in harvesting, i.e., $c_{n,t} = 0$, $\forall n \in \mathcal{N}$, $\forall t$.

9.4.3 Buyers

Every time-step, a new set of buyers appears at the market, with budgets and valuations drawn uniformly at random on $[0, 1]$.

To allocate resources to buyers (for the cases where the price vector $\boldsymbol{p}$ is computed by the policymaker, i.e., the non-market equilibrium scenario), we solve the constraint optimization program of (9.9) that assigns each buyer a bundle based on their budget constraints, aiming to maximize the social welfare of the buyers:

$$\begin{aligned} \max \quad & \sum_{b\in\mathcal{B}} u_b(\mathbf{x}) \\ s.t. \quad & u_b(\mathbf{x}) = \sum_{r\in\mathcal{R}} x_{b,r} v_{b,r}, \quad \forall\, b \in \mathcal{B} \\ & \sum_{r\in\mathcal{R}} x_{b,r} p_r \le \beta_b, \quad \forall\, b \in \mathcal{B} \\ & \sum_{b\in\mathcal{B}} x_{b,r} \le e_r, \quad \forall\, r \in \mathcal{R} \\ & x_{b,r} \ge 0, \quad \forall\, b \in \mathcal{B}, \quad r \in \mathcal{R} \end{aligned} \tag{9.9}$$

9.4.4 Multi-Agent Deep Reinforcement Learning

We consider a decentralized multi-agent learning scenario in a partially observable general-sum Markov game (e.g., (Shapley, 1953; Littman, 1994)), where each agent is modeled as an independent deep reinforcement learning agent. At each time-step, agents take actions based on a partial observation of the state space, and receive an individual reward. See Section 7.1 for a detailed description.

[7] Yet, the problem of coordination remains far from trivial; see Chapter 8.

[8] In the simulations of Chapter 8, for $N = 8$ and $M_s \le 0.4$, the agents failed to find a sustainable strategy, and always depleted the resource.

9.4.5 Harvester Architecture

Each agent uses a two-layer (64 neurons each) neural network for the policy approximation. The input (observation $o^n = \mathcal{O}^n(S)$) is a tuple $\langle \boldsymbol{p}_{t-1}, \boldsymbol{\phi}_{n,t-1}, \boldsymbol{u}_{n,t-1}(\cdot) \rangle$ consisting of the vector of prices for the resources, the vector of individual effort exerted for every resource, and reward (cumulative out of all the resources) obtained in the previous time-step. The output is a vector of continuous action values $a_t = \boldsymbol{\phi}_{n,t} \in [0, \Phi_{max}]$ specifying the current effort level to exert for harvesting each resource. The reward received from the environment corresponds to the revenue, i.e., $r^n(\sigma_t, \boldsymbol{a}_t) = \sum_{r \in \mathcal{R}} u_{n,r,t}(\varepsilon_{n,r,t}, s_{r,t})$.

9.4.6 Policymaker Architecture

The policymaker also uses a two-layer (64 neurons each) neural network for the policy approximation. The input is a tuple $\langle \boldsymbol{\varepsilon}_t, \boldsymbol{s}_t, \boldsymbol{\beta_t}, G(\boldsymbol{v}_t) \rangle$, where $\boldsymbol{\varepsilon}_t$ is the efforts exerted by all the harvesters for all the resources, $\boldsymbol{s}_t$ is the current stock level of each resource, $\boldsymbol{\beta_t}$ is the budgets of the current set of buyers (recall that a random set of buyers appears at the market at each time-step), and finally, $G(\boldsymbol{v}_t)$ are the valuations of the buyers, obfuscated by a function $G(\cdot)$. The output is a vector of continuous action values $a_t = \boldsymbol{p}_t \in [0, \infty]$ that corresponds to the prices.

Valuation Obfuscation

To test the robustness of our policymaker in more realistic scenarios, we considered the case where the buyer's valuations are *obfuscated*. To put this into context, note that one of the idealized assumptions that allows the market equilibrium to be computed centrally is that all the information of the market is *completely and accurately* known. For good supplies and budgets, this assumption is reasonable, as these are typically observable or inferable, and qualify as 'hard' information (Liberti and Petersen, 2019) (see also (Brânzei and Filos-Ratsikas, 2019)). In contrast, the valuations of the agents are 'soft' information; they are hard to elicit, since they are expressed on a cardinal scale, and are possibly even accurately unknown to the agents themselves. The literature on computational social choice theory (Brandt et al., 2016) has been concerned with the effect of limited or noisy valuation information on the desired outcomes of a system.

We considered three different obfuscation functions for the buyers' valuations: (i) the identity function $G(x) = x$ (no obfuscation) – which we used in the majority of the simulations – (ii) a function that splits $[0, 1]$ into k bins, and each valuation value is replaced by the midpoint of the bin interval (average value of the endpoints), and (iii) a function that adds uniform noise on $(0, y)$, i.e., $G(x, y) = x + \mathcal{U}(0, y)$.

The bins approach corresponds to the case where the agents are not asked to provide accurate cardinal values, but instead they provide scores that somehow encode their

actual values. As the literature of the distortion in computational social choice suggests, such an elicitation device is cognitively much more conceivable (see (Anshelevich et al., 2021) and references therein). The added noise approach corresponds to the case where agents are uncertain about their own values, so they end up reporting noisy estimates of their true value. This approach is clearly reminiscent of the literature on noisy estimates of a ground truth, pioneered by Mallows (1957) but in fact dating back to the works of Marquis de Condorcet, more than two centuries ago.

Multi-objective Optimization

Finally, the policymaker's reward is the weighted average of the desired objectives, specifically:

$$w_h \frac{1}{|\mathcal{N}|} \sum_{n \in \mathcal{N}} \boldsymbol{u}_{n,t}(\cdot) + w_b \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} \boldsymbol{u}_{b,t}(\cdot) + w_s \min_{r \in \mathcal{R}} (\min(s_{r,t} - S_r^{eq}, 0)) + w_f Fair(\mathbf{x})$$

where w_h, w_b, w_s, and $w_f \in [0, 1]$ correspond to the weights for the harvesters' social welfare objective (sum of utilities, $\sum_{n \in \mathcal{N}} \boldsymbol{u}_{n,t}(\cdot)$), the buyers' social welfare objective (sum of utilities, $\sum_{b \in \mathcal{B}} \boldsymbol{u}_{b,t}(\cdot)$), the sustainability objective (defined in this work as the maximum negative deviation from the equilibrium stock, $\min_{r \in \mathcal{R}} (\min(s_{r,t} - S_r^{eq}, 0))$), and the fairness objective ($Fair(\mathbf{x})$) – which we evaluate using three different well-established fairness indices (see Section 9.4.9). It is important to note that the proposed technique is not limited to our choice of objectives; rather it can be used for *any combination of objectives*.

9.4.7 Learning Algorithm

The policies for all agents (harvesters and the policymaker) are trained using the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017). PPO was chosen because it avoids large policy updates, ensuring a smoother training, and avoiding catastrophic failures. As a reminder, the buyers are not learning agents; see Section 9.4.3.

9.4.8 Reproducibility and Reporting of Results

Reproducibility

Reproducibility is a major challenge in (MA)DRL due to different sources of stochasticity, e.g., hyper-parameters, model architecture, implementation details, etc. (Henderson et al., 2018; Hernandez-Leal et al., 2019; Engstrom et al., 2020). Recent work has demonstrated that code-level optimizations play an important role in performance, both in terms of achieved reward and underlying algorithmic behavior (Engstrom et al., 2020). To

Table 9.1: List of hyper-parameters.

Parameter	Value
Learning Rate (α)	0.0001
Clipping Parameter	0.3
Value Function Clipping Parameter	10.0
KL Target	0.01
Discount Factor (γ)	0.99
GAE Parameter Lambda	1.0
Value Function Loss Coefficient	1.0
Entropy Coefficient	0.0

minimize those sources of stochasticity – and given that the focus of this work is in the performance of the introduced policymaker and not of the training algorithm – we opted to use RLlib[9] as our implementation framework. All the hyper-parameters were left to the default values specified in Ray and RLlib.[10] For completeness, Table 9.1 presents a list of the most relevant of them.[11]

Termination Condition

An episode terminates when either (a) a fixed number of time-steps $T_{max} = 500$ is reached, or (b) any of the resources gets depleted, i.e., the stock falls below a threshold $\delta = 10^{-4}$. We trained our agents for 2400 episodes.

Statistical Significance

All simulations were *repeated* 8 *times*. The graphs depict the average values over those 8 trials, and the shaded area represents one standard deviation of uncertainty. The reported numerical results (Table 9.2) are the average values of the last 400 episodes over those trials. (MA)DRL also lacks common practices for statistical testing (Henderson et al., 2018; Hernandez-Leal et al., 2019). In this work, we opted to use the Student's T-test (Student, 1908) due to it's robustness (Colas et al., 2019); p-values can be found in Appendix C. All of the reported results that improve the baseline have p-values < 0.05.

9.4.9 Fairness Metrics

We employed three well established fairness metrics: the Jain index (Jain et al., 1998), the Gini coefficient (Gini, 1912), and the Atkinson index (Atkinson, 1970) (see Section

[9] RLlib (https://docs.ray.io/en/latest/rllib.html) is an open-source library on top of Ray (https://docs.ray.io/en/latest/index.html) for Multi-Agent Deep Reinforcement Learning (Liang et al., 2017).

[10] See https://docs.ray.io/en/latest/rllib-algorithms.html#ppo.

[11] The source code can be found in https://github.com/panayiotisd.

2.5). For brevity and to improve readability, we only report results on the Jain index in the main text. Similar results were obtained for the other indices too. We refer the interested reader to Appendix C for the complete results.

9.5 Simulation Results

In what follows we study the effect – with regard to diverse objectives such as sustainability and resource wastefulness, fairness, buyers' and sellers' welfare, etc. – of introducing the proposed policymaker to our complex economic system, compared to having the market equilibrium prices (as given by solving the convex optimization program of (9.8)).

We evaluated the 'vanilla' policymaker, where $w_h = w_b = w_s = w_f = 1$, and four extreme cases where we optimize only one of the objectives, i.e., (i) $w_h = 1$ and $w_b = w_s = w_f = 0$, (ii) $w_b = 1$ and $w_h = w_s = w_f = 0$, (iii) $w_s = 1$ and $w_h = w_b = w_f = 0$, and (iv) $w_f = 1$ and $w_h = w_b = w_s = 0$. The latter offers clear-cut results, but – as we will show in Section 9.5.2 – it can potentially lead to adverse effects. In practice, use of simulations can enable the testing of economic policies at *large-scale*, and the the ability to evaluate a range of different parameters, allowing the *designer to ultimately select the weights that optimize the desired combination of objectives.*

9.5.1 Comparing the 'Vanilla' Policymaker to the Market Equilibrium Prices (MEP)

Figures 9.1a and 9.1b depict the per-harvester mean reward and per-buyer mean utility, respectively, while rows 1 and 2 of Table 9.2 show the relative difference of the achieved social welfare (sum of utilities), as compared to the market equilibrium prices (MEP). The vanilla policymaker (blue line in the figures and first column of the table) achieves results comparable to the equilibrium prices in both cases, with a loss of only $\approx 7\%$ of social welfare. Similar results are achieved in the case of fairness – both for the sellers and buyers (last two rows of Table 9.2) – with both the MEP and the policymaker achieving a fair allocation (Jain index ≥ 0.98[12]). This is particularly important, as the market equilibrium is geared by design to optimize the aforementioned metrics, i.e., fairness for the participants and economic efficiency. Notably, the vanilla policymaker significantly outperforms the MEP when it comes to sustainability, as we describe in more detail in Section 9.5.4.

[12]The higher the better; an allocation is considered totally fair iff the Jain index is 1. See Appendix C for the other metrics.

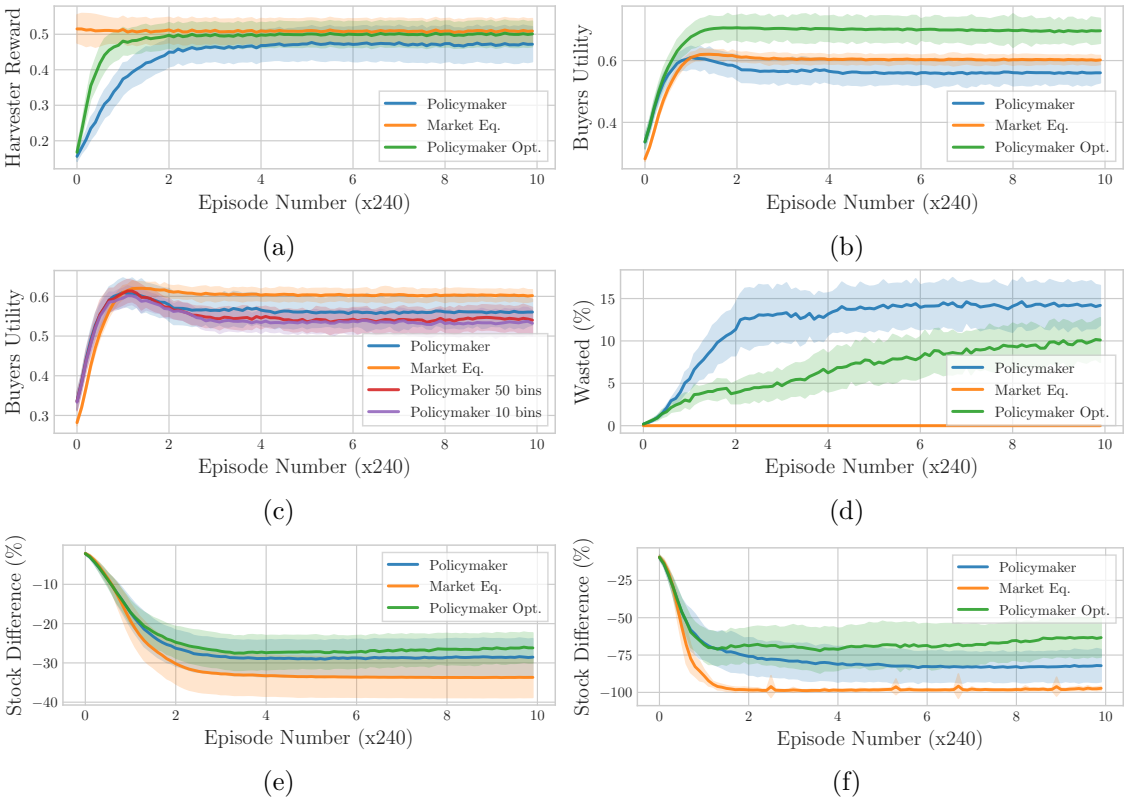


Figure 9.1: Evolution of several metrics over the number of training episodes. The orange line is the baseline, where the prices are the market equilibrium prices. The blue line refers to the vanilla policymaker where each objective in the reward function has the same weight (see Section 9.5). The green line refers to the policymaker that only optimizes the specific objective of each figure (i.e., in 9.1a we set $w_h = 1$ and $w_b = w_s = w_f = 0$, in 9.1b we set $w_b = 1$ and $w_h = w_s = w_f = 0$, and in 9.1d, 9.1e, and 9.1f we set $w_s = 1$ and $w_h = w_b = w_f = 0$). The red and purple lines in 9.1c refer to a policymaker with obfuscated valuations (see Section 9.4.6). Finally, in 9.1f we have a scarce resource setting (i.e., $M_s = 0.45$, see Section 9.4.1). Shaded areas represents one standard deviation.

9.5.2 Harvesters' Revenue, Buyers' Utility, and Social Welfare (SW)

Optimizing specifically for the harvesters' revenue or the buyers' utility (setting $w_h = 1$ or $w_b = 1$, respectively, and the remaining weights to 0), results in the policymaker closing the gap, or even significantly outperforming the MEP (green line in Figures 9.1a and 9.1b and second and third column of Table 9.2). The harvesters' Social Welfare (SW) improves from -7.44% to -1.74%, while the buyers' SW exhibits a dramatic improvement from -7% to $+15.42\%$.

It is important to note, though, that contrary to the case of optimizing the sustainability or the fairness, exclusively optimizing the harvesters' SW has detrimental effects to the the buyers' SW and vice versa (see Table 9.2). This is because these two objectives

Table 9.2: Numerical results of the last 400 episodes of each training trial (averaged over the 8 trials). Each column represents the relative difference (%) of the particular configuration of the policymaker, as compared to the market equilibrium prices ($100(X_{\text{policymaker}} - Y_{\text{market eq.}})/Y_{\text{market eq.}}$), for each of the metrics presented in each row. The first column refers to the vanilla policymaker, where each objective in the reward function has the same weight (see Section 9.5), and each of the following 4 columns refers to a policymaker that only optimizes the specific objective in the title (having weight 0 for the rest). Finally, the last two columns refer to a vanilla policymaker with obfuscated valuations (valuations split into 50 and 10 bins respectively, see Section 9.4.6).

	Policymaker						
	vanilla	$w_h = 1$	$w_b = 1$	$w_s = 1$	$w_f = 1$	Noisy (50)	Noisy (10)
Harvesters' Social Welfare	-7.44	-1.74	-72.91	-31.37	-34.14	-11.35	-9.71
Buyers' Social Welfare	-7.01	-24.71	15.42	1.23	2.88	-9.73	-11.51
Stock Difference[13]	-15.30	-2.64	-10.58	-21.83	-12.99	-23.40	-21.73
Harvesters' Fairness	-0.61	-0.05	-0.64	-0.72	-0.14	-1.16	-1.04
Buyers' Fairness	-0.12	-0.18	-0.05	-0.09	-0.07	-0.27	-0.31

are somewhat orthogonal; low prices lead to high buyers SW but low harvesters SW, and vice versa (although money do not have an intrinsic value in Fisher markets). In this work we showcase the potential of a vanilla policymaker, and the extreme cases of optimizing just one objective; it is up the designer to ultimately select the weights that best serve the desired combination of objectives.

Finally, we report results on noisy buyers' valuations (see Section 9.4.6), split into 50 and 10 bins (last two columns of Table 9.2, and Figure 9.1c; see Appendix C for the rest). Noisy valuations lead to only a small drop in the buyers' and harvesters' SW ($\approx 2-4\%$), the fairness remains the same, while sustainability improves significantly (up to 8% compared to the vanilla policymaker). This comes to show that the policymaker is robust to noisy valuations, which are much easier and practical to elicit.

9.5.3 Fairness

The MEP are geared towards optimizing fairness; it is important to ensure that the introduction of the policymaker does not result in an exploiter-exploitee situation. All of the evaluated versions of the policymaker achieve a fair allocation (Jain index ≥ 0.98,[12] see Appendix C for the other metrics). The relative values (Table 9.2) show a consistent improvement when specifically optimizing for fairness ($w_f = 1$) but, in absolute terms, all versions actually result in fair allocations.

9.5.4 Sustainability

The question of sustainability in the use of common-pool resources constitutes a critical open problem. Individuals face strong incentives to appropriate, which results in *overuse* and even *depletion* of the resource. The partial observability and the inherent pitfalls of MADRL (e.g., non-stationarity, credit assignment, global exploration, etc. (Hernandez-Leal et al., 2019; Matignon et al., 2012; Wiegand and Jong, 2004)) further increase this challenge.

We measure sustainability as the maximum negative deviation from the equilibrium stock (see Section 9.4.6). The introduced policymaker results in the *emergence of significantly and consistently more sustainable harvesting strategies*. Figure 9.1e shows that the MEP maintain a population stock that is 34% below the equilibrium population (on average), while the policymaker is only 28.5%. Optimizing for sustainability ($w_s = 1$, green line) improves the difference to 26%.

More interesting is Figure 9.1f, where we simulate a *scarce resource* environment (see Section 9.4.1). In this setting, the introduction of the policymaker results in a *dramatic improvement in sustainability*. The MEP maintain a population stock that is 97.3% below the equilibrium population (on average), while the policymaker is 82.1% and optimizing for sustainability improves the difference to 63.3%; almost 35% improvement compared to MEP. In this setting, the MEP fail to result in a sustainable strategy and permanently *deplete* the resources in 9.79% of the episodes, with episodes lasting as low as 48 time-steps (out of 500). In contrast, the vanilla policymaker fails in 4.59% of the episodes (min episode length of 180 time-steps), and the version that optimizes sustainability fails in only 2.24% of the episodes (min episode length of 258 time-steps).

Importantly, optimizing for sustainability does not have detrimental effects to most other objectives, as seen in Table 9.2.[13] The harvesters' and buyers' fairness improve as well, and so does the buyers' welfare. Only the harvesters' welfare degrades; but, as mentioned, it is up the designer to ultimately select the weights that best serve the desired combination of objectives.

Finally, we also measured the percentage of wasted resources (harvested resources that remain unsold), see Figure 9.1d. Of course, by design, the MEP sell the entire harvest. Optimizing for sustainability results in a decrease of the wasted resources from 14% to 10% (blue vs. green line).

[13]Note that the stock difference has negative values (negative deviation from the equilibrium stock) thus, in this metric, large negative numbers are *in favor* of the policymaker.

9.6 Societal Impact

Our approach can actively facilitate social mobility, sustainability, and fairness. As a potential negative social impact, the introduction of learning agents in socio-economic systems might bring forth an 'arms-race' for the best means of production, which now shift from traditional, to computational resources and technological know-how. This can increase social inequality.

9.7 Chapter Conclusion

We proposed a practical approach to computing market prices and allocations via *deep reinforcement learning*, allowing us to optimize a host of diverse objectives such as sustainability and resource wastefulness, fairness, and buyers' and sellers' welfare. We evaluate our approach in a realistic environment that combines an established market model with a common-pool appropriation component, and we study the emergent behaviors as a group of deep learners interact. We demonstrate significant improvements, especially towards solving the challenge of sustainability of limited resources. Our work constitutes an important first step in studying markets composed of *learning* agents, which are becoming ubiquitous in recent years.

10 Concluding Remarks

In this thesis, we presented new tools, algorithms, and theory for allocation problems and resource sharing in multi-agent systems. Two common themes motivated our research: *scalability*, and providing *practical* solutions for real-world applications.

Starting with allocation problems, while weighted matching has been studied extensively in the related literature, there is still a lot of ground for innovative research, especially when viewing the problem under the prism of multi-agent systems, due to the three challenges that arise: (i) complexity, (ii) communication, and (iii) privacy. In the first part of the thesis (Part I), we set about tackling those challenges. We presented ALMA (and its variants); a constant time (Chapter 3), privacy-preserving (Chapter 4) algorithm for weighted matching in massively large multi-agent environments. ALMA is decentralized, requires no inter-agent communication, and agents observe only their own action/reward pairs, making it ideal for on-device implementation.

Our research shows that due to real-time constraints, cumbersome centralized algorithms often provide diminishing returns as the problem size grows; instead, lightweight, decentralized heuristics (like ALMA) might be more desirable for large-scale applications. We demonstrate this in a variety of real-world applications using real data. In the largest, and most comprehensive – to the best of our knowledge – study on ridesharing and dynamic vehicle relocation (Chapter 3, and Appendix A), ALMA performed on par with the best algorithms, across the board of our comprehensive set of objectives, despite the on-line nature and massive scale of the problem. Importantly, the privacy-preserving version, PALMA (which combines ALMA with the proposed Piecewise Local Differential Privacy), manages to outperform even the centralized optimal solution (with a state-of-the-art, problem-specific privacy mechanism based on geo-indistinguishability) in the ridesharing test-case. Our work shows that harnessing the potential of intelligent systems on a large scale does not have to compromise privacy. Another test-case with significant real-world impact and applicability is the meeting scheduling problem (Chapter 5, and Appendix B),

where ALMA – and ALMA-Learning, the variant of ALMA that incorporates learning to increase the social welfare and fairness – were able to remove the scalability obstacles that made earlier attempts fail. Finally, we evaluated a paper assignment problem (Chapter 4) using real-data, and demonstrated high social welfare under strict, worst-case privacy guarantees. Our comprehensive evaluation of ALMA includes a variety of synthetic benchmarks as well. Overall, we demonstrated that ALMA: (i) is able to reach high social welfare, (ii) is orders of magnitude faster than the centralized, optimal algorithm, and (iii) provides strong worst-case privacy guarantees.

Finally, we also studied allocation problems under a rationality constraint, and showed that following such simple rules can constitute an approximate subgame-perfect equilibrium (in a setting with indivisible resources with binary utilities and access to a common environmental signal), i.e., an accessing pattern where no agent has an incentive to deviate (Chapter 6).

In the second part of the thesis (Part II), we examined the emergent behaviors in a group of deep reinforcement learning agents harvesting common-pool resources, and applied simple interventions to steer the population to desirable states. We were able to improve cooperation in large-scale, low observability, and high-stakes environments, and sustainability in the use of the common-pool resources.

To begin with – in order to better understand the impact of self-interested appropriation – we introduced a realistic common-pool resource appropriation game for multi-agent coordination (Chapters 8, and 9) – based on an abstracted bio-economic model for commercial fisheries – that exhibits the tragedy of the commons, and a complex market model (Chapter 9) which consists of all the established market parameters (i.e., a dynamic set of buyers and sellers, endowments, and utilities).

In line with the theme of having simple and scalable approaches, we proposed a really simple – yet robust – technique for sustainable cooperation: allow agents to observe an arbitrary common signal from the environment in order to couple their policies and learn temporal harvesting conventions (Chapter 8). We demonstrated that agents can now, for example, learn to harvest in turns, and with varying efforts per signal value, or allow for fallow periods. The benefit is twofold: the agents learn to not only avoid depletion, but also to maintain a healthy stock which allows for large harvest and, thus, higher social welfare. The proposed approach requires no communication, no extrinsic incentive mechanism, and it does not change the underlying architecture or learning algorithm. It is also important to stress that we do not assume any a priori relation between the signal space and the problem at hand.

Finally, we developed a practical approach to computing market prices and allocations via a deep reinforcement learning policymaker agent (Chapter 9). Besides the obvious

advantage of its applicability in markets where the equilibrium prices can not be efficiently computed, we demonstrated that we are able to tune the prices with regard to diverse exogenous objectives such as the sustainability of the resources, resource wastefulness (both of which constitute critical open problems), fairness, and buyers' and sellers' welfare.

10.1 Future Work

There are several directions to be explored with regard to all of the aforementioned research topics. In what follows, I will briefly highlight some of them.

Approximation bounds: While ALMA has been shown to perform well in a wide variety of domains, it would be interesting to formally characterize the expected loss in the utilitarian, or the egalitarian welfare, or in fairness, compared to the optimal – with respect to each of the aforementioned objectives – solution. Additionally, selecting the optimal (or a 'good enough' depending on the domain) back-off and resource selection function remains an open question as well.

Rationality: ALMA is applicable in cooperative settings. As discussed in Chapter 6, a rational agent could keep selecting the most preferred resource until every other agent backs-off ('bully' strategy). In the same chapter, we showed how one can use ALMA in conjunction with an arbitrary, periodic environmental signal to devise an accessing strategy that constitutes an approximate subgame-perfect equilibrium, but only for binary utilities. It remains open for future work to investigate if ALMA can constitute a rational strategy under more general utility functions / settings. Accessing a resource incurs a cost for the agents, thus we believe the fast convergence of ALMA – which can outweigh the loss in utility – might play a key role in meeting this goal.

Social Conventions: Behavioral conventions are a fundamental part of human societies, and allow humans to routinely and robustly coordinate on a large scale, and under dynamic and unpredictable demand. In this thesis, we demonstrated that such simple conventions can provide significant improvements in several objectives (e.g., convergence speed, robustness, social welfare, etc.) for large-scale multi-agent systems, from the pre-defined altruistic convention of ALMA (Chapter 3) and the courteous convention of the CA^{3}NONY framework (Chapter 6), to the learned temporal ordering conventions observed in deep reinforcement learning agents harvesting a common-pool resource (Chapter 8). As such, an interesting line of research would be to further investigate the adoption of human conventions in multi-agent systems for coordination at scale, ad-hoc coordination, etc. Signaling and cheap talk – like the common environmental signals we employed for CA^{3}NONY and the harvesting of common fisheries – may be instrumental for the emergence of conventions.

Adopting Market Prices: In order to tackle real-world socio-economic problems

– such as the problem of sustainable production (Chapters 8, and 9) – we need the ability to design and test novel economic policies. However, in practice, it is often quite hard to experiment with real-world pricing policies. Traditional work in economics often results in simplifying assumptions that are hard to validate. Our approach (the deep reinforcement learning based policymaker of Chapter 9) provides an alternative route, enabling experimentation to find the best possible policies (via tuning of the parameters and simulating the multi-agent environment). Specifically, in this work, we compute market prices that maintain comparable performance to the market equilibrium prices, while allowing us to account for exogenous objectives, like the goal of sustainable production. Knowing such prices is an important tool by itself (just like being able to compute the market equilibrium prices, or, similarly, the Nash equilibrium of a game). For example, it allows us to study the emergent state of the system, and examine how closely the market reaches the market equilibrium prices (recall that the tâtonnment process for reaching a market equilibrium is highly dependent on several initial parameters). Most importantly, though, one can then nudge agents to move the system closer to some desirable equilibrium state. The latter question of nudging agents to adopt the prices computed by our policymaker agent has been left open for future work. Demonstrating empirically that such prices satisfy both the sellers' and buyers' objectives along with the exogenous ones – as we did – is the first step. Depending on the market, the government, for example, might be able to enforce prices, but a more reasonable approach would be to reach the desirable state through subsidies, or by influencing the supply/demand.

Learning Agents as a Rationality Model: Common knowledge and the perfectly rational agent of neoclassical economics can be really strong, unattainable assumptions in the real world. The emergence of machine learning has brought forth the use of learning agents as forms of bounded rationality, and the rise of machina economica – a counterpart of homo economicus given computational barriers. In this new regime, agents act based on observational data alone, without the use of any modeling assumptions. Deep reinforcement learning agents are ideal candidates for modeling such machinae economicae agents, able to interact, learn, cooperate, coordinate, and compete in ever more complex, non-stationary environments (where agents can be impacted by, and influence the learning process). Yet, the introduction of learning agents in socio-economic systems might bring forth an 'arms race' for the best means of production, which now shift from traditional to computational resources and technological know-how. With AI agents playing an ever greater role in our lives, and called upon to solve increasingly more complex tasks and to regulate many aspects of our social, economic, and technological world, it is important to study this new form of rationality, both from a theoretical standpoint – in, perhaps, simplified environments, and agent models – and empirically, using detailed, realistic environments, like our common fisheries model (Chapters 8, and 9).

Appendices Part

A Putting Ridesharing to the Test

A.1 Preface

This appendix is based on (Danassis et al., 2022b). Detailed individual contributions are listed in Section 3.1.

A.2 Introduction

The emergence and widespread use of *Mobility-on-Demand* systems in recent years has had a profound impact on urban transportation in a variety of ways. Amongst other advantages, these systems have the potential to mitigate congestion costs (such as commute times, fuel usage, accident propensity, etc.), enable marketplace optimization for both passengers and drivers, and provide great environmental benefits. A prominent such example is ridesharing[1]. Ridesharing however results in some passenger disruption as well, due to compromise in flexibility, increased travel time, and loss of privacy and convenience. Thus, in the core of any ridesharing platform lies the need for an efficient balance between the incentives[2] of the passengers, the drivers, and those of the platform.

Optimizing the usage of transportation resources is not an easy task, especially for cities like New York, with more than 13000 taxis and 270 ride requests per minute. For example, (Buchholz, 2018) estimates that 45000 customer requests remain unmet each day in New York, despite the fact that approximately 5000 taxis are vacant at any time. In fact, on aggregate, drivers spend about 47% of their time not serving any passengers (Buchholz, 2018). Moreover, up to 80% of the taxi rides in Manhattan could be shared by two riders, with only a few minutes increase in travel time (Alonso-Mora et al., 2017). A *more sophisticated matching policy* could mitigate these costs

[1] Throughout the chapter, we use the term 'ridesharing' to refer to passengers (potentially) using the same vehicle at the same time, also referred to as 'ridepooling' (Shaheen and Cohen, 2019).

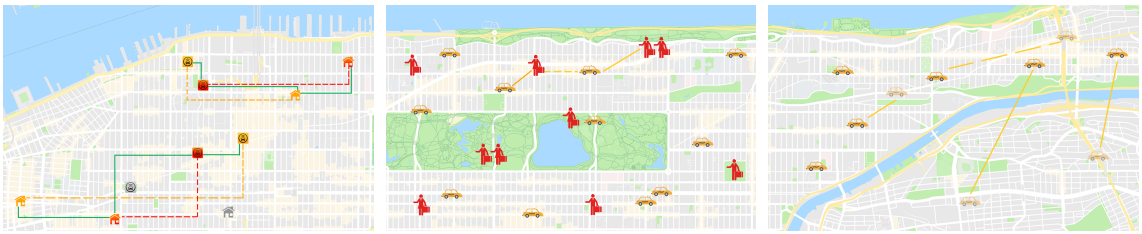
[2] We remark that the term 'incentive' is not used in the strict game-theoretic sense.

by better allocating available supply to demand. As a second example, *coordinated vehicle relocation* could also be employed to bridge the gap on the spatial supply/demand imbalance and improve passenger satisfaction and Quality of Service (QoS) metrics. Drivers often relocate to find passengers: 61.3% of trips begin in a different neighborhood than the drop-off location of the last passenger (Buchholz, 2018), yet currently drivers move without any coordinated search behavior, resulting in spatial search frictions.

Given the importance of the problem for transportation and the economy, it is not surprising that the related literature is populated with a plethora of papers, proposing different solutions along different axes, such as efficiency (Santi et al., 2014; Alonso-Mora et al., 2017; Agatz et al., 2011; Ashlagi et al., 2017; Huang et al., 2019; Bienkowski et al., 2018; Dickerson et al., 2018; Fagnant and Kockelman, 2018; Lokhandwala and Cai, 2018), platform revenue (Banerjee et al., 2017; Chen et al., 2019), driver incentives (Ma et al., 2019; Yuen et al., 2019; Garg and Nazerzadeh, 2020), fairness (Lesmana et al., 2019; Sühr et al., 2019; Xu and Xu, 2020; Nanda et al., 2020), reliability (Fielbaum and Alonso-Mora, 2020; Alonso-González et al., 2020), or analyzing the effects on sharing economies (Kooti et al., 2017; Jiang et al., 2018; Ghili and Kumar, 2020; Asadpour et al., 2020).

It is well-documented (e.g., (Lesmana et al., 2019)) that all these different desiderata are often contrasting (e.g., fairness vs. revenue), and therefore we should not expect a single algorithm for ridesharing to be superior for all of them; rather, the design of such algorithms should be contingent on the goals of the designer, and which of those properties they consider to be more important for the application at hand. Thus, we want a *flexible* and *adaptable* design, able to work best with respect to any set of such objectives with 'a few tweaks'.

To enable this, we propose a *modular approach to algorithm design in ridesharing*, in which an algorithm consists of three different *components*, namely (a) matching passengers with other passengers, (b) assigning rides to vehicles and (c) vehicle relocation, in which the taxis move, when they do not serve passengers, close to positions where requests are *expected to appear* in the near future. Each component can then be seen as a different (sub)-algorithm, and those algorithms can be appropriately chosen to be geared towards the specific objectives of the designer. As a matter of fact, our approach draws inspiration from several successful algorithms in the ridesharing literature, such as the well-known *High Capacity* algorithm of (Alonso-Mora et al., 2017), or the recent algorithm of (Riley et al., 2020), who can both be cast as examples of algorithms in this modular design setting.



(a) Request – Request (b) (Shared) Ride – Taxi (c) Vehicle Relocation

Figure A.1: The three components of a CAR.

A.2.1 Our Contributions

CARs

We initiate the *systematic* study of ***Component Algorithms for Ridesharing* (CARs)**. A CAR is an algorithm consisting of three sub-algorithms, each solving one of the following components of the ridesharing problem (Figure A.1).

- *Matching passengers to other passengers.* For this component, the underlying algorithmic problem is that of *Online Maximum Weight Matching*, where the "online" part stems from the fact that passenger requests appear at different points in time, and we have to *account for the future* when deciding which passengers to match. As such, we have a lot of classic as well as modern matching algorithms at our disposal.

- *Assigning rides to vehicles.* For this component, the underlying algorithmic problem can either be seen as an *Online Maximum Weight Bipartite Matching*, or as an instance of the *k-Taxi Problem* and by extension as the famous *k-Server problem* from the literature of online algorithms. Similarly to above, there is a large set of classic and modern solutions that one can plug-in as components for this part.

- *Vehicle Relocation.* For this component, the objective is to use *historical data* to *predict* the location of future requests and move idle taxis closer to those locations. From an algorithmic standpoint, this problem can be cast as either as *k-Facility Location* problem, concerned with the optimal placement of facilities (taxis) to minimize transportation costs, or as an *Online Maximum Weight Matching* problem on the *history of requests.*

Evaluation Platform

While several papers in the literature provide evaluations on realistic datasets, (e.g., see (Riley et al., 2020; Santi et al., 2014; Alonso-Mora et al., 2017; Agatz et al., 2011; Santos and Xavier, 2013; Danassis et al., 2019)), they either (a) only consider parts of the ridesharing problem and therefore do not propose end-to-end solutions, (b) only evaluate

a few newly-proposed algorithms against some basic baselines, (c) only consider a limited number of performance metrics, predominantly with regard to the overall efficiency and often without regard to QoS metrics, or (d) perform evaluations on a much smaller scale, thus not capturing the real-life complexity of the problem. On the contrary, *our work provides a comprehensive evaluation of a large number of proposed algorithms, over multiple different metrics, and for real-world scale, end-to-end problems.* Specifically:

- We meticulously design an experimental setting to *resemble reality as close as possible* in *every* aspect of the problem. To the best of our knowledge, this is the *first end-to-end experimental evaluation of this magnitude*, and could serve as a *common-ground* for evaluating future work in a setting designed to capture real-world challenges.
- We evaluate our CARs for a host of different objectives (12 metrics) related to *global efficiency*, *complexity*, *passenger*, *driver*, and *platform* incentives (see Table A.2).

Results

Applying the modular approach we advocated above, we design a large set of CARs, based on different classic and modern algorithms for the different components (14 in total, see Table A.1). The main take-away is the following:

- CARs based on off-line, in-batches maximum-weight matching approaches perform well on global efficiency and passenger related metrics.
- CARs based on k-server algorithms perform well on driver related metrics (e.g., the Balance algorithm (Manasse et al., 1990)).
- Lightweight CARs perform better in real-world, large-scale settings since real-time constraints dictate short planning windows which can diminish the benefit of cumbersome optimization techniques compared to myopic approaches.
- Simple, lightweight relocation schemes can significantly improve Quality of Service metrics by up to 50%.
- We identify a *scalable*, *on-device* CAR based on ALMA that performs well *across the board*.

Our findings provide convincing evidence to a ridesharing platform as to which combination of components would be most suitable for a given set of objectives.

A.3 Discussion and Related Work

The literature on ridesharing is rather extensive; here we only highlight the key algorithmic principles in our design of CARs.

The dynamic ridesharing – and the closely related *dynamic dial-a-ride* (see (Agatz et al., 2012)) – problem has drawn the attention of diverse disciplines over the past few years, from operations research to transportation engineering, and computer science. Solution approaches include constrained optimization (Qian et al., 2017; Simonetto et al., 2019; Agatz et al., 2011; Alonso-Mora et al., 2017; Riley et al., 2020), weighted matching (Ashlagi et al., 2017; Bei and Zhang, 2018; Dickerson et al., 2018; Zhao et al., 2019; Danassis et al., 2019), other heuristics (Qian et al., 2017; Santos and Xavier, 2015; Bathla et al., 2018; Lowalekar et al., 2019; Santos and Xavier, 2013; Pelzer et al., 2015; Gao et al., 2017; Shah et al., 2020), reinforcement learning (Guériau and Dusparic, 2018; Li et al., 2019; He and Shin, 2019), or model predictive control (Chen and Cassandras, 2019; Riley et al., 2020; Tsao et al., 2019), among others. We refer the interested reader to the following surveys (Agatz et al., 2012; Silwal et al., 2019; Furuhata et al., 2013; Ho et al., 2018; Mourad et al., 2019; Cordeau and Laporte, 2007) for a review on the optimization challenges, various algorithmic designs adopted over the years, a classification of existing ridesharing systems, models and algorithms for shared mobility, and finally models and solution methodologies for the dial-a-ride problem, respectively.

As we mentioned in the introduction, the key algorithmic components of ridesharing are the following. First, it is an *online* problem, as the decisions made at some point in time clearly affect the possible decisions in the future, and therefore the the literature of *online algorithms* and competitive analysis (Borodin and El-Yaniv, 2005; Manasse et al., 1988) offers clear-cut candidates for CARs. Second, all of the components can be seen as some type of *matching* both for bipartite graphs (for matching passengers with taxis, or idle taxis with 'future' requests) and for general graphs (for matching passengers to shared rides). In fact, several of the algorithms that have been proposed in the literature for the problem are for different variants of online matching.

Finally, ridesharing displays an inherent connection to the *k-taxi problem* (Coester and Koutsoupias, 2019; Buchbinder et al., 2020; Fiat et al., 1994; Kosoresow, 1996), which, in turn, is a generalization of the well-known *k-server problem* (Koutsoupias and Papadimitriou, 1995; Koutsoupias, 2009).[3] In the k-taxi problem, once a request appears (with a source and a destination), one of the k taxis at the platform's disposal must serve the request. Viewing shared rides (multiple passengers that have already been matched in a previous step) as requests, one can clearly apply the k-taxi (and k-server algorithms) to the ridesharing setting. Granted, the k-server algorithms have been designed to operate in a more challenging setting in which (a) the requests have to be served *immediately*, whereas normally there is some leeway in that regard, often at the expense of customer satisfaction, and (b) the positions of requests are typically *adversarially* chosen, rather than following some distribution, as is the case in reality. Despite those facts, the fundamental idea behind these algorithms is a pivotal part of

[3] In fact the latter two problems are quite closely connected, and algorithms for the k-server problem can be used to solve the k-taxi problem. See (Coester and Koutsoupias, 2019) for more details.

ridesharing, as it aims to *serve existing requests efficiently, but at the same time place the vehicles as well as possible to serve future requests.* This is also the main principle of the relocation strategies for idle taxis.

The algorithms that we consider are appropriate modifications of the most significant ones that have been proposed for the aforementioned key algorithmic primitives of the ridesharing problem, as well as heuristic approaches which are based on the same principles, but were specifically designed with the ridesharing application in mind. We emphasize that such modifications are needed, primarily because many of these algorithms were tailored for sub-problems of the ridesharing setting, and end-to-end solutions in the literature are rather scarce.

Much of the related work in the literature focuses on approaches that are inherently centralized and require knowledge of the full ridesharing network, which makes them rather computationally intensive. As an additional goal of our investigation, we would like to identify solutions that are lightweight, decentralized, and which ideally run *on-device*. Of course, some hybrid and decentralized approaches for the ridesharing problem have been proposed (e.g., (Simonetto et al., 2019; Guériau and Dusparic, 2018)), and several of the algorithms that we include in our experimental evaluation can be implemented in a decentralized manner (e.g., (Giordani et al., 2010; Ismail and Sun, 2017; Zavlanos et al., 2008; Bürger et al., 2012)), but that would typically require a larger amount of communication between the agents; in this case, the vehicles. As it turns out though, the ALMA algorithm (see Chapter 3), which we designed with precisely these objectives in mind (low computational complexity, scalability, and low communication cost), performs very well across the board with respect to our objectives.

The third component of our CARs is the relocation of idle taxis. Relocation is an important component of a successful ridesharing application. Many studies in shared mobility systems have shown that the adoption of a relocation strategy can help improve the system performance for their specific context (Guériau and Dusparic, 2018; Vosooghi et al., 2019; Martínez et al., 2017; Bélanger et al., 2016; Ruch et al., 2018; Alonso-Mora et al., 2017; Buchholz, 2018; Lioris et al., 2016; Spieser et al., 2014; Tsao et al., 2019; van Engelen et al., 2018; Wen et al., 2017; Wallar et al., 2018). Strategies include using a short window of known active requests (Alonso-Mora et al., 2017), historical demand (Guériau and Dusparic, 2018), or techniques to predict future demand (Spieser et al., 2016). Yet, relocation by nature increases vehicle travel distance, leading to undesirable consequences (economical, environmental, maintenance, management of human resources, etc.), thus a balance needs to be struck. Most of the employed relocation approaches are course-grained; the network is generally divided into several zones, blocks, etc. (Guériau and Dusparic, 2018; Vosooghi et al., 2019; Martínez et al., 2017) and the entities (e.g., the vehicles) move between the zones. However, compared to other shared mobility systems, dynamic ridesharing posses unique challenges, meaning that such coarse-grained approaches are not appropriate: most of them are centralized – thus computationally intensive and not

scalable –, they might not take into account the behavior of other drivers, potentially leading to over-saturation of high demand areas, and, most importantly, they are *slow to adapt* to the highly dynamic nature of the problem (e.g., responding to high demand generated by a concert, or the fact that vehicles remain free for only a few minutes at a time). The problem clearly calls for fine-grained solutions, yet such approaches in the literature are still rather scarce. In this work, we employ such a fine-grained relocation scheme (similarly to (Alonso-Mora et al., 2017)), based on matching between the idle taxis and the *potential* requests, which is better suited for the problem at hand.

Relocation can be either viewed as the k-center or k-Facility Location Problem (Guha and Khuller, 1999), or as an *Online Maximum Weight Matching* problem on the *history* of requests. Given the high complexity of the former problems (they are both NP-hard, in fact, APX-hard (Hsu and Nemhauser, 1979; Feder and Greene, 1988)), we have opted for the latter interpretation.

A.4 Problem Statement & Modeling

In this section we formally present the Ridesharing problem. To avoid introducing unnecessary notation, we only present the description of the model here; precise notation and details are provided in the respective sections where they are used.

In the Ridesharing problem there is a (potentially infinite) metric space $\mathcal{X}$ representing the topology of the environment, equipped with a distance function $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$. Both are known in advance. At any moment, there is a (*dynamic*) set of available taxi vehicles $\mathcal{V}_t$, ready to serve customer requests (i.e., drive to the pick-up, and subsequently to the destination location). Between serving requests, vehicles can *relocate* to locations of potentially higher demand, to mitigate spatial search frictions between drivers. Customer requests appear in an online manner at their respective pick-up locations, *wait* to potentially be matched to a shared ride, and finally are served by a taxi to their respective destination. In order for two requests to be able to share a ride, they must satisfy *spatial*, and *temporal* constraints. The former dictates that requests should be matched only if there is good spatial overlap among their routes. Yet, due to the latter constraint, requests cannot be matched even if they have perfect spatial overlap, if they are not both 'active' at the same time. Finally, ridesharing is an inherently *online* problem, as we are unaware of the requests that will appear in the future, and need to make decisions before the requests expire, while taking into account the dynamics of the fleet of taxis.

A.4.1 Performance Metrics

The goal is to *minimize the cumulative distance driven* by the fleet of taxis, while maintaining *high Quality of Service (QoS)*, given that we *serve all requests* (service guarantee). Serving all requests improves passenger satisfaction, and, most importantly, allows us to ground our evaluation to a common scenario, ensuring a fair comparison.

Global Metrics

Distance Driven: Minimize the cumulative distance driven by all vehicles for serving all the requests. We chose this objective as it directly correlates to passenger, driver, company, and environmental objectives (minimize operational cost, delay, CO_2 emissions, maximize the number of shared rides, improve QoS, etc.). All of the evaluated algorithms have to *serve all the requests*, either as shared, or single rides.

Complexity: Real-world time constraints dictate that the employed solution produces results in a reasonable time-frame.[4]

Passenger Specific Metrics – Quality of Service (QoS)

Time to Pair: Expected time to be paired in a shared ride, i.e., $\mathbb{E}[t_{\text{paired}} - t_{\text{open}}]$, where $t_{\text{open}}, t_{\text{paired}}$ denote the time the request appeared, and was paired as a shared ride, respectively. If the request is served as a single ride, then t_{paired} refers to the time the algorithm chose to serve it as such.

Time to Pair with Taxi: Expected time to be paired with a taxi, i.e., $\mathbb{E}[t_{\text{taxi}} - t_{\text{paired}}]$, where t_{taxi} denotes the time the (shared) ride was paired with a taxi.

Time to Pick-up: Expected time to passenger pickup, i.e., $\mathbb{E}[t_{\text{pickup}} - t_{\text{taxi}}]$, where t_{pickup} denotes the time the request was picked-up.

Delay: Additional travel time over the expected direct travel time (when served as a single ride, instead of a shared ride), i.e., $\mathbb{E}[(t_{\text{dest}} - t_{\text{pickup}}) - (t'_{\text{dest}} - t_{\text{pickup}})]$. t_{dest}, and t'_{dest} denote the time the request reaches, and would have reached as a single ride, its destination.

[4] For example UberPool has a waiting period of at most 2 minutes until you get a match (https://www.uber.com/au/en/ride/uberpool/), thus any algorithm has to run in under that time to be applicable in real life.

Research conducted by ridesharing companies shows that passengers' satisfaction level remains sufficiently high as long as the pick-up time is less than a certain threshold. The latter is corroborated by data on booking cancellation rate against pick-up time (Tang et al., 2017). In other words, passengers would rather have a short pick-up time and long detour, than vice-versa (Brown, 2016b). This also suggests that an effective relocation scheme can considerably improve passenger satisfaction by reducing the average pick-up time (see Section A.8.2).

Given the importance of short pick-up times in passengers' satisfaction, we opted to distinguish and study each segment of the waiting process separately ('Time to Pair', 'Time to Pair with Taxi', and 'Time to Pick-up'). To the best of our knowledge, we are the first to do so. Such analysis can provide a clear picture of sources of inefficiency to a ridesharing platform, and improve the overall satisfaction which in turn correlates to the growth of the company.

Driver Specific Metrics

Driver Profit: Total revenue earned minus total travel costs.

Number of Shared Rides: Related to the profit. By carrying more than one passenger at a time, drivers can serve more requests in a day, which consequently, increases their income (Widdows et al., 2017).

Frictions: Waiting time experienced by drivers between serving requests (i.e., time between dropping-off a ride, and getting matched with another). Search frictions occur when drivers are unable to locate rides due to spatial supply and demand imbalance. Even though in our scenario matchings are performed automatically, without any searching involved by the drivers, lower frictions indicate lower regret by the drivers, thus lower temptation to potentially switch to an alternative ridesharing platform.

Platform Specific Metrics

Platform Profit: Usually a commission on the driver's fee,[5] and passenger fees (which, given that we serve all the requests, the latter would be constant across all the employed algorithms).

[5]E.g., Uber charges partners 25% fee on all fares (https://www.uber.com/en-GH/drive/resources/payments/).

Quality of Service (QoS): Refer to the aforementioned, passenger specific metrics. Improving the QoS to their costumers correlates to the growth of the company.

Number of Shared Rides: The matching rate is important especially in the nascent stage of a ridesharing platform (Dutta and Sholley, 2018).

We do not report separate values on the aforementioned metrics, as they directly correlate to their respective passenger, and driver specific ones.

A.4.2 Modeling

Our evaluation setting is *meticulously designed to resemble reality as closely as possible*, in every aspect of the problem. We achieve this by (a) using actual data from the NYC's yellow taxi trip records,[6] both for modeling customer requests and taxis (b) following closely the pricing model employed by ridesharing platforms and (c) running our simulations to the scale of the actual problem faced by the ridesharing platforms (we run simulations with more than $390,000$ requests and $12,000$ taxis). Moreover, we have exhaustively designed every detail of the problem, such as speed of the vehicles, initial positions, distance function, pricing, etc. In what follows, we describe each design aspect in detail.

Dataset

We have used the yellow taxi trip records of 2016, provided by the NYC Taxi and Limousine Commission.[6] The dataset was cleaned to remove requests with travel time shorter than 1 minute, or invalid geo-locations (e.g., outside Manhattan, Bronx, Staten Island, Brooklyn, or Queens). For every request, the dataset provides amongst others the pick-up and drop-off times, and geo-location coordinates. Time is discrete, with granularity of 1 minute (same as the dataset). On average, there are 272 new requests per minute, totaling to 391479 requests in the broader NYC area (352455 in Manhattan) on the evaluated day (Jan, 15). Figure A.2 depicts the number of request per minute on the aforementioned day.

Taxi Vehicles

A unique feature of the NYC Yellow taxis is that they may only be hailed from the street and are not authorized to conduct pre-arranged pick-ups. This provides an ideal setting for a counter-factual analysis for several reasons: (1) We can assume a realistic position

[6] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

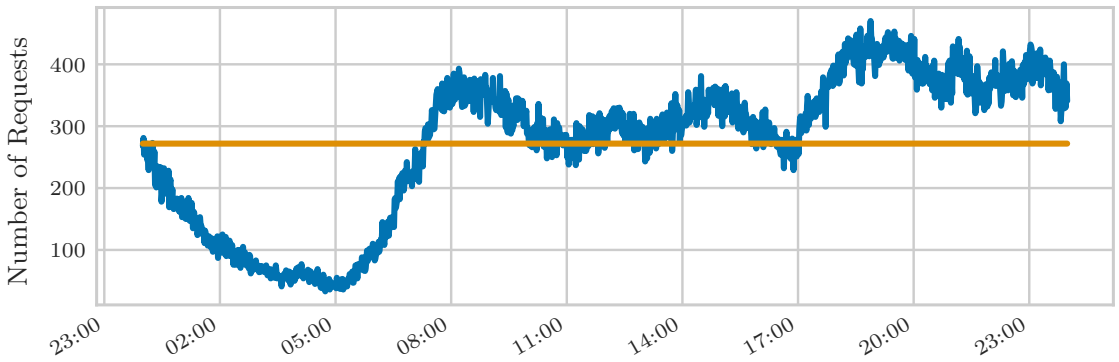


Figure A.2: Request per minute on Jan. 15, 2016 (blue line). Mean = 272 requests (yellow line).

of each taxi at the beginning of the simulation (last drop-off location). (2) Door-to-door service can be inefficient (Fielbaum et al., 2021; Stiglic et al., 2015), thus users may be requested to walk to/from a nearby fast street. Given that users have presumably hailed the taxis from larger streets, this results to a more accurate modeling of the origins of supply and demand. Finally, (3) all observed rides are obtained through search, thus – assuming reasonable prices, and delays – customers do not have nor are willing to take an alternative means of transportation. The latter validates our choice that all of the algorithms considered will have to eventually serve all the requests.

By law, there are $13,587$ taxis in NYC.[14] The majority of the results presented in this work use a much lower number of vehicles (what we call *base number*) for three reasons: (1) to reduce the complexity of the problem, given that most of the employed algorithms can not handle such a large number of vehicles, (2) to evaluate under resource scarcity – making the problem harder – to better differentiate between the results, and (3) to investigate the possibility of a more efficient utilization of resources, with minimal cost to the consumers. However, we still present simulations for a wide range of vehicles, up to close to the total number.

The number, initial location, and speed of the taxi vehicles were calculated as follows:

- We calculated the *base number* of taxis, as the minimum number of taxis required to serve all requests as single rides (no ridesharing). If a request appears, and all taxis are occupied serving other requests, we increase the required number of taxis by one. This resulted to around $4000 - 5000$ vehicles (depending on the size of the simulation, see Section A.8.2). Simulations were conducted for $\{\times 0.5, \times 0.75, \times 1.0, \times 2.0, \times 3.0\}$ the base number.

- Given a number of taxis, V, the initial position of each taxi is the drop-off location of the last V requests, prior to the starting time of the simulation. To avoid cold start, we compute the drop-off time of each request, and assume the vehicle occupied until

then.

- The vehicles' average speed is estimated to 6.2 m/s (22.3 km/h), based on the trip distance and time per trip as reported in the dataset, and corroborated by the related literature (in (Santi et al., 2014) the speed was estimated at $5.5 - 8.5$ m/s depending on the time of day).

Customer Requests

A request, r, is a tuple $\langle t_r, s_r, d_r, k_r \rangle$. Request r appears (becomes *open*) at its respective pick-up time (t_r), and geo-location (s_r). Let d_r denote the destination. Each request admits a willingness to wait (k_r) to find a match (rideshare), i.e., we assume *dynamic* waiting periods per request. The rationale behind k_r is that requests with longer trips are more willing to wait to find a match than requests with destinations near-by. After k_r time-steps we call request r, *critical*. If a critical request is not matched, it has to be served as a single ride. Recall that in our setting *all* of the requests must be served. Let $\mathcal{R}_t^{\text{open}}, \mathcal{R}_t^{\text{critical}}$ denote the sets of open, and critical requests respectively, and let $\mathcal{R}_t = \mathcal{R}_t^{\text{open}} \cup \mathcal{R}_t^{\text{critical}}$.

We calculate k_r as in related literature (Danassis et al., 2019). Let $w_{\min}$, and $w_{\max}$ be the minimum and maximum possible waiting time, i.e., $w_{\min} \leq k_r \leq w_{\max}, \forall r$. Knowing s_r, d_r, we can compute the expected trip time ($\mathbb{E}[t_{\text{trip}}]$). Assuming people are willing to wait proportional to their trip time, let $k_r = q \times \mathbb{E}[t_{\text{trip}}]$, where $q \in [0, 1]$. $w_{\min}, w_{\max}$, and q can be set by the ridesharing company, based on customer satisfaction (following (Danassis et al., 2019), let $w_{\min} = 1, w_{\max} = 3$, and $q = 0.1$).

Rides

A (shared)ride, ρ, is a pair $\langle r_1, r_2 \rangle$, composed of two requests. If a request r is served as a single ride, then $r_1 = r_2 = r$. Let $\mathcal{P}_t$ denote the set of rides waiting to be matched to a taxi at time t. Contrary to some recent literature on high capacity ridesharing (e.g., (Alonso-Mora et al., 2017; Lowalekar et al., 2019)), we *purposefully restricted ourselves to rides of at most two requests* for two reasons: *complexity*, and *passenger satisfaction*. The complexity of the problem grows rapidly as the number of potential matches increases, while most of the proposed/evaluated approaches already struggle to tackle matchings of size two on the scale of a real-world application. Moreover, even though a fully utilized vehicle would ultimately be a more efficient use of resources, it diminishes passenger satisfaction (a frequent worry being that the ride will become interminable, according to internal research by ridesharing companies) (Widdows et al., 2017; Brown, 2016a). Given that a hard constraint is the serving of all requests, we do not assume a time limit on matching rides with taxis; instead we treat it as a QoS metric.

Distance Function

The optimal choice for a distance function would be the actual driving distance. Yet, our simulations require trillions of distance calculations, which is not attainable. Given that the locations are given in latitude and longitude coordinates, it is tempting to use the Haversine formula[7] to estimate the Euclidean distance, as in related literature (Santos and Xavier, 2013; Brown, 2016a). We have opted to use the Manhattan distance, given that the simulation takes place mostly in Manhattan. To evaluate our choice, we collected more than 12 million actual driving distances using the Open Source Routing Machine (project-osrm.org), which computes the shortest path in road networks. Manhattan distance's standard and mean squared error, compared to the actual driving distance, was -0.5 ± 2.9 km, and 1.7 ± 2.4 km respectively, while Euclidean distance's was -3.2 ± 3.8 km, and 3.2 ± 3.8 km respectively.

Pricing

A combination of an one-time flag drop fee[8] ($\beta = 2.2$ \$), distance fare[8] ($\pi_{I} = 0.994$ \$/km for a single ride, $\pi_{II} = 0.8$ \$/km shared), fuel price[9] (3.2 \$/gal), and vehicle mileage (46.671 km/gal (Buchholz, 2018)). The aforementioned fuel price and mileage result in a cost per km $c = 0.0686$ \$/km. The revenue $M(\rho)$ of a taxi driver from serving ride ρ is given by (Buchholz, 2018):

$$M(\rho) = \begin{cases} \beta + \pi_{I}\delta(s_r, d_r) - c\delta(s_v, s_r, d_r) & \text{, if } \rho \text{ single} \\ 2\beta + \pi_{II}\delta(s_{r_1}, d_{r_1}|r_2) + \pi_{II}\delta(s_{r_2}, d_{r_2}|r_1) & \\ \quad -c\delta(s_v, s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2}) & \text{, if } \rho \text{ shared} \end{cases}$$

where, with some slight abuse of notation, $\delta(s_v, s_r, d_r)$ denotes the distance from the current location of the taxi s_v, to the pick-up and subsequently drop-off location of the ride, $\delta(s_{r_1}, d_{r_1}|r_2)$ denotes the distance driven from the pick-up to the destination of r_1, given that r_1 will share the ride with r_2 (similarly $\delta(s_{r_2}, d_{r_2}|r_1)$ for r_2), and finally, $\delta(s_v, s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2})$ denotes the total driving distance of the taxi for serving the two requests starting from s_v.

Embedding into HSTs

A starting point of many of the employed k-server algorithms is embedding the input metric space $\mathcal{X}$ into a distribution μ over σ-hierarchically well-separated trees (HSTs), with separation $\sigma = \Theta(\log|\mathcal{X}| \log(k \log|\mathcal{X}|))$, where $|\mathcal{X}|$ denotes the number of points. It

[7] https://en.wikipedia.org/wiki/Haversine_formula
[8] https://www.uber.com/us/en/price-estimate/
[9] https://www.eia.gov/dnav/pet/pet_pri_gnd_dcus_sny_m.htm

Table A.1: Evaluated CARs.

	Step (a)	Step (b)	Step (c)
Maximum Weight Matching (MWM)	MWM	MWM	MWM/ALMA/Greedy
ALtruistic MAtching Heuristic (ALMA) (Danassis et al., 2019)	ALMA	ALMA	ALMA
Greedy	Greedy	Greedy	Greedy
Approximation (Appr) (Bei and Zhang, 2018)	Appr	Appr	-
Postponed Greedy (PG) (Ashlagi et al., 2018)	PG	MWM	-
Greedy Dual (GD) (Bienkowski et al., 2018)	GD	MWM	-
Balance (Bal) (Manasse et al., 1990)	MWM	Bal	-
Harmonic (Har) (Raghavan and Snir, 1989)	MWM	Har	-
Double Coverage (DC) (Chrobak et al., 1990)	MWM	DC	-
Work Function (WFA) (Koutsoupias and Papadimitriou, 1995)	MWM	WFA	-
k**-Taxi** (Coester and Koutsoupias, 2019)	MWM	k-Taxi	-
High Capacity (HC) (Alonso-Mora et al., 2017)	HC	HC	(HC)
Baseline: Single Ride	-	MWM	-
Baseline: Random	-	Random	-

has been shown that solving the problem on HSTs suffices, as any finite metric space can be embedded into a probability distribution over HSTs with low distortion (Fakcharoenphol et al., 2004). The distortion is of order $\mathcal{O}(\sigma \log_\sigma |\mathcal{X}|)$, and the resulting HSTs have depth $\mathcal{O}(\log_\sigma \Delta)$, where Δ is the diameter of $\mathcal{X}$ (Bansal et al., 2015).

Given the popularity of the aforementioned method, it is worth examining the size of the resulting trees. Given that the geo-coordinate system is a discrete metric space, we could directly embed it into HSTs. Yet, the size of the space is huge, thus for better discretization we have opted to generate the graph of the street network of NYC. To do so, we used data from openstreetmap.org. Similarly to (Santi et al., 2014), we filtered the streets selecting only primary, secondary, tertiary, residential, unclassified, road, and living street classes, using those as undirected edges and street intersections as nodes. The resulting graph for NYC contains 66543 nodes, and 95675 edges (5018, and 8086 for Manhattan). Given that graph, we generate the HSTs (Santi et al., 2014).

A.5 Component Algorithms for Ridesharing

In this section, we describe our design choices for developing *Component Algorithms for Ridesharing (CARs)*. Each CAR is composed of three parts (Figure A.1): (a) request – request matching to create a (shared) ride, (b) ride to taxi matching, and (c) relocation of the idle fleet. Each of these components is a significant problem in its own right. Complexity issues make the simultaneous consideration of all three problems impractical. Instead, a more realistic approach is to tackle each component individually, under minimum consideration of the remaining two.[10] The algorithms that we consider are appropriate modifications of the most significant ones that have been proposed for

[10]To have a comprehensive analysis, we have also evaluated the HC algorithm, a highly cited approach that tackles steps (a), and (b) simultaneously. Yet, this results in a prohibitively large ILP (see Sections A.5.1 and A.6).

the key algorithmic primitives of the ridesharing problem (see Sections A.2.1 and A.3), i.e., *online and offline matching algorithms, with or without delays* for steps (a), (b), and (c), *k-taxi/server algorithms* for step (b), as well as *heuristic approaches* that were specifically designed with the ridesharing application in mind.

A list of all the CARs that we designed and evaluated (14 in total) can be found in Table A.1, while in the following sections we provide a detailed description of each CAR component.

A.5.1 CAR components

We have evaluated a variety of approaches ranging from offline maximum weight matching (MWM), and greedy solutions, to online MWM, k-Taxi/Server algorithms, and linear programming. Offline algorithms (e.g., MWM, ALMA, Greedy) can be run either in a just-in-time (JiT) manner – i.e., when a request becomes critical – or in batches, i.e., every x minutes (given that our dataset has granularity of 1 minute, we run in batches of 1, and 2 minutes).

Matching Graphs: At time t, let $\mathcal{G}_a = (\mathcal{R}_t, \mathcal{E}_t^a)$, where $\mathcal{E}_t^a$ denotes the weighted edges between requests. With a slight abuse of notation, let $\delta(s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2})$ denote the minimum distance required to serve both r_1, and r_2 (as a shared ride, i.e., excluding the case of first serving one of them and then the other) with a single taxi located either in s_1, or s_2. The weight w_{r_1,r_2} of an edge $(r_1, r_2) \in \mathcal{E}_t^a$ is defined as $w_{r_1,r_2} = \delta(s_1, d_1) + \delta(s_2, d_2) - \delta(s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2})$ (similarly to (Danassis et al., 2019; Alonso-Mora et al., 2017)). If $r_1 = r_2$, let $w_{r_1,r_2} = 0$ (single passenger ride). Intuitively, this number represents an approximation (given that it is impossible to know in advance the location of the taxi that will serve the ride) on the travel distance saved by matching requests r_1, and r_2.[11]

Similarly, at time t, let $\mathcal{G}_b = (\mathcal{V}_t \cup \mathcal{P}_t, \mathcal{E}_t^b)$, where $\mathcal{E}_t^b$ denotes the weighted edges between rides and taxis. With a slight abuse of notation, let $\delta(s_v, s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2})$ denote the minimum distance required (out of all the possible pick-up and drop-off combinations) to serve both requests r_1, and r_2 (that compose the (shared) ride ρ) with a single taxi located at s_v. The weight $w_{v,\rho}$ of an edge $(v, \rho) \in \mathcal{E}_t^b$ is defined as $w_{v,\rho} = 1/\delta(s_v, s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2})$. If $r_1 = r_2$ (single passenger ride), let $\delta(s_v, s_{r_1}, s_{r_2}, d_{r_1}, d_{r_2}) = \delta(s_v, s_{r_1}, d_{r_1})$. For the step (b) of the Ridesharing problem, we run the offline algorithms every time the set of rides ($\mathcal{P}_t$) is not empty.

[11] It also ensures that the shared ride will cost less than the single ride option.

Maximum Weight Matching (MWM)

The maximum weight matching algorithm finds a matching with maximum total edge weight in a graph. We use a maximum wieght matching algorithm to

- match requests into shared rides (step (a) of the Ridesharing problem), i.e., find a matching on $\mathcal{G}^a$ that maximizes the quantity $\sum_{(r_1,r_2)\in\mathcal{E}_t^a} w_{r_1,r_2}$.
- match rides with taxis (step (b) of the Ridesharing problem), i.e., find a matching on $\mathcal{G}_b$ that maximizes the quantity $\sum_{(v,\rho)\in\mathcal{E}_t^b} w_{v,\rho}$.

In both cases we use the well-known *blossom algorithm* of Edmonds (1965). Not surprisingly, MWM results in high quality allocations, but that comes with an overhead in running time, compared to simpler, 'local' solutions (see Section A.8.2). This is because blossom's worst-case time complexity – on a graph (V, E) – is $\mathcal{O}(|E||V|^2)$, and we have to run it three times, one for each step of the Ridesharing problem. Additionally, the MWM algorithm inherently requires a global view of the whole request set in a time window, and is therefore not a good candidate for the fast, decentralized solutions that are more appealing for real-life applications.

ALtruistic MAtching Heuristic (ALMA)

A distinctive characteristic of ALMA is that agents (in our context: requests / rides) make decisions locally, based solely on their own utilities. In particular, while contesting for a resource (in our context: request / taxi), each agent will back-off with probability that depends on their own utility loss of switching to their next most preferred resource. E.g., for step (b) of the Ridesharing problem, suppose that for the agent representing ride ρ, the next most preferred taxi to v is v', then $loss = w_{v,\rho} - w_{v',\rho}$. The back-off probability ($P(\cdot)$) is computed individually and locally, based on Equation[12] A.1. For a detailed description of ALMA, see Chapter 3.

$$P(loss) = \begin{cases} 1-\epsilon, & \text{if } loss \leq \epsilon \\ \epsilon, & \text{if } 1-loss \leq \epsilon \\ 1-loss, & \text{otherwise} \end{cases} \tag{A.1}$$

Greedy

Greedy is a very simple algorithm, which selects a node $i \in V$ of a graph $G = (V, E)$ uniformly at random, considers all the edges (i, j) with endpoint i, and matches i with a node j^* that is the endpoint of the edge with the largest weight among those, i.e., $(i, j^*) \in$

[12]The parameter ϵ places a threshold on the minimum / maximum back-off probability.

$\arg\max(w_{i,j})$. Greedy approaches are appealing,[13] not only due to their low complexity, but also because real-time constraints dictate short planning windows which diminish the benefit of batch optimization solutions compared to myopic approaches (Widdows et al., 2017).

Approximation (Appr), (Bei and Zhang, 2018)

Approximation (Appr) is a recently-proposed offline algorithm due to Bei and Zhang (2018) which can be used to solve steps (a), and (b) of the Ridesharing problem. The algorithm takes a two-phase approach which is also based on maximum weight matchings (or more accurately, the equivalent notion of minimum cost matchings), but on a set of different weights (to the ones we defined for the MWM algorithm). In particular:

- First, it matches requests to shared rides using minimum cost matching based on the shortest distance to serve any request pair but on the worst pickup choice. Formally, the algorithm defines the quantities:

$$w_{ij} = \min\{\delta(s_1, s_2) + \delta(s_2, d_1) + \delta(d_1, d_2), \delta(s_1, s_2) + \delta(s_2, d_2) + \delta(d_2, d_1)\}$$

$$w_{ji} = \min\{\delta(s_2, s_1) + \delta(s_1, d_1) + \delta(d_1, d_2), \delta(s_2, s_1) + \delta(s_1, d_2) + \delta(d_2, d_1)\}$$

  and then chooses $w^1(i, j) = \max\{w_{ij}, w_{ji}\}$. Intuitively, w_{ij} is the distance of the shortest path that picks up request r_1 first (at its source location s_1), and similarly, w_{ji} is the distance of the shortest path that picks up request r_2 first.

- Then it matches rides to taxis using again minimum cost matching, and assuming the weight to be the distance of the closest pick-up location of the two. Formally, let $w^2(v, \langle r_i, r_j\rangle) = \min\{\delta(s_v, s_i), \delta(s_v, s_j)\}$, where s_v is the position of taxi v, and compute a minimum cost matching in the bipartite graph defined by pairs $\langle r_i, r_j\rangle$ matched in the previous step and taxis, with weights defined by u^2.

Bei and Zhang (2018) prove a worst-case approximation guarantee of 2.5 for the algorithm.

Postponed Greedy (PG), (Ashlagi et al., 2018)

Postponed Greedy (PG) is another very recently proposed, algorithm for the maximum weight online matching problem with deadlines (step (a) of the Ridesharing problem). The algorithm is online, meaning that it considers the potential requests that might appear in the future when making decisions about the present; its competitive ratio was proven to be 1/4 by Ashlagi et al. (2018). Contrary to our setting, the algorithm was designed for fixed deadlines, i.e., $k_r = c, \forall r \in \mathcal{R}$.

[13] (Widdows et al., 2017) reports that GrabShare's scheduling component has used an entirely greedy approach to allocate bookings to drivers. Lyft also started with a greedy system (Brown, 2016a).

The algorithm is best described in terms of an *auction* environment (Ashlagi et al., 2018) as follows. Let S_t and B_t be the sets of *virtual sellers* and *virtual buyers* at time t respectively. When a request r appears at time t, the algorithm creates a virtual seller s_r and a virtual buyer b_r for that request, and adds them to the aforementioned sets, i.e., $S_t \leftarrow S_{t-1} \cup \{s_r\}$ and $B_t \leftarrow B_{t-1} \cup \{b_r\}$. In other words, every request has two copies: a buyer and a seller. These are then placed in a virtual weighted bipartite graph $G = (S_t, B_t, E_t)$, where the edge weights are defined in the same manner as the weights of $\mathcal{G}_a$ (see 'Matching Graphs' in Section A.5.1). The algorithm proceeds to match the newly added buyer b_r with a seller s_{r^*} in a greedy manner, i.e., $(b_r, s_{r^*}) \in \arg\max_{r' \in S_{t-1}}(w_{r,r'})$. This choice remains fixed for subsequent time steps. When the request r becomes critical (i.e., the deadline is about to be met), the 'role' of the request as either a seller or a buyer is conclusively chosen (uniformly at random). If r is a seller, and a subsequent buyer was matched with r, the match is finalized and is included in the output matching.

The major difference between the setting consider by Ashlagi et al. (2018) and our setting is that for us, requests become critical out-of-order, and a critical request cannot be matched later. Thus, we apply the following modification: when a request becomes critical, if determined to be a seller, the match is finalized (if one has been found), otherwise the request is treated as a single ride.

Greedy Dual (GD), (Bienkowski et al., 2018)

Greedy Dual is an online algorithm for solving the minimum cost (bipartite) perfect matching with delays, i.e., both steps (a), and (b) of the Ridesharing problem, which is based on the popular primal-dual technique (Goemans and Williamson, 1997). The weight (cost) of an edge in this setting includes arrival times as well, specifically:

$$w_{r_1,r_2} = \frac{(\delta(s_1, s_2) + \delta(d_1, d_2))}{u_{\text{average}}} + |t_1 - t_2|,$$

where u_{average} is the average speed (see Section A.4.2). The algorithm partitions all the requests into *active sets*, starting with the singleton $\{r\}$ for a newly arrived request r. As is typical in the primal-dual approach, at every time-step t these actives sets 'grow', until the weight of the edges of different active sets make the dual constraints of the problem tight (i.e., satisfied with equality). At this point the active sets merge, and the algorithm matches as many pairs of free requests in these sets as possible.

The algorithm has a competitive ratio of $\mathcal{O}(|\mathcal{R}|)$ and works with infinite metric spaces, potentially making the algorithm better suited for applications like the Ridesharing problem. Yet, in terms of our setup, it does not take into account the willingness to wait (k_r), thus missing matches of requests that became critical. Despite being designed for bipartite matchings as well, we opted out from using it for step (b) since it would require to create a new node every time a taxi vehicle drops-off a ride and becomes available.

Balance (Bal), (Manasse et al., 1990)

Balance is a simple and classic algorithm for the k-server problem from the literature of competitive analysis. The rationale behind the algorithm is that it tries to balance out the distance traveled by taxis over the course of their operation, trying to maintain the workload as equal as possible. In particular, a ride is served by the taxi that has the minimum sum of the distance traveled so far plus its distance to the source of the ride (chosen uniformly at random between the sources of the two requests composing the ride). Specifically, ride ρ will be matched to taxi v:

$$(v, \rho) = \arg\min_{v \in \mathcal{V}_t} (\text{driven}(v) + \delta(s_v, s_\rho)) \tag{A.2}$$

where driven(v) denotes the distance driven by taxi v so far, and s_ρ is selected equiprobably among s_1 and s_2. The algorithm is min-max fair, i.e., it greedily minimizes the maximum accumulated distance among the taxis. The competitive ratio of the algorithm is $|\mathcal{X}| - 1$ in arbitrary metric spaces with $|\mathcal{X}|$ points (Manasse et al., 1990).

Harmonic (Har), (Raghavan and Snir, 1989)

The Harmonic algorithm (Har) is another classic randomized algorithm from the k-server problem literature, which is simple and memoryless (i.e., it does not need to 'remember' the decisions that it took in previous steps). The algorithm matches a taxi with a ride with probability inversely proportional to the distance from its source (chosen uniformly at random between the sources of the two requests composing the ride). Specifically, ride ρ will be matched to taxi v with probability:

$$P(v, \rho) = \frac{\frac{1}{\delta(s_v, s_\rho)}}{\sum_{\rho' \in \mathcal{P}_t} \frac{1}{\delta(s_v, s_{\rho'})}} \tag{A.3}$$

where s_ρ and $s_{\rho'}$ are both selected equiprobably among s_1, s_2 and $s_{1'}$, $s_{2'}$, respectively. The trade-off for its simplicity is the high competitive ratio, which is $\mathcal{O}(2^{|\mathcal{V}|} \log |\mathcal{V}|)$ (Bartal and Grove, 2000).

Double Coverage (DC), (Chrobak et al., 1990)

Double Coverage (DC) is one of the two most famous k-server algorithms in the literature. The algorithm is designed to run on a specific type of metric space called an HST (Hierarchical Separated Tree, see Section A.4.2). For a general metric spaces $\mathcal{X}$, the algorithm can be applied by first embedding $\mathcal{X}$ to an HST (a process which is referred to as an 'HST embedding'). This process 'simulates' the general space $\mathcal{X}$ by an HST, in the sense that the HST approximately captures the properties of the original space $\mathcal{X}$. The

points of $\mathcal{X}$ are the leaves of the HST.

Given an HST, the algorithm works as follows. To determine which taxi will serve a ride, all *unobstructed* taxis move towards its source, i.e., a leaf of the HST (chosen randomly between the sources of the two requests sharing the ride) with equal speed. Initially, all taxis are unobstructed. During this movement process, a taxi becomes *obstructed* when its path from its current location to the leaf corresponding to the ride is 'blocked' by another taxi, meaning that it would have to move through the same position in the tree that another taxi has already been at, to reach the leaf. In this case, the taxi stops (as the 'blocking' taxi is closer to serving the ride), while the remaining taxis keep moving as before. When some taxi reaches the leaf corresponding to the ride, the process stops, and each taxi maintains its current position on the HST.

To implement the algorithm, we first appropriately discretize our metric space and then perform the HST embedding as described in (Bartal, 1996; Fakcharoenphol et al., 2004) (see Section A.4.2 for more details). Given that only leaves correspond to locations on $\mathcal{X}$, we chose to implement the *lazy* version of the algorithm (which is worst-case equivalent to the original definition e.g., see (Koutsoupias, 2009)), i.e., only the taxi serving the ride will move on $\mathcal{X}$; one can envision a process in which the taxis 'virtually' move as described above, but once the ride has been served, all taxis are restored to their original positions. This is also on par with the main goal of minimizing the distance driven. The algorithm is k-competitive on all tree metrics (Chrobak and Larmore, 1991a).

Work Function (WFA), (Chrobak and Larmore, 1991b; Koutsoupias and Papadimitriou, 1995)

The Work Function algorithm (WFA) is perhaps the most important k-server algorithm, as it provides the best competitive ratio to date, due to the celebrated result of (Koutsoupias and Papadimitriou, 1995). Intuitively, to decide which taxi (or server) will be the one to serve a ride that just appeared at time t, and, more generally, the movement of the other taxis, the algorithm:

- computes the (offline) optimal solution until time $t-1$, meaning the best possible allocation of rides to taxis using the information from the beginning of the algorithm until the appearance of the ride at time t,
- computes a *greedy cost* for switching between configurations,
- chooses the new taxi positions that minimize the sum of the two aforementioned costs.

More formally, let $L^t = (l_1^t, l_2^t, \ldots, l_{|\mathcal{V}|}^t)$ denote the *configuration* of the fleet of taxis $\mathcal{V}$ at time-step t, i.e., a vector of taxi locations, where l_v^t specifies the location of taxi v. Let

$\mathrm{OPT}_t(L)$ be the optimal (total distance-minimizing) way of serving rides that appear at times 1 through t, such that the taxis end up at configuration L. To choose configuration L^t, it uses the following rule:

$$L^t = \arg\min_L \{\mathrm{OPT}_t(L) + d(L^{t-1}, L)\}$$

The WFA serves ride ρ_t at time-step t by switching from the current taxi configuration L^{t-1}, to a new configuration L^t. Specifically, it selects L^t which minimizes (a) the minimum total cost of starting from L^0, serving in turn $\rho_0, \rho_1, \ldots, \rho_{t-1}$, and ending up in L^t, plus (b) the distance traveled by a taxi to move from its position in L^{t-1} to that in L^t.

An obvious obstacle that makes the algorithm intractable in practice is that the complexity increases from step to step, resulting in computation and/or memory issues. To circumvent this obstacle, we implemented an efficient variant using network flows, as described in (Rudec et al., 2013). Yet, as the authors of (Rudec et al., 2013) state as well, the only practical way of using the WFA is switching to its window version w-WFA, where we only optimize for the last w rides. Even though the complexity of w-WFA does not change between time-steps, it does change with the number of taxis. The resulting network has $2|\mathcal{P}| + 2|\mathcal{V}| + 2$ nodes, and we have to run the Bellman–Ford algorithm (Bellman, 1958) at least once to compute the potential of nodes and make the costs positive (Bellman–Ford runs in $\mathcal{O}(|\mathcal{P}||\mathcal{V}|)$. We refer the reader to (Bertsekas, 1998) for more details on network optimization. As before, the source of the ride is chosen randomly between the sources of the two requests composing the ride.

k-Taxi, (Coester and Koutsoupias, 2019)

This is a very recent algorithm for the k-taxi problem, which provides the best possible competitive ratio. The algorithm operates on HSTs, where the rides and taxis at any time are placed at its leaves. First, it generates a Steiner tree that spans the leaves that have taxis or rides, and then uses this tree to schedule rides, by simulating an electrical circuit. In particular, whenever a ride appears at a leaf, the algorithm interprets the edges of the tree with length R as resistors with resistance R, which determine the fraction of the current flow that will be routed from the node corresponding to the taxi towards the ride. These fractions are then interpreted as probabilities which determine which taxi will be chosen to pick up the ride.

High Capacity (HC), (Alonso-Mora et al., 2017)

This algorithm comes from a highly-cited paper, and is the only one in our evaluated approaches that addresses vehicle relocation (step (c)). Contrary to our approach, it tackles steps (a), and (b) simultaneously, leaving step (c) as a separate sub-problem. The

algorithm consists of five steps:

(i) Computing a pairwise request-vehicle shareability graph (RV-graph) (Santi et al., 2014). The RV-graph represents which requests and vehicles might be pairwise-shared, with edges connecting all possible requests to pair and all possible vehicles to serve a request.

(ii) Computing a graph consisting of feasible (candidate) trips and the set of vehicles that can execute them (RTV-graph). This is a tripartite graph with edges connecting requests to trips (a request is connected to a trip if it is part of it), and edges connecting trips to vehicles (an edge between vehicle and a trip exists if the vehicle is able to serve it).

(iii) Computing a greedy solution for the RTV-graph. In this step, rides are assigned to vehicles iteratively in decreasing size of the trip (in our case, we first assign shared rides (two requests), and then single rides) and increasing cost (e.g., delay).

(iv) Solving an ILP to compute the best assignment of vehicles to trips, using the previously computed greedy solution as an initial solution.

(v) (optional) Rebalancing of free vehicles. If there remain any unassigned requests, it solves an ILP to optimally assign them to idle vehicles based on travel times.

We use CPLEX (Bliek et al., 2014) to solve the ILPs.

Baseline: Single Ride

Uses MWM to schedule the serving of single rides to taxis (there is no ridesharing, i.e., we omit step (a) of the Ridesharing problem).

Baseline: Random

Makes random matches, provided that the edge weight is non-negative.

While our evaluation contains many recently proposed algorithms for matching, the observant reader might notice that, with the exception of k-taxi, our k-server algorithms are from the classical literature. We did consider more recent k-server algorithms (e.g., (Dehghani et al., 2017; Lee, 2018; Bansal et al., 2015)), but their complexity turns out to be prohibitive. This is mainly because they proceed via an 'online rounding' of an LP-relaxation of the problem, which maintains a variable for every (time-step, point in the metric space) pair. Even for one hour (3600 time-steps) and our discretization of Manhattan (5018 nodes), we need more than 18 million variables (230 million for NYC).

A.6 Scalability Challenges

To highlight the challenges in the design of CARs, we will be referring to our evaluation setting (see Section A.4.2), which accurately models a real-world application, in terms of both *scale* and *detail*. Let $\mathcal{V}$, $\mathcal{R}$ denote the set of vehicles / requests, respectively. Recall that in our setting, which involves real data from NYC taxi records, there are 272 new requests per minute on average, totaling to 391479 requests in the broader NYC area (352455 in Manhattan) on the evaluated day (Jan, 15, 2006). By law, there are $13,587$ taxis in NYC.[14]

A.6.1 ILP Approaches

A natural approach would be to try to use Integer Linear Programs (ILPs) for matching passengers to other passengers or rides, under spatial and temporal constraints, similarly to the High Capacity algorithm of (Alonso-Mora et al., 2017) (which can be seen as a CAR with steps (a) and (b) intertwined). As is commonly the case with ILPs, the problem is scalability; the number of variables can be as large as $\mathcal{O}(|\mathcal{V}||\mathcal{R}|^2)$ – which results in 27 - 216 million variables, given that every time-step we have approximately 300 - 600 requests, and as many taxis – and the number of constraints is $|\mathcal{V}| + |\mathcal{R}|$. This makes ILP approaches prohibitive as components in CARs. The latter make hard to even compute the initial greedy solution in real-time. Alonso-Mora et al. circumvent this issue by enforcing delay constraints, specifically they ignore requests that are not matched to any vehicle within a maximum waiting time. This is not possible in our model since we have to serve all requests (service guarantee).[15]

A.6.2 MWM Approaches

Given that all three parts of the ridesharing problem can be viewed as matching problems, a natural approach would be to run maximum-weight matching (MWM) *in batches* (e.g., (Bei and Zhang, 2018)), meaning that we serve the requests that have accumulated over a pre-specified time window. The MWM problem can be solved via the classic *blossom algorithm* (Edmonds, 1965) with run time – on a graph (V, E) – of $\mathcal{O}(|E||V|^2)$.

A.6.3 k-server/taxi Algorithms

Many of these algorithms operate by embedding the input metric space $\mathcal{X}$ into a distribution μ over Hierarchical Separated Trees (HSTs) (e.g., the classic *double-coverage* (Chrobak et al., 1990)), and thus to apply them in practice, it is necessary to examine the size of

[14] https://www1.nyc.gov/site/tlc/businesses/yellow-cab.page

[15] For the sake of completeness we have evaluated the High Capacity algorithm on much smaller test cases; see Section A.10.

these trees. Given that the geo-coordinate system is a discrete metric space, we could directly embed it into HSTs. Yet, the size of the space is huge, and hence for better discretization we have opted to generate the graph of the street network of NYC (see Section A.4.2). The resulting graph for NYC contains 66543 nodes, and 95675 edges (5018, and 8086 for Manhattan). Here, there is an obvious interplay between the accuracy of the embedding and the algorithm's complexity.

More recent k-server algorithms (e.g., (Dehghani et al., 2017; Lee, 2018; Bansal et al., 2015)) use sophisticated 'online rounding' techniques; these however require maintaining variables for every (time-step, point in the metric space) pair, which makes them prohibitive for any large-scale real-world application; even for one hour (3600 time-steps) and our discretization of Manhattan (5018 nodes), we would need more than 18 million variables (230 million for NYC).

A.6.4 Observability

Most approaches are centralized, and require a *global* view of the *entire window*, which is hard to scale. As autonomous agents proliferate, a practical and applicable CAR must be distributed and ideally run *on-device*.

A.7 Vehicle Relocation Challenges

There are two ways to enforce relocation: *passive*, and *active*. Ridesharing platforms, like Uber and Lyft, have implemented market-driven pricing as a passive form of relocation. Counterfactual analysis performed in (Buchholz, 2018) shows that implementing pricing rules can result in daily net surplus gains of up to 232000 and 93000 additional daily taxi-passenger matches. While the gains are substantial, the market might be slow to adapt, and drivers and passengers do not always follow equilibrium policies. Contrary to that, our approach is *active*, in the sense that we directly enforce relocation. Moreover, we adopt a more *anthropocentric* approach: in our setting, the demand is fixed, thus the goal is not to increase revenue as a result of serving more rides, but rather to improve the QoS.[16]

There are many ways to approach dynamic relocation. Most of the employed relocation approaches are course-grained; the network is generally divided into several zones, blocks, etc. (Guériau and Dusparic, 2018; Vosooghi et al., 2019; Martínez et al., 2017) and the entities (e.g., the vehicles) move between the zones. However, compared to other shared mobility systems, dynamic ridesharing posses unique challenges, meaning that such coarse-grained approaches are not appropriate:[17] most of them are centralized –

[16] Decreased delays can also in turn improve revenue by serving more requests in a fixed time window.

[17] As a matter of fact, we tried zone based relocation (generating zones based on historical data using the OPTICS clustering algorithm (Xu and Tian, 2015), or using pre-defined clusters based on population

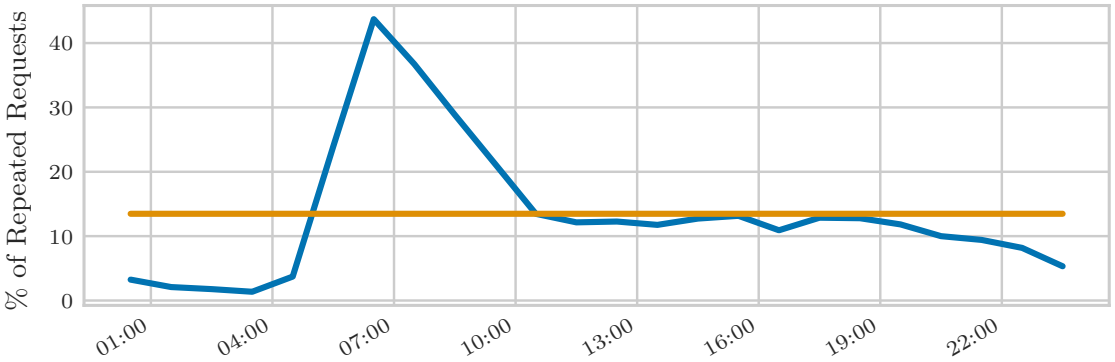


Figure A.3: Percentage of similar trips per hour in Manhattan, January 15, 2016 (blue line). Mean value = 13.3% (yellow line).

thus computationally intensive and not scalable –, they might not take into account the behavior of other drivers, potentially leading to over-saturation of high demand areas, and, most importantly, they are *slow to adapt* to the highly dynamic nature of the problem (e.g., responding to high demand generated by a concert, or the fact that vehicles remain free for only a few minutes at a time). The problem clearly calls for fine-grained solutions, yet such approaches in the literature are still rather scarce. High Capacity (HC) employs fine-grained relocation. HC solves an ILP, which could reach high quality results, but it is not scalable nor practical. Ideally, we would like a solution that can run *on-device*. The k-server algorithms perform an implicit relocation, yet they are primarily developed for adversarial scenarios, and do not utilize the plethora of historic data.[18] In reality, requests follow patterns that emerge due to human habituality (e.g., during the first half of the day in Manhattan, there are many more drop-offs in Midtown compared to pickups (Buchholz, 2018)).

A.7.1 Patterns in Customer Requests

To confirm the existence of transportation patterns, we performed the following analysis: For each request r on January 15,[19] we searched the past three days for requests r' such that $|t_r - t_{r'}| \leq 10$, $\delta(s_r, s_{r'}) \leq 250$, and $\delta(d_r, d_{r'}) \leq 250$. The results are depicted in Figure A.3. On average, 13.3% of the trips are repeated across all three previous days, peaking at 43.7% on rush hours (e.g., 6-8 in the morning).

density according to the NYC census data (https://guides.newman.baruch.cuny.edu/nyc_data/nbhoods)). Due to the vast number of requests, the only discernible clusters were of large regions (Manhattan, Bronx, Staten Island, Brooklyn, or Queens), which does not allow for fine-grained relocation. As a result, we achieved significantly inferior results.

[18] NYC TLC has been proving data on yellow taxi trips since 2009.

[19] January 15, 2016 was selected as a representative date for our simulations since it is not a holiday, and it is a Friday thus sampling for past requests results in a representative pattern (contrary to sampling on a weekend for example).

A.7.2 Relocation Matching Graph

Given the high density of the requests, and the low frictions of the taxis (i.e., taxis remain free for relocation only for a short time window), we opted for a simple, fine-grained, matching approach. We use the history to predict a set of *expected future requests*. Specifically, let D, and T be the sampling windows, in days and minutes respectively (we used $D = 3$, and $T = 2$). Let t denote the current time-step. The set of past requests on our sampling window is $\mathcal{R}_{\text{past}} = \{r : t_r - t \leq T\}$, as long as r appeared at most D number of days prior to t. The set of expected future requests $\mathcal{R}_{\text{future}}$ is generated by sampling from $\mathcal{R}_{\text{past}}$. Relocation is performed in a just-in-time manner, every time the set of idle vehicles is not empty. We generate similar matching graphs as in Section A.5.1, and then we proceed to match requests to shared rides, and rides to idle taxis. The difference being that now the set of nodes of $\mathcal{G}_a$ is $\mathcal{R}_{\text{future}} \cup \mathcal{R}_t$. Finally, each idle taxi starts moving towards the source of its match (given that these are expected rides, the source is picked at random between the sources of the two requests composing the ride).

A.8 Evaluation

A.8.1 Employed CARs

Evaluating all of the possible combinations of CAR components is infeasible. To make the evaluation tractable, we first consider only the first two steps of the ridesharing problem (i.e., no relocation). When possible, we use the same component for both steps (a) and (b). k-Taxi/Server algorithms, though, can not solve step (a), thus we opted to use the best performing component for step (a) (namely the offline maximum-weight matching (MWM) run in batches). Then, we move to evaluate step (c), testing only the most promising components (namely the MWM and ALMA, plus the Greedy as a baseline). We begin by isolating step (c); we fix the component for (a) and (b) to MWM, to have a common-ground for evaluating relocation. Finally, we present results on *end-to-end* solutions. A list of all the evaluated CARs can be found in Table A.1, while Table A.2 contains a summary of all the evaluated metrics.

A.8.2 Simulation Results

In this section we present the results of our evaluation. For every metric we report the average value out of 8 runs. In what follows we shortly detail only the most relevant results. Please refer to Section A.10 for the complete results including larger test-cases on the broader NYC area and *omitted metrics, standard deviation values, algorithms* (e.g., WFA, and HC had to be evaluated in smaller test-cases), etc. To improve readability, and present the results in a comprehensive manner, we have opted to repeat part of the main text of Section 3.6.

Table A.2: Evaluated performance metrics (global, passenger (Quality of Service), driver, and platform specific).

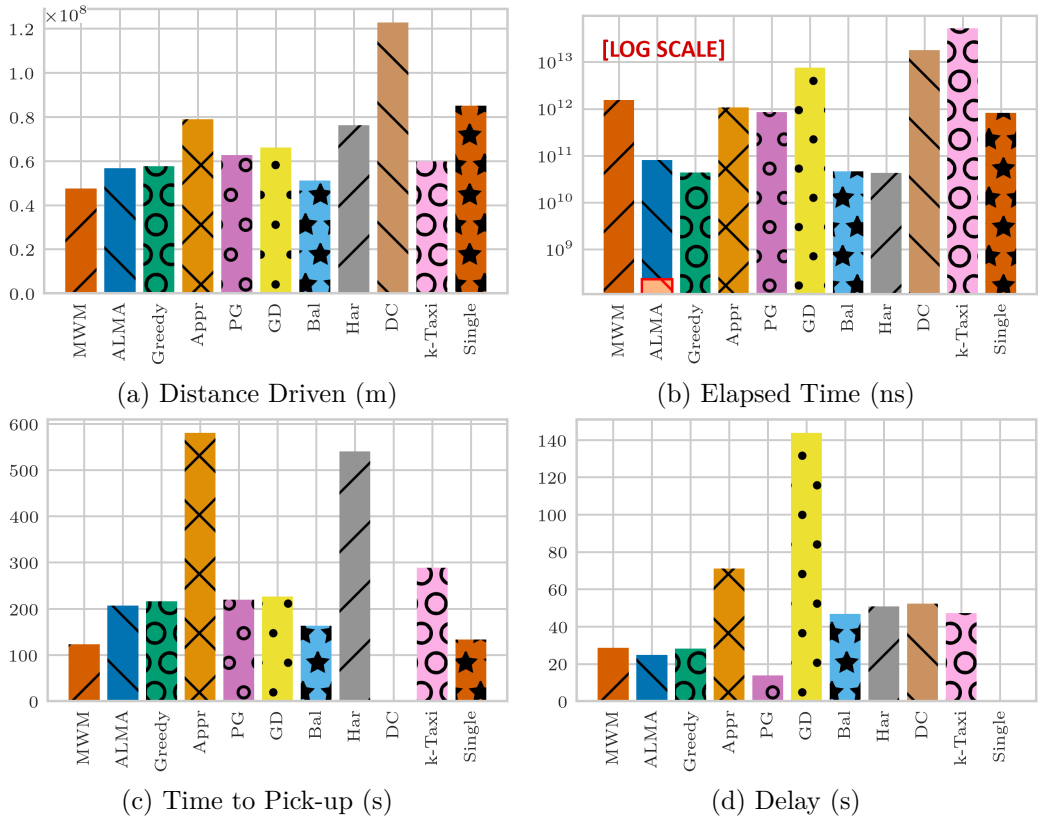
Distance Driven	Minimize the cumulative distance driven by all vehicles for serving all the requests. We chose this objective as it directly correlates to passenger, driver, company, and environmental objectives.
Complexity	Real-world time constraints dictate that the employed solution produces results in a reasonable time-frame.[4]
Time to Pair	Expected time to be paired in a shared ride.
Time to Pair with Taxi	Expected time to be paired with a taxi.
Time to Pick-up	Expected time to passenger pickup.
Delay	Additional travel time over the expected direct travel time (when served as a single, instead of a shared ride).
Driver Profit	Total revenue earned minus total travel costs.
Number of Shared Rides	Related to the profit. By carrying more than one passenger at a time, drivers can serve more requests in a day.
Frictions	Waiting time experienced by drivers between serving requests (i.e., time between dropping-off a ride, and getting matched with another). Lower frictions indicate lower regret by the drivers.
Platform Profit	A commission on the driver's fee, and passenger fees.
Quality of Service (QoS)	Refer to the passenger metrics. Improving the QoS to their customers correlates to the growth of the company.
Number of Shared Rides	The matching rate is important especially in the nascent stage of the platform (Dutta and Sholley, 2018).

Figures A.4, A.7, A.5, and A.6 present the results without relocation. We first present results on one hour (Figures A.4 and A.7) and base number of taxis (see Section A.4.2). Then, we show that the results are robust at a larger time-scale[20] (Figure A.5), and varying number of vehicles[21] (2138 - 12828) (Figure A.6). Finally, we present results on the step (c) of the Ridesharing problem: dynamic relocation (Table A.4, Figure A.8).

Distance Driven:

In the small test-case (Figure A.4a) MWM performs the best, followed by Bal (+7%). ALMA comes second (+19%), and then Greedy (+21%). The high performance of Bal in this metric is because it uses MWM for step (a), which has a more significant impact on the distance driven. Similar results are observed for the whole day (Figure A.5a), with Bal, ALMA, and Greedy achieving +4%, +18%, and +22% compared to MWM,

[20] Missing components were too computationally expensive to simulate for an entire day.
[21] We only present the most promising solutions.



(a) Distance Driven (m)

(b) Elapsed Time (ns)

(c) Time to Pick-up (s)

(d) Delay (s)

Figure A.4: 08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

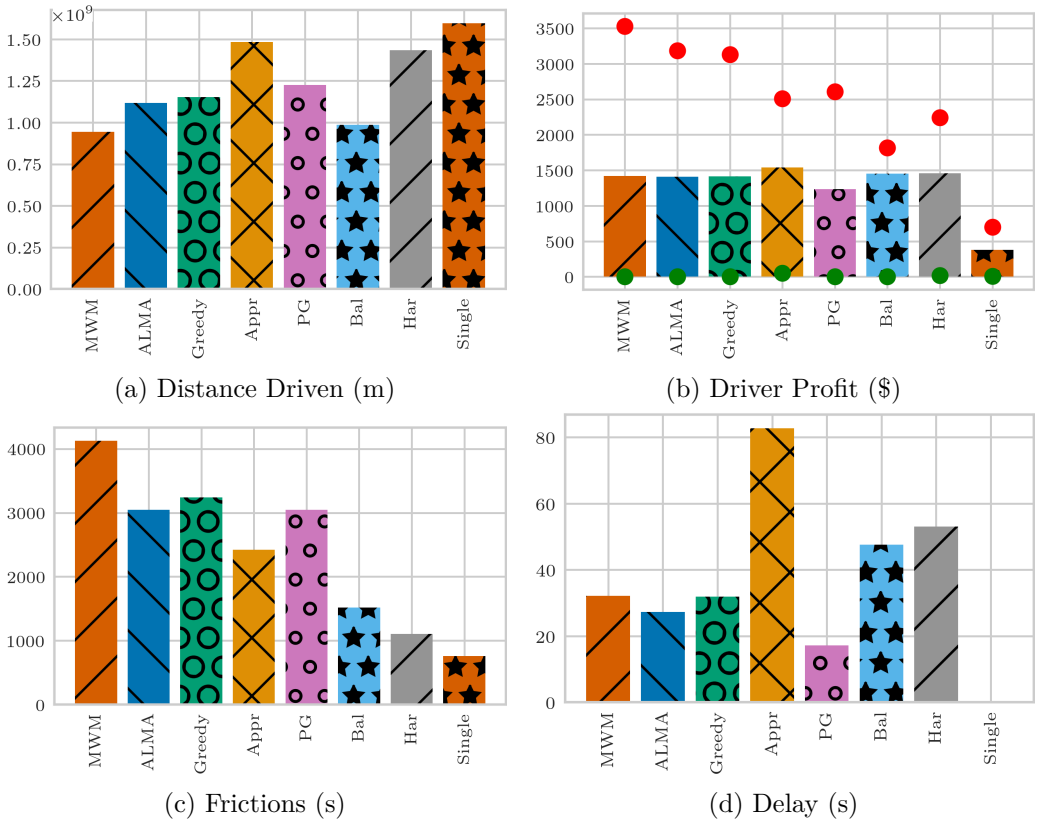
respectively. Figure A.6a shows that as we decrease the number of taxis, Bal loses its advantage, Greedy is pulling away from ALMA (9% worse than ALMA), while ALMA closes the gap to MWM (+17%).

Complexity:

To estimate the complexity, we measured the elapsed time of each algorithm. Greedy is the fastest one (Figure A.4b), closely followed by Har, Bal, and ALMA. ALMA is inherently decentralized. The red overlay denotes the parallel time for ALMA, which is 2.5 orders of magnitude faster than Greedy.

Time to Pick-up:

MWM exhibits exceptionally low time to pick-up (Figure A.4c), lower than the single ride baseline. ALMA, Greedy, and Bal have +69%, +76%, and +33% compared to MWM, respectively. As before, Figure A.6b shows that as we decrease the number of taxis, Bal loses its advantage, and Greedy is pulling further away from ALMA. Note that to improve visualization, we removed DC's pick-up time as it was one order of magnitude



(a) Distance Driven (m)

(b) Driver Profit ($)

(c) Frictions (s)

(d) Delay (s)

Figure A.5: 00:00 - 23:59 (full day), #Taxis = 5081 (base number). Manhattan, January 15, 2016

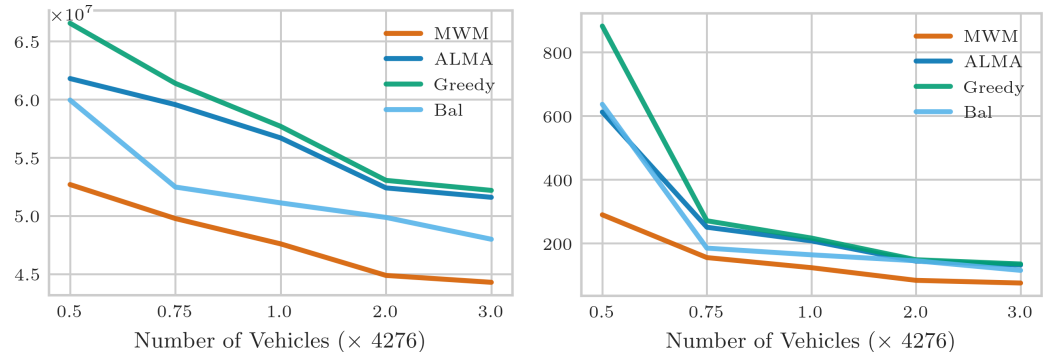
larger than Appr.

Delay:

PG exhibits the lowest delay (Figure A.4d), but this is because it makes 26% fewer shared rides than the rest of the high performing algorithms. ALMA has the smallest delay (−13% compared to MWM), with Greedy following at −1%, while Bal has +63% (both compared to MWM). As the number of taxis decrease (Figure A.6c), ALMA's gains increase further (−22% compared to MWM).

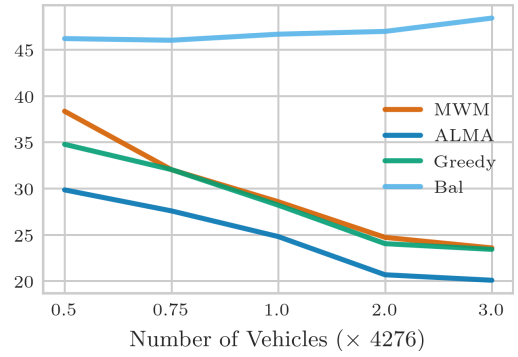
Figure A.6d depicts the cumulative delay, which is the sum of all delays described in Section A.4.1, namely the time to pair, time to pair with taxi, time to pick-up, and delay. An interesting observation is that reducing the fleet size from 12828 (×3.0 of the base number) to just 3207 (×0.75 of the base number) vehicles (75% reduction) results in only approximately 2 minutes of additional delay. This goes to show the great potential for efficiency gains such technologies have to offer.

Finally, we wanted to investigate the distribution of the achieved QoS metrics and,

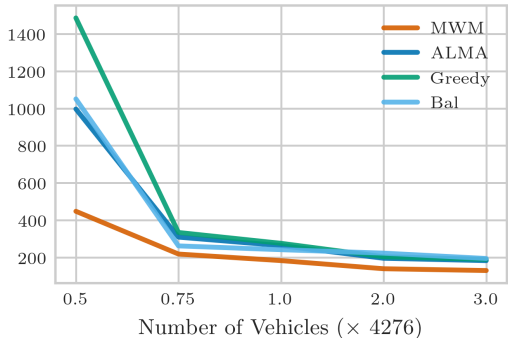


(a) Distance Driven (m)

(b) Time to Pick-up (s)

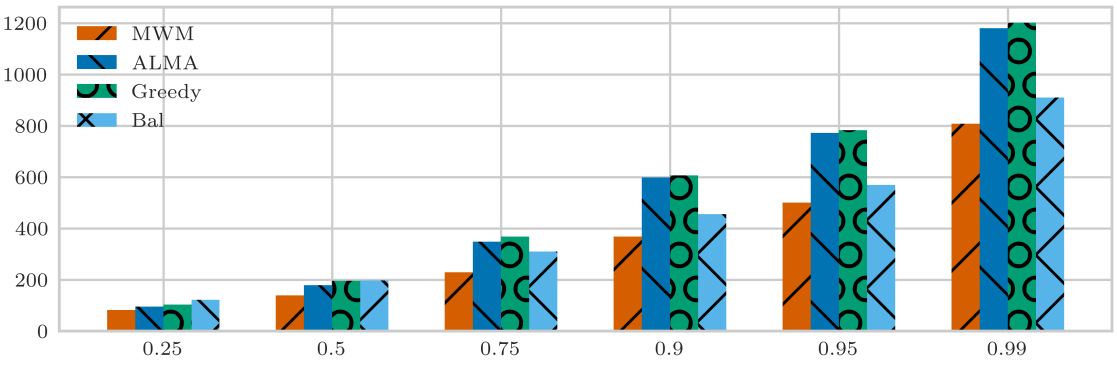


(c) Delay (s)



(d) Cumul. Delay (s)

Figure A.6: 08:00 - 09:00, #Taxis = $\{2138, 3207, 4276, 8552, 12828\}$. Manhattan, January 15, 2016



(a) Sequence of Percentiles for Cumulative Delay (s)

Figure A.7: 08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

consequently, the reliability/fairness of each CAR. As such, we plotted in Figure A.7a the sequence of percentiles[22] for the cumulative delay. As shown, the vast majority of the users (75%) experience cumulative delay close to the average value (only 46, 85, 92, 69 additional seconds of cumulative delay than the average value for MWM, ALMA, Greedy,

[22]Given a vector V of cumulative delays per request, the q-th percentile of V is the value $q/100$ of the way from the minimum to the maximum in a sorted copy of V.

Table A.3: Fairness of the Drivers' Profit.
08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

	Jain Index	Relative Diff. to MWM
MWM	0.71	0.0%
ALMA	0.75	6.0%
Greedy	0.75	6.9%
Appr	0.86	21.4%
PG	0.75	7.0%
GD	0.77	9.0%
Bal	0.90	28.2%
Har	0.84	19.1%
DC	0.19	-73.0%
k-Taxi	0.71	0.3%
Single	0.92	30.5%

and BAL, respectively). Of course, some of the users experiences high cumulative delay, but this is a small percentage of them. Specifically, less than 5% of requests experience a delay of more than 8.5, 13, 13, and 9.5 minutes for MWM, ALMA, Greedy, and BAL, respectively. Given the size and the average speed of taxi vehicles in Manhattan, such delays could be expected and, thus, acceptable; ultimately, it is up to the ridesharing platform to impose hard constraints and reject requests with potentially high delay.

Profit & Frictions:

Contrary to their performance in QoS metrics, GD, and Appr achieve the highest driver profit, 12% and 8% higher than MWM, respectively (although the low QoS and increased distance driven suggest low quality matchings, which can explain the higher revenue, yet deems them undesirable). Bal, and Har follow with $+2-3\%$. ALMA and Greedy achieve the similar profit to MWM. PG exhibits significantly worse results (-13%), due to the lower number of shared rides it matches.

Small differences in driver profit can have a significant impact on the platform's profit. There are 13587 taxis in NYC,[14] $67-85\%$ of which are on the road at one time (i.e., 9103 - 11549 taxis). The additional 2% profit of Bal translates to $32.3 additional revenue in a day. Multiplied by the total number of taxis, and assuming that the platform keeps 25% as commission,[5] this results in $73506 - $93258 additional revenue per day for the platform.

Figure A.5b also depicts the maximum (red dot), and minimum (green dot) value of a driver's profit. Closer to the mean maximum value suggests a fairer algorithm for the drivers. Moreover, it is worth noting that the minimum value for all the algorithms is zero, meaning that there are taxis which remain unutilized (in spite of the fact that the

number of taxis – in this scenario 5081 – is considerably lower than the current fleet size of yellow taxis).

In order to investigate the fairness of the distribution of profits amongst drivers, we calculated the Jain index (Jain et al., 1998), a well-established fairness metric (see Section 2.5). Table A.3 shows the Jain index, and the relative difference compared to MWM. Bal achieves the most fair allocation (excluding the single ride baseline[23]), with a Jain index of 0.9, closely followed by Appr with 0.86. MWM, ALMA, and Greedy all achieve relatively fair allocations, with the latter two achieving a 6% and 7% improvement over MWM.

Figure A.5c shows the driver frictions. Just like with the profit, k-server algorithms seem to outperform matching algorithms by far. Compared to MWM, Bal and Har achieve a 63% and 73% decrease, respectively, while ALMA and Greedy achieve a 26%, and 21% decrease, respectively. Given that we have a fixed supply, lower frictions indicate a more even distribution of rides amongst taxis.

It is important to note that while the results for all the other metrics are consistent when moving from the one hour test-case to the full day test-case, this is not true for the frictions (see Figures A.9j and A.12j and Tables A.7 and A.11). This is because taxis that serve zero or one rides are assumed to have zero friction by definition. Algorithms like Bal – which attempts to balance the distance driven by each taxi – will utilize each vehicle multiple times, even for the short time window of one hour. This results to a deceivingly high number in the frictions in the one hour test-case. As a matter of fact, the number of taxis that served less than two rides (and, thus, had zero friction) in the one hour test-case for Bal were 483. For MWM this number is 1368 (almost 3 times larger), for ALMA it is 1181, and for Greedy 1120. This is why we opted to present the frictions for the full day test-case in Figure A.5c.

Time to Pair with Taxi & Number of Shared Rides:

Excluding the test-case with the smallest taxi fleet ($\times 0.5$ the base number), the time to pair with taxi was zero, or close to zero, for all the evaluated algorithms. The latter comes to show the potential for efficiency gains and better utilization of resources using smart technologies. The reason for the low time to pair with a taxi is that, for the step (b) of the ridesharing problem (matching (shared) rides to taxis), we run the offline algorithms in a just-in-time (JiT) manner, i.e., every time the set of rides ($\mathcal{P}_t$) is not empty (see Section A.5.1). We opted to do so for simplicity – the alternative would require to run all combinations of batch sizes for both steps (a) and (b). Results from step (a), though, suggest that running in batches is more beneficial (running in batch size

[23]It is expected that the single ride baseline would result in a fair allocation of the profit, as it constantly utilizes the entire fleet of vehicles (see the definition of the base number of taxis in Section A.4.2).

Table A.4: Relocation Gains.

	MWM	ALMA	Greedy
Time to Pick-up	-48.95%	-55.18%	-55.03%
Time to Pick-up SD	-52.97%	-58.22%	-58.21%
Delay	-15.95%	-17.79%	-17.73%
Delay SD	-19.25%	-20.96%	-20.98%
Cumulative Delay	-38.37%	-43.23%	-43.11%
Total Distance	5.48%	6.25%	6.24%

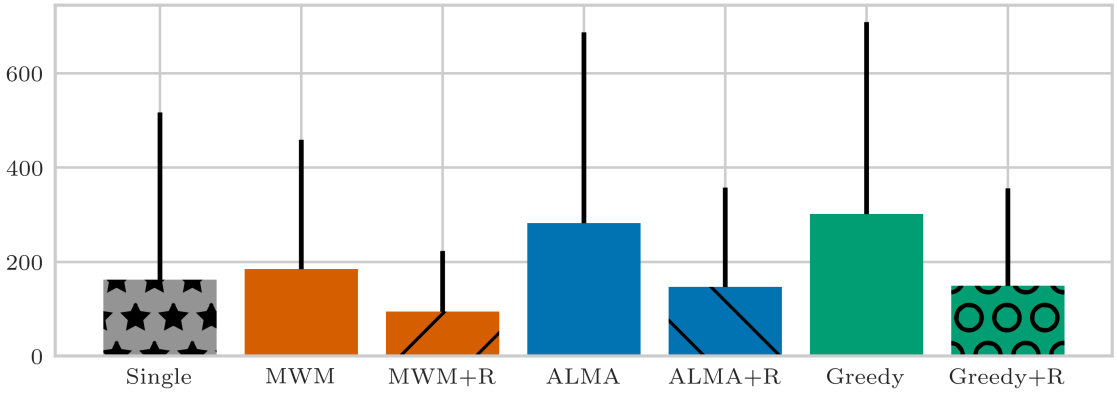


Figure A.8: Time to Pick-up (s) – End-To-End Solution
January 15, 2016 – 00:00 - 23:59 – Manhattan – #Taxis = 5081

of two minutes consistently outperformed the JiT version, see Section A.10). There is a clear trade-off: match with a taxi as soon as possible (JiT), and have a vehicle moving to pick-up the ride earlier, or wait (match in batches every x minutes), potentially allowing for better matches? Answering this question remains open for future work.

The number of shared rides is approximately the same for all the employed algorithms, with notable exception the PG which makes 26% fewer shared rides.

Relocation

The aim of any relocation strategy is to improve the spatial allocation of supply. Serving requests redistributes the taxis, resulting in an inefficient allocation. One can assume a 'lazy' approach, relocating vehicles only to serve requests. While this minimizes the cost of serving a request (e.g., distance driven, fuel, etc.), it results in sub-optimal QoS. Improving the QoS (especially the time to pick-up, since it highly correlates to passenger satisfaction, see Section A.4.1) plays a vital role in the growth of a company. Thus, *a crucial trade-off of any relocation scheme is improving the QoS metrics, while minimizing the excess distance driven.*

CARs with relocation successfully balance this trade-off (Table A.4). In particular, ALMA – the best performing overall – radically improves the QoS metrics by more than 50% (e.g., it decreases the pick-up time by 55%, and its standard deviation (SD) by 58%), while increasing the driving distance by only 6%. The cumulative delay is decreased by 43%.

As a final step, we evaluate *end-to-end* solutions, using MWM, ALMA, and Greedy to solve all three steps of the ridesharing problem. Figure A.8 depicts the time to pick-up (error bars denote one SD of uncertainty), a metric highly correlated to passenger satisfaction level (Tang et al., 2017; Brown, 2016b). We compare against the single ride baseline (no delay due to sharing a ride, see Section A.5.1). Once more, the proposed relocation scheme results in radical improvements, as the time to pick-up drops (compared to the single ride) from +14.09% to −41.76% for MWM, from +74.14% to −9.33% for ALMA, and from +86.10% to −7.97% for Greedy. This comes to show that *simple relocation schemes can eliminate the negative effects of ridesharing on the QoS.*

ALMA as an end-to-end CAR

While MWM seems to perform the best in the total distance driven, and most QoS metrics – which is reasonable since it makes optimal matches amongst passengers – it hard to scale and requires a centralized solution. In contrast, greedy approaches are appealing.[13] not only due to their low complexity, but also because real-time constraints dictate short planning windows which can diminish the benefit of batch optimization solutions compared to myopic approaches (Widdows et al., 2017) In fact, ALMA is of a greedy nature as well, albeit it utilizes a more intelligent backing-off scheme, thus there are scenarios where ALMA significantly outperforms the greedy, as proven by the simulation results. For example, in more challenging scenarios (smaller taxi fleet, or potentially different types of taxis) the smarter back off mechanism results in a more profound difference. Most importantly, ALMA was inherently developed for multi-agent applications. Agents make decisions locally, using completely uncoupled learning rules, and require only a 1-bit partial feedback, making it an ideal candidate for an *on-device* implementation. This is fundamentally different than a decentralized implementation of the Greedy algorithm for example. Even in decentralized algorithms, the number of communication rounds required grows with the size of the problem. However, in practice the real-time constraints impose a limit on the number of rounds, and thus on the size of the problem that can be solved within them.

A.8.3 High-level Analysis

Applying the modular approach we advocate, allowed us to thoroughly test a wide variety of state-of-the-art algorithms for ridesharing. When dealing with a multi-objective optimization problem, it is unreasonable to expect to identify an approach that outperforms

Table A.5: High level (qualitative) ranking of the evaluated CARs.
For each metric, the best performing CAR receives four stars ($\star\star\star\star$). Then, for the rest of the CARs, we compute the relative difference to the best performing one, i.e., $r = (x - OPT)/OPT$, where x is the value for the CAR considered, and OPT is the value achieved by the best performing CAR in the specific metric. If the relative difference is < 0.1, this CAR also receives four stars ($\star\star\star\star$), if it is $0.1 \leq r < 0.5$, the CAR receives three stars ($\star\star\star$), if it is $0.5 \leq r < 1$, the CAR receives two stars ($\star\star$), and finally, if $r \geq 1$, the CAR receives one star ($\star$).
The ranking is primarily based on the one hour test-case and base number of taxis (08:00 - 09:00, Manhattan, #Taxis = 4276, see Tables A.6, and A.7).
The same ranking holds in most cases for the full day test-case and base number of taxis (00:00 - 23:59, Manhattan, #Taxis = 5081, see Tables A.11, and A.12). The only exceptions are the ones awarded one additional star in the full day test-case, which is denote inside a parenthesis when relevant.
As mentioned in Section A.8.2, reporting frictions for the one hour test-case can be deceiving, thus we report the ranking based on the full day test-case. To make the distinction clear, the awarded stars are also inside a parenthesis. Algorithms that were too computationally heavy to run for a full day lack a ranking for the frictions.
The ranking for the WFA and HC (which we were able to run only in much smaller test-cases, see Tables A.8, and A.9 for WFA, and Tables A.17, and A.18 for HC) was extrapolated based on their performance against the baseline CARs that were common in all test-cases (MWM, ALMA, and Greedy).
The Time to Pair with a Taxi is not included as it was zero for most CARs (see Section A.8.2).
It is important to note that since the ranking is based on the relative difference in performance compared to the best performing CAR in each metric, it can be misleading in cases where the change is insignificant in terms of absolute values. Such a case is the Time to Pick-up, where the difference between MWM vs. Bal, or ALMA, or Greedy is only 1-2 minutes.
‡ALMA run in a decentralized manner is orders of magnitude faster than any other CAR. In order to allow for a clear ranking between the rest of the CARs, we performed the ranking based on the second best CAR (i.e., Greedy).
†Again, to allow for a proper ranking between the CARs, PG was not included in the delay metric because its notably low delay is only a result of making significantly fewer shared rides (26% less, see Section A.8.2).

	Operational Efficiency			Quality of Service				Drivers' Metrics	
	Distance Driven	Computational Complexity	Number of Shared Rides	Time to Pair	Time to Pick-up	Delay	Cumulative Delay	Profit	Frictions
Maximum Weight Matching (MWM)	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	(⋆)
ALtruistic MAtching Heuristic (ALMA)	⋆⋆⋆	⋆⋆⋆⋆ ‡	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	⋆⋆⋆	(⋆)
Greedy	⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆	⋆⋆⋆	⋆⋆(⋆)	⋆⋆⋆	(⋆)
Approximation (Appr)	⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆	⋆	⋆⋆⋆⋆	(⋆)
Postponed Greedy (PG)	⋆⋆⋆	⋆	⋆⋆⋆	⋆	⋆⋆	†	⋆⋆(⋆)	⋆⋆⋆	(⋆)
Greedy Dual (GD)	⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆	⋆⋆	⋆	⋆	⋆⋆⋆⋆	-
Balance (Bal)	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆(⋆)	⋆⋆	⋆⋆⋆	⋆⋆⋆⋆	(⋆⋆⋆)
Harmonic (Har)	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆(⋆)	⋆	⋆⋆⋆⋆	(⋆⋆⋆⋆)
Double Coverage (DC)	⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆	⋆	⋆⋆⋆⋆	-
Work Function (WFA)	⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆	⋆	⋆⋆⋆	⋆	⋆⋆⋆	-
***k*-Taxi**	⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆⋆⋆⋆	⋆	⋆⋆	⋆	⋆⋆⋆⋆	-
High Capacity (HC)	⋆⋆⋆⋆	⋆	⋆⋆⋆⋆	⋆	⋆⋆⋆	⋆⋆	⋆⋆⋆⋆	⋆⋆⋆	-

the competition across the board. Nevertheless, our findings provide convincing evidence to a ridesharing platform as to which CARs would be most suitable for a given set of objectives. Specifically: (i) CARs that rely on off-line (in-batches) maximum-weight matching solutions perform well on global efficiency and passenger related metrics, (ii) CARs based on k-server algorithms perform well on driver related metrics (e.g., Bal), (iii) lightweight CARs perform better in real-world, large-scale settings due to short planning windows imposed by the requirement to run in real-time, (iv) a simple, fine-grained relocation scheme based on the history of requests can significantly improve Quality of Service metrics by up to 50%, and finally, (v) we identify a scalable, on-device CAR based on ALMA that performs well across the board. A summary of the results can be found in Table A.5.

A.9 Conclusion

Managing transportation resources on a large scale remains a critical open problem. We initiate the *systematic* study of *Component Algorithms for Ridesharing* (CARs), a modular design methodology for ridesharing. To gain insight into the intricate dynamics of the problem, it is highly important to evaluate a diverse set of candidate solutions in settings designed to closely resemble reality. We evaluate a diverse set of candidate CARs (14 in total) – focused on the *key algorithmic components* of ridesharing – over 12 metrics, in settings designed to *closely resemble reality* in every aspect of the problem. To the best of our knowledge, this is the *first end-to-end evaluation of this magnitude*. We show the capacity of *simple relocation schemes* to improve QoS metrics radically, eliminating the negative effects of ridesharing, and identify an ALMA-based CAR that offers an *efficient* (across all metrics), *scalable*, *on-device*, end-to-end solution.

A.10 Simulation Results in Detail

We present in detail the results of Section A.8.2 including, but not limited to, larger test-cases (broader NYC area), and the omitted algorithms, graphs, and tables. For every metric we report the average value out of 8 runs.

Section A.10.1 08:00 - 09:00 – Manhattan: We begin with our small test-case: one hour (08:00 - 09:00), base number of taxis (i.e., 4276, see Section A.4.2), limited to Manhattan. Figure A.9, and Table A.6 depict all the evaluated metrics, while the latter also includes the standard deviation of each value. Finally, Table A.7 presents the relative difference (percentage of gain or loss) compared to MWM (first line of the table). In what follows, we will adhere to the same pattern, i.e., presenting two tables for the same evaluation, one containing the absolute values, and one presenting the relative difference compared to the algorithm in the first line of the table. We were able to run most of the algorithms in this test-case, except for WFA which we run only for $\{\times 0.5, \times 0.75\}$ the base number of taxis, and HC which is so computationally heavy, that we had to run a separate test-case of only 10 minutes (see Section A.10.5).

Offline algorithms (e.g., MWM, ALMA, Greedy) can be run either in a just-in-time (JiT) manner – i.e., when a request becomes critical – or in batches. The following two tables (Tables A.8, and A.9) evaluate the performance of each algorithm for each option. Given that our dataset has granularity of one minute, we run in batches of one, and two minutes. Moreover, due to the large number of requests, at least one request turns critical in every time-step. Thus, JiT and in batches of one minute produced the exact same results. To allow for the evaluation of every algorithm (except HC), we run the evaluation in a smaller scale, i.e., 2138 taxis ($\{\times 0.5\}$ the base number of taxis). These tables also include the results for the WFA algorithm. Every other result presented in this work assumes the best performing option for each of the algorithms (usually batch size of two minutes).

Figure A.10 shows the sequence of percentiles for the various delays introduced in Section A.4.1, while Table A.10 presents the complete results.

Finally, Figure A.11 shows that our results are robust to a varying number of vehicles (2138 - 12828).

Section A.10.2 00:00 - 23:59 (full day) – Manhattan: We continue to show that the results are robust to a larger time-scale. As before, Figure A.12, and Tables A.11, and A.12 depict all the evaluated metrics.

Sections A.10.3 08:00 - 09:00, and A.10.4 00:00 - 23:59 (full day) – Broader

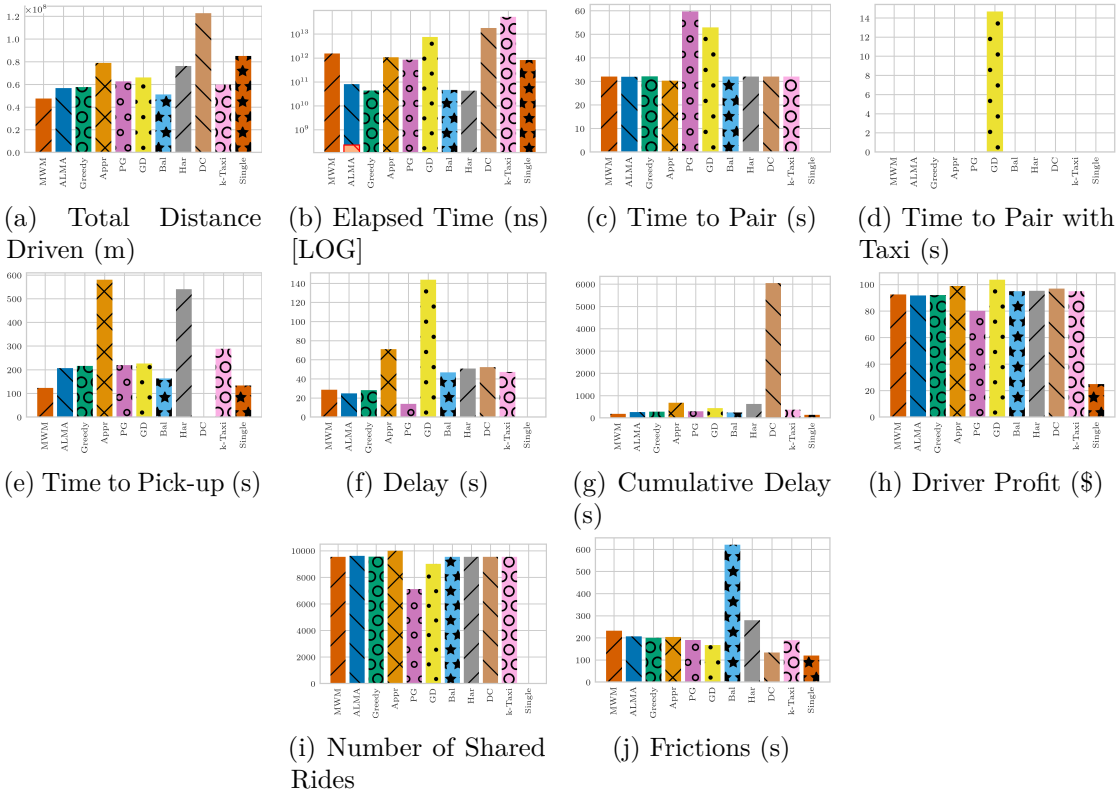
NYC Area: In the following two sections, we show that our results are robust to larger geographic areas, specifically in the broader NYC Area, including Manhattan, Bronx, Staten Island, Brooklyn, and Queens. Figure A.13, and Tables A.13, and A.14, and Figure A.14, and Tables A.15, and A.16 depict all the evaluated metrics, for one hour, and one day respectively.

Section A.10.5 08:00 - 08:10 – Manhattan: This is a limited test-case aimed to evaluate the HC algorithm, due to its high computational complexity. Figure A.15, and Tables A.17, and A.18 depict all the evaluated metrics.

Section A.10.6 Dynamic Vehicle Relocation – 00:00 - 23:59 (full day) – Manhattan: In this section, we present results on the step (c) of the Ridesharing problem: dynamic relocation. We fix an algorithm for steps (a), and (b) – specifically MWM – to allow for a common ground and a fair comparison, focused only on the relocation part. Figure A.16, and Tables A.19, and A.20 depict all the evaluated metrics.

Section A.10.7 End-To-End Solution – 00:00 - 23:59 (full day) – Manhattan: As a final step, we evaluate end-to-end solutions, using MWM, ALMA, and Greedy to solve all three of the steps of the Ridesharing problem. Figure A.17, and Tables A.21, and A.22 present all the evaluated metrics.

A.10.1 08:00 - 09:00 – Manhattan



(a) Total Distance Driven (m)
(b) Elapsed Time (ns) [LOG]
(c) Time to Pair (s)
(d) Time to Pair with Taxi (s)
(e) Time to Pick-up (s)
(f) Delay (s)
(g) Cumulative Delay (s)
(h) Driver Profit ($)
(i) Number of Shared Rides
(j) Frictions (s)

Figure A.9: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 4276 (base number).

Table A.6: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 4276 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	4.76E+07	0.00E+00	1.54E+12	0.00E+00	32.05	30.90	0.00	0.00	122.78	146.36	28.56	76.94	183.39	92.42	59.75	9.54E+03	0.00	232.30	420.13
ALMA	5.67E+07	1.14E+05	8.09E+10	4.36E+09	31.95	30.99	0.00	0.00	206.93	246.41	24.79	77.61	263.67	91.65	53.22	9.61E+03	10.46	206.59	387.78
Greedy	5.77E+07	8.97E+04	4.37E+10	2.73E+09	32.16	31.02	0.00	0.00	215.82	249.18	28.19	79.89	276.17	92.05	52.55	9.55E+03	18.52	200.17	381.71
Appr	7.90E+07	0.00E+00	1.06E+12	0.00E+00	30.29	30.19	0.00	0.00	580.48	427.45	71.13	133.34	681.90	98.89	40.52	1.00E+04	0.00	203.84	315.85
PG	6.27E+07	1.05E+05	8.55E+11	2.18E+10	59.69	43.21	0.00	0.00	219.16	282.64	13.77	61.12	292.62	80.08	45.69	7.11E+03	28.92	190.24	384.91
GD	6.62E+07	0.00E+00	7.54E+12	9.01E+10	52.91	32.09	14.67	18.27	225.96	267.47	143.82	313.95	437.36	103.65	56.84	9.01E+03	0.00	166.78	348.97
Bal	5.11E+07	5.16E+04	4.60E+10	2.09E+09	32.05	30.89	0.00	0.00	163.20	156.45	46.67	120.62	241.91	94.98	30.95	9.54E+03	0.00	621.14	490.55
Har	7.61E+07	2.47E+05	4.28E+10	2.41E+09	32.05	30.89	0.00	0.00	540.38	479.35	50.84	129.03	623.27	95.18	41.59	9.54E+03	0.00	279.55	282.88
DC	1.23E+08	5.41E+06	1.79E+13	1.05E+12	32.05	30.89	0.00	0.00	0.00	10458.42	52.29	125.51	6051.60	96.94	196.20	9.54E+03	0.00	133.54	349.94
k-Taxi	5.97E+07	2.31E+05	5.23E+13	3.00E+12	32.05	30.89	0.00	0.00	288.89	372.65	47.09	120.88	368.02	94.90	62.00	9.54E+03	0.00	188.68	343.92
Single	8.51E+07	0.00E+00	8.12E+11	0.00E+00	0.00	0.00	0.02	1.04	133.36	201.19	0.00	0.00	133.38	24.88	7.30	0.00E+00	0.00	119.80	291.01

Table A.7: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 4276 (base number). Each column presents the relative difference compared to the first line, i.e., the MWM (algorithm - MWM) / MWM, for each metric.

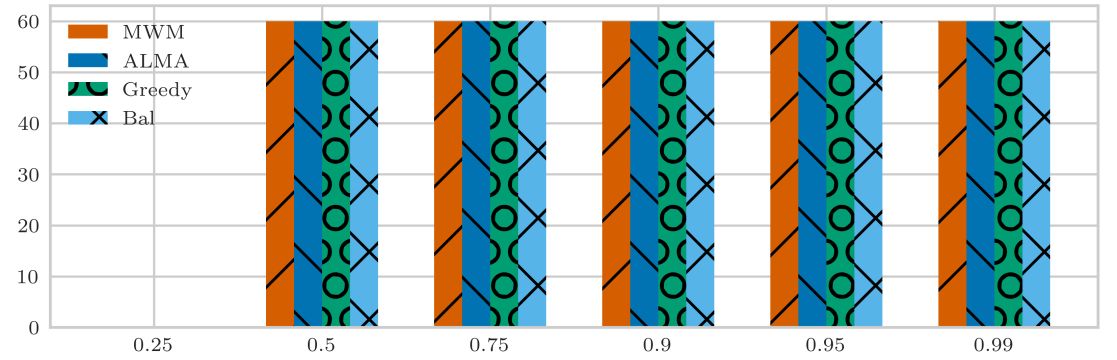
	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	0.00%	–	0.00%	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
ALMA	19.15%	–	-94.75%	–	-0.29%	0.30%	–	–	68.54%	68.36%	-13.19%	0.88%	43.78%	-0.84%	-10.93%	0.72%	–	-11.07%	-7.70%
Greedy	21.24%	–	-97.16%	–	0.35%	0.40%	–	–	75.78%	70.25%	-1.30%	3.84%	50.59%	-0.41%	-12.06%	0.08%	–	-13.83%	-9.14%
Appr	65.95%	–	-30.85%	–	-5.47%	-2.28%	–	–	372.79%	192.06%	149.01%	73.30%	271.83%	6.99%	-32.18%	4.81%	–	-12.25%	-24.82%
PG	31.69%	–	-44.50%	–	86.27%	39.86%	–	–	78.51%	93.12%	-51.81%	-20.56%	59.57%	-13.35%	-23.54%	-25.48%	–	-18.10%	-8.38%
GD	39.03%	–	389.79%	–	65.11%	3.87%	–	–	84.04%	82.75%	403.50%	308.04%	138.49%	12.15%	-4.87%	-5.63%	–	-28.21%	-16.94%
Bal	7.41%	–	-97.01%	–	0.00%	0.00%	–	–	32.92%	6.90%	63.39%	56.77%	31.91%	2.76%	-48.19%	0.00%	–	167.39%	16.76%
Har	59.98%	–	-97.22%	–	0.00%	0.00%	–	–	340.13%	227.52%	78.00%	67.70%	239.87%	2.98%	-30.39%	0.00%	–	20.34%	-32.67%
DC	158.13%	–	1061.10%	–	0.00%	0.00%	–	–	-100.00%	7045.82%	83.09%	63.13%	3199.91%	4.88%	228.37%	0.00%	–	-42.51%	-16.71%
k-Taxi	25.48%	–	3293.85%	–	0.00%	0.00%	–	–	135.29%	154.62%	64.85%	57.11%	100.68%	2.68%	3.76%	0.00%	–	-18.78%	-18.14%
Single	78.81%	–	-47.28%	–	-100.00%	-100.00%	–	–	8.62%	37.46%	-100.00%	-100.00%	-27.27%	-73.08%	-87.78%	-100.00%	–	-48.43%	-30.73%

Table A.8: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 2138.
Offline algorithms are run either in Just-in-Time (JiT) manner, or in batches (with batch size 1, or 2 min). Because of the density of the dataset, requests become critical every time-step, thus JiT is the same as in batches with batch size 1.

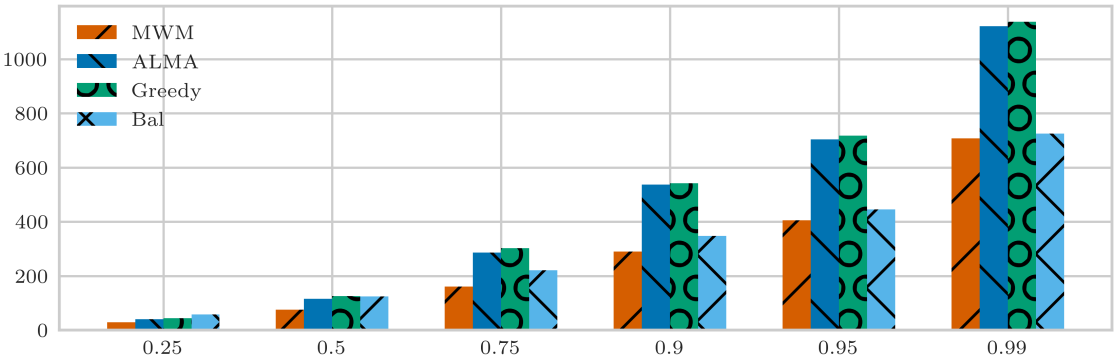
	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM (1)	5.46E+07	0.00E+00	9.91E+10	0.00E+00	5.17	18.08	102.26	407.94	318.23	542.59	45.92	123.45	471.57	190.44	52.46	9.72E+03	0.00	29.68	12.38
MWM (2)	5.27E+07	0.00E+00	1.45E+11	0.00E+00	32.05	30.90	88.64	308.08	288.94	441.85	38.34	112.20	447.97	187.50	49.91	9.54E+03	0.00	34.32	16.06
ALMA (1)	6.30E+07	1.38E+05	3.96E+10	4.12E+09	3.97	15.71	356.07	614.16	654.41	806.25	39.48	119.40	1053.93	188.79	36.01	9.85E+03	8.36	29.43	9.91
ALMA (2)	6.18E+07	1.02E+05	6.02E+10	7.57E+09	31.91	30.92	323.09	566.04	611.59	758.26	29.84	102.38	996.43	184.66	35.59	9.62E+03	6.88	30.00	10.28
Greedy (1)	6.76E+07	2.28E+05	6.78E+09	1.68E+09	4.41	16.52	577.85	706.95	932.92	813.98	44.64	121.06	1559.82	189.99	36.36	9.82E+03	6.13	29.54	9.76
Greedy (2)	6.66E+07	1.01E+05	1.31E+10	3.75E+09	32.14	31.04	536.36	668.99	881.67	783.43	34.77	106.58	1484.94	185.72	36.99	9.55E+03	17.72	29.97	10.00
Appr (1)	8.04E+07	0.00E+00	5.41E+11	0.00E+00	27.94	30.07	852.44	1048.80	1454.91	1185.15	71.59	137.37	2406.88	198.00	36.05	1.00E+04	0.00	45.28	18.08
Appr (2)	8.05E+07	0.00E+00	5.42E+11	0.00E+00	30.29	30.19	804.67	995.86	1410.40	1145.77	69.81	128.89	2315.17	197.35	35.61	1.00E+04	0.00	29.69	10.13
PG	6.74E+07	1.17E+05	5.20E+11	1.81E+10	59.53	43.27	297.99	764.61	664.30	1053.38	19.90	107.57	1041.73	161.90	39.97	7.12E+03	31.39	29.56	8.29
GD	6.92E+07	0.00E+00	7.11E+12	3.28E+11	52.91	32.09	358.38	801.79	626.96	945.13	153.49	333.21	1191.73	210.00	54.47	9.01E+03	0.00	30.21	9.70
Bal (1)	6.22E+07	9.71E+04	1.35E+10	4.56E+09	5.17	18.08	403.38	535.51	725.18	637.49	55.53	134.36	1189.26	192.96	41.38	9.72E+03	0.00	30.23	11.05
Bal (2)	5.99E+07	1.38E+05	3.42E+10	7.09E+09	32.05	30.89	336.45	466.17	635.90	569.04	46.20	120.74	1050.59	189.53	41.81	9.54E+03	0.00	31.76	12.20
Har (1)	8.10E+07	2.36E+05	1.37E+10	4.70E+09	5.17	18.08	946.56	1058.46	1555.42	1190.40	61.98	146.65	2569.12	194.22	49.97	9.72E+03	0.00	29.45	9.67
Har (2)	7.96E+07	2.87E+05	3.36E+10	6.90E+09	32.05	30.89	906.49	1024.29	1501.74	1153.03	51.28	130.04	2491.55	190.37	51.20	9.54E+03	0.00	29.65	9.65
DC (1)	1.41E+08	1.90E+06	1.16E+13	5.48E+11	5.17	18.08	0.00	0.00	7660.58	10890.62	62.70	142.71	7728.45	192.45	280.85	9.72E+03	0.00	85.76	269.59
DC (2)	1.38E+08	1.71E+06	8.99E+12	3.74E+11	32.05	30.89	0.00	0.00	7290.07	10196.66	51.77	124.76	7373.89	188.37	272.89	9.54E+03	0.00	93.37	287.47
k-Taxi (1)	7.10E+07	3.00E+05	3.41E+12	2.12E+11	5.17	18.08	646.48	528.54	1099.19	702.35	58.32	141.50	1809.15	193.48	46.97	9.72E+03	0.00	29.33	9.74
k-Taxi (2)	6.92E+07	2.04E+05	4.32E+12	4.05E+11	32.05	30.89	586.22	491.05	1020.87	670.76	48.51	126.93	1687.65	189.90	48.16	9.54E+03	0.00	30.14	10.10
WFA (1)	8.48E+07	3.25E+05	1.28E+14	6.77E+12	5.17	18.08	0.00	0.00	30966.54	40847.98	64.04	145.43	31035.75	194.69	597.69	9.72E+03	0.00	63.79	254.48
WFA (2)	8.09E+07	0.00E+00	9.69E+13	0.00E+00	32.05	30.90	0.00	0.00	28964.94	39066.93	51.85	127.05	29048.84	190.49	593.79	9.54E+03	0.00	69.68	264.10
Single	8.63E+07	0.00E+00	2.02E+12	0.00E+00	0.00	0.00	700.50	1317.51	843.80	1444.94	0.00	0.00	1544.30	49.72	6.90	0.00E+00	0.00	29.41	6.11
Random	1.14E+08	2.56E+05	6.26E+09	1.42E+09	3.61	15.05	1963.08	1457.54	2903.86	1580.37	161.07	328.92	5031.62	222.47	47.27	9.88E+03	6.54	29.56	9.37

Table A.9: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 2138.
Offline algorithms are run either in Just-in-Time (JiT) manner, or in batches (with batch size 1, or 2 min). Because of the density of the dataset, requests become critical every time-step, thus JiT is the same as in batches with batch size 1.
Each column presents the relative difference compared to the first line, i.e., MWM of batch size one (algorithm - MWM(1)) / MWM(1), for each metric.

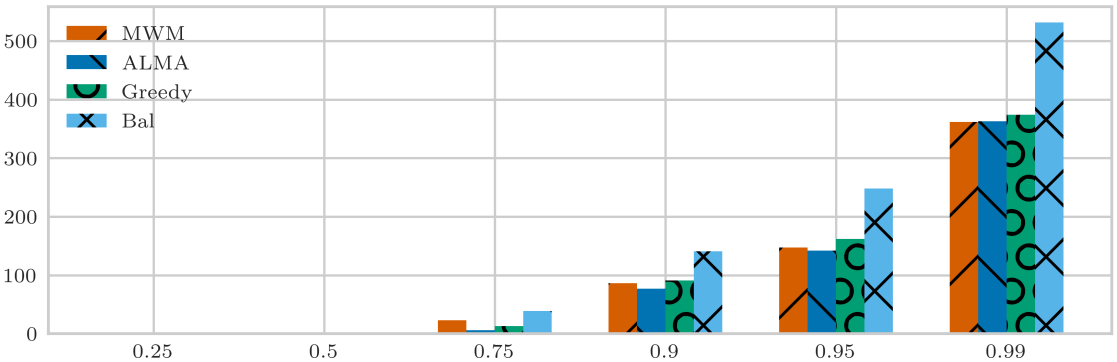
	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM (1)	0.00%	–	0.00%	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
MWM (2)	-3.44%	–	46.61%	–	520.26%	70.89%	-13.32%	-24.48%	-9.20%	-18.57%	-16.49%	-9.12%	-5.00%	-1.54%	-4.86%	-1.82%	–	15.65%	29.68%
ALMA (1)	15.50%	–	-60.03%	–	-23.20%	-13.08%	248.20%	50.55%	105.64%	48.59%	-14.01%	-3.29%	123.49%	-0.87%	-31.36%	1.36%	–	-0.84%	-20.00%
ALMA (2)	13.27%	–	-39.29%	–	517.65%	71.03%	215.95%	38.76%	92.19%	39.75%	-35.02%	-17.07%	111.30%	-3.04%	-32.16%	-1.03%	–	1.08%	-16.96%
Greedy (1)	23.91%	–	-93.16%	–	-14.58%	-8.61%	465.07%	73.30%	193.16%	50.02%	-2.78%	-1.94%	230.77%	-0.24%	-30.70%	1.01%	–	-0.47%	-21.15%
Greedy (2)	21.98%	–	-86.81%	–	522.14%	71.70%	424.50%	63.99%	177.06%	44.39%	-24.29%	-13.67%	214.89%	-2.48%	-29.48%	-1.72%	–	0.99%	-19.22%
Appr (1)	47.32%	–	445.49%	–	440.80%	66.34%	733.59%	157.10%	357.19%	118.42%	55.92%	11.28%	410.39%	3.97%	-31.28%	2.92%	–	52.57%	46.03%
Appr (2)	47.52%	–	447.16%	–	486.30%	66.99%	686.88%	144.12%	343.20%	111.17%	52.04%	4.40%	390.95%	3.63%	-32.13%	2.90%	–	0.02%	-18.16%
PG	23.56%	–	424.86%	–	1052.23%	139.31%	191.40%	87.43%	108.75%	94.14%	-56.65%	-12.87%	120.90%	-14.99%	-23.81%	-26.70%	–	-0.41%	-33.08%
GD	26.84%	–	7071.20%	–	924.12%	77.50%	250.45%	96.55%	97.01%	74.19%	234.27%	169.91%	152.71%	10.27%	3.83%	-7.35%	–	1.79%	-21.63%
Bal (1)	14.02%	–	-86.42%	–	0.00%	0.00%	294.47%	31.27%	127.88%	17.49%	20.94%	8.83%	152.19%	1.33%	-21.13%	0.00%	–	1.87%	-10.75%
Bal (2)	9.88%	–	-65.49%	–	520.26%	70.88%	229.01%	14.28%	99.83%	4.87%	0.61%	-2.20%	122.78%	-0.48%	-20.31%	-1.82%	–	7.00%	-1.50%
Har (1)	48.42%	–	-86.16%	–	0.00%	0.00%	825.64%	159.47%	388.77%	119.39%	34.98%	18.79%	444.80%	1.98%	-4.75%	0.00%	–	-0.76%	-21.90%
Har (2)	45.93%	–	-66.14%	–	520.26%	70.88%	786.45%	151.09%	371.90%	112.50%	11.68%	5.34%	428.35%	-0.04%	-2.41%	-1.82%	–	-0.08%	-22.10%
DC (1)	159.34%	–	11636.41%	–	0.00%	0.00%	-100.00%	-100.00%	2307.25%	1907.15%	36.55%	15.59%	1538.87%	1.05%	435.35%	0.00%	–	188.95%	2076.91%
DC (2)	152.49%	–	8969.76%	–	520.26%	70.88%	-100.00%	-100.00%	2190.83%	1779.25%	12.75%	1.06%	1463.68%	-1.09%	420.17%	-1.82%	–	214.60%	2221.34%
k-Taxi (1)	30.13%	–	3339.41%	–	0.00%	0.00%	532.19%	29.56%	245.41%	29.44%	27.00%	14.61%	283.64%	1.60%	-10.47%	0.00%	–	-1.18%	-21.38%
k-Taxi (2)	26.82%	–	4258.51%	–	520.26%	70.88%	473.26%	20.37%	220.80%	23.62%	5.65%	2.81%	257.88%	-0.28%	-8.20%	-1.82%	–	1.54%	-18.41%
WFA (1)	55.42%	–	129083.20%	–	0.00%	0.00%	-100.00%	-100.00%	9630.90%	7428.32%	39.48%	17.80%	6481.32%	2.23%	1039.29%	0.00%	–	114.92%	1954.90%
WFA (2)	48.26%	–	97654.76%	–	520.26%	70.89%	-100.00%	-100.00%	9001.91%	7100.07%	12.93%	2.91%	6059.99%	0.03%	1031.86%	-1.82%	–	134.77%	2032.61%
Single	58.20%	–	1935.09%	–	-100.00%	-100.00%	585.01%	222.97%	165.16%	166.30%	-100.00%	-100.00%	227.48%	-73.89%	-86.85%	-100.00%	–	-0.91%	-50.64%
Random	108.44%	–	-93.69%	–	-30.10%	-16.76%	1819.69%	257.30%	812.51%	191.26%	250.78%	166.43%	966.99%	16.82%	-9.89%	1.60%	–	-0.40%	-24.38%



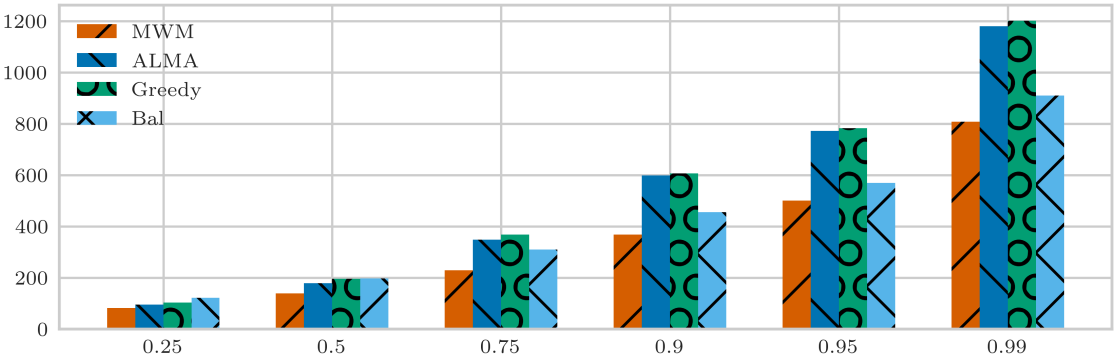
(a) Sequence of Percentiles for Time to Pair (s)



(b) Sequence of Percentiles for Time to Pick-up (s)



(c) Sequence of Percentiles Delay (s)



(d) Sequence of Percentiles for Cumulative Delay (s)

Figure A.10: 08:00 - 09:00, #Taxis = 4276 (base number). Manhattan, January 15, 2016

Table A.10: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 4276 (base number).
(Given a vector V, the q-th percentile of V is the value $q/100$ of the way from the minimum to the maximum in a sorted copy of V.)

(a) Sequence of Percentiles for Time to Pair (s).

	0.25	**0.5**	**0.75**	**0.9**	**0.95**	**0.99**
MWM	0	60	60	60	60	60
ALMA	0	60	60	60	60	60
Greedy	0	60	60	60	60	60
Appr	0	60	60	60	60	60
PG	60	60	60	120	120	180
GD	40	52	59	105	118	173
Bal	0	60	60	60	60	60
Har	0	60	60	60	60	60
DC	0	60	60	60	60	60
k-Taxi	0	60	60	60	60	60
Single	0	0	0	0	0	0

(b) Sequence of Percentiles for Time to Pair with Taxi (s).

	0.25	**0.5**	**0.75**	**0.9**	**0.95**	**0.99**
MWM	0	0	0	0	0	0
ALMA	0	0	0	0	0	0
Greedy	0	0	0	0	0	0
Appr	0	0	0	0	0	0
PG	0	0	0	0	0	0
GD	2	8	20	39	55	84
Bal	0	0	0	0	0	0
Har	0	0	0	0	0	0
DC	0	0	0	0	0	0
k-Taxi	0	0	0	0	0	0
Single	0	0	0	0	0	0

(c) Sequence of Percentiles for Time to Pick-up (s).

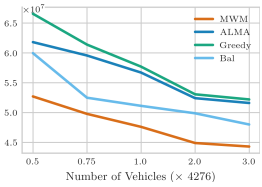
	0.25	**0.5**	**0.75**	**0.9**	**0.95**	**0.99**
MWM	29	76	161	290	406	707
ALMA	40	115	286	537	704	1121
Greedy	44	126	302	542	718	1138
Appr	288	454	737	1148	1470	2126
PG	38	101	298	594	803	1300
GD	56	133	296	539	749	1311
Bal	58	124	221	347	445	725
Har	197	406	734	1185	1521	2223
DC	714	2811	7017	13954	21352	57870
k-Taxi	65	162	348	676	939	1705
Single	27	59	137	373	575	910

(d) Sequence of Percentiles for Delay (s).

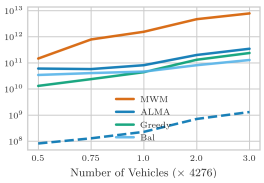
	0.25	**0.5**	**0.75**	**0.9**	**0.95**	**0.99**
MWM	0	0	23	86	147	362
ALMA	0	0	6	77	142	363
Greedy	0	0	13	91	162	374
Appr	0	17	100	197	280	568
PG	0	0	0	27	82	279
GD	0	0	146	413	743	1608
Bal	0	0	39	141	248	532
Har	0	0	45	157	269	556
DC	0	0	49	166	274	537
k-Taxi	0	0	39	142	250	537
Single	0	0	0	0	0	0

(e) Sequence of Percentiles for Cumulative Delay (s).

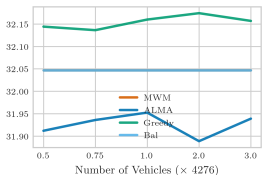
	0.25	**0.5**	**0.75**	**0.9**	**0.95**	**0.99**
MWM	82	139	229	369	501	808
ALMA	95	179	348	599	772	1181
Greedy	103	196	368	607	783	1202
Appr	372	547	842	1288	1626	2345
PG	99	180	386	690	906	1398
GD	187	318	538	911	1249	1971
Bal	122	198	310	456	570	910
Har	266	487	826	1296	1644	2364
DC	803	2892	7099	14036	21446	57946
k-Taxi	133	240	440	775	1052	1846
Single	27	59	137	373	575	910



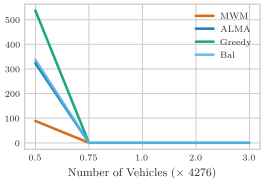
(a) Total Distance Driven (m)



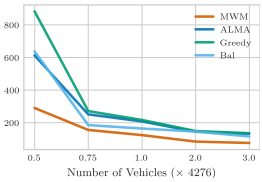
(b) Elapsed Time (ns) [LOG]



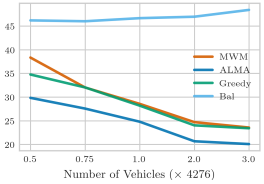
(c) Time to Pair (s)



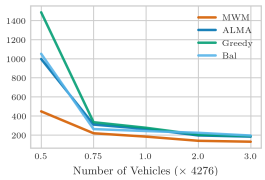
(d) Time to Pair with Taxi (s)



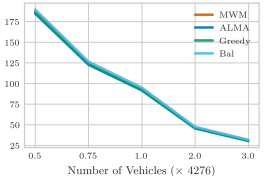
(e) Time to Pick-up (s)



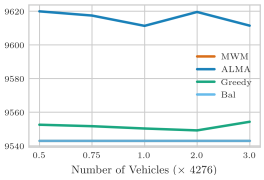
(f) Delay (s)



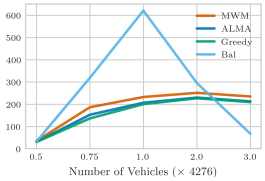
(g) Cumulative Delay (s)



(h) Driver Profit ($)



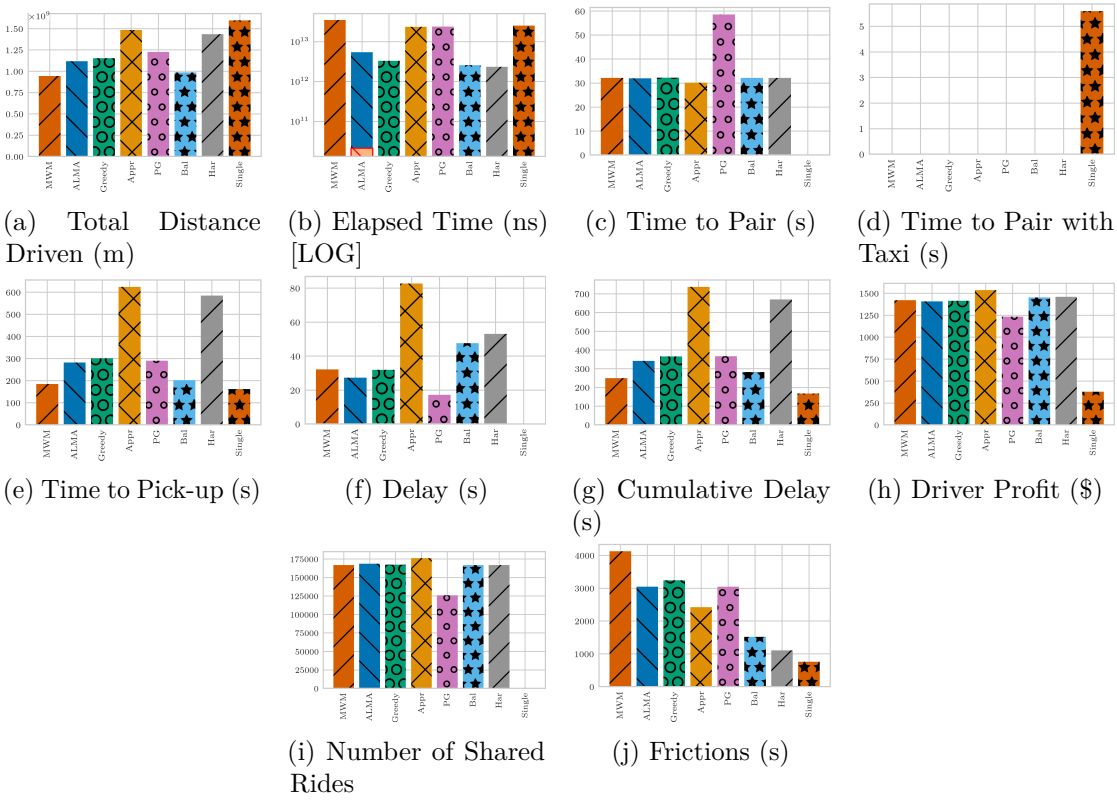
(i) Number of Shared Rides



(j) Frictions (s)

Figure A.11: January 15, 2016 – 08:00 - 09:00 – Manhattan – Varying #Taxis = $\{2138, 3207, 4276, 8552, 12828\}$.

A.10.2 00:00 - 23:59 (full day) – Manhattan



(a) Total Distance Driven (m)
(b) Elapsed Time (ns) [LOG]
(c) Time to Pair (s)
(d) Time to Pair with Taxi (s)
(e) Time to Pick-up (s)
(f) Delay (s)
(g) Cumulative Delay (s)
(h) Driver Profit ($)
(i) Number of Shared Rides
(j) Frictions (s)

Figure A.12: January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number).

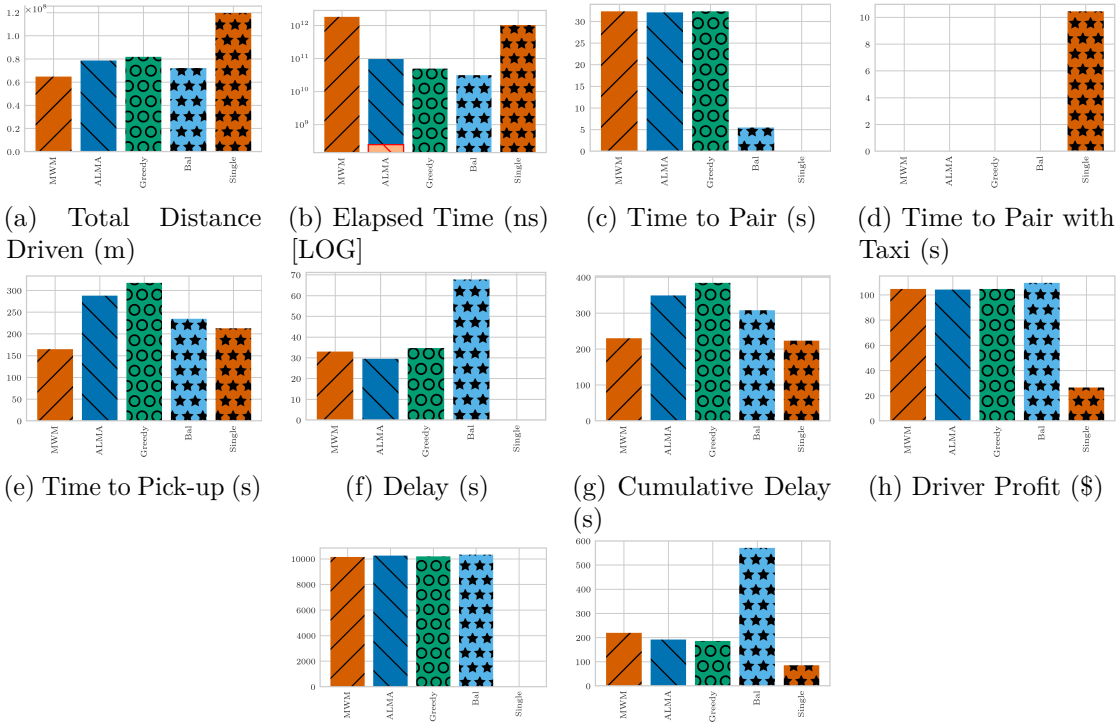
Table A.11: January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	9.45E+08	0.00E+00	3.48E+13	0.00E+00	32.10	30.84	0.00	0.00	184.55	274.34	32.14	87.69	248.79	1420.37	895.19	1.67E+05	0.00	4127.47	10597.84
ALMA	1.12E+09	2.73E+05	5.42E+12	3.31E+11	31.98	31.01	0.00	0.00	281.70	405.36	27.32	86.78	341.00	1406.70	736.46	1.68E+05	29.39	3047.45	7264.56
Greedy	1.15E+09	5.93E+05	3.30E+12	3.86E+11	32.18	31.05	0.00	0.00	301.04	407.70	31.93	90.56	365.15	1414.66	719.44	1.67E+05	26.29	3242.70	8167.00
Appr	1.48E+09	0.00E+00	2.37E+13	0.00E+00	30.14	30.18	0.00	0.00	624.13	473.31	82.75	156.13	737.02	1536.97	478.35	1.76E+05	0.00	2421.75	4505.68
PG	1.22E+09	4.64E+05	2.39E+13	1.50E+12	58.58	42.99	0.00	0.00	290.18	431.80	17.16	84.29	365.91	1234.17	603.66	1.26E+05	95.74	3044.98	8561.98
Bal	9.85E+08	2.42E+05	2.56E+12	1.39E+11	32.10	30.84	0.00	0.00	201.85	225.85	47.51	126.16	281.47	1452.66	107.31	1.67E+05	0.00	1516.33	221.62
Har	1.43E+09	1.46E+06	2.34E+12	3.85E+11	32.10	30.84	0.00	0.00	584.05	533.66	53.08	132.72	669.23	1458.51	187.05	1.67E+05	0.00	1106.16	246.76
Single	1.60E+09	0.00E+00	2.54E+13	0.00E+00	0.00	0.00	5.59	99.05	161.77	355.37	0.00	0.00	167.36	376.90	116.72	0.00E+00	0.00	756.43	1216.88

Table A.12: January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number). Each column presents the relative difference compared to the first line, i.e., the MWM (algorithm - MWM) / MWM, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	0.00%	–	0.00%	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
ALMA	18.29%	–	-84.43%	–	-0.39%	0.56%	–	–	52.64%	47.76%	-15.00%	-1.04%	37.06%	-0.96%	-17.73%	1.08%	–	-26.17%	-31.45%
Greedy	21.92%	–	-90.52%	–	0.23%	0.68%	–	–	63.12%	48.61%	-0.65%	3.27%	46.77%	-0.40%	-19.63%	0.41%	–	-21.44%	-22.94%
Appr	57.08%	–	-32.07%	–	-6.12%	-2.16%	–	–	238.19%	72.53%	157.50%	78.05%	196.24%	8.21%	-46.56%	5.69%	–	-41.33%	-57.48%
PG	29.57%	–	-31.48%	–	82.46%	39.38%	–	–	57.23%	57.40%	-46.61%	-3.87%	47.08%	-13.11%	-32.57%	-24.49%	–	-26.23%	-19.21%
Bal	4.24%	–	-92.64%	–	0.00%	0.00%	–	–	9.37%	-17.67%	47.85%	43.87%	13.13%	2.27%	-88.01%	0.00%	–	-63.26%	-97.91%
Har	51.67%	–	-93.28%	–	0.00%	0.00%	–	–	216.47%	94.53%	65.19%	51.36%	168.99%	2.68%	-79.10%	0.00%	–	-73.20%	-97.67%
Single	69.00%	–	-27.23%	–	-100.00%	-100.00%	–	–	-12.35%	29.54%	-100.00%	-100.00%	-32.73%	-73.46%	-86.96%	-100.00%	–	-81.67%	-88.52%

A.10.3 08:00 - 09:00 – Broader NYC Area (Manhattan, Bronx, Staten Island, Brooklyn, Queens)



(a) Total Distance Driven (m)
(b) Elapsed Time (ns) [LOG]
(c) Time to Pair (s)
(d) Time to Pair with Taxi (s)
(e) Time to Pick-up (s)
(f) Delay (s)
(g) Cumulative Delay (s)
(h) Driver Profit ($)
(i) Number of Shared Rides
(j) Frictions (s)

Figure A.13: January 15, 2016 – 08:00 - 09:00 – Broader NYC Area – #Taxis = 4972 (base number).

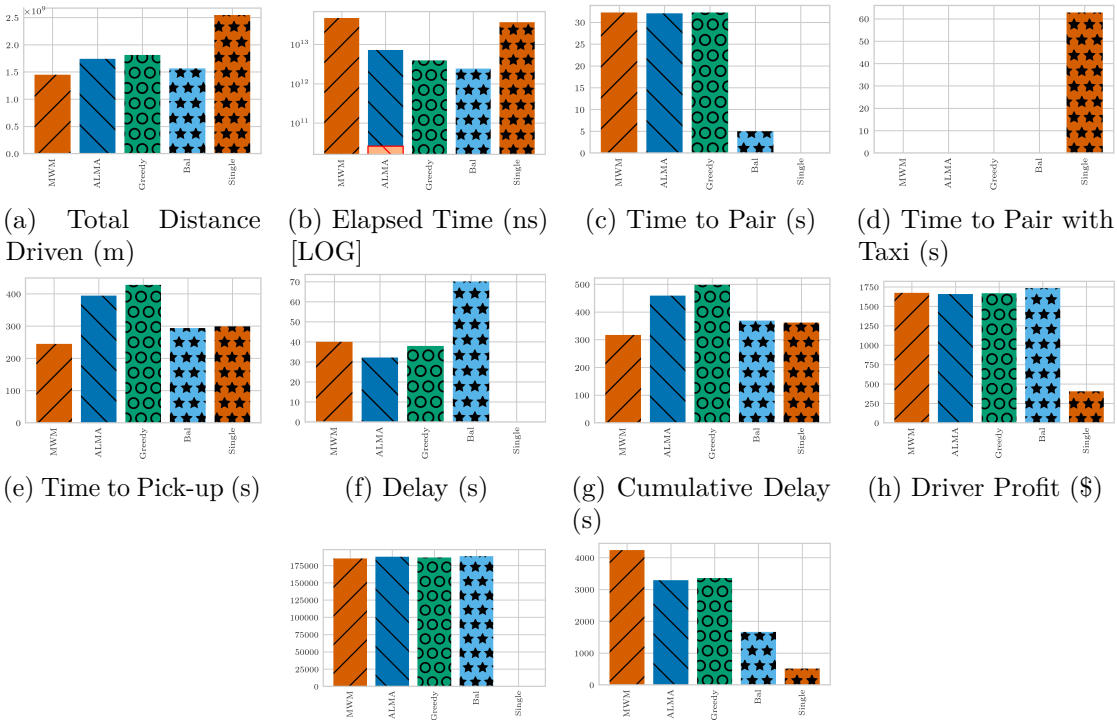
Table A.13: January 15, 2016 – 08:00 - 09:00 – Broader NYC Area – #Taxis = 4972 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	6.48E+07	0.00E+00	1.81E+12	0.00E+00	32.34	31.34	0.00	0.00	164.59	401.29	33.01	104.44	229.94	104.67	81.54	1.02E+04	0.00	219.19	415.07
ALMA	7.86E+07	2.48E+05	9.61E+10	1.02E+10	32.09	31.32	0.00	0.00	287.93	646.99	29.55	167.88	349.58	104.13	72.74	1.03E+04	5.88	191.41	379.85
Greedy	8.18E+07	2.96E+05	4.88E+10	7.00E+09	32.32	31.40	0.00	0.00	317.48	720.24	34.73	170.53	384.53	104.68	69.88	1.02E+04	12.28	185.35	374.04
Bal	7.22E+07	1.15E+05	3.09E+10	6.01E+09	5.49	18.97	0.00	0.00	234.85	428.34	67.79	219.01	308.14	109.57	65.24	1.03E+04	0.00	571.11	516.22
Single	1.20E+08	0.00E+00	1.03E+12	0.00E+00	0.00	0.00	10.44	83.19	212.61	577.74	0.00	0.00	223.06	26.37	9.21	0.00E+00	0.00	85.07	211.52

Table A.14: January 15, 2016 – 08:00 - 09:00 – Broader NYC Area – #Taxis = 4972 (base number). Each column presents the relative difference compared to the first line, i.e., the MWM (algorithm - MWM) / MWM, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	0.00%	–	0.00%	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
ALMA	21.37%	–	-94.68%	–	-0.75%	-0.08%	–	–	74.94%	61.23%	-10.48%	60.74%	52.03%	-0.52%	-10.79%	1.02%	–	-12.67%	-8.49%
Greedy	26.34%	–	-97.30%	–	-0.04%	0.20%	–	–	92.90%	79.48%	5.20%	63.27%	67.24%	0.01%	-14.31%	0.42%	–	-15.44%	-9.88%
Bal	11.44%	–	-98.29%	–	-83.01%	-39.48%	–	–	42.69%	6.74%	105.35%	109.70%	34.01%	4.68%	-20.00%	1.77%	–	160.56%	24.37%
Single	84.67%	–	-42.94%	–	-100.00%	-100.00%	–	–	29.18%	43.97%	-100.00%	-100.00%	-2.99%	-74.80%	-88.71%	-100.00%	–	-61.19%	-49.04%

A.10.4 00:00 - 23:59 (full day) – Broader NYC Area (Manhattan, Bronx, Staten Island, Brooklyn, Queens)



(a) Total Distance Driven (m)
(b) Elapsed Time (ns) [LOG]
(c) Time to Pair (s)
(d) Time to Pair with Taxi (s)
(e) Time to Pick-up (s)
(f) Delay (s)
(g) Cumulative Delay (s)
(h) Driver Profit ($)
(i) Number of Shared Rides
(j) Frictions (s)

Figure A.14: January 15, 2016 – 00:00 - 23:59 (full day) – Broader NYC Area – #Taxis = 6533 (base number).

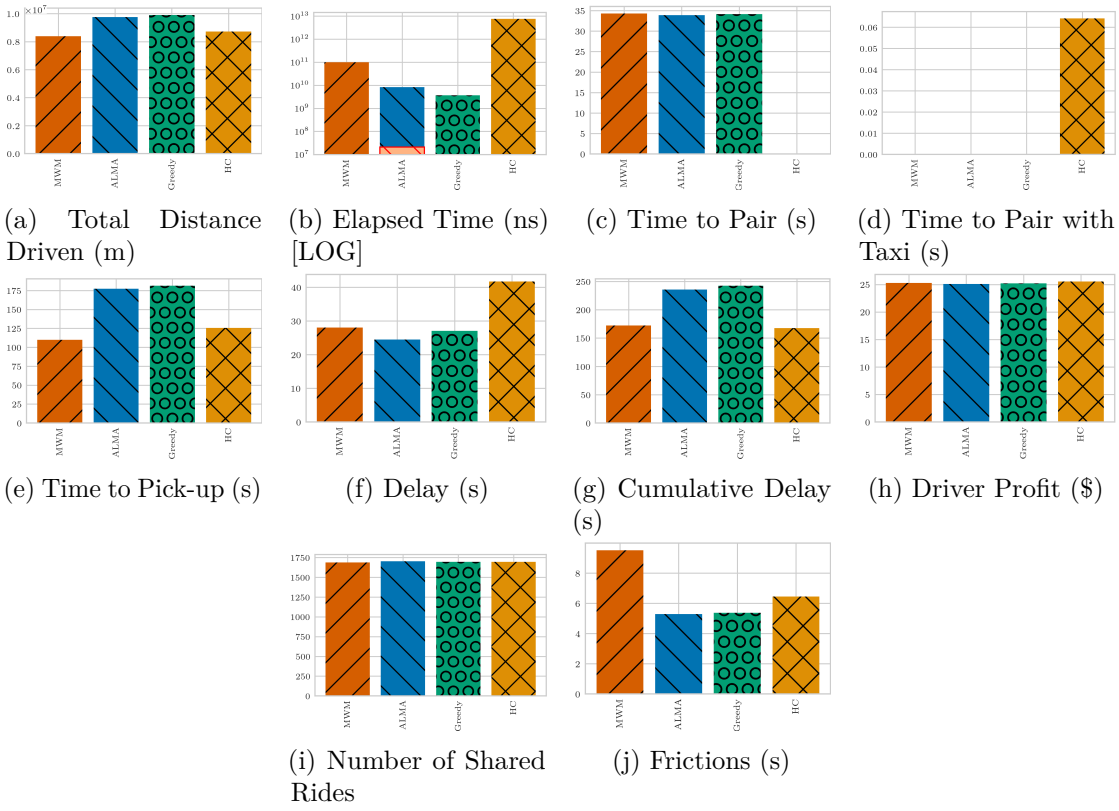
Table A.15: January 15, 2016 – 00:00 - 23:59 (full day) – Broader NYC Area – #Taxis = 6533 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	1.45E+09	0.00E+00	4.71E+13	0.00E+00	32.26	31.24	0.00	0.00	244.90	433.03	40.01	131.68	317.17	1675.21	949.14	1.85E+05	0.00	4238.58	10671.94
ALMA	1.74E+09	5.28E+05	7.20E+12	1.37E+11	32.09	31.44	0.00	0.00	394.30	701.06	32.17	133.83	458.57	1659.01	741.69	1.88E+05	22.08	3286.78	8808.76
Greedy	1.81E+09	2.36E+06	3.92E+12	3.31E+11	32.28	31.49	0.00	0.00	427.74	750.89	37.92	137.07	497.93	1667.74	672.62	1.87E+05	16.76	3357.00	9464.19
Bal	1.57E+09	2.60E+05	2.42E+12	1.60E+11	4.97	18.17	0.00	0.00	293.91	457.48	70.17	216.02	369.04	1736.02	153.93	1.89E+05	0.00	1666.22	570.20
Single	2.55E+09	0.00E+00	3.70E+13	0.00E+00	0.00	0.00	62.85	753.27	298.52	1147.60	0.00	0.00	361.37	405.82	80.73	0.00E+00	0.00	512.34	437.98

Table A.16: January 15, 2016 – 00:00 - 23:59 (full day) – Broader NYC Area – #Taxis = 6533 (base number). Each column presents the relative difference compared to the first line, i.e., the MWM (algorithm - MWM) / MWM, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	0.00%	–	0.00%	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
ALMA	20.13%	–	-84.69%	–	-0.53%	0.64%	–	–	61.01%	61.90%	-19.59%	1.63%	44.58%	-0.97%	-21.86%	1.34%	–	-22.46%	-17.46%
Greedy	25.04%	–	-91.66%	–	0.05%	0.80%	–	–	74.66%	73.41%	-5.22%	4.09%	56.99%	-0.45%	-29.13%	0.68%	–	-20.80%	-11.32%
Bal	8.15%	–	-94.85%	–	-84.61%	-41.85%	–	–	20.01%	5.65%	75.38%	64.05%	16.35%	3.63%	-83.78%	1.83%	–	-60.69%	-94.66%
Single	75.86%	–	-21.44%	–	-100.00%	-100.00%	–	–	21.90%	165.02%	-100.00%	-100.00%	13.94%	-75.78%	-91.49%	-100.00%	–	-87.91%	-95.90%

A.10.5 08:00 - 08:10 – Manhattan



(a) Total Distance Driven (m)
(b) Elapsed Time (ns) [LOG]
(c) Time to Pair (s)
(d) Time to Pair with Taxi (s)
(e) Time to Pick-up (s)
(f) Delay (s)
(g) Cumulative Delay (s)
(h) Driver Profit ($)
(i) Number of Shared Rides
(j) Frictions (s)

Figure A.15: January 15, 2016 – 08:00 - 08:10 – Manhattan – #Taxis = 2779 (base number).

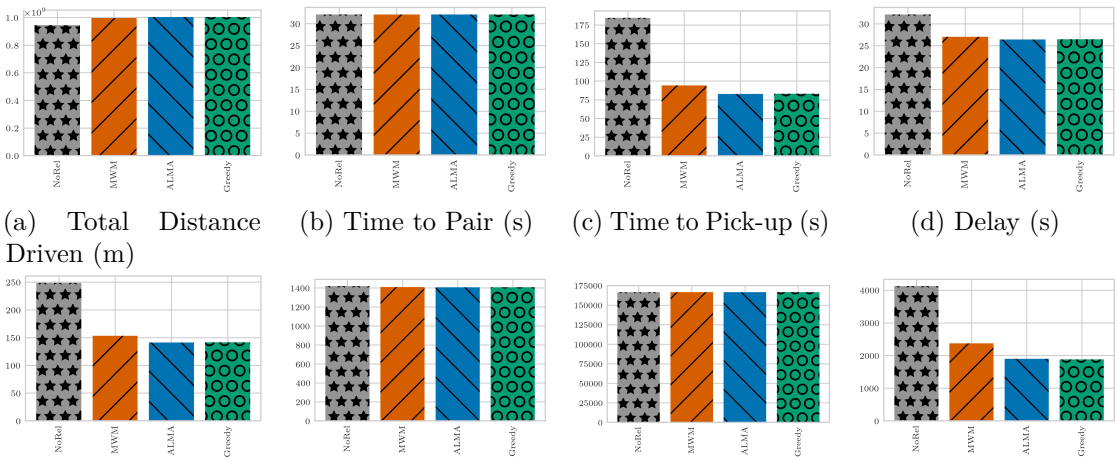
Table A.17: January 15, 2016 – 08:00 - 09:00 – Manhattan – #Taxis = 2779 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	8.38E+06	0.00E+00	9.92E+10	0.00E+00	34.32	30.93	0.00	0.00	109.83	125.56	28.08	80.29	172.23	25.30	29.59	1.69E+03	0.00	9.51	31.75
ALMA	9.76E+06	6.10E+04	8.28E+09	1.93E+09	33.88	30.71	0.00	0.00	177.21	216.99	24.49	75.40	235.58	25.09	26.38	1.70E+03	9.76	5.29	22.26
Greedy	9.91E+06	1.06E+04	3.66E+09	7.61E+08	34.16	30.88	0.00	0.00	181.43	216.19	27.05	74.92	242.64	25.19	26.14	1.70E+03	7.55	5.37	21.50
HC	8.72E+06	0.00E+00	7.52E+12	0.00E+00	0.13	4.02	0.06	2.78	125.69	155.09	41.77	106.45	167.65	25.56	29.61	1.70E+03	0.00	6.45	27.19

Table A.18: January 15, 2016 – 08:00 - 08:10 – Manhattan – #Taxis = 2779 (base number). Each column presents the relative difference compared to the first line, i.e., the MWM (algorithm - MWM) / MWM, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
MWM	0.00%	–	0.00%	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
ALMA	16.40%	–	-91.65%	–	-1.28%	-0.70%	–	–	61.34%	72.81%	-12.80%	-6.09%	36.78%	-0.83%	-10.87%	0.95%	–	-44.37%	-29.90%
Greedy	18.17%	–	-96.31%	–	-0.45%	-0.15%	–	–	65.18%	72.18%	-3.67%	-6.68%	40.88%	-0.43%	-11.65%	0.44%	–	-43.57%	-32.29%
HC	3.98%	–	7474.61%	–	-99.61%	-87.01%	–	–	14.44%	23.52%	48.74%	32.59%	-2.66%	1.02%	0.05%	0.47%	–	-32.16%	-14.37%

A.10.6 Dynamic Vehicle Relocation – 00:00 - 23:59 (full day) – Manhattan



(a) Total Distance Driven (m) (b) Time to Pair (s) (c) Time to Pick-up (s) (d) Delay (s)

(e) Cumulative Delay (s) (f) Driver Profit ($) (g) Number of Shared Rides (h) Frictions (s)

Figure A.16: Dynamic Vehicle Relocation – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number).

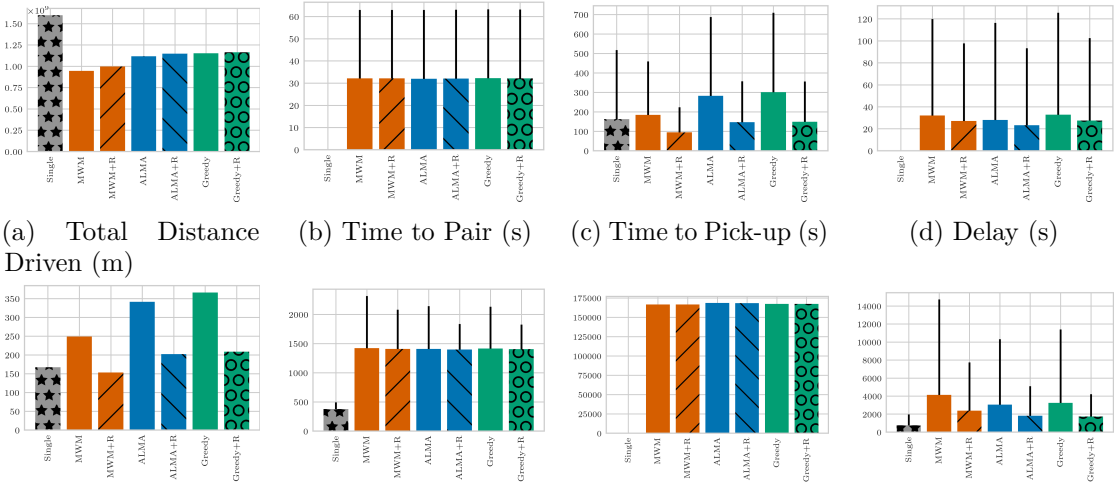
Table A.19: Dynamic Vehicle Relocation – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
NoRel	9.45E+08	–	–	–	32.10	30.84	0.00	0.00	184.55	274.34	32.14	87.69	248.79	1420.37	895.19	1.67E+05	0.00	4127.47	10597.84
MWM	9.97E+08	–	–	–	32.10	30.84	0.00	0.00	94.21	129.01	27.01	70.81	153.33	1408.73	674.38	1.67E+05	0.00	2372.57	5366.92
ALMA	1.00E+09	–	–	–	32.10	30.84	0.00	0.00	82.72	114.61	26.42	69.31	141.24	1407.37	447.49	1.67E+05	0.00	1898.47	2739.70
Greedy	1.00E+09	–	–	–	32.10	30.84	0.00	0.00	82.99	114.65	26.44	69.29	141.53	1407.41	440.30	1.67E+05	0.00	1880.72	2962.94

Table A.20: Dynamic Vehicle Relocation – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number). Each column presents the relative difference compared to not using relocation (algorithm - NoRel) / NoRel, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
NoRel	0.00%	–	–	–	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	–	0.00%	0.00%
MWM	5.48%	–	–	–	0.00%	0.00%	–	–	-48.95%	-52.97%	-15.95%	-19.25%	-38.37%	-0.82%	-24.67%	0.00%	–	-42.52%	-49.36%
ALMA	6.25%	–	–	–	0.00%	0.00%	–	–	-55.18%	-58.22%	-17.79%	-20.96%	-43.23%	-0.92%	-50.01%	0.00%	–	-54.00%	-74.15%
Greedy	6.24%	–	–	–	0.00%	0.00%	–	–	-55.03%	-58.21%	-17.73%	-20.98%	-43.11%	-0.91%	-50.81%	0.00%	–	-54.43%	-72.04%

A.10.7 End-To-End Solution – 00:00 - 23:59 (full day) – Manhattan



(a) Total Distance Driven (m) (b) Time to Pair (s) (c) Time to Pick-up (s) (d) Delay (s)

(e) Cumulative Delay (s) (f) Driver Profit ($) (g) Number of Shared Rides (h) Frictions (s)

Figure A.17: End-To-End Solution – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number)

Table A.21: End-To-End Solution – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number).

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
Single	1.60E+09	–	–	–	0.00	0.00	5.59	99.05	161.77	355.37	0.00	0.00	167.36	376.90	116.72	0.00E+00	–	756.43	1216.88
MWM	9.45E+08	–	–	–	32.10	30.84	0.00	0.00	184.55	274.34	32.14	87.69	248.79	1420.37	895.19	1.67E+05	–	4127.47	10597.84
MWM+R	9.97E+08	–	–	–	32.10	30.84	0.00	0.00	94.21	129.01	27.01	70.81	153.33	1408.73	674.38	1.67E+05	–	2372.57	5366.92
ALMA	1.12E+09	–	–	–	31.98	31.01	0.00	0.00	281.70	405.36	28.02	88.30	341.70	1406.70	736.46	1.68E+05	–	3047.45	7264.56
ALMA+R	1.15E+09	–	–	–	32.02	31.04	0.00	0.00	146.68	210.60	23.22	70.06	201.92	1397.14	440.45	1.68E+05	–	1815.54	3295.10
Greedy	1.15E+09	–	–	–	32.18	31.05	0.00	0.00	301.04	407.70	32.85	92.71	366.07	1414.66	719.44	1.67E+05	–	3242.70	8167.00
Greedy+R	1.16E+09	–	–	–	32.17	31.02	0.00	0.00	148.88	207.00	27.46	75.18	208.52	1404.95	422.37	1.67E+05	–	1726.67	2487.20

Table A.22: End-To-End Solution – January 15, 2016 – 00:00 - 23:59 (full day) – Manhattan – #Taxis = 5081 (base number). Each column presents the relative difference compared to the Singe Ride baseline (algorithm - Singe) / Singe, for each metric.

	Distance Driven (m)	SD	Elapsed Time (ns)	SD	Time to Pair (s)	SD	Time to Pair with Taxi (s)	SD	Time to Pick-up (s)	SD	Delay (s)	SD	Cumulative Delay (s)	Driver Profit ($)	SD	Number of Shared Rides	SD	Frictions (s)	SD
Single	0.00%	–	–	–	–	–	0.00%	0.00%	0.00%	0.00%	–	–	0.00%	0.00%	0.00%	–	–	0.00%	0.00%
MWM	-40.83%	–	–	–	–	–	-100.00%	-100.00%	14.09%	-22.80%	–	–	48.66%	276.85%	666.95%	–	–	445.65%	770.90%
MWM+R	-37.59%	–	–	–	–	–	-100.00%	-100.00%	-41.76%	-63.70%	–	–	-8.39%	273.76%	477.77%	–	–	213.66%	341.04%
ALMA	-30.01%	–	–	–	–	–	-100.00%	-100.00%	74.14%	14.07%	–	–	104.17%	273.23%	530.96%	–	–	302.88%	496.98%
ALMA+R	-28.17%	–	–	–	–	–	-100.00%	-100.00%	-9.33%	-40.74%	–	–	20.65%	270.69%	277.36%	–	–	140.02%	170.78%
Greedy	-27.86%	–	–	–	–	–	-100.00%	-100.00%	86.10%	14.73%	–	–	118.73%	275.34%	516.38%	–	–	328.69%	571.14%
Greedy+R	-27.17%	–	–	–	–	–	-100.00%	-100.00%	-7.97%	-41.75%	–	–	24.59%	272.76%	261.87%	–	–	128.27%	104.39%

B The Meeting Scheduling Problem

B.1 Introduction

In this appendix we provide a detailed description of our modeling of the meeting scheduling problem of Section 5.5, along with implementation details, and additional simulation results.

B.2 Problem Formulation

Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ denote the set of events we want to schedule and $\mathcal{P} = \{P_1, \ldots, P_m\}$ the set of participants. Additionally, we define a function mapping each event to the set of its participants $\text{part} : \mathcal{E} \to 2^{\mathcal{P}}$, where $2^{\mathcal{P}}$ denotes the power set of $\mathcal{P}$. Let *days* and *slots* denote the number of days and time slots per day of our calendar (e.g., $days = 7, slots = 24$ would define a calendar for one week where each slot is 1 hour long). In order to add length to each event we define an additional function $\text{len} : \mathcal{E} \to \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural numbers (excluding 0). We do not limit the length; this allows for events to exceed a single day and even the entire calendar if needed. Finally, we assume that each participant has a preference for attending certain events at a given starting time, given by:

$$\text{pref} : \mathcal{E} \times \text{part}(\mathcal{E}) \times \{1, \ldots, days\} \times \{1, \ldots, slots\} \to [0, 1].$$

For example, $\text{pref}(E_1, P_1, 2, 6) = 0.7$ indicates that participant P_1 has a preference of 0.7 to attend event E_1 starting at day 2 and slot 6. The preference function allows participants to differentiate between different kinds of meetings (personal, business, etc.), or assign priorities. For example, one could be available in the evening for personal events while preferring to schedule business meetings in the morning.

Finding a schedule consists of finding a function that assigns each event to a given

starting time, i.e.,

$$\text{sched} : \mathcal{E} \to (\{1, \ldots, days\} \times \{1, \ldots, slots\}) \cup \emptyset$$

where $\text{sched}(E) = \emptyset$ means that the event E is not scheduled. For the schedule to be *valid*, the following hard constraints need to be met:

1. Scheduled events with common participants must not overlap.
2. An event must not be scheduled at a (day, slot) tuple if any of the participants is not available. We represent an unavailable participant as one that has a preference of 0 (as given by the function pref) for that event at the given (day, slot) tuple.

More formally the hard constraints are:

$$\begin{gathered}\forall E_1 \in \mathcal{E}, \forall E_2 \in \mathcal{E} \setminus \{E_1\} : \\ (\text{sched}(E_1) \neq \emptyset \wedge \text{sched}(E_2) \neq \emptyset \wedge \text{part}(E_1) \cap \text{part}(E_2) \neq \emptyset) \\ \Rightarrow (\text{sched}(E_1) > \text{end}(E_2) \vee \text{sched}(E_2) > \text{end}(E_1))\end{gathered}$$

and

$$\begin{gathered}\forall E \in \mathcal{E} : \\ (\exists P \in \mathcal{P}, \exists d \in [1, days], \exists s \in [1, slots] : \text{pref}(E, P, d, s) = 0 \\ \Rightarrow \text{sched}(E) \neq (d, s))\end{gathered}$$

where $\text{end}(E)$ returns the ending time (last slot) of the event E as calculated by the starting time $\text{sched}(E)$ and the length $\text{len}(E)$.

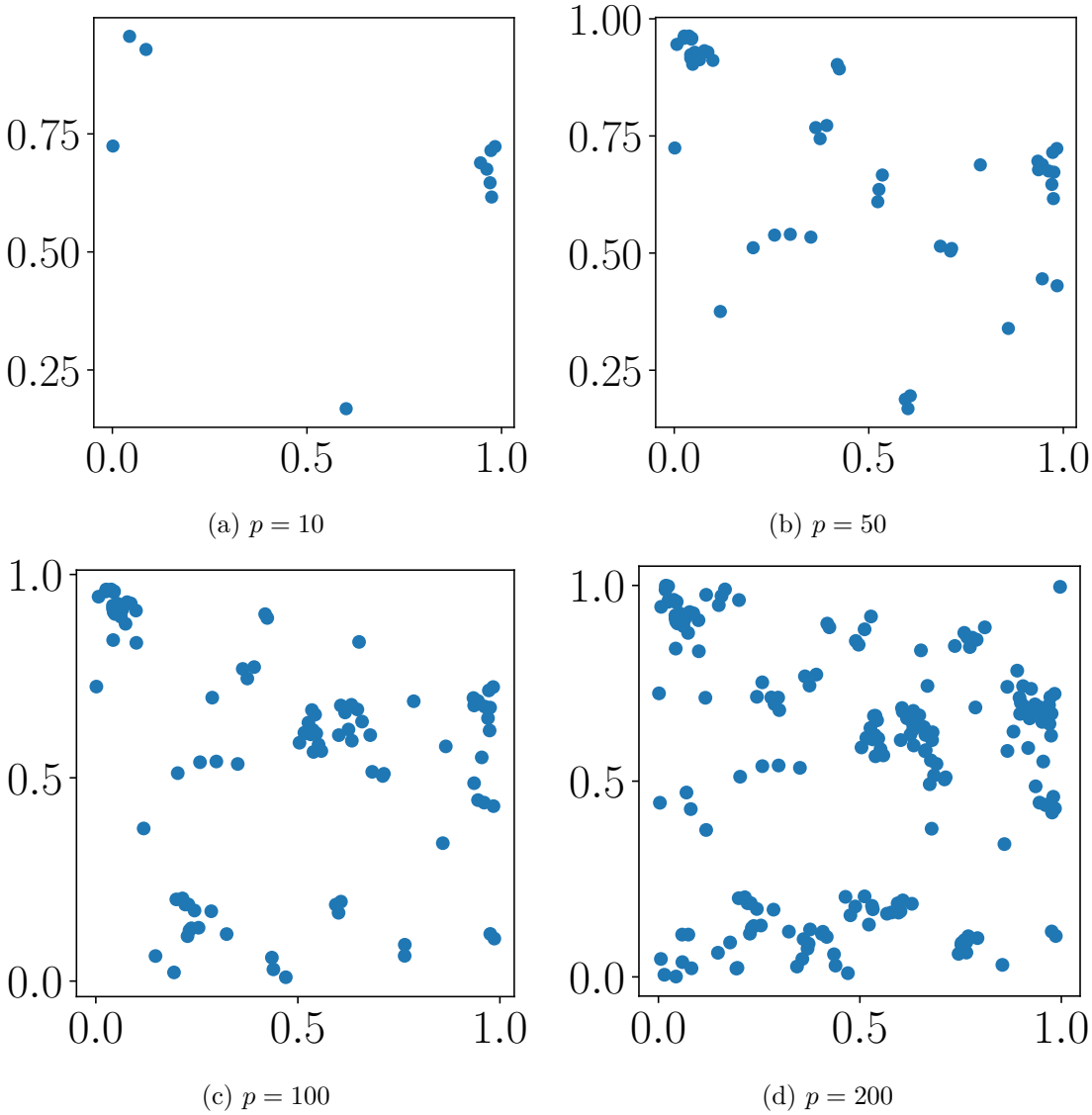
In addition to finding a valid schedule, we focus on maximizing the social welfare, i.e., the sum of the preferences for all scheduled meetings:

$$\sum_{\substack{E \in \mathcal{E} \\ \text{sched}(E) \neq \emptyset}} \sum_{P \in \mathcal{P}} \text{pref}(E, P, \text{sched}(E))$$

B.3 Modeling Events, Participants, and Utilities

B.3.1 Event Length

To determine the length of each generated event, we used information on real meeting lengths in corporate America in the 80s (Romano and Nunamaker, 2001). That data were then used to fit a logistic curve, which in turn was used to yield probabilities for an



(a) $p = 10$ (b) $p = 50$

(c) $p = 100$ (d) $p = 200$

Figure B.1: Generated datapoints on the 1×1 plane for p number of people.

event having a given length. The function was designed such that the maximum number of hours was 11. According to the data less than 1% of meetings exceeded that limit.

B.3.2 Participants

To determine the number of participants in an event, we used information on real meeting sizes (Romano and Nunamaker, 2001). As before, we used the data to fit a logistic curve, which we used to sample the number of participants. The curve was designed such that no more than 90 people were chosen for an event at a time[1] (only 3% of the meetings

[1]The median is significantly lower.

exceeds this number). Finally, for instances where the number of people in the entire system was below 90, that bound was reduced accordingly.

In order to simulate the naturally occurring clustering of people (see also Section B.5.5) we assigned each participant to a point on a 1×1 plane in a way that enabled the emergence of clusters. Participants that are closer on the plane, are more likely to attend the same meeting. In more detail, we generated the points in an iterative way. The first participant was assigned a uniformly random point on the plane. For each subsequent participant, there is a 30% probability that they also get assigned a uniformly random point. With a 70% probability the person would be assigned a point based on a normal distribution centered at one of the previously created points. The selection of the latter point is based on the time of creation; a recently created point is exponentially more likely to be chosen as the center than an older one. This ensures the creation of clusters, while the randomness and the preference on recently generated points prohibits a single cluster to grow excessively. Figure B.1 displays an example of the aforedescribed process.

B.3.3 Utilities

The utility function has two independent components. The first was designed to roughly reflect availability on an average workday. This function depends only on the time and is independent of the day. For example, a participant might prefer to schedule meetings in the morning rather than the afternoon (or during lunch time). A second function decreases the utility for an event over time. This function is independent of the time slot and only depends on the day and reflects the desire to schedule meetings sooner rather than days or weeks into the future. We generate the utility value for each participant / event combination using a normal distribution with the product of the aforementioned preference functions as mean and 0.1 as standard deviation. Figure B.2 displays the distributions.

In addition to the above, we set a number of slots to 0 for each participant to simulate already scheduled meetings or other obligations. A maximum of b slots were blocked this way each day. We chose which blocks to block using the same preference function for a day described above (Figure B.2, left). In other words, a slot with a high utility is also more likely to get blocked. Finally, for each event, we only consider the 24 most valuable slots.

B.4 Mapping the Meeting Scheduling Problem to the Allocation Problem

We can map the meeting scheduling problem to the assignment problem of Section 2.2 if we consider each event as an agent and each starting time, i.e., (day,slot) tuple, as

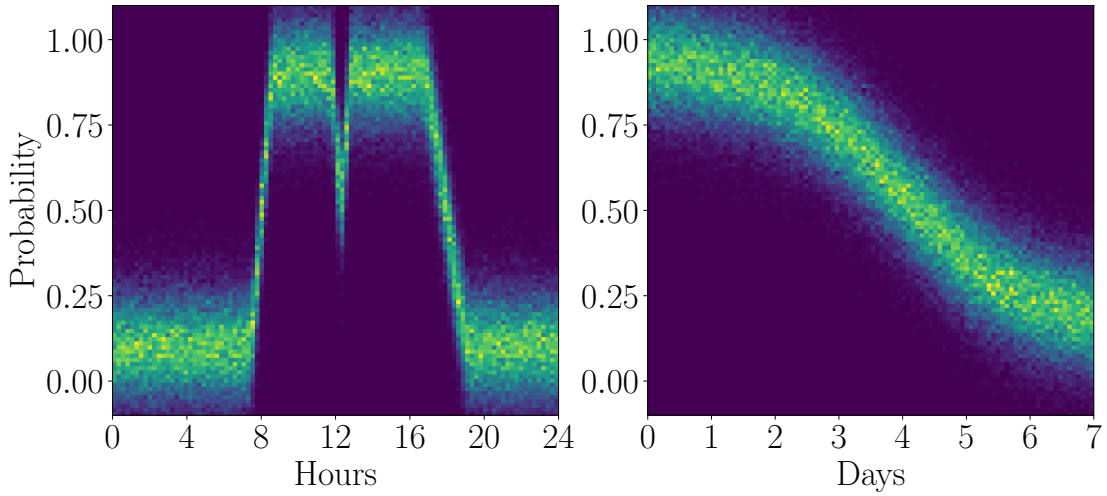


Figure B.2: Distribution used for generating preference data. The image in the left displays the distribution of preference throughout a workday. The image on the right displays the distribution of preference throughout a week.

a resource. The utility for scheduling an event to a slot can be considered as the sum of the utilities (preferences) of the participants. There are, however, some important differences. For one, two events can be scheduled at the same time as long as their sets of respective participants are disjoint. On the other hand, two events can be conflicting even if they are not assigned to the same starting slot, if they overlap. However, if we detect these conflicts, we can apply ALMA (and ALMA-Learning) to solve the meeting scheduling problem. We describe how to do so in Section B.5.

B.5 Implementation Details

B.5.1 Event-agents and Representation-agents

Each participant and each event is represented by an agent. We call those agents *representation-agents* and *event-agents*, respectively. For simplicity we use the variables introduced in Section B.2 to describe these agents. I.e., $\mathcal{E}$ is the set of agents representing events and $\mathcal{P}$ the set of agents representing participants. For simplicity, we hereafter refer to a (*day*,*slot*) tuple simply as a slot. Initially each agent in $\mathcal{P}$ knows the events he wants to attend, their length, as well as the corresponding personal preferences. Additionally, each agent in $\mathcal{P}$ creates an empty personal calendar for events to be added later. The agents in $\mathcal{E}$ on the other hand know their set of participants.

B.5.2 Aggregation of Preferences

Each event-agent in $\mathcal{E}$ needs to be able to aggregate the utilities of their attendees and compute the joint utility for the event. To do this, the representation-agents attending an event can simply relay their utilities to the corresponding event agents.[2] The agents in $\mathcal{E}$ will then combine the received data (e.g., sum the utilities for each slot). Since we defined slots with a preference of 0 as unavailable, slots where one of the preferences is 0 will assume the value 0 instead of the sum. The slot entries will then be converted into a list $\mathcal{L} = [L_1, \ldots, L_{days \cdot slots}]$, sorted in descending order, where each entry is a tuple of the form $L_i = \langle day_i, slot_i, preference_i \rangle$. Entries with a preference of 0 can simply be dropped.

B.5.3 ALMA: Collision Detection

With the aforedescribed setup we can now apply ALMA. Each event-agent has a list of resources (i.e., slots) and the corresponding utility values. Since collisions are based on the attendees of each event, we let the representation-agents detect possible collisions. The event-agents send a message to all their attendees informing them of the currently contested slot. In turn, each representation-agent will check for possible collisions and relay that information back to the corresponding event-agents. Possible collisions are checked against all simultaneous attempts to acquire slots and an internal calendar for each representation-agent that contains already scheduled slots. This leaves the decision of collision detection to the representation-agents, which does not only make sure that the relevant information for scheduling a meeting stays within the group of participants, but also allows to enforce additional personal constraints, which were not considered in the problem description in Section B.2. For example, if two events are impossible to attend in succession due to locational constraints, the attendee could simply appear unavailable by reporting a collision. If there are no collisions, the event-agent will 'acquire' the slot and inform his attendees. All informed representation-agents will delete the event-agent from their list of working agents.

B.5.4 ALMA: Normalizing the Utilities

The utilities for an event-agent are not in the range $[0, 1]$, as required by ALMA; instead they are bounded by the number of attendees, as a result of summing up the individual utilities. Therefore, the loss is also not in $[0, 1]$. There are several approaches for normalizing the utilities. One option is to simply divide by the number of attendees. This has the important downside that it distorts 'the comparability of loss' between

[2] Alternatively one could increase privacy by adding noise to the utilities or even by submitting a ranking of possible slots instead of actual utilities. The latter has the added benefit that the corresponding event-agent can choose a scale and does not rely on all users having the same metric when it comes to their utilities.

event-agents, since events with fewer attendees would be considered as important as ones with more participants even though the latter usually contributes more to social welfare. In other words, we want event-agents with more participants to be less likely to back-off in case of a conflict. Another option is to find a global variable to use for normalization such as the maximum number of attendees over all events or the highest utility value for any event. We chose the latter option.

B.5.5 ALMA: Computational and Communication Complexity

Let R^* denote the maximum number of resources any agent is interested in, and N^* denote the maximum number of agents competing for any resource. Corollary 2 proves that by bounding these two quantities (i.e., we consider R^* and N^* to be constant functions of N, R), the expected number of steps any individual agent running ALMA requires to converge is independent of the total problem size (i.e., N, and R). For the meeting scheduling problem:

- R^*: This corresponds to the maximum number of possible slots for any event. That number is bounded by the number of all available slots, given the calendar. However, in practice, meetings must be scheduled within a specific time-frame, thus each event would have a small, bounded number of slots available.

- N^*: This corresponds to the maximum number of event-agents with overlapping attendee sets competing for a slot. Assuming that we can cluster the total set of participants into smaller groups, such that either inter-group events are rare or preferred meeting times of groups do not overlap, then we could describe N^* as the maximal number of events of any such group. This form of clustering naturally occurs in many real-life situations, e.g., people in the corporate world often build clusters by their work hours or their respective departments.

To summarize, the number of rounds is bounded by the number of possible slots per event and the number of events per cluster, if a clustering exists.

In addition to the number of agents and resources, the complexity bound proven in Theorem 1 depends on the worst-case back-off probability, p^*. To mitigate this problem and further speed-up the run-time in real-world applications, we can slowly reduce the steepness of the back-off curve over time.

Finally, each round requires $\mathcal{O}(E^* + A^*)$ messages, where E^* is the maximum number of events for any single attendee and A^* the maximum number of attendees for any single event.

B.5.6 MSRAC Baseline

The problem formulation of (BenHassine and Ho, 2007) requires each meeting to be associated with a 'degree of importance'. To calculate this value, we take the average utility of all participants and multiply it with the number of participants.[3] This makes it highly unlikely that two events have the same importance value, thus reducing collisions of equally important events.

We modified the MSRAC algorithm (BenHassine and Ho, 2007) accordingly, to account for events of varying length and preferences per event/slot combinations. In short, MSRAC works as follows. The event-agent asks all participants for the relevant utility information. These are then summed up and sorted. Unavailable slots, as well as slots occupied by more important events, are dropped. The event-agent proposes the first slot in the sorted list. Then, each participant considers all the proposed events for that slot and keeps the one with the highest importance value, specifically:

- If the slot is free, accept the proposal.
- If the slot is occupied by a less important event, accept the proposal and invite the agent of the less important event to reschedule.
- If the slot is occupied by a more important event, reject proposal and invite agent to propose new slot.
- If the slot is occupied by an equally important event, keep the one with a higher utility and reject and reschedule the other one.

If an event-agent receives a message to reschedule, it deletes the current proposal and proposes the next slot to its participants. This process is repeated until a stable state is reached.

B.6 Simulation Results in Detail

In this section we present in more detail the simulation results of Section 5.5. Figures B.3, B.4, and B.5 present the results for the relative difference in social welfare (compared to CPLEX), and the Jain index and Gini coefficient, respectively. Moreover, in Figure B.6 we depict the metrics for the hand-crafted larger instance. Finally, Tables B.1, B.2, and B.3 aggregate the results.

[3] Since meetings with more participants contribute more to social welfare.

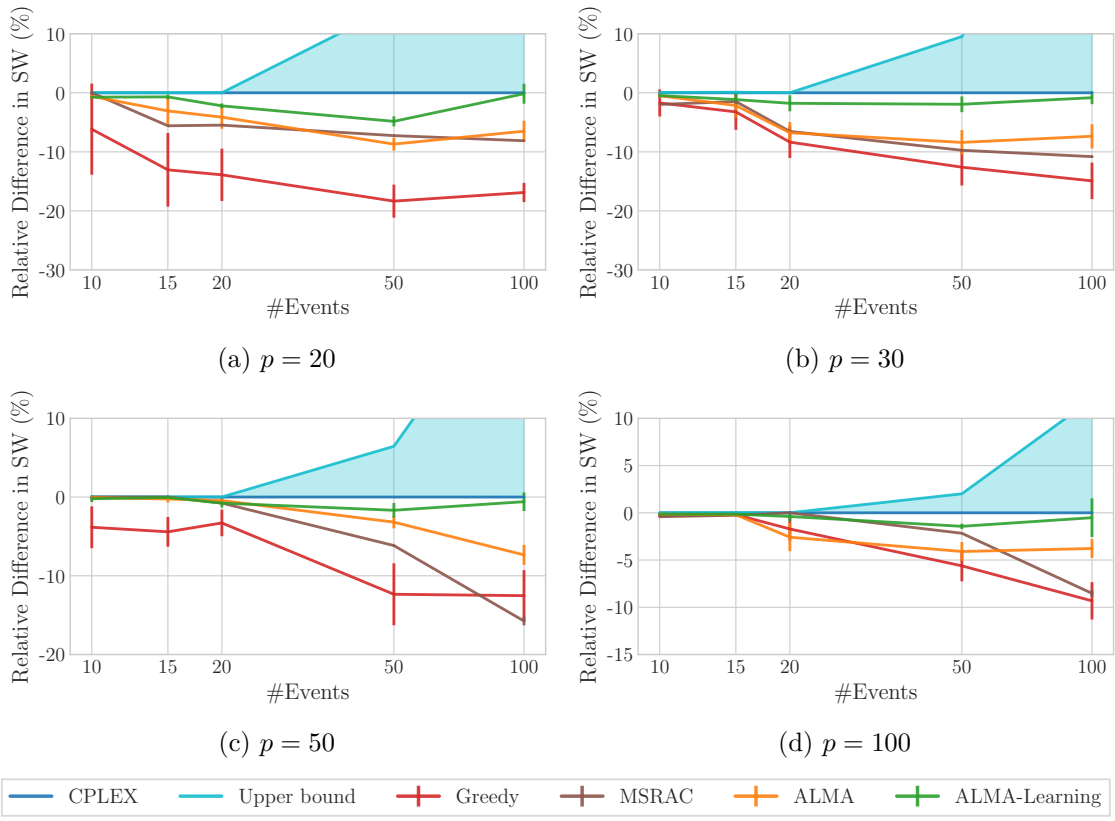


Figure B.3: Relative difference in social welfare (compared to CPLEX), for increasing number of events (x-axis in log scale). Results for various number of participants ($\mathcal{P}$). ALMA-Learning was trained for 512 time-steps.

Table B.1: Range of the average loss (%) in social welfare compared to the CPLEX for increasing number of participants, $\mathcal{P}$ ($|\mathcal{E}| \in [10, 100]$). The last line corresponds to the loss compared to the upper bound for the optimal solution for the hand-crafted large test-case with $|\mathcal{P}| = 100, |\mathcal{E}| = 280$.

	CPLEX	Greedy	MSRAC	ALMA	**ALMA-Learning**
$\|\mathcal{P}\| = 20$	N/A	$6.16 \pm 7.71\% - 18.35 \pm 2.80\%$	$0.00 \pm 0.00\% - 8.12 \pm 0.06\%$	$0.59 \pm 0.15\% - 8.69 \pm 1.10\%$	$0.16 \pm 1.68\% - 4.84 \pm 0.85\%$
$\|\mathcal{P}\| = 50$	N/A	$1.72 \pm 2.29\% - 14.92 \pm 3.09\%$	$1.47 \pm 0.00\% - 10.81 \pm 0.16\%$	$0.50 \pm 0.24\% - 8.40 \pm 2.09\%$	$0.47 \pm 0.37\% - 1.94 \pm 1.34\%$
$\|\mathcal{P}\| = 50$	N/A	$0.00 \pm 0.00\% - 15.74 \pm 0.55\%$	$3.29 \pm 1.69\% - 12.52 \pm 3.25\%$	$0.07 \pm 0.07\% - 7.34 \pm 1.27\%$	$0.05 \pm 0.05\% - 1.68 \pm 0.93\%$
$\|\mathcal{P}\| = 100$	N/A	$0.19 \pm 0.11\% - 9.32 \pm 1.98\%$	$0.00 \pm 0.00\% - 8.52 \pm 0.37\%$	$0.14 \pm 0.00\% - 4.09 \pm 0.99\%$	$0.14 \pm 0.11\% - 1.43 \pm 0.28\%$
$\|\mathcal{E}\| = 280$	$0.00\% - 4.39\%$	$0.00 \pm 0.00\% - 15.31 \pm 1.00\%$	$0.00 \pm 0.00\% - 22.07 \pm 0.34\%$	$0.00 \pm 0.00\% - 10.81 \pm 2.02\%$	$0.00 \pm 0.00\% - 8.84 \pm 0.68\%$

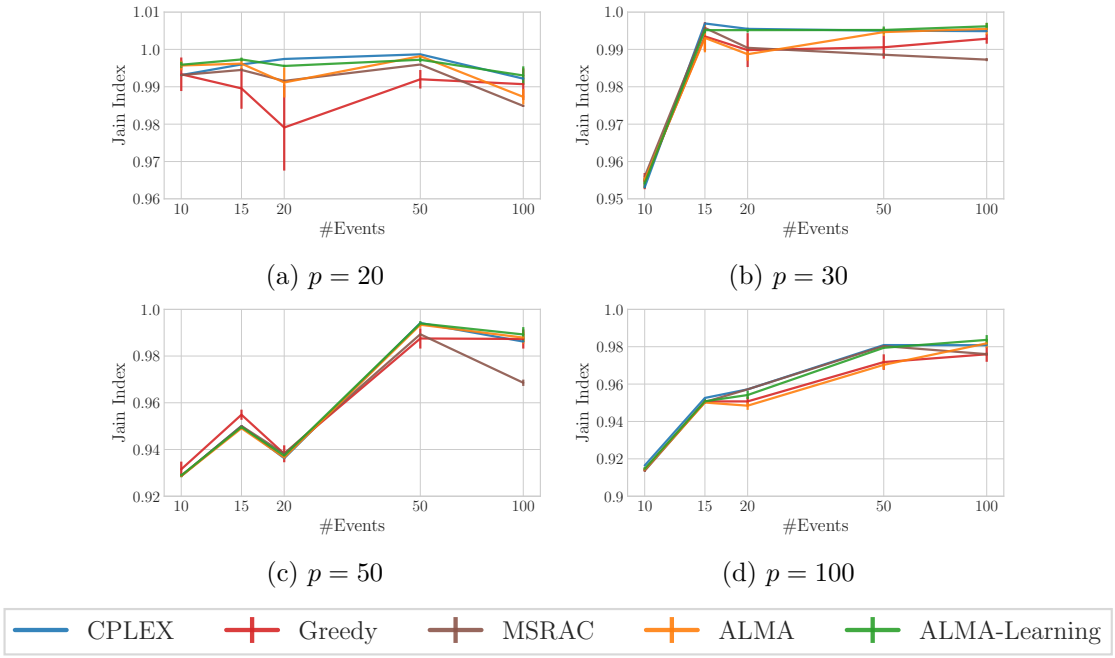


Figure B.4: Fairness – Jain index (the higher, the better), for increasing number of events, $\mathcal{E}$ (x-axis in log scale). Results for various number of participants ($\mathcal{P}$). ALMA-Learning was trained for 512 time-steps.

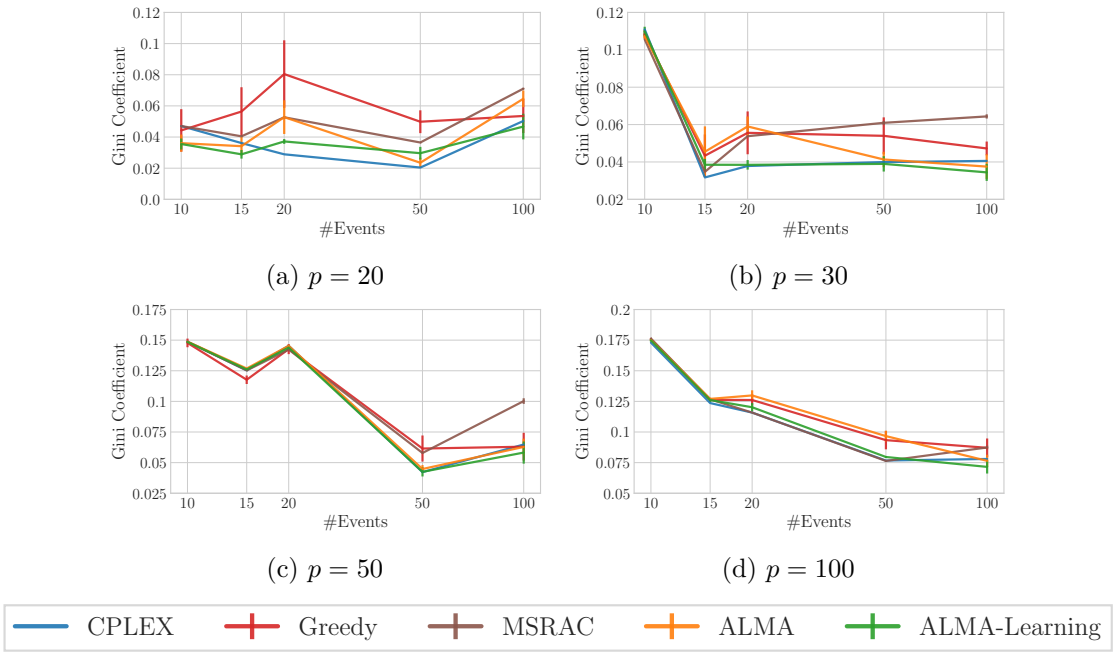
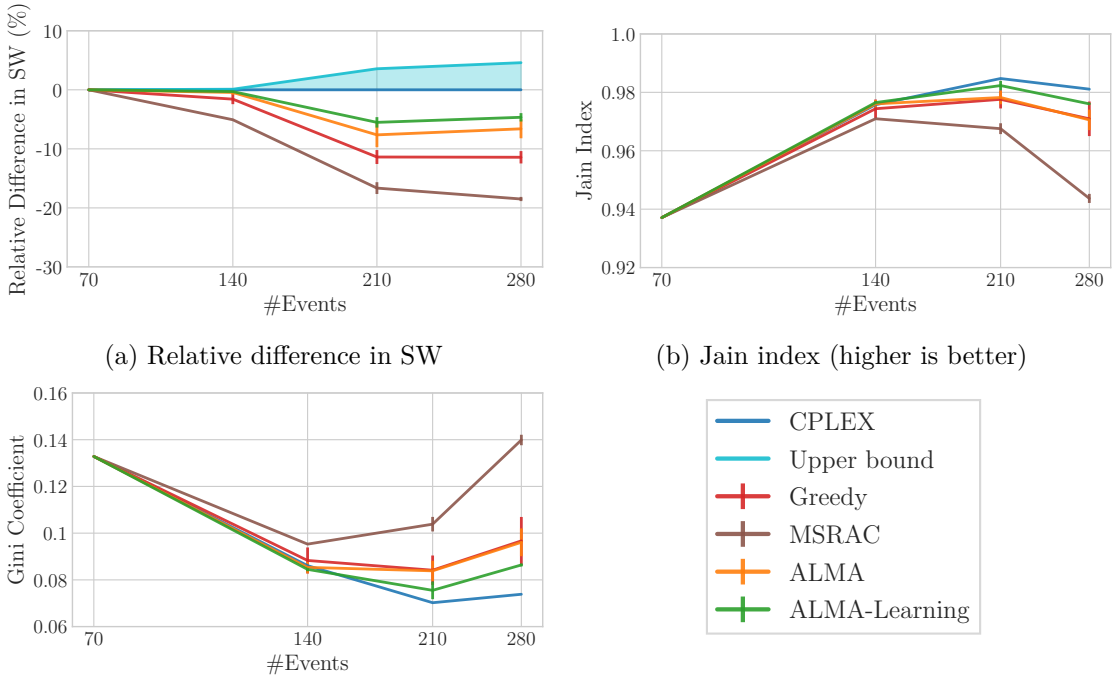


Figure B.5: Fairness – Gini coefficient (the lower, the better), for increasing number of events, $\mathcal{E}$ (x-axis in log scale). Results for various number of participants ($\mathcal{P}$). ALMA-Learning was trained for 512 time-steps.



(a) Relative difference in SW

(b) Jain index (higher is better)

(c) Gini coefficient (lower is better)

Figure B.6: Hand-crafted, large instances. Results for increasing number of events, $\mathcal{E}$ (x-axis in log scale), and 100 participants ($|\mathcal{P}| = 100$). ALMA-Learning was trained for 512 time-steps.

Table B.2: Fairness – Jain index (the higher, the better), for increasing number of participants, $\mathcal{P}$ ($|\mathcal{E}| \in [10, 100]$). The last line corresponds to the hand-crafted large test-case with $|\mathcal{P}| = 100, |\mathcal{E}| = 280$. In parenthesis we include the average improvement in fairness compared to CPLEX.

	CPLEX	Greedy	MSRAC	ALMA	**ALMA-Learning**
$\|\mathcal{P}\| = 20$	$0.99 - 1.00$	$0.98 \pm 0.01 - 0.99 \pm 0.00 (98.16 \pm 1.16\% - 100.02 \pm 0.45\%)$	$0.98 \pm 0.00 - 1.00 \pm 0.00 (99.26 \pm 0.02\% - 100.00 \pm 0.00\%)$	$0.99 \pm 0.00 - 1.00 \pm 0.00 (99.37 \pm 99.37\% - 100.26 \pm 0.13\%)$	$0.99 \pm 0.00 - 1.00 \pm 0.00 (99.81 \pm 0.03\% - 100.28 \pm 0.08\%)$
$\|\mathcal{P}\| = 50$	$0.95 - 1.00$	$0.95 \pm 0.00 - 0.99 \pm 0.00 (99.43 \pm 0.46\% - 100.17 \pm 0.23\%)$	$0.96 \pm 0.00 - 1.00 \pm 0.00 (99.23 \pm 0.04\% - 100.30 \pm 0.00\%)$	$0.96 \pm 0.00 - 1.00 \pm 0.00 (99.32 \pm 0.17\% - 100.21 \pm 0.06\%)$	$0.95 \pm 0.00 - 1.00 \pm 0.00 (99.82 \pm 0.08\% - 100.13 \pm 0.09\%)$
$\|\mathcal{P}\| = 50$	$0.93 - 0.99$	$0.93 \pm 0.00 - 0.99 \pm 0.00 (99.34 \pm 0.43\% - 100.51 \pm 0.23\%)$	$0.93 \pm 0.00 - 0.99 \pm 0.00 (98.21 \pm 0.14\% - 100.21 \pm 0.00\%)$	$0.93 \pm 0.00 - 0.99 \pm 0.00 (99.90 \pm 99.90\% - 100.17 \pm 0.26\%)$	$0.93 \pm 0.00 - 0.99 \pm 0.00 (99.95 \pm 0.03\% - 100.31 \pm 0.32\%)$
$\|\mathcal{P}\| = 100$	$0.92 - 0.98$	$0.91 \pm 0.00 - 0.98 \pm 0.00 (99.07 \pm 0.42\% - 99.81 \pm 0.10\%)$	$0.91 \pm 0.00 - 0.98 \pm 0.00 (99.51 \pm 0.08\% - 100.00 \pm 0.00\%)$	$0.91 \pm 0.00 - 0.98 \pm 0.00 (98.92 \pm 98.92\% - 100.10 \pm 0.24\%)$	$0.91 \pm 0.00 - 0.98 \pm 0.00 (99.68 \pm 0.23\% - 100.29 \pm 0.26\%)$
$\|\mathcal{E}\| = 280$	$0.94 - 0.98$	$0.94 \pm 0.00 - 0.98 \pm 0.00 (98.97 \pm 0.60\% - 100.00 \pm 0.00\%)$	$0.94 \pm 0.00 - 0.97 \pm 0.0096.19 \pm 0.16\% - 100.00 \pm 0.00\%)$	$0.94 \pm 0.00 - 0.98 \pm 0.00 (98.93 \pm 98.93\% - 100.03 \pm 0.14\%)$	$0.94 \pm 0.00 - 0.98 \pm 0.00 (99.49 \pm 0.01\% - 100.08 \pm 0.04\%)$

Table B.3: Fairness – Gini coefficient (the lower, the better), for increasing number of participants, $\mathcal{P}$ ($|\mathcal{E}| \in [10, 100]$). The last line corresponds to the hand-crafted large test-case with $|\mathcal{P}| = 100, |\mathcal{E}| = 280$. In parenthesis we include the average improvement in fairness compared to CPLEX.

	CPLEX	Greedy	MSRAC	ALMA	**ALMA-Learning**
$\|\mathcal{P}\| = 20$	$0.02 - 0.05$	$0.04 \pm 0.01 - 0.08 \pm 0.02 (94.07 \pm 28.94\% - 278.19 \pm 75.03\%)$	$0.04 \pm 0.00 - 0.07 \pm 0.00 (100.00 \pm 0.00\% - 182.22 \pm 0.00\%)$	$0.02 \pm 0.00 - 0.06 \pm 0.01 (76.63 \pm 76.63\% - 182.55 \pm 37.40\%)$	$0.03 \pm 0.00 - 0.05 \pm 0.01 (75.48 \pm 7.50\% - 145.14 \pm 19.93\%)$
$\|\mathcal{P}\| = 50$	$0.03 - 0.11$	$0.04 \pm 0.01 - 0.11 \pm 0.00 (97.62 \pm 2.09\% - 146.89 \pm 30.29\%)$	$0.03 \pm 0.00 - 0.11 \pm 0.00 (95.15 \pm 0.00\% - 158.71 \pm 2.79\%)$	$0.04 \pm 0.01 - 0.11 \pm 0.00 (92.52 \pm 92.52\% - 155.91 \pm 14.12\%)$	$0.03 \pm 0.00 - 0.11 \pm 0.00 (84.89 \pm 11.24\% - 121.39 \pm 9.84\%)$
$\|\mathcal{P}\| = 50$	$0.04 - 0.15$	$0.06 \pm 0.01 - 0.15 \pm 0.00 (93.90 \pm 2.76\% - 145.61 \pm 25.23\%)$	$0.06 \pm 0.00 - 0.15 \pm 0.00 (97.86 \pm 0.00\% - 154.72 \pm 3.55\%)$	$0.04 \pm 0.00 - 0.15 \pm 0.00 (96.95 \pm 96.95\% - 106.06 \pm 7.60\%)$	$0.04 \pm 0.00 - 0.15 \pm 0.00 (89.84 \pm 13.96\% - 100.62 \pm 0.45\%)$
$\|\mathcal{P}\| = 100$	$0.08 - 0.17$	$0.09 \pm 0.01 - 0.18 \pm 0.00 (101.53 \pm 0.62\% - 121.70 \pm 9.79\%)$	$0.08 \pm 0.00 - 0.18 \pm 0.00 (99.56 \pm 0.99\% - 112.00 \pm 1.91\%)$	$0.08 \pm 0.00 - 0.18 \pm 0.00 (98.04 \pm 98.04\% - 125.99 \pm 5.74\%)$	$0.07 \pm 0.01 - 0.17 \pm 0.00 (91.70 \pm 6.95\% - 103.77 \pm 0.98\%)$
$\|\mathcal{E}\| = 280$	$0.07 - 0.13$	$0.08 \pm 0.01 - 0.13 \pm 0.00 (100.00 \pm 0.00\% - 130.96 \pm 13.86\%)$	$0.10 \pm 0.00 - 0.14 \pm 0.00 (100.00 \pm 0.00\% - 189.42 \pm 3.03\%)$	$0.08 \pm 0.00 - 0.13 \pm 0.00 (99.14 \pm 99.14\% - 130.12 \pm 8.07\%)$	$0.08 \pm 0.00 - 0.13 \pm 0.00 (98.15 \pm 0.93\% - 116.99 \pm 0.83\%)$

C Detailed Simulation Results of Chapters 8 and 9

In this appendix we provide numerical values for all the simulations of Chapters 8 and 9. Specifically, Tables C.1 - C.19 include the results for Chapter 8: the absolute values with and without the introduced signal, the relative difference, and the Student's T-test p-values. Finally, the last table (Table C.20) includes the results for Chapter 9 for every configuration of the policymaker (comparison to the market equilibrium prices and the Student's T-test p-values).

Table C.1: Social Welfare
Results (averaged over 8 trials) for increasing population size ($N \in [2, 64]$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$
$M_s = 0.2$	0.36	0.36	0.76	0.76	1.50	1.51	2.92	2.92	5.65	5.65	10.96	10.96
$M_s = 0.3$	0.60	0.60	1.28	1.28	2.64	2.64	5.38	4.81	8.26	8.61	16.38	15.29
$M_s = 0.4$	3.68	43.46	2.30	2.89	4.70	160.05	5.77	130.37	10.66	10.73	20.45	21.10
$M_s = 0.5$	122.30	111.57	179.20	186.65	141.89	193.53	70.10	154.00	12.81	39.35	25.67	25.81
$M_s = 0.6$	143.23	135.73	206.14	232.03	174.52	215.93	176.30	157.74	15.20	130.34	30.37	33.12
$M_s = 0.7$	172.69	179.28	181.15	223.18	121.33	211.70	196.26	202.63	34.14	196.57	41.49	52.57
$M_s = 0.8$	176.33	199.11	200.64	275.35	200.10	291.44	247.90	304.74	86.95	316.08	62.28	80.49
$M_s = 0.9$	197.49	207.15	244.93	316.75	327.91	395.14	341.24	499.66	165.75	637.30	101.12	1517.30
$M_s = 1.0$	223.28	229.01	339.33	371.72	482.25	524.64	585.71	687.65	592.26	1717.98	297.34	5350.52
$M_s = 1.1$	252.41	250.56	418.38	429.24	674.72	707.17	1315.68	1348.68	2611.96	3415.66	5225.12	8317.41
$M_s = 1.2$	280.61	278.61	523.86	532.54	1046.36	1051.10	2091.35	2120.13	4180.33	4784.53	8367.87	9900.67

Table C.2: Social Welfare (Relative difference & p-values)
(i) Relative difference in Social Welfare when signal of cardinality $G = N$ is introduced ((Result$_{G=N}$ −Result$_{G=1}$)/Result$_{G=1}$, where Result$_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
for varying population size ($N \in [2, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p
$M_s = 0.2$	0.0	0.87876	0.2	0.57508	0.2	0.47454	-0.1	0.47064	0.1	0.36608	0.0	0.93297
$M_s = 0.3$	-0.3	0.48043	0.2	0.57516	0.1	0.82779	-10.5	0.02451	4.2	0.00000	-6.6	0.00000
$M_s = 0.4$	1082.1	0.01102	25.8	0.16861	3305.9	0.00000	2158.1	0.00000	0.7	0.74501	3.2	0.01327
$M_s = 0.5$	-8.8	0.00453	4.2	0.42501	36.4	0.00261	119.7	0.00566	207.2	0.14710	0.6	0.48706
$M_s = 0.6$	-5.2	0.01401	12.6	0.01795	23.7	0.00033	-10.5	0.48519	757.6	0.00001	9.1	0.00000
$M_s = 0.7$	3.8	0.00161	23.2	0.00000	74.5	0.00000	3.2	0.83672	475.7	0.00000	26.7	0.00000
$M_s = 0.8$	12.9	0.00032	37.2	0.00013	45.6	0.00105	22.9	0.00886	263.5	0.00000	29.2	0.00000
$M_s = 0.9$	4.9	0.06719	29.3	0.00008	20.5	0.12549	46.4	0.00355	284.5	0.00000	1400.5	0.00000
$M_s = 1.0$	2.6	0.03517	9.5	0.02099	8.8	0.01232	17.4	0.00014	190.1	0.00000	1699.5	0.00000
$M_s = 1.1$	-0.7	0.41309	2.6	0.12172	4.8	0.01402	2.5	0.00002	30.8	0.00000	59.2	0.00000
$M_s = 1.2$	-0.7	0.56215	1.7	0.00477	0.5	0.09644	1.4	0.00000	14.5	0.00000	18.3	0.00000

Table C.3: Episode Length (#time-steps)
Results (averaged over 8 trials) for increasing population size ($N \in [2, 64]$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$
$M_s = 0.2$	2.04	2.04	2.14	2.14	2.05	2.06	2.01	2.01	1.99	2.00	1.96	1.96
$M_s = 0.3$	2.75	2.72	2.83	2.82	3.63	3.66	5.17	3.61	1.98	2.67	2.20	1.19
$M_s = 0.4$	25.17	276.52	6.50	8.50	11.26	491.08	6.15	474.22	2.25	1.95	1.44	2.07
$M_s = 0.5$	499.79	499.94	463.01	500.00	481.93	500.00	240.27	496.08	1.90	119.85	2.62	2.32
$M_s = 0.6$	499.98	499.96	498.51	499.49	499.90	499.72	499.85	495.69	2.36	438.24	2.72	6.78
$M_s = 0.7$	500.00	500.00	499.24	500.00	499.49	500.00	500.00	500.00	70.78	500.00	10.00	31.33
$M_s = 0.8$	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	200.63	500.00	22.00	66.68
$M_s = 0.9$	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	275.00	500.00	53.00	500.00
$M_s = 1.0$	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
$M_s = 1.1$	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00
$M_s = 1.2$	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00	500.00

Table C.4: Episode Length (Relative difference & p-values)
(i) Relative difference in Episode Length when signal of cardinality $G = N$ is introduced $((\text{Result}_{G=N} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
for varying population size ($N \in [2, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).
NaN values in the p-values column are due to having only a single data point; both cases (with and without the signal) have the same episode length in all the trials.

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p
$M_s = 0.2$	0.0	0.99280	0.0	0.98122	0.7	0.43808	-0.5	0.34488	0.6	0.32379	0.0	0.97624
$M_s = 0.3$	-0.9	0.37060	-0.2	0.83869	0.9	0.48086	-30.2	0.02482	34.7	0.00000	-46.0	0.00000
$M_s = 0.4$	998.6	0.00924	30.7	0.16075	4261.1	0.00000	7610.6	0.00000	-13.3	0.52170	44.2	0.04255
$M_s = 0.5$	0.0	0.36593	8.0	0.04688	3.8	0.20338	106.5	0.01043	6209.4	0.14849	-11.8	0.56968
$M_s = 0.6$	0.0	0.58271	0.2	0.36518	0.0	0.56152	-0.8	0.02847	18503.0	0.00001	149.4	0.00001
$M_s = 0.7$	0.0	NaN	0.2	0.13837	0.1	0.10198	0.0	NaN	606.4	0.00001	213.3	0.00000
$M_s = 0.8$	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	149.2	0.00418	203.1	0.00000
$M_s = 0.9$	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	81.8	0.01919	843.4	0.00000
$M_s = 1.0$	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN
$M_s = 1.1$	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN
$M_s = 1.2$	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN	0.0	NaN

Table C.5: Training Time (#episodes)
Results (averaged over 8 trials) for increasing population size ($N \in [2, 64]$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$
$M_s = 0.2$	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00
$M_s = 0.3$	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00	5000.00
$M_s = 0.4$	5000.00	5000.00	5000.00	5000.00	5000.00	4990.13	5000.00	4960.13	5000.00	5000.00	5000.00	5000.00
$M_s = 0.5$	4955.38	4340.63	4971.25	3408.75	5000.00	3389.50	5000.00	4847.88	5000.00	5000.00	5000.00	5000.00
$M_s = 0.6$	3037.38	2268.00	3822.75	2013.75	3903.00	3051.63	4583.13	2797.50	5000.00	4081.13	5000.00	5000.00
$M_s = 0.7$	1536.13	1008.13	2478.63	1328.00	3104.50	1880.00	3213.00	2680.00	4937.00	3026.00	5000.00	5000.00
$M_s = 0.8$	1004.00	1124.00	1968.00	1296.00	3088.00	2064.00	3778.00	2116.00	4887.00	2551.00	5000.00	5000.00
$M_s = 0.9$	1248.00	936.00	2168.00	1416.00	3025.00	1412.00	3810.00	2100.00	4536.00	2896.00	5000.00	2581.00
$M_s = 1.0$	864.00	776.00	1692.00	1252.00	2116.00	1612.00	2740.00	1788.00	3393.00	2334.00	1484.00	1965.00
$M_s = 1.1$	684.00	800.00	1028.00	1376.00	1284.00	1436.00	1164.00	1536.00	1218.00	1832.00	1193.00	1172.00
$M_s = 1.2$	840.00	780.00	1132.00	1040.00	1004.00	1212.00	932.00	1116.00	949.00	844.00	897.00	816.00

Table C.6: Training Time (Relative difference & p-values)
(i) Relative difference in Training Time when signal of cardinality $G = N$ is introduced ($(\text{Result}_{G=N} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
for varying population size ($N \in [2, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p
$M_s = 0.2$	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000
$M_s = 0.3$	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000	0.0	0.00000
$M_s = 0.4$	0.0	0.00000	0.0	0.00000	-0.2	0.00000	-0.8	0.00000	0.0	0.00000	0.0	0.00000
$M_s = 0.5$	-12.4	0.00000	-31.4	0.00000	-32.2	0.00000	-3.0	0.00000	0.0	0.00000	0.0	0.00000
$M_s = 0.6$	-25.3	0.00000	-47.3	0.00006	-21.8	0.00013	-39.0	0.00003	-18.4	0.00000	0.0	0.00000
$M_s = 0.7$	-34.4	0.00000	-46.4	0.02193	-39.4	0.00396	-16.6	0.00666	-38.7	0.00000	0.0	0.00000
$M_s = 0.8$	12.0	0.01448	-34.1	0.00266	-33.2	0.00228	-44.0	0.00128	-47.8	0.00000	0.0	0.00000
$M_s = 0.9$	-25.0	0.00462	-34.7	0.01569	-53.3	0.00583	-44.9	0.00221	-36.2	0.00001	-48.4	0.00000
$M_s = 1.0$	-10.2	0.00137	-26.0	0.02400	-23.8	0.01508	-34.7	0.03351	-31.2	0.00211	32.4	0.00000
$M_s = 1.1$	17.0	0.00000	33.9	0.00024	11.8	0.00006	32.0	0.00000	50.4	0.00001	-1.8	0.00000
$M_s = 1.2$	-7.1	0.00847	-8.1	0.00002	20.7	0.00000	19.7	0.00000	-11.1	0.00000	-9.0	0.00000

Table C.7: Jain Index (higher is fairer)
Results (averaged over 8 trials) for increasing population size ($N \in [2, 64]$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$
$M_s = 0.2$	0.99984	0.99973	0.99976	0.99984	0.99963	0.99966	0.99967	0.99961	0.99943	0.99947	0.99932	0.99941
$M_s = 0.3$	0.99973	0.99991	0.99984	0.99981	0.99967	0.99974	0.99963	0.99959	0.99958	0.99930	0.99936	0.99942
$M_s = 0.4$	0.99210	0.99299	0.99910	0.99846	0.99119	0.98808	0.98691	0.99370	0.98603	0.99526	0.99063	0.99648
$M_s = 0.5$	0.98061	0.98251	0.97149	0.97895	0.97910	0.98978	0.98376	0.99416	0.98946	0.99694	0.99489	0.99773
$M_s = 0.6$	0.98252	0.99911	0.98507	0.99479	0.97847	0.99246	0.98147	0.99599	0.99898	0.99628	0.99935	0.99831
$M_s = 0.7$	0.99317	0.99996	0.98657	0.99239	0.98647	0.99458	0.98862	0.99601	0.99635	0.99660	1.00000	0.99787
$M_s = 0.8$	0.99705	0.99872	0.98081	0.99269	0.97903	0.99534	0.98679	0.99645	0.99023	0.99733	1.00000	0.99864
$M_s = 0.9$	0.99780	0.99535	0.97744	0.99304	0.97841	0.99581	0.98236	0.99704	0.99234	0.99810	0.99999	0.99912
$M_s = 1.0$	0.98674	0.99740	0.98892	0.99454	0.99094	0.99715	0.99441	0.99876	0.99578	0.99924	1.00000	0.99929
$M_s = 1.1$	0.99620	0.99498	0.99770	0.99737	0.99990	0.99976	0.99999	0.99999	1.00000	0.99945	0.99999	0.99909
$M_s = 1.2$	0.99855	0.99956	0.99998	0.99993	0.99999	0.99999	1.00000	0.99998	0.99999	0.99917	0.99999	0.99876

Table C.8: Jain Index (Relative difference & p-values)
(i) Relative difference in Jain Index when signal of cardinality $G = N$ is introduced $((\text{Result}_{G=N} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
for varying population size ($N \in [2, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p
$M_s = 0.2$	0.0	0.62689	0.0	0.19181	0.0	0.73995	0.0	0.43584	0.0	0.67344	0.0	0.12180
$M_s = 0.3$	0.0	0.19431	0.0	0.74755	0.0	0.40216	0.0	0.68556	0.0	0.00316	0.0	0.24335
$M_s = 0.4$	0.1	0.85540	-0.1	0.47459	-0.3	0.31067	0.7	0.00312	0.9	0.00014	0.6	0.00000
$M_s = 0.5$	0.2	0.88279	0.8	0.27542	1.1	0.08226	1.1	0.00027	0.8	0.00105	0.3	0.00025
$M_s = 0.6$	1.7	0.05974	1.0	0.02806	1.4	0.00501	1.5	0.00055	-0.3	0.00000	-0.1	0.00000
$M_s = 0.7$	0.7	0.16061	0.6	0.25648	0.8	0.00610	0.7	0.00002	0.0	0.94743	-0.2	0.00000
$M_s = 0.8$	0.2	0.33436	1.2	0.03393	1.7	0.00015	1.0	0.00049	0.7	0.16118	-0.1	0.00000
$M_s = 0.9$	-0.2	0.36223	1.6	0.00338	1.8	0.00011	1.5	0.00000	0.6	0.07503	-0.1	0.00000
$M_s = 1.0$	1.1	0.07827	0.6	0.08720	0.6	0.00834	0.4	0.00017	0.3	0.00165	-0.1	0.00000
$M_s = 1.1$	-0.1	0.49041	0.0	0.64625	0.0	0.11982	0.0	0.18597	-0.1	0.00006	-0.1	0.00000
$M_s = 1.2$	0.1	0.05751	0.0	0.15660	0.0	0.38672	0.0	0.00003	-0.1	0.00000	-0.1	0.00000

Table C.9: Gini Coefficient (lower is fairer)
Results (averaged over 8 trials) for increasing population size ($N \in [2, 64]$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$	$G = 1$	$G = N$
$M_s = 0.2$	0.00583	0.00551	0.00764	0.00663	0.01048	0.00956	0.00991	0.01087	0.01323	0.01282	0.01459	0.01356
$M_s = 0.3$	0.00674	0.00390	0.00644	0.00656	0.00983	0.00855	0.01040	0.01085	0.01135	0.01468	0.01416	0.01345
$M_s = 0.4$	0.03897	0.03083	0.01409	0.01667	0.05032	0.05879	0.06336	0.04329	0.06617	0.03841	0.05399	0.03328
$M_s = 0.5$	0.05571	0.05590	0.08869	0.07415	0.07528	0.05419	0.06839	0.04166	0.05646	0.03109	0.03890	0.02668
$M_s = 0.6$	0.05350	0.01113	0.06094	0.03547	0.07879	0.04711	0.07156	0.03444	0.01466	0.03405	0.01272	0.02298
$M_s = 0.7$	0.02754	0.00215	0.05777	0.04123	0.06279	0.03842	0.05894	0.03514	0.01252	0.03259	0.00038	0.02537
$M_s = 0.8$	0.02160	0.01446	0.07129	0.04442	0.07916	0.03539	0.06264	0.03255	0.03503	0.02884	0.00012	0.02059
$M_s = 0.9$	0.01901	0.02562	0.08016	0.03768	0.07998	0.03427	0.07229	0.02973	0.03275	0.02440	0.00027	0.01639
$M_s = 1.0$	0.04736	0.02190	0.05277	0.03522	0.04490	0.02877	0.03175	0.01899	0.01802	0.01524	0.00012	0.01480
$M_s = 1.1$	0.02741	0.03299	0.02376	0.02583	0.00392	0.00692	0.00119	0.00191	0.00035	0.01244	0.00038	0.01670
$M_s = 1.2$	0.01585	0.00951	0.00138	0.00358	0.00102	0.00154	0.00078	0.00235	0.00083	0.01604	0.00113	0.01958

Table C.10: Gini Coefficient (Relative difference & p-values)
(i) Relative difference in Gini Coefficient when signal of cardinality $G = N$ is introduced ($(\text{Result}_{G=N} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
for varying population size ($N \in [2, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	N =2		$N = 4$		$N = 8$		$N = 16$		$N = 32$		$N = 64$	
	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p	(%)	p
$M_s = 0.2$	-5.5	0.89907	-13.2	0.38327	-8.7	0.54803	9.7	0.30792	-3.2	0.66529	-7.0	0.10096
$M_s = 0.3$	-42.1	0.18044	1.9	0.93581	-13.0	0.32875	4.4	0.75014	29.3	0.00649	-5.0	0.29447
$M_s = 0.4$	-20.9	0.55214	18.3	0.65124	16.8	0.32715	-31.7	0.00172	-42.0	0.00005	-38.3	0.00000
$M_s = 0.5$	0.3	0.99318	-16.4	0.29791	-28.0	0.07229	-39.1	0.00045	-44.9	0.00016	-31.4	0.00013
$M_s = 0.6$	-79.2	0.01921	-41.8	0.01744	-40.2	0.00115	-51.9	0.00000	132.2	0.00000	80.6	0.00000
$M_s = 0.7$	-92.2	0.05186	-28.6	0.19899	-38.8	0.00318	-40.4	0.00002	160.3	0.12355	6619.1	0.00000
$M_s = 0.8$	-33.1	0.33714	-37.7	0.01429	-55.3	0.00008	-48.0	0.00005	-17.7	0.71927	16702.9	0.00000
$M_s = 0.9$	34.8	0.51138	-53.0	0.00151	-57.2	0.00001	-58.9	0.00000	-25.5	0.51399	6016.9	0.00000
$M_s = 1.0$	-53.8	0.08247	-33.3	0.06736	-35.9	0.00585	-40.2	0.00000	-15.4	0.16010	12183.2	0.00000
$M_s = 1.1$	20.3	0.40907	8.7	0.53123	76.5	0.05059	60.5	0.01015	3416.3	0.00000	4296.4	0.00000
$M_s = 1.2$	-40.0	0.13271	159.9	0.05210	50.1	0.29175	201.1	0.00000	1839.3	0.00000	1640.3	0.00000

Table C.11: CIC values
Results (averaged over the 8 trials and the agents in the population) for increasing population size ($N \in [4, 64]$) and environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

	$N = 4$	$N = 8$	$N = 16$	$N = 32$	$N = 64$
$M_s = 0.2$	0.037	0.043	0.046	0.047	0.048
$M_s = 0.3$	0.037	0.043	0.047	0.050	0.054
$M_s = 0.4$	0.424	1.008	0.611	0.122	0.188
$M_s = 0.5$	0.908	0.841	0.542	0.225	0.144
$M_s = 0.6$	0.798	0.658	0.281	0.530	0.130
$M_s = 0.7$	0.710	0.710	0.454	0.292	0.210
$M_s = 0.8$	0.741	0.668	0.513	0.267	0.210
$M_s = 0.9$	0.465	0.429	0.378	0.251	0.266
$M_s = 1.0$	0.406	0.513	0.265	0.382	0.331
$M_s = 1.1$	0.141	0.180	0.135	0.221	0.296
$M_s = 1.2$	0.080	0.095	0.150	0.218	0.301

Table C.12: Social Welfare
Results (averaged over 8 trials) for varying signal size ($G = \{1, \frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, where $N = 32$) and equilibrium stock multiplier (M_s values of 0.7, 0.8 and 0.9). The following results include:
(i) Absolute values,
(ii) Relative difference (%), i.e., $(\text{Result}_{G=X} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality $X \in \{\frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, and
(iii) Student's T-test p-values with respect to $G = 1$

	Absolute Values					
	$G = 1$	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	34.14	184.82	158.95	196.57	219.33	221.43
$M_s = 0.8$	86.95	249.51	260.94	316.08	376.67	423.03
$M_s = 0.9$	165.75	431.43	497.05	637.30	784.72	970.48

	Relative Difference (%)				
	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	441.3	365.6	475.7	542.4	548.5
$M_s = 0.8$	187.0	200.1	263.5	333.2	386.5
$M_s = 0.9$	160.3	199.9	284.5	373.4	485.5

	p-values				
	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	0.00000	0.00014	0.00000	0.00000	0.00000
$M_s = 0.8$	0.00005	0.00004	0.00000	0.00000	0.00000
$M_s = 0.9$	0.00030	0.00007	0.00000	0.00000	0.00000

Table C.13: Episode Length (#time-steps)
Results (averaged over 8 trials) for varying signal size ($G = \{1, \frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, where $N = 32$) and equilibrium stock multiplier (M_s values of 0.7, 0.8 and 0.9). The following results include:
(i) Absolute values,
(ii) Relative difference (%), i.e., $(\text{Result}_{G=X} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality $X \in \{\frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, and
(iii) Student's T-test p-values with respect to $G = 1$

Absolute Values

	$G=1$	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s = 0.7$	70.78	500.00	440.16	500.00	500.00	500.00
$M_s = 0.8$	200.63	500.00	500.00	500.00	500.00	500.00
$M_s = 0.9$	275.00	500.00	500.00	500.00	500.00	500.00

Relative Difference (%)

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s = 0.7$	606.4	521.9	606.4	606.4	606.4
$M_s = 0.8$	149.2	149.2	149.2	149.2	149.2
$M_s = 0.9$	81.8	81.8	81.8	81.8	81.8

p-values

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s = 0.7$	0.00001	0.00072	0.00001	0.00001	0.00001
$M_s = 0.8$	0.00418	0.00418	0.00418	0.00418	0.00418
$M_s = 0.9$	0.01919	0.01919	0.01919	0.01919	0.01919

Table C.14: Training Time (#episodes)
Results (averaged over 8 trials) for varying signal size ($G = \{1, \frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, where $N = 32$) and equilibrium stock multiplier (M_s values of 0.7, 0.8 and 0.9). The following results include:
(i) Absolute values,
(ii) Relative difference (%), i.e., (Result$_{G=X}$ − Result$_{G=1}$)/Result$_{G=1}$, where Result$_{G=X}$ denotes the achieved result using a signal of cardinality $X \in \{\frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, and
(iii) Student's T-test p-values with respect to $G = 1$

	Absolute Values					
	$G = 1$	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	4757.00	4039.00	3493.00	3279.00	3147.00	3767.00
$M_s = 0.8$	4875.75	3572.00	2933.00	2890.00	2680.00	3228.00
$M_s = 0.9$	4511.13	3479.00	2386.00	2896.00	2575.00	2123.00

	Relative Difference (%)				
	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	-15.1	-26.6	-31.1	-33.8	-20.8
$M_s = 0.8$	-26.7	-39.8	-40.7	-45.0	-33.8
$M_s = 0.9$	-22.9	-47.1	-35.8	-42.9	-52.9

	p-values				
	$G = \frac{N}{2}$	$G = 23$	$G = N$	$G = 41$	$G = \frac{3N}{2}$
$M_s = 0.7$	0.00001	0.01975	0.00001	0.00000	0.00001
$M_s = 0.8$	0.01919	0.00102	0.01919	0.00000	0.01919
$M_s = 0.9$	0.01919	0.00004	0.01919	0.00003	0.01919

Table C.15: Jain Index (higher is better)
Results (averaged over 8 trials) for varying signal size ($G = \{1, \frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, where $N = 32$) and equilibrium stock multiplier (M_s values of 0.7, 0.8 and 0.9). The following results include:
(i) Absolute values,
(ii) Relative difference (%), i.e., $(\text{Result}_{G=X} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality $X \in \{\frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, and
(iii) Student's T-test p-values with respect to $G = 1$

Absolute Values

	$G=1$	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	0.99635	0.99489	0.99594	0.99660	0.99744	0.99686
$M_s=0.8$	0.99023	0.99517	0.99665	0.99733	0.99742	0.99780
$M_s=0.9$	0.99234	0.99718	0.99777	0.99810	0.99863	0.99884

Relative Difference (%)

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	-0.1	0.0	0.0	0.1	0.1
$M_s=0.8$	0.5	0.6	0.7	0.7	0.8
$M_s=0.9$	0.5	0.5	0.6	0.6	0.7

p-values

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	0.69758	0.91342	0.94743	0.77056	0.89174
$M_s=0.8$	0.32098	0.20210	0.16118	0.15617	0.13698
$M_s=0.9$	0.12907	0.09127	0.07503	0.05402	0.04728

Table C.16: Gini Coefficient
Results (averaged over 8 trials) for varying signal size ($G = \{1, \frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, where $N = 32$) and equilibrium stock multiplier (M_s values of 0.7, 0.8 and 0.9). The following results include:
(i) Absolute values,
(ii) Relative difference (%), i.e., $(\text{Result}_{G=X} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality $X \in \{\frac{N}{2}, 23, N, 41, \frac{3N}{2}\}$, and
(iii) Student's T-test p-values with respect to $G = 1$

Absolute Values

	$G=1$	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	0.01252	0.04010	0.03525	0.03259	0.02801	0.03110
$M_s=0.8$	0.03503	0.03873	0.03253	0.02884	0.02838	0.02590
$M_s=0.9$	0.03275	0.02992	0.02646	0.02440	0.02065	0.01885

Relative Difference (%)

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	220.3	181.5	160.3	123.7	148.4
$M_s=0.8$	10.6	-7.1	-17.7	-19.0	-26.1
$M_s=0.9$	-8.7	-19.2	-25.5	-37.0	-42.4

p-values

	$G=\frac{N}{2}$	$G=23$	$G=N$	$G=41$	$G=\frac{3N}{2}$
$M_s=0.7$	0.04090	0.08751	0.12355	0.22435	0.15164
$M_s=0.8$	0.82971	0.88448	0.71927	0.69934	0.59813
$M_s=0.9$	0.82380	0.62158	0.51399	0.34755	0.28352

Table C.17: Social Welfare, Episode Length, Training Time, Jain Index, Gini Coefficient
Results (averaged over 8 trials) for higher growth rate ($r = 2$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$), and population size $N = 64$.

	Social Welfare		**Episode Length**		**Training Time**		**Jain Index**		**Gini Coefficient**	
	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$
$M_s=0.2$	7.52	7.53	1.04	1.05	5000.00	5000.00	0.99955	0.99959	0.01193	0.01129
$M_s=0.3$	11.65	11.57	1.15	1.13	5000.00	5000.00	0.99953	0.99963	0.01215	0.01084
$M_s=0.4$	21.00	19.89	2.54	2.24	5000.00	5000.00	0.99776	0.99795	0.02611	0.02549
$M_s=0.5$	23.52	22.52	2.52	2.19	5000.00	5000.00	0.99467	0.99724	0.04075	0.02941
$M_s=0.6$	23.91	26.33	1.71	2.72	5000.00	5000.00	0.99243	0.99696	0.04894	0.03108
$M_s=0.7$	26.62	354.47	3.08	306.34	5000.00	5000.00	0.99066	0.99728	0.05172	0.02909
$M_s=0.8$	219.16	922.99	190.03	453.91	4957.13	5000.00	0.97867	0.99834	0.06876	0.02285
$M_s=0.9$	1320.48	7171.94	500.00	500.00	4900.00	2053.00	0.97524	0.99911	0.07791	0.01666
$M_s=1.0$	2490.16	16849.49	500.00	500.00	3003.00	909.00	0.99858	0.99890	0.00612	0.01856
$M_s=1.1$	17286.26	18737.69	500.00	500.00	853.00	1193.00	0.99997	0.99914	0.00192	0.01640
$M_s=1.2$	20740.25	19354.80	500.00	500.00	781.00	1427.00	0.99996	0.99930	0.00220	0.01480

Table C.18: Social Welfare, Episode Length, Training Time, Jain Index, Gini Coefficient
(i) Relative difference in the achieved result when signal of cardinality $G = N$ is introduced ($(\text{Result}_{G=N} - \text{Result}_{G=1})/\text{Result}_{G=1}$, where $\text{Result}_{G=X}$ denotes the achieved result using a signal of cardinality X), and
(ii) Student's T-test p-values,
Results (averaged over 8 trials) for higher growth rate ($r = 2$), with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$), and population size $N = 64$.

	Social Welfare		**Episode Length**		**Training Time**		**Jain Index**		**Gini Coefficient**	
	(%)	**p-value**	(%)	**p-value**	(%)	**p-value**	(%)	**p-value**	(%)	**p-value**
$M_s = 0.2$	0.1	0.19975	0.2	0.17555	0.0	NaN	0.0	0.21075	-5.4	0.20205
$M_s = 0.3$	-0.7	0.00261	-1.7	0.00274	0.0	NaN	0.0	0.00050	-10.8	0.00096
$M_s = 0.4$	-5.3	0.66393	-11.8	0.63772	0.0	NaN	0.0	0.41637	-2.4	0.65179
$M_s = 0.5$	-4.2	0.52530	-13.1	0.53060	0.0	NaN	0.3	0.00004	-27.8	0.00001
$M_s = 0.6$	10.1	0.01996	59.3	0.02644	0.0	NaN	0.5	0.00000	-36.5	0.00000
$M_s = 0.7$	1231.4	0.00000	9862.3	0.00000	0.0	NaN	0.7	0.00005	-43.8	0.00001
$M_s = 0.8$	321.1	0.00006	138.9	0.01041	0.9	0.33428	2.0	0.00097	-66.8	0.00204
$M_s = 0.9$	443.1	0.00000	0.0	NaN	-58.1	0.00000	2.4	0.00000	-78.6	0.00000
$M_s = 1.0$	576.6	0.00000	0.0	NaN	-69.7	0.00000	0.0	0.51998	203.5	0.00000
$M_s = 1.1$	8.4	0.00000	0.0	NaN	39.9	0.00117	-0.1	0.00000	753.5	0.00000
$M_s = 1.2$	-6.7	0.00000	0.0	NaN	82.7	0.00000	-0.1	0.00000	573.6	0.00000

Table C.19: Average number of agents in each bin (i.e., harvesting with effort $\epsilon \in [0-0.33)$ ('idle'), $[0.33-0.66)$ ('moderate'), and $[0.66-1]$ ('active')). The presented values start from the first equilibrium stock multiplier (M_s) where a non-depleting strategy was achieved in each setting. Results (averaged over 8 trials) for increasing population size, with ($G = N$) and without ($G = 1$) the introduced signal, for environments of decreasing difficulty (increasing $S_{eq} \propto M_s$).

Number of 'idle' agents: $\epsilon \in [0-0.33)$

	$N=8$		$N=16$		$N=32$		$N=64$, $r=1$		$N=64$, $r=2$	
	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$
$M_s=0.2$										
$M_s=0.3$										
$M_s=0.4$		**4.9**		**9.4**						
$M_s=0.5$	**0.4**	3.7		6.9						
$M_s=0.6$	0.0	2.8	**0.0**	5.0		**12.1**				
$M_s=0.7$	0.0	1.8	0.0	3.3		6.6				**31.6**
$M_s=0.8$	0.0	1.0	0.0	2.0		3.6				5.9
$M_s=0.9$	0.0	0.4	0.0	0.8		1.4		**2.0**		1.8
$M_s=1.0$	0.0	0.2	0.0	0.2	**0.0**	0.3	**0.0**	1.2	**0.0**	1.6
$M_s=1.1$	0.0	0.0	0.0	0.0	0.0	0.2	0.0	1.6	0.0	1.3
$M_s=1.2$	0.0	0.0	0.0	0.0	0.0	0.2	0.0	1.9	0.0	1.1

Number of 'moderate' agents: $\epsilon \in [0.33-0.66)$

	$N=8$		$N=16$		$N=32$		$N=64$, $r=1$		$N=64$, $r=2$	
	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$
$M_s=0.2$										
$M_s=0.3$										
$M_s=0.4$		**0.7**		**2.0**						
$M_s=0.5$	**6.8**	1.0		2.5						
$M_s=0.6$	6.4	1.3	**13.4**	3.0		**5.8**				
$M_s=0.7$	3.6	1.5	6.1	3.0		5.6				**6.6**
$M_s=0.8$	1.6	1.5	1.5	2.3		4.8				11.6
$M_s=0.9$	0.4	0.9	0.6	1.6		3.3		**7.8**		7.8
$M_s=1.0$	0.1	0.5	0.1	0.5	**0.4**	1.5	**0.0**	6.3	**0.1**	6.6
$M_s=1.1$	0.0	0.0	0.0	0.0	0.0	0.9	0.0	6.4	0.0	5.9
$M_s=1.2$	0.0	0.0	0.0	0.0	0.0	1.4	0.0	7.0	0.0	5.7

Number of 'active' agents: $\epsilon \in [0.66-1]$

	$N=8$		$N=16$		$N=32$		$N=64$, $r=1$		$N=64$, $r=2$	
	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$	$G=1$	$G=N$
$M_s=0.2$										
$M_s=0.3$										
$M_s=0.4$		**2.4**		**4.5**						
$M_s=0.5$	**0.9**	3.3		6.6						
$M_s=0.6$	1.6	3.9	**2.6**	8.0		**14.1**				
$M_s=0.7$	4.4	4.7	9.9	9.8		19.8				**25.7**
$M_s=0.8$	6.4	5.5	14.5	11.7		23.6				46.6
$M_s=0.9$	7.6	6.6	15.4	13.6		27.3		**54.3**		54.4
$M_s=1.0$	7.9	7.4	15.9	15.3	**31.6**	30.2	**64.0**	56.5	**63.9**	55.8
$M_s=1.1$	8.0	8.0	16.0	16.0	32.0	30.9	64.0	56.0	64.0	56.8
$M_s=1.2$	8.0	8.0	16.0	16.0	32.0	30.4	64.0	55.2	64.0	57.2

Table C.20: Numerical results of the last 400 episodes of each training trial (averaged over the 8 trials).
Each odd column represents the relative difference (%) of the particular configuration of the policymaker, as compared to the market equilibrium prices ($100(X_{\text{policymaker}} - Y_{\text{market eq.}})/Y_{\text{market eq.}}$), for each of the metrics presented in each row.
Each even column shows the Student's T-test p-values.
The first two columns refers to the vanilla policymaker, where each objective in the reward function has the same weight (see Section 9.5), and each of the following 8 columns refers to a policymaker that only optimizes the specific objective in the title (having weight 0 for the rest).
Finally, the last 8 columns refer to a vanilla policymaker with obfuscated valuations (see Section 9.4.6). The first 4 of them split the valuations into 50 and 10 bins, respectively, while the last 4 add uniform noise (5% and 10%, respectively).
The p-values are computed as follows: The p-value for the vanilla policymaker is calculated using the results from the market equilibrium prices (i.e., we measure the significance of the difference of the policymaker results compared to the MEP). The p-value for any of the following policymakers is calculated using the results from the vanilla policymaker (i.e., we measure if there is a statistically significant change between the vanilla and the optimized policymaker).
Finally, note that the stock difference has negative values (negative deviation from the equilibrium stock) thus, in this metric, large negative numbers are *in favor* of the policymaker.

	Policymaker																	
	vanilla	p-value	$w_h = 1$	p-value	$w_b = 1$	p-value	$w_s = 1$	p-value	$w_f = 1$	p-value	Noisy (50)	p-value	Noisy (10)	p-value	Uni (0.05)	p-value	Uni (0.1)	p-value
Harvesters' Social Welfare	-7.44	1.21e-08	-1.74	4.16e-07	-72.91	4.87e-17	-31.37	7.00e-06	-34.14	1.67e-06	-11.35	2.38e-05	-9.71	2.95e-03	-8.20	2.50e-01	-10.07	9.92e-04
Buyers' Social Welfare	-7.01	2.26e-06	-24.71	6.65e-08	15.42	9.72e-13	1.23	2.28e-04	2.88	2.14e-05	-9.73	9.88e-03	-11.51	2.25e-04	-13.68	3.89e-06	-11.56	2.00e-04
Stock Difference	-15.30	2.72e-09	-2.64	2.66e-08	-10.58	2.34e-02	-21.83	2.36e-03	-12.99	1.53e-01	-23.40	2.72e-06	-21.73	3.42e-05	-24.68	4.90e-07	-22.28	1.41e-05
Harvesters' Fairness Jain	-0.61	3.71e-03	-0.05	6.94e-03	-0.64	8.96e-01	-0.72	6.81e-01	-0.14	2.13e-02	-1.16	6.44e-03	-1.04	2.70e-02	-1.33	1.02e-03	-1.76	1.15e-05
Harvesters' Fairness Gini	-2.78	2.87e-04	-0.54	2.09e-03	-2.86	9.16e-01	-3.08	6.99e-01	-1.29	2.48e-02	-4.57	7.44e-03	-4.09	3.85e-02	-4.75	4.01e-03	-5.52	2.87e-04
Harvesters' Fairness Atkinson	-0.29	4.45e-03	-0.03	8.55e-03	-0.29	9.53e-01	-0.33	7.54e-01	-0.07	2.32e-02	-0.59	3.16e-03	-0.48	3.67e-02	-0.66	6.50e-04	-0.93	2.68e-06
Buyers' Fairness Jain	-0.12	5.48e-05	-0.18	1.32e-01	-0.05	1.30e-02	-0.09	4.07e-01	-0.07	6.31e-02	-0.27	7.60e-06	-0.31	9.04e-07	-0.31	3.82e-07	-0.31	1.14e-06
Buyers' Fairness Gini	-1.49	3.26e-07	-1.96	1.16e-01	-0.84	7.07e-03	-1.29	4.04e-01	-1.06	4.31e-02	-2.57	1.83e-05	-2.74	6.28e-06	-2.81	1.80e-06	-2.78	2.97e-06
Buyers' Fairness Atkinson	-0.06	5.50e-05	-0.09	1.38e-01	-0.02	1.19e-02	-0.05	4.16e-01	-0.03	5.61e-02	-0.13	8.68e-06	-0.15	1.19e-06	-0.15	4.24e-07	-0.15	1.29e-06

Bibliography

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Abel, D. (2019). Concepts in bounded rationality: Perspectives from reinforcement learning. Master's thesis, Brown University.

Adsul, B., Babu, C. S., Garg, J., Mehta, R., and Sohoni, M. (2010). Nash equilibria in fisher market. In *International Symposium on Algorithmic Game Theory*, pages 30–41. Springer.

Agassounon, W. and Martinoli, A. (2002). Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, AAMAS '02, page 1090–1097, New York, NY, USA. Association for Computing Machinery.

Agatz, N., Erera, A., Savelsbergh, M., and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303.

Agatz, N. A., Erera, A. L., Savelsbergh, M. W., and Wang, X. (2011). Dynamic ride-sharing: A simulation study in metro atlanta. *Procedia-Social and Behavioral Sciences*, 17:532–550.

Agussurja, L., Kumar, A., and Lau, H. C. (2018). Resource-constrained scheduling for maritime traffic management. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI18)*.

Ahmed, F., Dickerson, J. P., and Fuge, M. (2017). Diverse weighted bipartite b-matching. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Ijcai'17, page 35–41. AAAI Press.

Albrecht, S. V., Crandall, J. W., and Ramamoorthy, S. (2016). Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94.

Alonso-González, M. J., van Oort, N., Cats, O., Hoogendoorn-Lanser, S., and Hoogendoorn, S. (2020). Value of time and reliability for urban pooled on-demand services. *Transportation Research Part C: Emerging Technologies*, 115:102621.

Alonso-Mora, J., Samaranayake, S., Wallar, A., Frazzoli, E., and Rus, D. (2017). On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*.

Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., and Palamidessi, C. (2013). Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, Ccs '13.

Anshelevich, E., Filos-Ratsikas, A., Shah, N., and Voudouris, A. A. (2021). Distortion in social choice problems: The first 15 years and beyond. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI-21)*.

Arrow, K. J. and Debreu, G. (1954). Existence of an equilibrium for a competitive economy. *Econometrica: Journal of the Econometric Society*, pages 265–290.

Asadpour, A., Lobel, I., and van Ryzin, G. (2020). Minimum earnings regulation and the stability of marketplaces. In *Proceedings of the 21st ACM Conference on Economics and Computation*, Ec '20. Acm.

Ashlagi, I., Azar, Y., Charikar, M., Chiplunkar, A., Geri, O., Kaplan, H., Makhijani, R., Wang, Y., and Wattenhofer, R. (2017). Min-cost bipartite perfect matching with delays. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Ashlagi, I., Burq, M., Dutta, C., Jaillet, P., Saberi, A., and Sholley, C. (2018). Maximum weight online matching with deadlines. *arXiv preprint arXiv:1808.03526*.

Atkinson, A. B. (1970). On the measurement of inequality. *Journal of Economic Theory*, 2(3):244–263.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Aumann, R. J. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96.

Banerjee, S., Freund, D., and Lykouris, T. (2017). Pricing and optimization in shared vehicle systems: An approximation framework. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. Acm.

Bansal, N., Buchbinder, N., Madry, A., and Naor, J. (2015). A polylogarithmic-competitive algorithm for the k-server problem. *Journal of the ACM*, 62(5):1–49.

Barrett, S., Rosenfeld, A., Kraus, S., and Stone, P. (2017). Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence*, 242:132–171.

Barrett, S. and Stone, P. (2011). Ad hoc teamwork modeled with multi-armed bandits: An extension to discounted infinite rewards. In *Proceedings of 2011 AAMAS Workshop on Adaptive and Learning Agents*, pages 9–14.

Bartal, Y. (1996). Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. of 37th Conference on Foundations of Computer Science*. Ieee.

Bartal, Y. and Grove, E. (2000). The harmonic k-server algorithm is competitive. *Journal of the ACM (JACM)*.

Bathla, K., Raychoudhury, V., Saxena, D., and Kshemkalyani, A. D. (2018). Real-time distributed taxi ride sharing. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2044–2051. Ieee.

Baumann, T., Graepel, T., and Shawe-Taylor, J. (2020). Adaptive mechanism design: Learning to promote cooperation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. Ieee.

Bei, X., Garg, J., Hoefer, M., and Mehlhorn, K. (2016). Computing equilibria in markets with budget-additive utilities. In *24th Annual European Symposium on Algorithms, ESA 2016*, page 8. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing.

Bei, X. and Zhang, S. (2018). Algorithms for trip-vehicle assignment in ride-sharing. In *Thirty-Second AAAI*.

Bélanger, V., Kergosien, Y., Ruiz, A., and Soriano, P. (2016). An empirical comparison of relocation strategies in real-time ambulance fleet management. *Computers & Industrial Engineering*, 94:216–229.

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90.

Bellman, R. (2013). *Dynamic programming*. Courier Corporation.

BenHassine, A. and Ho, T. B. (2007). An agent-based approach to solve dynamic meeting scheduling problems with preferences. *Engineering Applications of Artificial Intelligence*, 20(6):857–873.

Bernstein, D. S., Hansen, E. A., and Zilberstein, S. (2005). Bounded policy iteration for decentralized pomdps. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, Ijcai'05, page 1287–1292, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bertsekas, D. P. (1979). A distributed algorithm for the assignment problem. *Lab. for Information and Decision Systems Working Paper, MIT*.

Bertsekas, D. P. (1998). *Network optimization continuous and discrete models*. Athena Scientific Belmont.

Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R. (2011). Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 19–26.

Bienkowski, M., Kraska, A., Liu, H.-H., and Schmidt, P. (2018). A primal-dual online deterministic algorithm for matching with delays. In *International Workshop on Approximation and Online Algorithms*. Springer.

Birnbaum, B., Devanur, N. R., and Xiao, L. (2011). Distributed algorithms via gradient descent for fisher markets. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 127–136.

Bliek, C., Bonami, P., and Lodi, A. (2014). Solving mixed-integer quadratic programming problems with ibm-cplex: a progress report. In *Proceedings of the twenty-sixth RAMP symposium*, pages 16–17.

Bonabeau, E., Sobkowski, A., Theraulaz, G., and Deneubourg, J.-L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. In *Biocomputing and Emergent Computation: Proceedings of BCEC97*, page 36–45. World Scientific Press.

Borchardt, C. W. and Jacobi, C. G. (1865). De investigando ordine systematis aequationum differentialium vulgarium cujuscunque. *Journal für die reine und angewandte Mathematik*, 64:297–320.

Borodin, A. and El-Yaniv, R. (2005). *Online computation and competitive analysis*. cambridge university press.

Borowski, H. P., Marden, J. R., and Shamma, J. S. (2014). Learning efficient correlated equilibria. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6836–6841. Ieee.

Brainard, W. C. and Scarf, H. E. (2005). How to compute equilibrium prices in 1891. *American Journal of Economics and Sociology*, 64(1):57–83.

Bramoullé, Y., López-Pintado, D., Goyal, S., and Vega-Redondo, F. (2004). Network formation and anti-coordination games. *International Journal of Game Theory*, 33(1):1–19.

Brand, A., Allen, L., Altman, M., Hlava, M., and Scott, J. (2015). Beyond authorship: attribution, contribution, collaboration, and credit. *Learned Publishing*, 28(2):151–155.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., and Procaccia, A. D. (2016). *Handbook of computational social choice*. Cambridge University Press.

Brandt, F., Fischer, F., and Holzer, M. (2009). Symmetries and the complexity of pure nash equilibrium. *J. Comput. Syst. Sci.*, 75(3):163–177.

Brânzei, S., Chen, Y., Deng, X., Filos-Ratsikas, A., Frederiksen, S., and Zhang, J. (2014). The fisher market game: Equilibrium and welfare. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Brânzei, S. and Filos-Ratsikas, A. (2019). Walrasian dynamics in multi-unit markets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1812–1819.

Brown, T. (2016a). Matchmaking in lyft line — part 1. eng.lyft.com/matchmaking-in-lyft-line-9c2635fe62c4.

Brown, T. (2016b). Matchmaking in lyft line — part 2. eng.lyft.com/matchmaking-in-lyft-line-691a1a32a008.

Buchbinder, N., Coester, C., Joseph, and Naor (2020). Online k-taxi via double coverage and time-reverse primal-dual.

Buchholz, N. (2018). Spatial equilibrium, search frictions and dynamic efficiency in the taxi industry. Technical report, mimeo, Princeton University.

Budescu, D. V., Au, W. T., and Chen, X.-P. (1997). Effects of protocol of play and social orientation on behavior in sequential resource dilemmas. *Organizational Behavior and Human Decision Processes*, 69(3):179–193.

Bürger, M., Notarstefano, G., Bullo, F., and Allgöwer, F. (2012). A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. *Automatica*, 48(9):2298–2304.

Burkard, R., Dell'Amico, M., and Martello, S. (2012). *Assignment Problems*. Society for Industrial and Applied Mathematics.

Busoniu, L., Babuska, R., and De Schutter, B. (2008). A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172.

Cai, Q., Filos-Ratsikas, A., Tang, P., and Zhang, Y. (2018). Reinforcement mechanism design for e-commerce. In *Proceedings of the 2018 World Wide Web Conference*, pages 1339–1348.

Calhoun, J. B. (1962). Population density and social pathology. *Scientific American*, 206(2):139–149.

Calhoun, J. B. (1973). Death squared: The explosive growth and demise of a mouse population. *Proceedings of the Royal Society of Medicine*, 66(1p2):80–88.

Casari, M. and Plott, C. R. (2003). Decentralized management of common property resources: experiments with a centuries-old institution. *Journal of Economic Behavior & Organization*, 51(2):217–247.

Chakrabarty, D., Devanur, N., and Vazirani, V. V. (2006). New results on rationality and strongly polynomial time solvability in eisenberg-gale markets. In *International Workshop on Internet and Network Economics*, pages 239–250. Springer.

Chakraborty, M., Chua, K. Y. P., Das, S., and Juba, B. (2017). Coordinated versus decentralized exploration in multi-agent multi-armed bandits. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 164–170.

Challet, D., Marsili, M., Zhang, Y.-C., et al. (2013). Minority games: interacting agents in financial markets. *OUP Catalogue*.

Charness, G. and Rabin, M. (2002). Understanding social preferences with simple tests*. *The Quarterly Journal of Economics*, 117(3):817–869.

Chatzikokolakis, K., Andrés, M. E., Bordenabe, N. E., and Palamidessi, C. (2013). Broadening the scope of differential privacy using metrics. In *Privacy Enhancing Technologies*.

Chen, M., Shen, W., Tang, P., and Zuo, S. (2019). Dispatching through pricing: modeling ride-sharing and designing dynamic prices. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*.

Chen, N., Deng, X., Tang, B., and Zhang, H. (2016). Incentives for strategic behavior in fisher market games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Chen, R. and Cassandras, C. G. (2019). Optimization of ride sharing systems using event-driven receding horizon control. *arXiv:1901.01919*.

Chen, X., Dai, D., Du, Y., and Teng, S.-H. (2009). Settling the complexity of arrow-debreu equilibria in markets with additively separable utilities. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 273–282. Ieee.

Chen, X., Paparas, D., and Yannakakis, M. (2017). The complexity of non-monotone markets. *Journal of the ACM (JACM)*, 64(3):1–56.

Cheng, S.-F. and Nguyen, T. D. (2011). Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In *Ieee/wic/acm*, volume 2, pages 14–21.

Cheung, Y. K., Cole, R., and Tao, Y. (2018). Dynamics of distributed updating in fisher markets. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 351–368.

Chrobak, M., Karloff, H., Payne, T., and Vishwanathan, S. (1990). New results on server problems. *SIAM Journal on Discrete Mathematics*, pages 291–300.

Chrobak, M. and Larmore, L. L. (1991a). An optimal on-line algorithm for k servers on trees. *SIAM Journal on Computing*.

Chrobak, M. and Larmore, L. L. (1991b). The server problem and on-line games. *On-line algorithms*.

Cigler, L. and Faltings, B. (2013). Decentralized anti-coordination through multi-agent learning. *Jair*, 47:441–473.

Clark, C. W. (2006). *The worldwide crisis in fisheries: economic models and human behavior*. Cambridge University Press.

Codenotti, B., Saberi, A., Varadarajan, K., and Ye, Y. (2006). Leontief economies encode nonzero sum two-player games. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 659–667.

Coester, C. and Koutsoupias, E. (2019). The online k-taxi problem. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*. Acm.

Colas, C., Sigaud, O., and Oudeyer, P.-Y. (2019). A hitchhiker's guide to statistical comparisons of reinforcement learning algorithms. *arXiv preprint arXiv:1904.06979*.

Cooper, R. (1999). *Coordination Games*. Cambridge University Press.

Cordeau, J.-F. and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of operations research*, 153(1):29–46.

Crandall, J. W. (2014). Towards minimizing disappointment in repeated games. *Journal of Artificial Intelligence Research*, 49:111–142.

Crawford, E. and Veloso, M. (2005). Learning dynamic preferences in multi-agent meeting scheduling. In *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 487–490. Ieee.

Cuccu, G., Togelius, J., and Cudré-Mauroux, P. (2019). Playing atari with six neurons. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '19, page 998–1006, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Danassis, P., Erden, Z. D., and Faltings, B. (2021a). Improved cooperation by exploiting a common signal. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS-21*, page 395–403. International Foundation for Autonomous Agents and Multiagent Systems.

Danassis, P., Erden, Z. D., and Faltings, B. (2022a). Exploiting environmental signals to enable policy correlation in large-scale decentralized systems. *Autonomous Agents and Multi-Agent Systems*.

Danassis, P. and Faltings, B. (2019). Courtesy as a means to coordinate. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS-19*, page 665–673. International Foundation for Autonomous Agents and Multiagent Systems.

Danassis, P. and Faltings, B. (2020). Efficient allocations in constant time: Towards scalable solutions in the era of large scale intelligent systems. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J., editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 2895–2896. IOS Press.

Danassis, P., Filos-Ratsikas, A., and Faltings, B. (2019). Anytime heuristic for weighted matching through altruism-inspired behavior. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 215–222. International Joint Conferences on Artificial Intelligence Organization.

Danassis, P., Filos-Ratsikas, A., and Faltings, B. (2021b). Achieving diverse objectives with ai-driven prices in deep reinforcement learning multi-agent markets. *ArXiv: 2106.06060*.

Danassis, P., Sakota, M., Filos-Ratsikas, A., and Faltings, B. (2022b). Putting ridesharing to the test: Efficient and scalable solutions and the power of dynamic vehicle relocation. *Artificial Intelligence Review*.

Danassis, P., Triastcyn, A., and Faltings, B. (2022c). A distributed differentially private algorithm for resource allocation in unboundedly large settings. In *Proceedings of the 21th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS-22*. International Foundation for Autonomous Agents and Multiagent Systems.

Danassis, P., Wiedemair, F., and Faltings, B. (2021c). Improving multi-agent coordination by learning to estimate contention. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 125–131. International Joint Conferences on Artificial Intelligence Organization.

Dantzig, G. B. (1990). *Origins of the simplex method*. Acm.

Dehghani, S., Ehsani, S., Hajiaghayi, M., Liaghat, V., and Seddighin, S. (2017). Stochastic k-server: How should uber work? In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*.

Dickerson, J. P., Sankararaman, K. A., Srinivasan, A., and Xu, P. (2018). Allocation problems in ride-sharing platforms: Online matching with offline reusable resources. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Diekert, F. K. (2012). The tragedy of the commons from a game-theoretic perspective. *Sustainability*, 4(8):1776–1786.

Ding, W. and Lenhart, S. (2010). Introduction to optimal control for discrete time models with an application to disease modeling. In *Modeling Paradigms and Analysis of Disease Trasmission Models*, pages 109–120.

Duetting, P., Feng, Z., Narasimhan, H., Parkes, D., and Ravindranath, S. S. (2019). Optimal auctions through deep learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1706–1715. Pmlr.

Dutta, C. and Sholley, C. (2018). Online matching in a ride-sharing platform. *arXiv preprint arXiv:1806.10327*.

Dvijotham, K., Rabani, Y., and Schulman, L. J. (2020). Convergence of incentive-driven dynamics in fisher markets. *Games and Economic Behavior*.

Dwork, C. (2006). Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V., and Wegener, I., editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg. Springer Berlin Heidelberg.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56.

Edmonds, J. and Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *J. Acm*, 19(2):248–264.

Eisenberg, E. and Gale, D. (1959). Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168.

Elkin, M. (2004). Distributed approximation: A survey. *SIGACT News*, 35(4):40–57.

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2020). Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*.

Fagnant, D. J. and Kockelman, K. M. (2018). Dynamic ride-sharing and fleet sizing for a system of shared autonomous vehicles in austin, texas. *Transportation*, 45(1):143–158.

Fakcharoenphol, J., Rao, S., and Talwar, K. (2004). A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497. Special Issue on STOC 2003.

Feder, T. and Greene, D. (1988). Optimal algorithms for approximate clustering. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 434–444. Acm.

Fehr, E. and Rockenbach, B. (2004). Human altruism: economic, neural, and evolutionary perspectives. *Current Opinion in Neurobiology*, 14(6):784–790.

Fiat, A., Rabani, Y., and Ravid, Y. (1994). Competitive k-server algorithms. *Journal of Computer and System Sciences*.

Fielbaum, A. and Alonso-Mora, J. (2020). Unreliability in ridesharing systems: Measuring changes in users' times due to new requests. *Transportation Research Part C: Emerging Technologies*, 121:102831.

Fielbaum, A., Bai, X., and Alonso-Mora, J. (2021). On-demand ridesharing with optimized pick-up and drop-off walking locations. *Transportation Research Part C: Emerging Technologies*, 126:103061.

Fioretto, F., Lee, C., and Van Hentenryck, P. (2018). Constrained-based differential privacy for mobility services. In *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems*.

Fisher, I. (1892). *Mathematical Investigations in the Theory of Value and Prices*. PhD thesis, Yale University.

Foster, D. P. and Vohra, R. V. (1997). Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(1-2):40.

Franzin, M. S., Freuder, E., Rossi, F., and Wallace, R. (2002). Multi-agent meeting scheduling with preferences: efficiency, privacy loss, and solution quality. In *Proceedings of the AAAI Workshop on Preference in AI and CP*.

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., and Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57.

Ganapathi Subramanian, S., Poupart, P., Taylor, M. E., and Hegde, N. (2020). Multi type mean field reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '20, page 411–419, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Gao, J., Wang, Y., Tang, H., Yin, Z., Ni, L., and Shen, Y. (2017). An efficient dynamic ridesharing algorithm. In *2017 IEEE International Conference on Computer and Information Technology (CIT)*, pages 320–325.

Garg, N. and Nazerzadeh, H. (2020). Driver surge pricing. In *Proceedings of the 21st ACM Conference on Economics and Computation*, Ec '20. Acm.

Geng, Y. and Cassandras, C. G. (2013). New "smart parking" system based on resource allocation and reservations. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139.

Ghili, S. and Kumar, V. (2020). Spatial distribution of supply and the role of market thickness: Theory and evidence from ridesharing. In *Proceedings of the 21st ACM Conference on Economics and Computation*, Ec '20. Acm.

Gil, M., Alajaji, F., and Linder, T. (2013). Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences*, 249:124–131.

Gini, C. (1912). Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi.*

Gintis, H. (2000). Strong reciprocity and human sociality. *Journal of theoretical biology*, 206(2):169–179.

Giordani, S., Lujak, M., and Martinelli, F. (2010). A distributed algorithm for the multi-robot task allocation problem. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 721–730. Springer.

Goemans, M. X. and Williamson, D. P. (1997). The primal-dual method for approximation algorithms and its application to network design problems. *Approximation algorithms for NP-hard problems*, pages 144–191.

Grenager, T., Powers, R., and Shoham, Y. (2002). Dispersion games: general definitions and some specific learning results. In *Aaai/iaai*, pages 398–403.

Guériau, M. and Dusparic, I. (2018). Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Ieee.

Guha, S. and Khuller, S. (1999). Greedy strikes back improved facility location algorithms. *Journal of algorithms*.

Gunn, T. and Anderson, J. (2013). Dynamic heterogeneous team formation for robotic urban search and rescue. *Procedia Computer Science*, 19:22–31. The 4th Int. Conf. on Ambient Systems, Networks and Technologies (ANT 2013), the 3rd Int. Conf. on Sustainable Energy Information Technology (SEIT-2013).

Gupta, S. and Gentry, J. W. (2016). The behavioral responses to perceived scarcity – the case of fast fashion. *The International Review of Retail, Distribution and Consumer Research*, 26(3):260–271.

Hardin, G. (1968). The tragedy of the commons. *science*, 162(3859):1243–1248.

Hart, S. and Mas-Colell, A. (2000). A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150.

Hassine, A. B., Defago, X., and Ho, T. B. (2004). Agent-based approach to dynamic meeting scheduling problems. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, Aamas '04, page 1132–1139, Usa. IEEE Computer Society.

He, S. and Shin, K. G. (2019). Spatio-temporal capsule-based reinforcement learning for mobility-on-demand network coordination. In *The World Wide Web Conference, WWW 2019*, pages 2806–2813. Acm.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI18)*.

Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797.

Ho, S. C., Szeto, W., Kuo, Y.-H., Leung, J. M., Petering, M., and Tou, T. W. (2018). A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*.

Hsu, J., Huang, Z., Roth, A., Roughgarden, T., and Wu, Z. S. (2014). Private matchings and allocations. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, Stoc '14, page 21–30, New York, NY, USA. Association for Computing Machinery.

Hsu, W.-L. and Nemhauser, G. L. (1979). Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1(3):209–215.

Huang, T., Fang, B., Bei, X., and Fang, F. (2019). Dynamic trip-vehicle dispatch with scheduled and on-demand requests. In *The Conference on Uncertainty in Artificial Intelligence (UAI.*

Hughes, E., Leibo, J. Z., Phillips, M., Tuyls, K., Dueñez Guzman, E., Castañeda, A. G., Dunning, I., Zhu, T., McKee, K., Koster, R., Roff, H., and Graepel, T. (2018). Inequity aversion improves cooperation in intertemporal social dilemmas. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Nips'18, page 3330–3340, Red Hook, NY, USA. Curran Associates Inc.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In Coello, C. A. C., editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ismail, S. and Sun, L. (2017). Decentralized hungarian-based approach for fast and scalable task allocation. In *2017 Int. Conf. on Unmanned Aircraft Systems (ICUAS)*, pages 23–28.

Jain, K. and Vazirani, V. V. (2010). Eisenberg–gale markets: algorithms and game-theoretic properties. *Games and Economic Behavior*, 70(1):84–106.

Jain, R., Chiu, D., and Hawe, W. (1998). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. *CoRR*, cs.NI/9809099.

Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J. Z., and De Freitas, N. (2019). Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3040–3049, Long Beach, California, USA. Pmlr.

Jiang, S., Chen, L., Mislove, A., and Wilson, C. (2018). On ridesharing competition and accessibility: Evidence from uber, lyft, and taxi. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018*. Acm.

Kalra, N. and Martinoli, A. (2006). Comparative study of market-based and threshold-based task allocation. In *Distributed autonomous robotic systems 7*, pages 91–101. Springer.

Karimzadehgan, M. and Zhai, C. (2008). Data set for multi-aspect review assignment evaluation. http://sifaka.cs.uiuc.edu/ir/data/review.html. Accessed: 2021-01-14.

Karimzadehgan, M., Zhai, C., and Belford, G. (2008). Multi-aspect expertise matching for review assignment. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*.

Kollock, P. (1998). Social dilemmas: The anatomy of cooperation. *Annual review of sociology*, 24(1):183–214.

Kooti, F., Grbovic, M., Aiello, L. M., Djuric, N., Radosavljevic, V., and Lerman, K. (2017). Analyzing uber's ride-sharing economy. In *Proceedings of the 26th International Conference on World Wide Web Companion, 2017*. Acm.

Korte, B. and Vygen, J. (2012). *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition.

Kosoresow, A. P. (1996). *Design and Analysis of Online Algorithms for Mobile Server Applications*. PhD thesis, Stanford University, Stanford, CA, USA. Aai9702926.

Koster, R., Hadfield-Menell, D., Hadfield, G. K., and Leibo, J. Z. (2020). Silly rules improve the capacity of agents to learn stable enforcement and compliance behaviors. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '20, page 1887–1888, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Koutsoupias, E. (2009). The k-server problem. *Computer Science Review*, 3(2):105–118.

Koutsoupias, E. and Papadimitriou, C. H. (1995). On the k-server conjecture. *Journal of the ACM (JACM)*.

Kuhn, F., Moscibroda, T., and Wattenhofer, R. (2016). Local computation: Lower and upper bounds. *J. Acm*, 63(2).

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics*, 2(1-2):83–97.

Kun, J., Powers, B., and Reyzin, L. (2013). Anti-coordination games and stable graph colorings. In *International Symposium on Algorithmic Game Theory*, pages 122–133. Springer.

Laborie, P., Rogerie, J., Shaw, P., and Vilím, P. (2018). Ibm ilog cp optimizer for scheduling. *Constraints*, 23(2):210–250.

Lee, J. R. (2018). Fusible hsts and the randomized k-server conjecture. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. Ieee.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 464–473. Int. Foundation for Autonomous Agents and Multiagent Systems.

Lenhart, S. and Workman, J. T. (2007). *Optimal control applied to biological models*. CRC press.

Lesmana, N., Zhang, X., and Bei, X. (2019). Balancing efficiency and fairness in on-demand ridesourcing. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NEURIPS)*.

Lewis, D. (2008). *Convention: A philosophical study*. John Wiley & Sons.

Li, M., Qin, Z., Jiao, Y., Yang, Y., Wang, J., Wang, C., Wu, G., and Ye, J. (2019). Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference, WWW 2019*. Acm.

Liang, E., Liaw, R., Nishihara, R., Moritz, P., Fox, R., Gonzalez, J., Goldberg, K., and Stoica, I. (2017). Ray RLlib: A composable and scalable reinforcement learning library. In *Deep Reinforcement Learning symposium (DeepRL @ NeurIPS)*.

Liberti, J. M. and Petersen, M. A. (2019). Information: Hard and soft. *Review of Corporate Finance Studies*, 8(1):1–41.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lioris, J., Cohen, G., Seidowsky, R., and Salem, H. H. (2016). Dynamic evolution and optimisation of an urban collective taxis systems by discrete-event simulation. In *ITS World Congress 2016*, page 10p.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, Icml'94, page 157–163, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Littman, M. L. and Stone, P. (2002). Implicit negotiation in repeated games. In Meyer, J.-J. C. and Tambe, M., editors, *Intelligent Agents VIII: Agent Theories, Architectures, and Languages 8th International Workshop, ATAL 2001 Seattle,WA, USA, August 1–3, 2001 Revised Papers*, pages 393–404, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lokhandwala, M. and Cai, H. (2018). Dynamic ride sharing using traditional taxis and shared autonomous taxis: A case study of nyc. *Transportation Research Part C: Emerging Technologies*, 97:45–60.

Lovász, L. and Plummer, M. D. (2009). *Matching theory*, volume 367. American Mathematical Soc.

Lowalekar, M., Varakantham, P., and Jaillet, P. (2019). Zac: A zone path construction approach for effective real-time ridesharing. In *Icaps*.

Lowe, R., Foerster, J., Boureau, Y.-L., Pineau, J., and Dauphin, Y. (2019). On the pitfalls of measuring emergent communication. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '19, page 693–701, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Lupu, A. and Precup, D. (2020). Gifting in multi-agent reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '20, page 789–797, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Ma, H., Fang, F., and Parkes, D. C. (2019). Spatio-temporal pricing for ridesharing platforms. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 583–583. Acm.

Maheswaran, R. T., Tambe, M., Bowring, E., Pearce, J. P., and Varakantham, P. (2004). Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, Aamas '04, page 310–317, Usa. IEEE Computer Society.

Mallows, C. L. (1957). Non-null ranking models. i. *Biometrika*, 44(1/2):114–130.

Manasse, M., McGeoch, L., and Sleator, D. (1988). Competitive algorithms for on-line problems. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 322–333. Acm.

Manasse, M. S., McGeoch, L. A., and Sleator, D. D. (1990). Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230.

Martínez, L. M., Correia, G. H. d. A., Moura, F., and Mendes Lopes, M. (2017). Insights into carsharing demand dynamics: Outputs of an agent-based model application to lisbon, portugal. *International Journal of Sustainable Transportation*, 11(2):148–159.

Matignon, L., Laurent, G. J., and Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31.

Mihaylov, M., Tuyls, K., and Nowé, A. (2014). A decentralized approach for convention emergence in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 28(5):749–778.

Mironov, I. (2017). Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. Ieee.

Mourad, A., Puchinger, J., and Chu, C. (2019). A survey of models and algorithms for optimizing shared mobility. *Transportation Research Part B: Methodological*.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Nanda, V., Xu, P., Sankararaman, K. A., Dickerson, J. P., and Srinivasan, A. (2020). Balancing the tradeoff between profit and fairness in rideshare platforms during high-demand hours. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*. AAAI Press.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

Nekipelov, D., Syrgkanis, V., and Tardos, E. (2015). Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 1–18.

Nguyen, D. T., Kumar, A., and Lau, H. C. (2017). Collective multiagent sequential decision making under uncertainty. In *Aaai*.

Nigam, A. and Srivastava, S. (2020). Oddms: Online distributed dynamic meeting scheduler. In Fong, S., Dey, N., and Joshi, A., editors, *ICT Analysis and Applications*, pages 199–208, Singapore. Springer Singapore.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge.

Niv, Y. (2009). Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154.

Norris, J. R. (1998). *Markov chains.* Cambridge series in statistical and probabilistic mathematics. Cambridge University Press.

Nowak, M. A. and Sigmund, K. (2005). Evolution of indirect reciprocity. *Nature*, 437(7063):1291.

Ollivier, F. (2009). Looking for the order of a system of arbitrary ordinary differential equations. *Applicable Algebra in Engineering, Communication and Computing*, 20(1):7–32.

Ostrom, E. (1999). Coping with tragedies of the commons. *Annual review of political science*, 2(1):493–535.

Ostrom, E., Gardner, R., Walker, J., and Walker, J. (1994). *Rules, games, and common-pool resources*. University of Michigan Press.

Ottens, B., Dimitrakakis, C., and Faltings, B. (2017). Duct: An upper confidence bound approach to distributed constraint optimization problems. *ACM Trans. Intell. Syst. Technol.*, 8(5).

Papadimitriou, C. H. and Roughgarden, T. (2008). Computing correlated equilibria in multi-player games. *J. Acm*, 55(3):14:1–14:29.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Upper Saddle River, NJ, USA.

Parkes, D. C. and Wellman, M. P. (2015). Economic reasoning and artificial intelligence. *Science*, 349(6245):267–272.

Pelzer, D., Xiao, J., Zehe, D., Lees, M. H., Knoll, A. C., and Aydt, H. (2015). A partition-based match making algorithm for dynamic ridesharing. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2587–2598.

Perolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. (2017). A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652.

Peysakhovich, A. and Lerer, A. (2018a). Consequentialist conditional cooperation in social dilemmas with imperfect information. In *International Conference on Learning Representations*.

Peysakhovich, A. and Lerer, A. (2018b). Prosocial learning agents solve generalized stag hunts better than selfish ones. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '18, page 2043–2044, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Prorok, A. and Kumar, V. (2017). Privacy-preserving vehicle assignment for mobility-on-demand systems. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1869–1876. Ieee.

Qian, X., Zhang, W., Ukkusuri, S. V., and Yang, C. (2017). Optimal assignment and incentive design in the taxi group ride problem. *Transportation Research Part B: Methodological*, 103:208–226.

Raghavan, P. and Snir, M. (1989). Memory versus randomization in on-line algorithms. In *International Colloquium on Automata, Languages, and Programming*, pages 687–703. Springer.

Rego, V. (1992). Naive asymptotics for hitting time bounds in markov chains. *Acta Informatica*, 29(6):579–594.

Riley, C., van Hentenryck, P., and Yuan, E. (2020). Real-time dispatching of large-scale ride-sharing systems: Integrating optimization, machine learning, and model predictive control. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*.

Roelofsen, A. B. (2020). *An Institutional Approach to Normative Distributed Robotics for Mixed Societies of Humans and Robots*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne.

Romano, N. C. and Nunamaker, J. F. (2001). Meeting analysis: Findings from research and practice. In *Proceedings of the 34th annual Hawaii international conference on system sciences*, pages 13–pp. Ieee.

Roughgarden, T. (2016). *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 1st edition.

Rubinstein, A. and Dalgaard, C.-j. (1998). *Modeling bounded rationality*. MIT press.

Ruch, C., Hörl, S., and Frazzoli, E. (2018). Amodeus, a simulation-based testbed for autonomous mobility-on-demand systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Ieee.

Rudec, T., Baumgartner, A., and Manger, R. (2013). A fast work function algorithm for solving the k-server problem. *Central European Journal of Operations Research*, 21(1):187–205.

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., and Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*.

Santos, D. O. and Xavier, E. C. (2013). Dynamic taxi and ridesharing: A framework and heuristics for the optimization problem. In *Twenty-Third International Joint Conference on Artificial Intelligence*.

Santos, D. O. and Xavier, E. C. (2015). Taxi and ride sharing: A dynamic dial-a-ride problem with money as an incentive. *Expert Systems with Applications*, 42(19):6728–6737.

Schelling, T. C. (1960). The strategy of conflict. *Cambridge, Mass*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Shah, S., Lowalekar, M., and Varakantham, P. (2020). Neural approximate dynamic programming for on-demand ride-pooling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):507–515.

Shaheen, S. and Cohen, A. (2019). Shared ride services in north america: definitions, impacts, and the future of pooling. *Transport Reviews*, 39(4):427–442.

Shapley, L. S. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100.

Shen, W., Tang, P., and Zuo, S. (2019). Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 215–223.

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1):231–252. Computational Research on Interaction and Agency, Part 2.

Silwal, S., Gani, M. O., and Raychoudhury, V. (2019). A survey of taxi ride sharing system architectures. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 144–149. Ieee.

Simon, J. (2016). On the existence of altruistic value and utility functions. *Theory and Decision*, 81(3):371–391.

Simonetto, A., Monteil, J., and Gambella, C. (2019). Real-time city-scale ridesharing via linear assignment problems. *Transportation Research Part C: Emerging Technologies*, 101:208–232.

Smith, A. (1791). *An Inquiry into the Nature and Causes of the Wealth of Nations*, volume 1. Librito Mondi.

Spieser, K., Samaranayake, S., Gruel, W., and Frazzoli, E. (2016). Shared-vehicle mobility-on-demand systems: a fleet operator's guide to rebalancing empty vehicles. In *Transp. Research Board 95th Annual Meeting*.

Spieser, K., Treleaven, K., Zhang, R., Frazzoli, E., Morton, D., and Pavone, M. (2014). *Toward a Systematic Approach to the Design and Evaluation of Automated Mobility-on-Demand Systems: A Case Study in Singapore*, pages 229–245. Springer International Publishing, Cham.

Stiglic, M., Agatz, N., Savelsbergh, M., and Gradisar, M. (2015). The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological*, 82:36–53.

Stone, P., Kaminka, G. A., Kraus, S., and Rosenschein, J. S. (2010). Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Aaai*.

Student (1908). The probable error of a mean. *Biometrika*, pages 1–25.

Su, H.-H. (2015). *Algorithms for Fundamental Problems in Computer Networks*. PhD thesis, University of Michigan.

Sühr, T., Biega, A. J., Zehlike, M., Gummadi, K. P., and Chakraborty, A. (2019). Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Acm.

Talebi, M. S. (2013). Uncoupled learning rules for seeking equilibria in repeated plays: An overview. *CoRR*, abs/1310.5660.

Tang, M., Ow, S., Chen, W., Cao, Y., Lye, K., and Pan, Y. (2017). The data and science behind grabshare carpooling. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*.

Tardos, É. (2019). Learning and efficiency of outcomes in games, seminar slides.

Taxi, N. and Commission, L. (2016). Tlc trip record data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page. Accessed: 2019-11-10.

Tong, W., Hua, J., and Zhong, S. (2017). A jointly differentially private scheduling protocol for ridesharing services. *IEEE Transactions on Information Forensics and Security*, 12(10):2444–2456.

Triastcyn, A. (2020). *Data-Aware Privacy-Preserving Machine Learning*. PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne.

Triastcyn, A. and Faltings, B. (2019). Federated learning with bayesian differential privacy. In *IEEE International Conference on Big Data (Big Data)*. Ieee.

Triastcyn, A. and Faltings, B. (2020). Bayesian differential privacy for machine learning. In *37th International Conference on Machine Learning*.

Tsao, M., Milojevic, D., Ruch, C., Salazar, M., Frazzoli, E., and Pavone, M. (2019). Model predictive control of ride-sharing autonomous mobility-on-demand systems. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6665–6671.

van Engelen, M., Cats, O., Post, H., and Aardal, K. (2018). Enhancing flexible transport services with demand-anticipatory insertion heuristics. *Transportation Research Part E: Logistics and Transportation Review*, 110:110–121.

Van Erven, T. and Harremos, P. (2014). Rényi divergence and kullback-leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820.

Varakantham, P. (2016). Sequential decision making for improving efficiency in urban environments. *AAAI Press*.

Varakantham, P., Cheng, S.-F., Gordon, G., and Ahmed, A. (2012). Decision support for agent populations in uncertain and congested environments. *AAAI Press*.

Vosooghi, R., Puchinger, J., Jankovic, M., and Vouillon, A. (2019). Shared autonomous vehicle simulation and service design. *Transportation Research Part C: Emerging Technologies*, 107:15–33.

Walker, A. and Wooldridge, M. J. (1995). Understanding the emergence of conventions in multi-agent systems. In *Icmas95*, pages 384–389, San Francisco, CA.

Wallar, A., Van Der Zee, M., Alonso-Mora, J., and Rus, D. (2018). Vehicle rebalancing for mobility-on-demand systems with ride-sharing. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4539–4546.

Walras, L. (1874). *Éléments d'Économie Politique Pure*. Librairie Droz.

Wang, J. X., Hughes, E., Fernando, C., Czarnecki, W. M., Duéñez Guzmán, E. A., and Leibo, J. Z. (2019). Evolving intrinsic motivations for altruistic behavior. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, Aamas '19, page 683–692, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Wang, L., Wu, K., Hamdi, M., and Ni, L. M. (2013). Attachment-learning for multi-channel allocation in distributed ofdma-based networks. *IEEE Transactions on Wireless Communications*, 12(4):1712–1721.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

Wasik, A., Tomic, S., Saffiotti, A., Pecora, F., Martinoli, A., and Lima, P. U. (2018). Towards norm realization in institutions mediating human-robot societies. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 297–304.

Wasik, A. B., Martinoli, A., and Lima, P. U. (2017). An institutional robotics approach to the design of socially aware multi-robot behaviors. In *Proceedings of the RO-MAN 2017 Workshop on Towards Intelligent Social Robots: Social Cognitive Systems in Smart Environments*, pages 2–7.

Wen, J., Zhao, J., and Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 220–225.

Widdows, D., Lucas, J., Tang, M., and Wu, W. (2017). Grabshare: The construction of a realtime ridesharing service. In *2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*.

Wiegand, R. P. and Jong, K. A. (2004). *An Analysis of Cooperative Coevolutionary Algorithms*. PhD thesis, George Mason University, Usa. Aai3108645.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*.

Xu, Y. and Xu, P. (2020). Trade the system efficiency for the income equality of drivers in rideshare. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*.

Yang, J., Li, A., Farajtabar, M., Sunehag, P., Hughes, E., and Zha, H. (2020). Learning to incentivize other learning agents. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15208–15219. Curran Associates, Inc.

Young, H. P. (1996). The economics of convention. *The Journal of Economic Perspectives*, 10(2):105–122.

Yuen, C. F., Singh, A. P., Goyal, S., Ranu, S., and Bagchi, A. (2019). Beyond shortest paths: Route recommendations for ride-sharing. In *The World Wide Web Conference*, pages 2258–2269. Acm.

Zavlanos, M. M., Spesivtsev, L., and Pappas, G. J. (2008). A distributed auction algorithm for the assignment problem. In *Decision and Control, 2008.*, pages 1212–1217. Ieee.

Zhang, L. (2011). Proportional response dynamics in the fisher market. *Theoretical Computer Science*, 412(24):2691–2698.

Zhao, B., Xu, P., Shi, Y., Tong, Y., Zhou, Z., and Zeng, Y. (2019). Preference-aware task assignment in on-demand taxi dispatching: An online stable matching approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2245–2252.

Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D. C., and Socher, R. (2020). The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332*.

Zhou, L. (2015). A survey on contextual multi-armed bandits. *arXiv preprint arXiv:1508.03326*.

Zunino, A. and Campo, M. (2009). Chronos: A multi-agent system for distributed automatic meeting scheduling. *Expert Systems with Applications*, 36(3):7011–7018.

PANAYIOTIS DANASSIS

panayiotis.danassis@epfl.ch

RESEARCH EXPERTISE

- Multi-agent Systems (Coordination & Cooperation), Reinforcement Learning, Game Theory

EDUCATION

Ph.D. in Computer Science — **École Polytechnique Fédérale de Lausanne (EPFL)** — **2016 Sep – 2022 Jan**

- Artificial Intelligence Laboratory, School of Computer and Communication Sciences, GPA 6.0/6.0
- Thesis: 'Scalable Multi-agent Coordination and Resource Sharing'
- Advisor: Professor Boi Faltings

Diploma of Engineering — **National Technical University of Athens (NTUA)** — **2009 – 2015**

- School of Electrical and Computer Engineering
- 5-year Diploma (Master Equivalent). Order of admission: 4^{th}/450. Overall GPA 8.14/10, in-major GPA 9.0/10
- Thesis: 'A Novel 3-D FPGA Placement Algorithm based on Ant Colony Optimization'
- Advisor: Associate Professor Dimitrios Soudris

RESEARCH EXPERIENCE

Ph.D. Researcher — **Artificial Intelligence Laboratory, EPFL** — **2016 Sep – Present**

- Large-scale multi-agent systems (cooperation & coordination), reinforcement learning, and game theory

Research Associate — **Microprocessors and Digital Systems Laboratory, NTUA** — **2015 Sep – 2016 Jul**

- 'AEGLE: An analytics framework for integrated and personalized healthcare services in Europe' (H2020)
- Application of machine learning techniques in embedded systems

Undergraduate Research Member — **Microprocessors and Digital Systems Laboratory, NTUA** — **2014 Nov – 2015 Jun**

- 3-D Reconfigurable Architectures (3-D FPGAs), and Swarm Intelligence Algorithms

TEACHING EXPERIENCE

Guest Lecturer

- **Intelligent Agents** (Fall 2017, 2019), School of Computer & Communication Sciences, EPFL

Teaching Assistant

- **Intelligent Agents** (Fall 2017, 2018, 2019, 2020), School of Computer & Communication Sciences, EPFL
- **Intelligence Artificielle** (Spring 2017, 2018, 2020), School of Computer & Communication Sciences, EPFL
- **Microprocessors Laboratory** (Fall 2015), School of Electrical & Computer Engineering, NTUA

Student Supervision

- Supervised 3 Summer Interns, 2 Master Theses, 3 Master Semester Projects, and 3 Bachelor Semester Projects

AWARDS & FELLOWSHIPS

- Teaching Assistant Award, 2019, School of Computer & Communication Sciences, EPFL
- Scholarship from the Greek State Scholarships Foundation (IKY)
- Award from the Greek Minister of Education and Lifelong Learning Ms Anna Diamantopoulou

PROGRAM COMMITTEES

- International Conference on Autonomous Agents and Multiagent Systems (AAMAS) 2022 (subreviewer)
- The Web Conference (formerly known as WWW) 2021 (subreviewer)
- International Joint Conference on Artificial Intelligence (IJCAI) 2020

ACADEMIC VISITS

- Apr 1 - Apr 5, 2019, Singapore Management University, Host: Associate Professor Akshat Kumar

Working Papers

- **P. Danassis**, A. Filos-Ratsikas, B. Faltings, 'Achieving Diverse Objectives with AI-driven Prices in Deep Reinforcement Learning Multi-agent Markets'

Journal Papers

- 2022, **P. Danassis**, M. Sakota, A. Filos-Ratsikas, B. Faltings, 'Putting Ridesharing to the Test: Efficient and Scalable Solutions and the Power of Dynamic Vehicle Relocation', Artificial Intelligence Review
- 2022, **P. Danassis**, Z. D. Erden, B. Faltings, 'Exploiting Environmental Signals to Enable Policy Correlation in Large-scale Decentralized Systems', Journal of Autonomous Agents and Multi-agent Systems

Conference Papers

- 2022, **P. Danassis**, A. Triastcyn, B. Faltings, 'A Distributed Differentially Private Algorithm for Resource Allocation in Unboundedly Large Settings', AAMAS 2022: Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems
- 2021, **P. Danassis**, F. Wiedemair, B. Faltings, 'Improving Multi-agent Coordination by Learning to Estimate Contention', IJCAI 2021: Proceedings of the 30th International Joint Conference on Artificial Intelligence
- 2021, **P. Danassis**, Z. D. Erden, B. Faltings, 'Improved Cooperation by Exploiting a Common Signal', AAMAS 2021: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems
- 2020, **P. Danassis**, B. Faltings, 'Efficient Allocations in Constant Time: Towards Scalable Solutions in the Era of Large Scale Intelligent Systems', ECAI 2020: Proceedings of the 24th European Conference on Artificial Intelligence, 2-page Highlight Paper
- 2019, **P. Danassis**, A. Filos-Ratsikas, B. Faltings, 'Anytime Heuristic for Weighted Matching Through Altruism-Inspired Behavior', IJCAI 2019: Proceedings of the 28th International Joint Conference on Artificial Intelligence
- 2019, **P. Danassis**, B. Faltings, 'Courtesy as a Means to Coordinate', AAMAS 2019: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems

Peer-Reviewed Workshops and Symposia

- 2021, **P. Danassis**, A. Triastcyn, B. Faltings, 'Differential Privacy Meets Maximum-weight Matching', ALA 2021: Adaptive Learning Agents Workshop at AAMAS
- 2021, **P. Danassis**, A. Triastcyn, B. Faltings, 'Differential Privacy Meets Maximum-weight Matching', AASG 2021: Autonomous Agents for Social Good Workshop at AAMAS
- 2021, **P. Danassis**, Z. D. Erden, B. Faltings, 'Improved Cooperation by Exploiting a Common Signal', AASG 2021: Autonomous Agents for Social Good Workshop at AAMAS
- 2021, **P. Danassis**, A. Triastcyn, B. Faltings, 'Differential Privacy Meets Maximum-weight Matching', PPAI 2021: Privacy-Preserving Artificial Intelligence at AAAI
- 2020, **P. Danassis**, B. Faltings, 'Learning to Persist or Switch: Efficient and Fair Allocations in Large-scale Multi-agent Systems', ALA 2020: Adaptive Learning Agents Workshop at AAMAS
- 2019, **P. Danassis**, A. Filos-Ratsikas, B. Faltings, 'Anytime Heuristic for Weighted Matching Through Altruism-Inspired Behavior', ALA 2019: Adaptive Learning Agents Workshop at AAMAS
- 2018, **P. Danassis**, B. Faltings, 'Courtesy as a Means to Anti-coordinate', ALA 2018: Adaptive Learning Agents Workshop at AAMAS
- 2018, **P. Danassis**, B. Faltings, 'Learning in Ad-hoc Anti-coordination Scenarios', AAAI Spring Symposium Series

Publications Prior to my Doctoral Studies

- 2017, **P. Danassis**, K. Siozios, C. Korkas, D. Soudris, E. Kosmatopoulos, 'A Low-Complexity Control Mechanism Targeting Smart Thermostats', Energy and Buildings, Elsevier
- 2017, K. Siozios, **P. Danassis**, N. Zompakis, C. Korkas, E. Kosmatopoulos and D. Soudris, 'Supporting Decision Making for Large-Scale IoTs: Trading Accuracy for Computational Complexity', Components and Services for IoT Platforms: Paving the Way for IoT Standards, Springer (invited book chapter)
- 2016, **P. Danassis**, K. Siozios, D. Soudris, 'ANT3D: Simultaneous Partitioning and Placement for 3-D FPGAs based on Ant Colony Optimization', IEEE Embedded Systems Letters, IEEE
- 2016, **P. Danassis**, K. Siozios, D. Soudris, 'Parallel Application Placement onto 3-D Reconfigurable Architectures', International Conference on Modern Circuits and Systems Technologies (MOCAST), IEEE sponsored