EPFL

Thèse n° 8133

# How to Match in Modern Computational Settings

Présentée le 19 novembre 2021

Faculté informatique et communications
Laboratoire de théorie du calcul 2
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

**Buddhima Ruwanmini Gamlath GAMLATH RALALAGE**

Acceptée sur proposition du jury

Prof. R. Urbanke, président du jury
Prof. O. N. A. Svensson, directeur de thèse
Prof. A. Gupta, rapporteur
Prof. M. Ghaffari, rapporteur
Prof. M. Göös, rapporteur

École polytechnique fédérale de Lausanne

2021

Mathematics my foot! Algorithms are mathematics too,
and often more interesting and definitely more useful.
— Doron Zeilberger

To my parents. . .

# Acknowledgements

Countless people have made my journey towards this Ph.D. a pleasant one, and this is an incomplete attempt to pay tribute to them.

My heartfelt gratitude goes to my advisor Ola Svensson for accepting me as a student, nurturing me as a researcher, and guiding me through every step of my Ph.D. With his unparalleled enthusiasm for solving problems, regular constructive feedback, and funny yet informative discussions on topics from Hemingway's six-word stories to non-fungible tokens, working with Ola for five long years has been a delightful experience. If you ask my advice on choosing an advisor, I cannot suggest anyone better than Ola for the Job.

I am very much thankful to the jury members of my private defense, Rüdiger Urbanke, Mika Göös, Anupam Gupta, and Mohsen Ghaffari, for carefully reading my thesis and for the insightful discussions we had during the defense.

I am grateful to Chantal Schneeberger for shielding us from all the bureaucracies, promptly taking care of all my petty requests and queries, and above all, being the most cheerful person I know. My thanks also go to Pauline Raffestin, Cecilia Chapuis, and all the other administrative staff of EPFL for making my time at EPFL a pleasant experience.

My Ph.D. would not have been a success without my coauthors Adam, Andreas, David, Michael, Sagar, Sangxia, Slobodan, Vadim, and Xinrui, and my colleagues Abbas, Aida, Amir, Ashkan, Christos, Etienne, Farah, Grzegorz, Jakab, Jakub, Justin, Kshiteej, Navid, and Paritosh, with whom I have had many insightful discussions. I especially thank Etienne for translating the abstract of this thesis into French on short notice.

I am thankful to my Sri Lankan friends in Switzerland, Udaranga, Tharindu, Kamalaruban, Nirmana, Ruchiranga, Anuradha, Wenuka, Yasara, Pasindu, and Pradeep, for their immense support and for making me feel at home.

I am eternally grateful to all my mentors, colleagues, and friends at Sri Lanka Olympiad Mathematics Foundation, Dr. Chanakya Wijeratne, Supun, Thameera, Kasun, Nipuna, Nuwan, Pasindu, Dileepa, Lajanugan, Kirshanthan, Tharindu, Sankalpa, Melanka, Kanchana, Mevan, Manuja, Madhura, Isuru, and the members of the IMC 2014 team Dr. Boralugoda, Thilina, Ravi, and Sabri, without whom I would not have been tempted to pursue a Ph.D. in a theoretical discipline.

I would also like to fondly remember my teachers and friends from the University of Moratuwa and my high school for constantly motivating me to pursue further studies.

I extend my sincere gratitude to all taxpayers in Sri Lanka, Switzerland, and the EU for facilitating my education from kindergarten to Ph.D.

Last but not least, I am profoundly grateful to my parents Vijitha and Amarasooriya, my wife Dewmini, and her parents Wansika and Kalyanapala, and my brothers and sisters Kassapa, Sasankara, Hasala, Wathmini, and Thathsara for their unconditional love and encouragement.

*Lausanne, 1 August 2021* Buddhima Gamlath

# Abstract

This thesis focuses on the maximum matching problem in modern computational settings where the algorithms have to make decisions with partial information.

First, we consider two stochastic models called *query-commit* and *price-of-information* where the algorithm only knows the distribution from which the edges are sampled. In the query-commit model, the algorithm must query edges to know if they exist and is committed to adding all queried edges that exist to its output. In the price-of-information model, the algorithm incurs costs for querying edges, and the total query cost is subtracted from the output matching's weight. For maximum weighted matching in these models, previously known best algorithms were greedy algorithms that achieve $1/2$ approximations. We improve the approximation ratio to $1 - 1/e$ in both models.

Next, we consider situations where the input graphs do not fit into the space available for an algorithm instance. We consider two such models: the *semi-streaming* model where the algorithm receives the input as a stream of edges and the algorithm has only sub-linear (in the number of edges) space, and the *massively parallel computation (MPC)* model where the input is distributed among several machines, each of which has sub-linear space, and algorithm instances running on different machines must communicate in synchronous rounds. We start with a particular case of the semi-streaming model where the edges arrive in uniformly random order, and the algorithm goes over the stream only once. For this setting, we give the first algorithm that finds a $(1/2 + c)$-approximate maximum weighted matching in expectation; such algorithms were previously known only for the unweighted graphs. We then show how to efficiently find $(1 - \varepsilon)$-approximate weighted matchings for any $\varepsilon > 0$ in *multi-pass* semi-streaming and MPC models by extending our algorithmic ideas used in the single-pass semi-streaming model with random order edge arrivals.

Finally, we study *online* algorithms for matching, where the input graph is gradually revealed over time. In the online *edge-arrival* setting, the graph is revealed one edge at a time, and an algorithm is forced to make irrevocable decisions on whether to add each edge to the output matching upon their arrival. We show that no online algorithm can achieve a competitive ratio of $1/2 + c$ for any constant $c > 0$ in this setting. In the online *vertex-arrival* setting, the graph is revealed one vertex at a time, together with its incident edges to already revealed vertices, and the algorithm must irrevocably decide to ignore the revealed vertex or match it to one of the available neighbors. In this setting, we show how to round a previously known fractional online matching algorithm [86] to get an integral online matching algorithm with a competitive ratio of $1/2 + c$ for some constant $c > 0$.

Key words: Matching, Stochastic, Query-commit, Price-of-information, Semi-streaming, Random-order, Multi-pass, Massively-parallel-computation, Online, Vertex-arrival, Edge-arrival.

# Résumé

Cette thèse porte sur le problème de couplage maximum dans des environnements informatiques modernes où les algorithmes doivent prendre des décisions à partir d'informations partielles.

Premièrement, nous considérons deux modèles stochastiques appelés *requête-engagement* et *prix-de-l'information* où l'algorithme ne connaît que la distribution à partir de laquelle les arêtes sont échantillonnées. Dans le modèle requête-engagement, l'algorithme doit effectuer une requête pour chaque arête afin de savoir si elle est présente dans le graphe ou non. Si l'arête est effectivement présente, l'algorithme s'engage à ajouter l'arête en question à la solution finale. Dans le modèle prix-de-l'information, l'algorithme doit payer un coût pour chaque requête effectuée et le coût total des requêtes est soustrait du poids total du couplage retourné en sortie. Dans les deux modèles, les meilleurs algorithmes connus jusqu'ici pour calculer un couplage de poids maximum étaient des algorithmes gloutons qui garantissent une $1/2$-approximation. Nous améliorons le facteur d'approximation pour obtenir $1 - 1/e$ dans les deux modèles.

Ensuite, nous considérons des situations où les graphes en entrée ne peuvent pas être stockés entièrement en mémoire à cause de leur taille. Nous considérons deux de ces modèles : le modèle *semi-streaming* où l'algorithme reçoit l'entrée sous forme de flux d'arêtes et l'algorithme n'a qu'un espace sous-linéaire (en nombre d'arêtes), et le modèle de *traitement massivement parallèle (TMP)* dans lequel l'entrée est répartie sur plusieurs machines, chacune ayant un espace sous-linéaire, et les instances de l'algorithme s'exécutant sur différentes machines doivent communiquer en tours synchronisés. Nous commençons par un cas particulier du modèle semi-streaming où les arêtes arrivent dans un ordre uniformément aléatoire, et l'algorithme ne peut lire le flux qu'une seule fois. Dans ce cas, nous donnons le premier algorithme qui garantit, en espérance, une approximation à facteur $(1/2 + c)$ du couplage pondéré maximum. Auparavant, de tels algorithmes n'étaient connus que pour les graphes non pondérés. Nous montrons ensuite comment trouver efficacement des couplages pondérés à un facteur $(1 - \varepsilon)$ du couplage maximum pour tout $\varepsilon > 0$ dans les modèles *multi-passe* semi-streaming et TMP en étendant nos idées algorithmiques utilisées dans le modèle semi-streaming avec une seule lecture du flux et des arêtes dans un ordre uniformément aléatoire.

Enfin, nous étudions des algorithmes *en ligne* de couplage, où le graphe en entrée est progressivement révélé au fil du temps. Dans le modèle en ligne *arrivée-d'arête*, le graphe est révélé une arête à la fois, et l'algorithme est obligé de prendre une décision irrévocable d'ajouter ou non au couplage final l'arête qui vient d'être révélée. Nous montrons qu'aucun algorithme en ligne ne peut garantir un ratio compétitif de $1/2 + c$ pour toute constante $c > 0$ dans ce modèle. Dans le modèle en ligne *arrivée-de-sommet*, le graphe est révélé un sommet à la fois, avec ses arêtes incidentes aux sommets déjà révélés, et l'algorithme doit irrévocablement décider d'ignorer le sommet révélé ou de l'apparier à l'un des voisins disponibles. Dans ce modèle, nous montrons comment arrondir un algorithme en ligne de couplage fractionnaire déjà connu [86] pour obtenir un algorithme en ligne de couplage entier avec un ratio compétitif de $1/2 + c$ pour une constante $c > 0$.

Mots clefs : Couplage, Stochastique, Requête-engagement, Prix-de-l'information, Semi-streaming, Ordre-aléatoire, Multi-passe, Traitement-massivement-parallèle, En-ligne, Arrivée-de-sommet, Arrivée-d'arête.

# Contents

# List of Figures

# 1 Introduction

The maximum matching problem is a classic combinatorial optimization problem with a rich history. Given an undirected graph $G$ whose set of edges is $E$, a *matching* in graph $G$ is a subset of edges $M \subseteq E$ such that each vertex has at most one incident edge in $M$. The maximum matching problem asks to find a matching with the largest possible size. It has found numerous practical applications in situations such as associating goods with buyers, organ donors with recipients, jobs with machines, vacancies with applicants, etc. At the same time, maximum matching has been the central problem studied in several monumental works in theoretical computer science, including the one that defined algorithm efficiency as the polynomial-time computability [32] and the one that pioneered the primal-dual framework [69].

The maximum matching problem has several variants depending on the type of the input graph. When the edges of the input graph are unweighted, the problem is referred to as the *maximum cardinality matching* problem (MCM). For graphs where the edges of the input graph have non-negative weights, a maximum matching means a matching such that the sum of the weights of its edges is maximized. In this setting, the problem is called the *maximum weighted matching* problem (MWM). If the input graph is restricted to be bipartite, we call the former task the *maximum cardinality bipartite matching* problem (MCBM) and the latter the *maximum weighted bipartite matching* problem (MWBM).

In the classical computational setting, the input graph is completely known in advance, and we consider the problem of computing a maximum matching in the Random Access Machine (RAM) model. In this case, both the MCM and the MWM problems can be solved exactly in polynomial time [32, 52, 69, 79]. However, many practical scenarios for matching impose constraints that are not captured in this setting. Often, the complete information about the input is *not* known in advance, forcing the matching algorithms to make decisions with only partial information. Sometimes, the algorithms have to incur additional costs to have a better knowledge of the input.

In this thesis, we study the maximum matching problem in several computational settings where the algorithms lack the complete view of the input graph. We first consider situations where the input is a bipartite stochastic graph, and an algorithm incurs implicit or explicit costs for knowing parts of the realization. Next, we consider graphs that are too large to keep in the memory of a single computational unit. In this case, the graph is either provided as a stream of edges or is split across multiple computing units. An algorithm can go over the edge stream one or more times before computing the output in the former setting. In the latter setting, an algorithm runs in a distributed fashion where the different instances can communicate with each other before computing the output. Finally, we focus on more general online settings in which either the vertices or the edges of not necessarily bipartite graphs arrive online.

## 1.1 Our Contributions

In Part I of this thesis, we consider the MWBM problem in stochastic graphs. Namely, we treat two models, the *query-commit (QC)* model and the *price-of-information (PoI)* model. Part I is based on a joint work with Sagar Kale and Ola Svensson that was published in SODA 2019 [40].

In the query-commit model, each edge $e$ in the input graph exists independently with probability $p_e$. An algorithm in this setting receives the edge-existence probabilities as input and is characterized by a sequence of edge-existence queries. Each query has an implicit cost — if the queried edge exists, the algorithm *must* include that edge in its output matching. In this model, the approximation ratio is the worst-case (i.e., minimum) ratio between the expected matching size output by the algorithm and the expected maximum matching size. For unweighted graphs, the RANKING algorithm of Karp, Vazirani, and Vazirani [65] readily gives an approximation ratio of $(1 - 1/e)$ (see Chapter 3 for details). However, for weighted graphs, the known best algorithm was greedy, which yields a $1/2$-approximation. In this thesis, we present a $(1 - 1/e)$-approximation algorithm for this problem.

There exists a $(1 - 1/e)$-approximation algorithm for MWBM in the query-commit model.

In the price-of-information model introduced by Singla [84], the weight $w(e)$ of each edge $e$ is an independent random variable. An algorithm in this model first has to query a subset of the input graph's edges to know the realization of their weights where querying $w(e)$ incurs a cost of $\pi_e$. It then has to output a matching which must be a subset of the queried edges. The query costs and the distribution of the edge weights are known to the algorithm at the beginning. The goal is to maximize the expected *utility* where the utility is defined as the difference between the sum of weights in the output matching and the sum of costs of queried edges. The approximation ratio of such an algorithm is the minimum (over all input instances) ratio between the expected utility of the algorithm and the expected utility of an optimum algorithm in the same model. Prior to our work, the known best algorithm was the $1/2$-approximation greedy algorithm. In this thesis, we improve the approximation ratio to $(1 - 1/e)$.

There exists a $(1 - 1/e)$-approximation algorithm for maximizing the utility of weighted bipartite matching in the price-of-information model.

Part II of this thesis focuses on MWM in the semi-streaming and massively parallel computation (MPC) models. These models are motivated by the need for solving such problems at a large scale while having space limitations that prohibit storing the complete input on any single computing unit. An additional motivation for the MPC model is the increasing demand for efficient parallel algorithms when the number of computing units can scale up polynomially in the input size. Part II is based on joint work with Sagar Kale, Slobodan Mitrović, and Ola Svensson that was published in PODC 2019 [39].

In the semi-streaming model, the input graph is provided as a stream of edges, and the algorithm is only allowed $O(n \operatorname{poly} \log n)$ bits of space, where $n$ is the number of vertices. For unweighted graphs, the greedy algorithm guarantees to return a $1/2$-approximate maximum matching in this setting. It remains a major open problem to improve upon this factor when the order of the stream is adversarial, but for streams where edges arrive in a uniformly random order, better

algorithms are known [68]. For weighted graphs, a $(1/2 - \varepsilon)$-approximation algorithm exists for adversarial-order streams [44, 83], but no better algorithm was known for streams of edges arriving in uniformly random order. In this thesis, we show how to overcome this barrier of $1/2$ for the approximation ratio under *uniformly random order edge arrivals* by reducing the task of finding weighted augmenting paths with a few edges to that of finding unweighted augmenting paths of the same length.

For some absolute positive constant $c$, there is a $(1/2 + c)$-approximation algorithm for MWM in the semi-streaming model if the edges arrive in a uniformly random order.

In the MPC model, the input graph is stored in multiple computing units where the space of each unit is $O(n \operatorname{poly} \log n)$ bits which is sublinear in terms of the graph size for dense graphs with polynomially bounded edge-weights. Note that this allows a single computing unit to store any matching. The computation occurs in synchronous *rounds*: Each round consists of a computation phase where each machine updates the local state followed by a communication phase where each machine shares messages with the other machines. In this model, we measure the efficiency of an algorithm using *round complexity* which is the number of rounds the algorithm needs to compute the output. In the multi-pass semi-streaming model, the algorithm only has $O(n \operatorname{poly} \log n)$ bits of space as before, but it is allowed to go over the stream multiple times. The efficiency, in this case, is measured by the number of passes the algorithm needs to compute the output. In this thesis, we extend our techniques for the aforementioned semi-streaming model under uniformly random order edge streams to a general approach that reduces the task of finding weighted matching to that of finding short, unweighted augmenting paths. We then show how to efficiently implement this reduction in multi-pass semi-streaming and MPC models. This reduction yields algorithms for $(1 - \epsilon)$-approximate MWM in the respective computational models for any $\varepsilon > 0$ where the efficiency is degraded only by an $f(\varepsilon)$ factor for some function $f$ compared to the algorithms for MCM in the respective settings. Prior to our results, the known best results for $(1 - \varepsilon)$-approximate MWM used $\Omega(\log n)$ passes in the semi-streaming model [3] and $\Omega(\log n)$ rounds in the MPC model [4] (unless memory per machine is $\Theta(n^{1+c})$ for some constant $c$).

The task of $(1 - \varepsilon)$-approximate MWM can be reduced to that of finding short, unweighted augmenting paths. The reduction can be efficiently implemented in both the multi-pass semi-streaming model and the MPC model.

In Part III, we turn to online matching where we consider edge arrivals and vertex arrivals of general (not necessarily bipartite) unweighted graphs. The results in this part are based on a joint work with Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc that was published in FOCS 2019 [41].

Under one-sided vertex arrival in bipartite graphs, (better-than-$1/2$)-competitive algorithms were known for MCBM. In fact, the RANKING algorithm [65] is $(1 - 1/e)$-competitive for this case as reproven in several works [15, 30, 31, 36, 46]. However, under general vertex arrivals where an arriving vertex reveals only its neighbors among the already arrived vertices and has to be immediately and irrevocably matched to an existing neighbor or ignored, algorithms better than greedy were known only for *fractional* matchings [86]. (A fractional matching is an assignment of non-negative values to edges such that the sum of assigned values over edges incident to any given vertex is at most one.) This thesis shows how to round such a fractional matching in an

online fashion which yields the first algorithm that beats the greedy guarantee for MCM in the online vertex arrival setting.

> For some absolute positive constant $c > 0$, there exists a $(1/2+c)$-competitive online algorithm for MCM under general vertex arrivals.

As a complementary result, we also show that greedy is the optimal algorithm if we further generalize the model into edge arrivals even if the input graph is bipartite. Previously such an impossiblity result was only known for competitive ratios better than $2 - \sqrt{2} \approx 0.585$ due to Huang et al. [53].

> For any constant $c > 0$, no online algorithm can be $(1/2+c)$-competitive maximum cardinality bipartite matching under edge arrivals.

## 1.2 Outline of the Thesis

In Chapter 2, we introduced the notations and formally define the computational settings considered in this thesis.

Part I is devoted to our results for MWBM in stochastic graphs. We start with a brief overview of related work and our techniques in Chapter 3. Then in Chapter 4 and Chapter 5, we present our $(1 - 1/e)$-approximation algorithms for MWBM in query-commit and price-of-information models respectively.

In Part II, we present our results for MWM in streaming and MPC models. We again start with a brief discussion on related work and a general description of our techniques in Chapter 6. Then, in Chapter 7, we present our (better-than-1/2)-competitive algorithm for MWM in the random-order edge arrival setting. We then generalize the reduction from weighted augmentations to unweighted ones in Chapter 8 and show how to implement this reduction efficiently in multi-pass semi-streaming and MPC models.

We treat the subject of online matching in Part III. We begin with an overview of the related work and discuss our techniques in Chapter 9. Then, in Chapter 10, we show that no online algorithm for MCBM can be $(1/2 + c)$-competitive in the edge arrival setting for any constant $c > 0$. In Chapter 11, we wrap up our treatement of online matching with a $(1/2 + c)$-competitive algorithm for MCM under vertex arrivals in general graphs.

Finally, in Chapter 12, we conclude the thesis with a short discussion on related open problems and other variants of the maximum matching problem.

# 2 Preliminaries

This chapter covers the preliminaries for the subsequent parts of the thesis. We first introduce commonly used notations in Section 2.1. Then in Section 2.2, we formally define the maximum matching problem in the computational settings considered in the remainder of the thesis.

## 2.1 Notation

Let $\mathbb{R}^+$ denote the set of non-negative real numbers. We denote an input graph for maximum matching by $G = (V, E)$ where $V$ is the set of vertices and $E$ is the set of edges. In this thesis, we consider undirected graphs. We use $n := |V|$ to denote the number of vertices of a graph and $m := |E|$ to denote the number of edges. When $G$ is bipartite, we write $G = (A \cup B, E)$ where $A$ and $B$ denotes the two sides of the vertices. Unless otherwise noted, we assume $|A| = |B|$. In this case, we use $n := |A|$ to denote the number of vertices on one side of the graph. When $G$ is weighted, we additionally associate a non-negative weight function $w : E \to \mathbb{R}^+$ with the graph, and we write $G = (V, E, w)$. In a weighted graph, for each edge $e \in E$, $w(e)$ denotes the weight of $e$. For a vertex $v \in V$, we denote the set of edges incident to $v$ by $\delta(v)$ and the set of neighbors of $v$, i.e., vertices $u$ such that $\{u, v\} \in E$, by $N(v)$.

We usually denote matchings of a graph by $M$ or one of its subscripted/superscripted versions. For example, we may use $M_1$ and $M_2$ to denote two matchings of a given graph. We use $M^*$ to denote the maximum matching (cardinality or weighted version depending on the context) of a graph. In weighted settings, for a subset $S \subseteq E$ of edges, we use $w(S) := \sum_{e \in S} w(e)$ to denote the sum of weights of edges in $S$. When $M$ is a matching, we call $w(M)$ the weight of matching $M$.

We use OPT to denote the size (cardinality or weight) of the maximum matching in a given graph. In the randomized settings, we use OPT to denote the *expected* size of the maximum matching (or, in the case of the price-of-information model, the expected utility of an optimum matching algorithm).

## 2.2 Matching in Different Computational Settings

We now formally define the maximum matching problem in different computational settings. The models we consider are the query-commit model, the price-of-information model, the semi-streaming model, the massively parallel computation model, and the online model.

### 2.2.1 Query-Commit Model

In the query-commit model, the input graph is a weighted bipartite graph $G = (A \cup B, E, w)$, where each edge exists only with probability $p_e$ independently from other edges.

An algorithm for maximum matching in this model receives $G$ and $\mathbf{p} = (p_e)_{e \in E}$ as input and (adaptively) generates, in polynomial time, a sequence of edges that are to be queried. If an edge $e$ is queried and found to exist, the algorithm must include $e$ in its output matching. Due to this constraint, the algorithm must *not* query any edge $e$ with $p_e > 0$ that is incident to edges in the already computed part of the output matching.

For a query-commit algorithm $\mathcal{A}$ and an input instance $I = (G, \mathbf{p})$, let $\mathcal{A}(I)$ denote the expected weight of the matching output by $\mathcal{A}$ on $I$. Here, the expectation is taken over the edge-existence probabilities and any internal randomness of $\mathcal{A}$. Let $\mathrm{OPT}_I$ denote the expected weight of the maximum weighted matching in $G$ with respect to the edge-existence probabilities. We say that $\mathcal{A}$ is an $\alpha$-approximation algorithm for MWBM in the query-commit model if $\mathcal{A}(I)/\mathrm{OPT}_I \geq \alpha$ for all input instances $I$.

### 2.2.2 Price-of-Information Model

In the price-of-information model, the input graph is again a weighted bipartite graph $G = (A \cup B, E, w)$. However, unlike the query-commit model, each $w(e)$ is an independent random variable drawn from some distribution $\mathcal{D}_e$. Additionally, each edge $e \in E$ has a cost $\pi_e$ that would be incurred if an algorithm ever queries the realization of $w(e)$.

An algorithm for maximum matching in this model receives $G$, $\mathcal{D} = (\mathcal{D}_e)_{e \in E}$, and $\pi = (\pi_e)_{e \in E}$ as input. The algorithm then has to (adaptively) query a subset $Q \subseteq E$ of edges to know their realized weights and subsequently compute a matching $M \subseteq Q$ as the output. The expected utility of such an algorithm $\mathcal{A}$ on some input instance $I = (G, \mathcal{D}, \pi)$ is defined as $\mathcal{A}(I) := \mathbb{E}[\sum_{e \in M} w(e) - \sum_{e \in Q} \pi_e]$, where the expectation is taken over the randomness of $\mathcal{D}$ and any internal randomness of $\mathcal{A}$. The algorithm aims to maximize $\mathcal{A}(I)$, and it must run in polynomial time.

Let $\mathcal{A}_{\mathrm{OPT}}$ denote an optimal algorithm for MWBM in this model. We say that an algorithm $\mathcal{A}$ is an $\alpha$-approximation algorithm for MWBM in the price-of-information model if $\mathcal{A}(I)/\mathcal{A}_{\mathrm{OPT}}(I) \geq \alpha$ for all input instances $I$. This is a natural way to define the approximation ratio in this model due to the presence of explicit query costs. Note that this measure of approximation quality is quite different from the one used in the query-commit model, where the approximation quality is measured with respect to the maximum matching size.

### 2.2.3 Semi-streaming Model

The semi-streaming model for graph problems was introduced by Feigenbaum et al. [37]. In this model, an algorithm is restricted to use at most $O(n \operatorname{poly} \log n)$ bits of space where $n$ is the number of vertices of the input graph. Note that this is strictly sublinear in the size of the input as a graph with $n$ vertices can have $\Theta(n^2)$ edges. On the other hand, $\Omega(n \log n)$ bits might be necessary to store a valid matching.

We consider the MWM problem in the semi-streaming model where the input is a weighted graph $G = (V, E, w)$. We assume that the edge weights are $O(\text{poly}(n))$. In this model, the edges (and their weights) are fed to the algorithm in some arbitrary order.

For a semi-streaming algorithm $\mathcal{A}$, let $\mathcal{A}(G)$ denote the minimum weight (expected weight in the case of randomized algorithms) of a matching output by $\mathcal{A}$ on an input graph $G$ where the minimum is taken over all possible orders of edge arrivals. Let $\text{OPT}_G$ denote the maximum weight of a matching in $G$. We say that $\mathcal{A}$ is an $\alpha$-approximation algorithm if $\mathcal{A}(G)/\text{OPT}_G \geq \alpha$ for all input graphs $G$.

In the multi-pass semi-streaming setting, an algorithm is allowed to go over the edge stream multiple times. In this setting, the order of the edges need not be the same over all the passes. We measure the efficiency of a multi-pass algorithm using the number of passes it requires to produce the output.

In the random-order edge arrival setting, the edges are fed into a semi-streaming algorithm in uniformly random order. In this case, let $\mathcal{A}^{\text{rand}}(G)$ denote the expected weight of a matching output by an algorithm $\mathcal{A}$ where the expectation is taken over the randomness of the order of the edges and the internal randomness of the algorithm. We say that $\mathcal{A}$ is an $\alpha$-approximation algorithm if $\mathcal{A}^{\text{rand}}(G)/\text{OPT}_G \geq \alpha$ for all input graphs $G$.

### 2.2.4 Massively Parallel Computation (MPC) Model

The MPC model was introduced by Karloff et al. [62], and it has been refined in later work [6, 12, 47]. In this model, the computation happens in synchronous rounds on $\Gamma$ machines, each having $S$ bits of memory. At the beginning of computation, the input is partitioned across the machines such that each machine receives at most $S$ bits. During a round, each machine processes the received data locally. After the local computation on all the machines is over, each machine outputs messages of total size at most $S$. Each machine can send messages to any other machine, as long as at most $S$ bits are sent and received by each machine. The messages received at the end of one round will be used to guide the local computations in the next round.

We consider the MWM problem in the MPC model where the input is a weighted graph $G = (V, E, w)$. As before, we assume that the edge weights are bounded by some polynomial in $n$. A natural assumption is that $S \cdot \Gamma \in \Omega(|G|)$ where $|G|$ denotes the size of the graph in bits. I.e., it is possible to partition the entire graph across the machines. We do not assume any structure on how the graph is partitioned across the machines before the computation begins. In this thesis, we assume that $S \cdot \Gamma \in O(|G| \operatorname{poly}\log |G|)$. Furthermore, we consider the regime in which memory-per-machine is nearly linear in the number of vertices in the graph, i.e., $S \in \Theta(n \operatorname{poly}\log n)$ where $n$ is the number of vertices in the graph. As noted before, this allows a single machine to store a maximum matching of the input graph.

The approximation ratio is defined similarly to that in the semi-streaming setting. Namely, an MPC algorithm $\mathcal{A}$ is an $\alpha$-approximation algorithm if, for all possible graphs, the expected size of the matching output by $\mathcal{A}$ is at least $\alpha$ times the size of the maximum matching in the graph.

### 2.2.5 Online Model

In the online model, we consider the problem of MCM where the input is an unweighted graph $G = (V, E)$ that is not necessarily bipartite.

In the online *vertex* arrival setting, the vertices in $V$ are revealed to the algorithm in some arbitrary order $v_1, \ldots, v_n$. When vertex $v_i$ arrives, it reveals all its neighbors among the already arrived vertices $v_1, \ldots, v_{i-1}$. Upon the arrival of $v_i$, an online algorithm must either discard or irrevocably match it to some so-far-unmatched (i.e., previously discarded) vertex among its revealed neighbors before seeing the remaining vertices.

In the online *edge* arrival setting, the edges in $E$ arrive in some arbitrary order $e_1, \ldots, e_m$. Upon the arrival of an edge $e_i$, an online algorithm must either permanently discard it or immediately and irrevocably add it to the output matching (provided that the two incident vertices are unmatched at the time).

In both settings, the goal is to maximize the competitive ratio. For an online algorithm $\mathcal{A}$ and a graph $G$, let $\mathcal{A}(G)$ denote the minimum expected cardinality of the output matching where the minimum is taken over all possible arrival orders. Let $\text{OPT}_G$ denote the cardinality of the maximum matching in $G$. We say that an online algorithm $\mathcal{A}$ is $\alpha$-competitive if $\mathcal{A}(G)/\text{OPT}_G \geq \alpha$ for all possible input graphs $G$.

# Part I

# Stochastic Bipartite Matching

## Techniques Beyond Greedy for Weighted Graphs

# 3 Overview of Stochastic Matching

This part of the thesis considers the MWBM problem under two stochastic computational models: the query-commit (QC) model and the price-of-information (PoI) model. As formally defined in Chapter 2, these settings model the situations where the input graph is random, but the algorithms can know specific parts of the input by incurring some costs.

For MCBM in the query-commit model, we can query edges in the order given by the classical algorithm of Karp, Vazirani, and Vazirani [64] to get a $(1 - 1/e)$ approximation. Namely, we first fix a uniformly random permutation for the vertices of one side of the graph. Then, we go over the vertices on the other side in any order, and for each vertex, we query incident edges in the order given by the fixed permutation of the other side. However, for the more general problem of MWBM in the query-commit model, it is not clear how to use such a strategy. In fact, prior to our work, the known best algorithm for the weighted setting was the basic greedy that sorts the edges by weight $w_e$ and then queries in that order to get a $1/2$-approximate matching.

Similarly, in the price-of-information setting, Singla gave a $1/2$-approximation algorithm based on the greedy approach.

In this and the subsequent chapters, we show how to beat the greedy algorithm using new techniques and give clean algorithms that are $(1 - 1/e)$-approximate (improving from $1/2$) for the MWBM problem in the query-commit and price-of-information models.

A key component of our approach is to upper bound the optimum achieved by any strategy using a linear program (LP). We then exploit the structural properties of this LP in the design of our algorithms. We now give a high-level description of these techniques.

## 3.1 Techniques

We first focus on the query-commit model and later extend these techniques to the price-of-information model.

As described in Section 2.2, for MWBM in the query-commit model, the input $I$ consists of a weighted bipartite graph $G = (A \cup B, E, w)$ and its edge-existence probabilities $\mathbf{p} = (p_e)_{e \in E}$.

Our goal is to design a polynomial-time algorithm that, given an input instance $I$, computes a sequence of edges to query such that after the last query, we end up with a matching of large weight. First, to get a handle on the expected value of an optimum strategy ($\mathcal{A}_{\mathrm{OPT}}$), consider

the LP below. Recall from Section 2.1 that $\delta(u)$ denotes the set of edges incident to a vertex $u$.

$$\begin{aligned} \text{Maximize } & \sum_{e \in E} x_e \cdot w(e), \\ \text{subject to } & \sum_{e \in F} x_e \leq \Pr[\text{an edge in } F \text{ exists}], \text{ for all } u \in A \cup B, \text{ for all } F \subseteq \delta(u), \\ & x_e \geq 0, \text{ for all } e \in E. \end{aligned}$$

Let $x'_e$ be the probability that the output matching $M^*$ computed by $\mathcal{A}_{\mathrm{OPT}}$ contains $e$. Since $M^*$ can have at most one edge incident to $u$, for any $F \subseteq \delta(u)$, the events in $\{M^* \text{ contains } e : e \in F\}$ are disjoint. Hence, $\sum_{e \in F} x'_e$ is the probability that $M^*$ contains an edge in $F$, which must be at most the probability that at least one edge in $F$ exists. Therefore, $(x'_e)_{e \in E}$ has to satisfy the above LP. We can solve this LP in polynomial time using a submodular-function-minimization algorithm as a separation oracle for the ellipsoid algorithm as we see in Chapter 4. Let $\mathbf{x}^* = (x^*_e)_{e \in E}$ be the solution of the LP, and let $\mathbf{x}^*_u = (x^*_e)_{e \in \delta(u)}$ be $\mathbf{x}^*$ restricted to edges in $\delta(u)$. We can write $\mathbf{x}^*_u$ as a convex combination of extreme points of the polytope

$$\left\{ x \in \mathbb{R}_+^{\delta(u)} : \sum_{e \in F} x_e \leq \Pr[\text{an edge in } F \text{ exists}] \ \forall F \subseteq \delta(u) \right\}. \tag{3.1}$$

A key part of our approach is the nice structural properties of the extreme points of this polytope. Let $S_1, \ldots, S_t$ be the subsets of edges (in the increasing order of cardinality) that correspond to tight constraints of an extreme point $\mathbf{y} = (y_e)_{e \in \delta(u)}$ of the polytope. Then using the submodularity of the right hand side of the constraints, we can show that these subsets form a chain $\emptyset \subsetneq S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_t$ such that each subset in the chain has exactly one edge that is not present in the preceding subset. (Formally, we need strict submodularity for this to hold, which can be achieved with small perturbations to the edge existence probabilities.) Hence such a chain defines an ordering of the edges in $S_t$.

Now, for an extreme point $\mathbf{y}$, if we query the edges in the order given by its chain, it can be proven that we commit to an edge $e \in \delta(u)$ with probability $y_e$. Since $\mathbf{x}^*_u$ can be written as a convex combination of such extreme points, if we select an extreme point with probability equal to its coefficient in the convex combination and query the edges in the order given by its chain, we commit to an edge $e \in \delta(u)$ with probability $x^*_e$. Figure 3.1 explains this with an example.

However, if we implement the above process independently for each vertex in $A$, we may end up with collisions on $B$. Namely, we may match two or more vertices in $A$ to the same vertex in $B$. To avoid such collisions and ensure that we always produce a valid matching, we perform contention resolution. Suppose that we consider the vertices of $A$ in uniformly random order. Consider a fixed vertex $v \in B$. From $v$'s perspective, we can view the outcomes of the aforementioned procedure as follows: Each of $v$'s neighbors $u \in A$ arrives independently with probability $x^*_{uv}$ and weight $w_{uv}$ in a uniformly random order, and we have to pick one neighbor so that its expected utility is close to $\sum_{uv} x^*_{uv} \cdot w_{uv}$. Then, the above setting is similar to the *prophet secretary* problem.

In the prophet secretary problem, we have one secretary position to be filled, and a set of secretaries arrive in random order. Each secretary has some random skill level which is revealed only upon arrival, and after observing the skill level, we have to either immediately and irrevocably

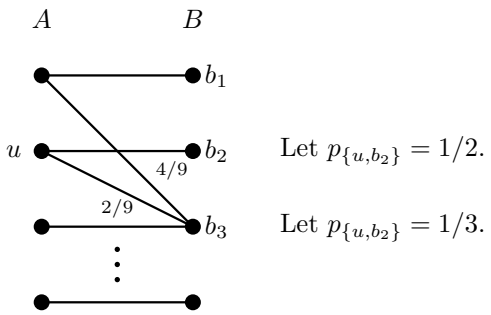



Figure 3.1 – An example bipartite graph for matching in the query-commit model. In this graph, we have $\delta(u) = \{\{u, b_2\}, \{u, b_3\}\}$. We write $\mathbf{x}^*_u = (4/9, 2/9)$ as a convex combination of extreme points of the polytope described in Equation (3.1): $\mathbf{x}^*_u = (2/3)(1/2, 1/6)+(1/3)(1/3, 1/3)$. For $(1/2, 1/6)$, the inequalities corresponding to sets $\{\{u, b_2\}\}$ and $\{\{u, b_2\}, \{u, b_3\}\}$ are tight, and for $(1/3, 1/3)$, the inequalities corresponding to sets $\{\{u, b_3\}\}$ and $\{\{u, b_2\}, \{u, b_3\}\}$ are tight. Observe that these form a chain. We note that it is possible that an inequality corresponding to a nonnegativity constraint is tight. Now, say we query the edges in the order given by the chain. For $(1/2, 1/6)$, we first query $\{u, b_2\}$ then $\{u, b_3\}$, so we select $\{u, b_2\}$ with probability $p_{\{u,b_2\}} = 1/2$ and $\{u, b_3\}$ with probability $(1 - p_{\{u,b_2\}})p_{\{u,b_3\}} = 1/6$, which does indeed correspond to the extreme point $(1/2, 1/6)$.

fill the position by hiring the secretary or continue to observe the next secretary in line. In their recent work, Ehsani et al. [34] gave a $(1 - 1/e)$-competitive algorithm for the prophet secretary problem based on dynamic thresholds that depend on the arrival times. We adapt their algorithm for resolving collisions using the aforementioned point of view together with a different set of dynamic thresholds.

In Chapter 4, we formally describe all these ideas and present our $(1 - 1/e)$-approximate algorithm for the MWBM problem in the query-commit model.

To derive an algorithm for the MWBM problem in the price-of-information model, we first generalize the query-commit algorithm to graphs with random edge weights. In this case, a query asks if the weight of an edge $e$ at least $c$, and if so, the algorithm has to add it to its output matching. To this end, we assume an edge $e$ has as many copies as the values its weight can take. But now, instead of being independent, the existence of these copies is correlated. To deal with these correlations, we write a more general LP with constraints corresponding to sets from a lattice family. This general LP can still be efficiently solved as we can solve submodular-function-minimization over a lattice family in polynomial time [48] which yields a polynomial-time separation oracle. As with the usual query-commit setting described earlier, the extreme points of the polytope defined by constraints for a vertex in $A$ again correspond to a chain with similar properties.

Once we have the generalized query-commit algorithm for input instances with random edge-weights, we can obtain a price-of-information algorithm by a clean reduction [84]: Namely, for each edge $e$ whose weight is a random variable $X_e$ and querying cost is $\pi_e$, let $\tau_e$ be the solution to the equation $\mathbb{E}[\max\{(X_e - \tau_e), 0\}] = \pi_e$, and let $Y_e = \min(X_e, \tau_e)$ be a new random variable. The idea behind $Y_e$ values is that whatever "excess" value we get over $Y_e$ can be used to pay the price $\pi_e$. We run the query-commit algorithm with weights $Y_e$, and whenever the algorithm queries any copy of an edge $e$, we probe $e$'s weight. However, we only pay $\pi_e$ the first time we probe that edge.

In Chapter 5, we formalize the generalization of our MWBM algorithm for the query-commit model to input graphs with random edge weights and subsequently present our $(1 - 1/e)$-approximate algorithm for the MWBM problem in the price-of-information model.

## 3.2 Related Work

As mentioned earlier, the algorithm of Karp et al. gives a $(1-1/e)$-approximation in the unweighted query-commit model for bipartite graphs. Note that $1 - 1/e \simeq 0.632$. Costello et al. [25] give a 0.573-approximation for general graphs and show that no algorithm can give an approximation better than 0.898 compared to the optimal offline algorithm (that knows all the outcomes before selecting the matching).

Motivated by applications in kidney exchange and online dating, Chen et al. [21] consider the matching problem in the query-commit model with the further constraint that for each vertex $v$, the algorithm can query at most $t_v$ edges incident to it ($t_v$ is a part of the input) and give a $(1/4)$-approximation algorithm. Bansal et al. [10] improve it to $(1/3)$ for bipartite graphs and to $(1/3.46)$ for general graphs, and also give a $(1/4)$-approximation in the weighted query-commit (for general graphs); both of these ratios for the unweighted case are further improved by Adamczyk et al. [1], who give a $(1/3.709)$-approximation for general graphs. Baveja et al. [11] improve this to $1/3.224$. We mention that this setting is more general than the setting we consider because $t_v = \deg(v)$ for us, i.e., we do not restrict on the number of edges incident to a vertex that we can query.

Molinaro and Ravi [80] give an optimal algorithm for a very special class of sparse graphs in the unweighted query-commit setting.

Blum et al. [16] consider the maximum matching problem, where, in the input graph, an edge $e$ exists with probability $p_e$, the algorithm can query the existence of an edge and does not have to commit, but needs to minimize the number of queries subject to outputting a good approximation. This model is considered in several follow-up works [7, 8, 13, 75] which ultimately showed that, for any $\varepsilon > 0$, there exist a $(1/2 - \varepsilon)$-approximate non-adaptive algorithm and a $(1 - \varepsilon)$-approximate adaptive algorithm that query $O_\varepsilon(1)$ edges per vertex.

Feldman et al. [38] consider an online variant of stochastic matching where the algorithm gets as input a bipartite graph $G = (A \cup B, E)$, and a distribution $\mathcal{D}$ over $B$, and $n$ elements are drawn i.i.d from $B$ according to $\mathcal{D}$ (so there may be repetitions) that the algorithm accesses online. When a copy $v \sim \mathcal{D}$ arrives online, we have to match it to an unmatched vertex $u$ in $A$ such that $\{u, v\} \in E$. Note that here the existence of an edge is not random in itself but that of a vertex is. Again, Karp et al.'s algorithm gives a $(1 - 1/e)$-approximation, and Feldman et al. give a 0.67-approximation. The subsequent work on this problem include [1, 10, 51, 58, 76], and the known best approximation ratio is 0.7299 due to Brubach et al. [18].

We also note the works of Dean et al. [28, 29], which considered stochastic problems where the cost of an input unit is only known as a probability distribution that is instantiated after the algorithm commits to including the item in the solution. Charikar et al. [20] and Katriel et al. [66] consider two-stage optimization problems, where the first stage is stochastic with a lower cost for decisions. In the second stage, with an increase in the decision cost, the actual input is known.

# 4 Weighted Matching in the Query-Commit Model

In this chapter, we present a $(1 - 1/e)$-approximation algorithm for the MWBM problem in the query-commit model.

We denote an input instance by $I = (G, \mathbf{p})$ where $G = (A \cup B, E, w)$ is a weighted bipartite graph with $|A| = |B| = n$ and $\mathbf{p} = (p_e)_{e \in E}$. Each edge $e \in E$ has weight $w(e)$ and exists independently with probability[1] $p_e$. Given such an input instance $I$, a query-commit algorithm for MWBM (adaptively) queries a sequence of edges $Q = (e_{q_1}, \ldots, e_{q_m})$ and outputs a valid matching $M \subseteq Q$. Since a query-commit algorithm is *committed* to add any queried edge that exists, its output is simply all the edges of $Q$ that exist.

For an algorithm $\mathcal{A}$ for MWBM in the query-commit model and an input instance $I$, recall that $\mathcal{A}(I)$ denote the expected weight of its output matching where the expectation is over the randomness of the existence of edges and any internal randomness of the algorithm. Also recall that we use $\mathrm{OPT}_I$ to denote the expected maximum weight of a matching in $G$. We give a query-commit algorithm Approx-QC such that, for any input instance $I$, the expected weight of the output matching, Approx-QC$(I)$, is at least $(1 - 1/e)\,\mathrm{OPT}_I$.

As described in Chapter 3, our approach consists of the following stages: First, in Section 4.1, we solve a linear program to bound the optimal expected query-commit utility $\mathrm{OPT}_I$. Then, in Section 4.2, we use the structural properties of our LP polytope to define a distribution over the permutations of edges and use this distribution to set the query order in our algorithm. There we show that we in fact match the performance of an optimal algorithm if we disregard the collisions on one side of the graph. Finally, in Section 4.3, we adapt ideas from the work of Ehsani et al. [34] on prophet secretary problem to resolve such collisions and present our $(1 - 1/e)$-competitive algorithm for MWBM in the query-commit model.

## 4.1 Upper-bounding the Optimal Expected Utility

Fix an instance $I = (G, \mathbf{p})$ where $G = (A \cup B, E, w)$ and $\mathbf{p} = (p_e)_{e \in E}$. For each vertex $u \in A \cup B$, recall that $\delta(u)$ denotes the set of edges incident to $u$. For a subset of edges $F \subseteq E$, let $f(F)$ be the probability that at least one edge in $F$ exists. Namely, $f(F) = 1 - \prod_{e \in F}(1 - p_e)$, because each edge $e$ exists independently with probability $p_e$.

[1] Alternatively, one may think of the weight of an edge $e$ as an independent random variable that takes value $w(e)$ with probability $p_e$ and value zero with probability $1 - p_e$. We consider a more general distribution of edge weights in Chapter 5.

Fix any query-commit algorithm $\mathcal{A}$ for the MWBM problem. For each edge $e \in E$, let $x_e$ be the probability that $\mathcal{A}$ includes $e$ in its output. Consider a vertex $u \in A \cup B$ and a subset $F \subseteq \delta(u)$. Since $\mathcal{A}$ outputs a valid matching, the events that edge $e$ being added to the output of $\mathcal{A}$ for each $e \in F$ are disjoint, and hence the probability that $\mathcal{A}$ adds one of the edges in $F$ to its output is $\sum_{e \in F} x_e$. But, for an edge to be added to the output, it must exist in the first place, and thus it must be the case that $\sum_{e \in F} x_e \leq f(F)$ (because $f(F)$ is the probability that at least one edge in $F$ exists). Therefore, $\mathbf{x} = (x_e)_{e \in E}$ is a feasible solution to the following linear program, which we call $\mathrm{LP}_{\mathrm{QC}}$.

$$\begin{aligned} \text{Maximize } & \sum_{e \in E} x_e \cdot w(e), \\ \text{subject to } & \sum_{e \in F} x_e \leq f(F)\,, \text{ for all } u \in A \cup B, \text{ for all } F \subseteq \delta(u)\,, \\ & x_e \geq 0\,, \text{ for all } e \in E\,. \end{aligned}$$

So, the expected weight of the output matching, $\mathcal{A}(I)$, is at most the value of $\mathrm{LP}_{\mathrm{QC}}$. In fact, the same reasoning is valid even if $\mathcal{A}$ knows all the random outcomes in advance, in which case it can directly pick the edges in the maximum weight matching of $G$. Hence, we have Lemma 4.1 below.

**Lemma 4.1.** *The expected weight of a maximum weighted matching in $I$, i.e., $\mathrm{OPT}_I$, is upper bounded by the value of $\mathrm{LP}_{\mathrm{QC}}$.*

## 4.2 Structure of the LP

Although $\mathrm{LP}_{\mathrm{QC}}$ has exponentially many constraints, we can solve it in polynomial time.

**Lemma 4.2.** *The linear program $\mathrm{LP}_{\mathrm{QC}}$ is polynomial-time solvable.*

*Proof.* Observe that for a fixed vertex $u \in A \cup B$, the constraints $\sum_{e \in F} x_e \leq f(F)$ for all $F \subseteq \delta(u)$, can be re-written as $0 \leq g_u(F)$ for all $F \subseteq \delta(u)$, where $g_u(F) = f(F) - \sum_{e \in F} x_e$ is a submodular function (notice that $f$ is submodular because it is a coverage function while $\sum_{e \in F} x_e$ is clearly modular). Thus we can minimize $g_u$ over all subsets of $\delta(u)$ for all $u \in A \cup B$ in polynomial time using $O(n)$ submodular minimizations to find a violating constraint. If none of the minimizations gives a negative value and if $x_e \geq 0$ for all $e \in E$, then the solution is feasible. Thus we can solve $\mathrm{LP}_{\mathrm{QC}}$ in polynomial-time using the ellipsoid method. □

For the rest of this section, we assume that $0 < p_e < 1$ for all $e \in E$. We can safely ignore those edges $e \in E$ for which $p_e = 0$, and for those with $p_e = 1$, we can scale down the probabilities (at a small loss in the objective value) due to the following lemma.

**Lemma 4.3.** *Let $\tilde{p}_e = (1-\gamma)p_e$ for all $e \in E$, and for a subset $F \subseteq E$, let $\tilde{f}(F) = 1 - \prod_{e \in F}(1-\tilde{p}_e)$ be the probability that at least one edge in $F$ exists under the scaled down probabilities $\tilde{p}_e$. If we replace $f(F)$ in $\mathrm{LP}_{\mathrm{QC}}$ by $\tilde{f}(F)$, the value of the resulting LP is at least $(1-\gamma)$ times the value of $\mathrm{LP}_{\mathrm{QC}}$.*

*Proof.* Fix a set $F \subseteq E$ and label the edges in $F$ from 1 through $|F|$. For $i = 1, \ldots, |F|$, let $Q_i = 1 - \prod_{e=1}^{i}(1-p_e)$ and let $\tilde{Q}_i = 1 - \prod_{e=1}^{i}(1-\tilde{p}_e)$. Then, for $i > 1$ we have that

$Q_i = Q_{i-1} + p_i(1 - Q_{i-1})$, and similarly, $\tilde{Q}_i = \tilde{Q}_{i-1} + \tilde{p}_i(1 - \tilde{Q}_{i-1})$. By definition, we have $f(F) = Q_{|F|}$ and $\tilde{f}(F) = \tilde{Q}_{|F|}$. We now prove that $\tilde{Q}_i \geq (1-\gamma)Q_i$ for $i = 1, \ldots, |F|$ by induction.

For the base case, we have $\tilde{Q}_1 = (1-\gamma)Q_1$. Notice that, by the definition of $\tilde{p}_i$'s, we have $Q_i \geq \tilde{Q}_i$. Thus, for $i > 1$ we have that

$$\begin{aligned}
\tilde{Q}_i &= \tilde{Q}_{i-1} + \tilde{p}_i(1 - \tilde{Q}_{i-1}) \\
&\geq (1-\gamma)Q_{i-1} + (1-\gamma)p_i(1 - \tilde{Q}_{i-1}) && \text{(by inductive hypothesis)} \\
&\geq (1-\gamma)Q_{i-1} + (1-\gamma)p_i(1 - Q_{i-1}) && \text{(because } Q_{i-1} \geq \tilde{Q}_{i-1}\text{)} \\
&= (1-\gamma)(Q_{i-1} + p_i(1 - Q_{i-1})) = (1-\gamma)Q_i.
\end{aligned}$$

Thus if we scale down the polytope defined by the constraints of $\mathrm{LP}_{\mathrm{QC}}$ by a factor of $(1-\gamma)$, the resulting polytope is contained inside the polytope defined by $\tilde{f}(F)$ constraints. Moreover, all extreme points of both the polytopes have non-negative coordinates and the objective function has non-negative coefficients. Hence the claim of Lemma 4.3 follows. □

**Remark.** *The expected utility of our proposed algorithm is* $(1 - 1/e) \cdot \mathrm{OPT}^* \geq (1 - 1/e) \cdot \mathrm{OPT}$*, where* $\mathrm{OPT}^*$ *is the optimal LP value of* $\mathrm{LP}_{\mathrm{QC}}$*. Thus, in the cases where the assumption* $p_e < 1$ *for all* $e \in E$ *does not hold, we can scale the probabilities down by* $(1-\gamma)$*, and consequently the guarantee on expected utility will at least be* $(1-\gamma)(1 - 1/e) \cdot \mathrm{OPT}$ *due to Lemma 4.3. We can choose* $\gamma$ *to be arbitrarily small. To implement the scaling down operation, we can simply replace each query made by an algorithm with a function that only queries with probability* $(1-\gamma)$*.*

The assumption that $0 < p_e < 1$ for all $e \in E$ yields the following lemma on the function $f$.

**Lemma 4.4.** *Fix a vertex* $u \in A \cup B$*, and suppose that* $0 < p_e < 1$ *for all* $e \in \delta(u)$*. Then the function* $f$ *is strictly submodular and strictly increasing on subsets of* $\delta(u)$*. That is:*

1. *For all subsets* $A, B \subseteq \delta(u)$ *such that* $A \setminus B \neq \emptyset$ *and* $B \setminus A \neq \emptyset$*,* $f(A) + f(B) > f(A \cup B) + f(A \cap B)$*.*
2. *For all* $A \subsetneq B \subseteq \delta(u)$*,* $f(A) < f(B)$*.*

*Proof.* Let $g(F) = 1 - f(F) = \prod_{e \in F}(1 - p_e)$ (note that $g(\emptyset) = 1$). Notice that for $F_1, F_2 \subseteq F$ such that $F_1 \cap F_2 = \emptyset$ and $F_1 \cup F_2 = F$, it holds that $g(F) = g(F_1) \cdot g(F_2)$. Let $A, B \subseteq \delta(u)$ be two sets such that $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$. It is sufficient to show that $g(A) + g(B) < g(A \cup B) + g(A \cap B)$. We have

$$g(A) + g(B) = g(A \cap B) \left( \underbrace{g(A \setminus B)}_{a} + \underbrace{g(B \setminus A)}_{b} \right), \tag{4.1}$$

and

$$g(A \cup B) + g(A \cap B) = g(A \cap B) \left( \underbrace{g(A \setminus B) \cdot g(B \setminus A)}_{a \cdot b} + 1 \right). \tag{4.2}$$

Since $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$, and $0 < p_e < 1$ for all $e \in E$, we have $g(A \cap B) > 0$ and both $a, b < 1$. Thus $a + b < 1 + a \cdot b$, because $(1 - a)(1 - b) > 0$. This combined with Equations (4.1) and (4.2) yields Property 1.

Now consider $A \subsetneq \delta(u)$ and any edge $e \in \delta(u) \setminus A$. To prove Property 2, it is sufficient to show that $f(A \cup \{e\}) > f(A)$, or equivalently, $g(A \cup \{e\}) < g(A)$. This is straightforward since $g(A \cup \{e\})/g(A) = 1 - p_e < 1$, because $p_e > 0$. □

Let $\mathbf{x}^* = (x_e^*)_{e \in E}$ be an optimal solution to $\mathrm{LP_{QC}}$. Fix a vertex $a \in A$. Then, $\mathbf{x}_a = (x_e^*)_{e \in \delta(a)}$, which is $\mathbf{x}^*$ restricted only to those coordinates that correspond to edges in $\delta(a)$, satisfy the following constraints:

$$\sum_{e \in F} x_e \leq f(F)\,, \qquad \text{for all } F \subseteq \delta(a)\,, \tag{4.3}$$

$$x_e \geq 0\,, \qquad \text{for all } e \in \delta(a)\,.$$

Notice that these constraints are only a subset of the constraints of $\mathrm{LP_{QC}}$.

Let $P_a^{\mathrm{QC}}$ denote the polytope defined by the above constrains. The extreme points of $P_a^{\mathrm{QC}}$ have a nice structure that becomes crucial when designing a good probability distribution $\mathcal{D}_a^{\mathrm{QC}}$ over permutations of edges. Namely, for any extreme point, the sets for which Constraint (4.3) is tight form a chain. Moreover, each set in the chain has exactly one more element than its predecessor and this element is non-zero coordinate of the extreme point. Formally, we have Lemma 4.5 below.

**Lemma 4.5.** *Let $\mathbf{y} = (y_e)_{e \in \delta(a)}$ be an extreme point of $P_a^{\mathrm{QC}}$ and let $Y = \{e \in \delta(a) : y_e > 0\}$ be the set of edges that correspond to the non-zero coordinates of $\mathbf{y}$. Then there exist $|Y|$ subsets $S_1, \ldots, S_{|Y|}$ of $\delta(a)$ such that $S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_{|Y|}$ with the following properties:*

1. *Constraint (4.3) is tight for all $S_1, S_2, \ldots S_{|Y|}$. That is $\sum_{e \in S_i} y_e = f(S_i)$ for all $i = 1, \ldots, |Y|$.*
2. *For each $i = 1, \ldots, |Y|$, the set $S_i \setminus S_{i-1}$ contains exactly one element $e_i$, and $y_{e_i}$ is non-zero (i.e., $e_i \in Y$).*

*Proof.* It is clear that at least $|Y|$ constraints in (4.3) are tight. If $|Y| = 1$, the claim of the lemma is obviously true. Suppose that $|Y| > 1$. Now let $A, B \subseteq \delta(a)$ be two different sets for which Constraint (4.3) is tight. Then we have

$$f(A) + f(B) = \sum_{e \in A} y_e + \sum_{e \in B} y_e = \sum_{e \in A \cup B} y_e + \sum_{e \in A \cap B} y_e \leq f(A \cup B) + f(A \cap B),$$

where the last inequality follows because $\mathbf{y}$ satisfies Constraint (4.3).

Observe that, if $A \not\subseteq B$ and $B \not\subseteq A$, then by Lemma 4.4, $f(A) + f(B) > f(A \cup B) + f(A \cap B)$. Thus, it must be the case that either $A \subsetneq B$ or $B \subsetneq A$, and consequently there exist $|Y|$ sets $S_1, S_2, \ldots, S_{|Y|}$ such that $S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_{|Y|}$, for which Constraint (4.3) is tight.

For each $i = 1, 2, \ldots, |Y|$, we thus have that $\sum_{e \in S_i \setminus S_{i-1}} y_e = f(S_i) - f(S_{i-1}) > 0$, where the inequality is due to the strictly increasing property of $f$. This implies that each $S_i \setminus S_{i-1}$ must

contain at least one edge $e_i$ such that $y_{e_i} > 0$, and since there are only $|Y|$ non-zero coordinates in $\mathbf{y}$, each $S_i \setminus S_{i-1}$ must contain exactly one such $e_i$. Now, suppose that some $S_i \setminus S_{i-1}$ contains some $e'_i$ such that $y_{e'_i} = 0$. Then $f(S_i) = \sum_{e \in S_i} y_e = \sum_{e \in S_i \setminus \{e'_i\}} y_e \leq f(S_i \setminus \{e'_i\}) < f(S_i)$ yields a contradiction. Here, the first inequality is due to Constraint (4.3) whereas the last inequality is due the strictly increasing property of $f$. □

Now fix a vertex $a \in A$ and consider the simple query algorithm given in Algorithm 4.1, which outputs at most one edge adjacent to vertex $a$. In Algorithm 4.1, $\mathcal{D}_a^{\mathrm{QC}}$ is a distribution over the permutations of edges in some subsets of $\delta(a)$ that, by Lemma 4.6, can be found in polynomial time. We have the following lemma considering Algorithm 4.1.

**Algorithm 4.1:** Query algorithm for selecting an edge adjacent to a fixed vertex $a \in A$.

```
Draw a permutation σ from D_a^QC for some fixed vertex a ∈ A.
foreach edge e in the order of σ do
    Query edge e to check whether it exists.
    If edge e exists, output e and terminate.
```

**Lemma 4.6.** *Let $\mathbf{x}^*$ be an optimal solution to $\mathrm{LP}_{\mathrm{QC}}$ and let $\mathbf{x}^*_a = (x^*_e)_{e \in \delta(a)}$ be its restriction to the coordinates that corresponds to edges in $\delta(a)$. Then there exists a distribution $\mathcal{D}_a^{\mathrm{QC}}$ over the permutations of subsets of $\delta(a)$ with the following property: When the permutation $\sigma$ is drawn from $\mathcal{D}_a^{\mathrm{QC}}$ in Algorithm 4.1, the probability that the algorithm outputs the edge $e$ is $x^*_e$, hence the expected weight of the edge output by Algorithm 4.1 is $\sum_{e \in E_a} x^*_e w(e)$. Moreover, a permutation of edges from $\mathcal{D}_a^{\mathrm{QC}}$ can be sampled in polynomial time.*

*Proof.* Let $\mathbf{y} = (y_e)_{e \in \delta(a)}$ be an extreme point of $P_a^{\mathrm{QC}}$ and let $|Y|$ be set of non-zero coordinates of $\mathbf{y}$. Let $\emptyset = S_0 \subsetneq S_1 \subsetneq \cdots \subsetneq S_{|Y|}$ be the chain of sets (for which Constraint (4.3) is tight) guaranteed by Lemma 4.5 for the extreme point $\mathbf{y}$. We can efficiently find the chain by first setting $S_Y = Y$, and iteratively recovering $S_{i-1}$ from $S_i$ by trying all possible $S_i \setminus \{e\}$ for $e \in S_i$ to check whether Constraint (4.3) is tight. For each $i = 1, \ldots, |Y|$, let $e_i$ be the unique element in $S_i \setminus S_{i-1}$ and let $\sigma_y = (e_1, \ldots, e_{|Y|})$. Notice that $S_i = \{e_1, \ldots, e_i\}$, and hence $y_{e_i} = \sum_{e \in S_i} y_e - \sum_{e \in S_{i-1}} y_e = f(S_i) - f(S_{i-1})$. Thus if we select $\sigma_y$ as the permutation in Algorithm 4.1 and query according to that order, the probability that it outputs the edge $e_i$ is exactly

$$\Pr[\text{some edge in } S_i \text{ appears}] - \Pr[\text{some edge in } S_{i-1} \text{ appears}] = f(S_i) - f(S_{i-1}) = y_{e_i}.$$

Note that the point $\mathbf{x}^*_a$ is contained in polytope $P_a^{\mathrm{QC}}$. Thus, using the constructive version of Caratheodary's theorem, we can efficiently find a convex combination $\mathbf{x}^*_a = \sum_{i \in [k]} a_i \cdot \mathbf{y}^{(i)}$, where $a_i \geq 0$ for all $i \in [k]$, $\mathbf{y}^{(i)}$ is an extreme points of $P_a^{\mathrm{QC}}$ for all $i \in [k]$, $\sum_{i \in [k]} a_i = 1$, and $k = \mathrm{poly}(|E_a|)$. This is because we can optimize a linear function over $P_a^{\mathrm{QC}}$ in polynomial time using submodular minimization as a separation oracle, and for such polytopes, the constructive version of Caratheodary's theorem holds (See Theorem 6.5.11 of [49]).

Define the distribution $\mathcal{D}_a^{\mathrm{QC}}$ such that it gives permutation $\sigma_{\mathbf{y}^{(i)}}$ with probability $a_i$. If follows that, if we sample according to this distribution in Algorithm 4.1, then for any fixed edge $e$, the probability that the algorithm outputs the edge $e$ is $\sum_{i \in [k]} a_i \cdot y^{(i)}_e = x^*_e$. Consequently, the expected weight of the output of Algorithm 4.1 is $\sum_{e \in E_a} x^*_e \cdot w(e)$. □

## 4.3 Proposed Algorithm and Analysis

Suppose that we run Algorithm 4.1 for all vertices $a \in A$ and let $M'$ be the set of all output edges. Then we have

$$\mathbb{E}\left[\sum_{e \in M'} w(e)\right] = \sum_{a \in A} \sum_{e \in \delta(a)} x_e^* \cdot w(e) = \sum_{e \in E} x_e^* \cdot w(e) \geq \mathrm{OPT}_I\,.$$

Furthermore, $M'$ contains at most one adjacent edge per each vertex $a \in A$. However $M'$ may contain more than one adjacent edge for some vertices $b \in B$, and hence it may not be valid matching.

Now again suppose that we run Algorithm 4.1 as described above for all vertices $a \in A$ in some arbitrary order. Consider some fixed vertex $b \in B$. By Lemma 4.6, from the perspective of $b$, an edge $e \in \delta(b)$ appears with probability $x_e^*$ (when we say an edge $e = (a, b)$ appears, it means that Algorithm 4.1, when run on vertex $a$, outputs the edge $e$). Viewing the vertices in $A$ as buyers, we think of the appearance of an edge $e = (a, b)$ as a buyer $a$ making a take-it or leave-it offer of value $w_e$ for item $b$. Thus if we use a uniformly random order of vertices in $A$, picking an edge adjacent to the fixed vertex $b$ can be viewed as an instance of the prophet secretary problem.

The $(1 - 1/e)$-competitive algorithm algorithm given by Ehsani et al. [34] for the prophet secretary problem first sets a base price for the item. If some buyer comes at time $t \in [0, 1]$, and if the item is not already sold, then the algorithm sells the item to this buyer if the offered price is at least $(1 - e^{t-1})$ times the base price. Since the prophet secretary problem deals with a single item, the goal is to choose the buyer with highest offer, and hence they set base price of the item as the expected value of the maximum offer.

However, rather than picking the maximum weighted edge adjacent to each $b$, we want to maximize the total weight of the matching constructed. Thus, we set the base price $c_b$ for each $b$, not as the the expectation of the offline secretary problem, but as the expected weight of the edge adjacent to $b$ in some optimal offline maximum-weight bipartite matching. To be concrete, we set $c_b = \sum_{e \in \delta(b)} x_e^* \cdot w(e)$ (recall that we can think of $x_e^*$ as the probability that some fixed optimal algorithm for maximum weighted bipartite matching in query-commit model adds edge $e$ to its output).

We present the pseudo-code of our algorithm Approx-QC in Algorithm 4.2. We start by independently assigning each $a \in A$ a uniformly random arrival time $t_a \in [0, 1]$, and then for each vertex $a \in A$ in the order of the arrival time, we run a slightly modified version of the query algorithm given in Algorithm 4.1. For each $b \in B$, we pick an edge $e = (a, b)$ if it appears and if its weight exceeds the threshold $(1 - e^{t_a - 1}) \cdot c_b$. Since a query-commit algorithm is committed to adding any queried edge that exists, we query an edge $e$ only if $e \cap B$ is not already assigned to some other vertex $a \in A$ and its weight $w(e)$ exceeds the threshold. But we still need to make sure that, for a fixed $b$, edges $e \in \delta(b)$ appears (in the sense that if we run Algorithm 4.1, it outputs the edge $e$) with probability $x_e^*$. Hence we have the **else** clause of the conditional in Algorithm 4.2 that simulates the behavior of Algorithm 4.1 in the cases we decide not to actually query an edge.

We conclude this section with Theorem 4.7 which shows that our algorithm Approx-QC is $(1-1/e)$-approximate. The proof follows exactly the same lines (except for the definition of base price $c_b$) as

**Algorithm 4.2:** Outline of Approx-QC.

```
Solve LP_QC to get x* and find the permutation distributions D_a^QC for all a ∈ A.
For each vertex a ∈ A, select t_a ∈ [0,1] (arrival time) independently and uniformly at
  random.
For each vertex b ∈ B, set the base price c_b = Σ_{e∈E_b} x*_e · w(e).
Let M be an empty matching.
foreach vertex a ∈ A in the increasing order of t_a do
    Draw a permutation σ of edges from D_a^QC.
    foreach e = (a,b) in the order of σ do
        if w(e) ≥ (1 − e^{t_a − 1}) · c_b and b is not matched then
            Query edge e to check whether it exists.
            If it exists, add it to M and continue to next vertex in A.
        else
            Flip a coin that give Heads with probability p_e.
            If Heads, continue to next vertex in A.
return the matching M.
```

in Ehsani et al. [34] to show that the expected weight of the edge adjacent to a fixed vertex $b \in B$ in the output of Approx-QC is at least $(1 - 1/e) \cdot c_b$. Then by the linearity of expectation, the expected utility of Approx-QC is at least $(1 - 1/e) \cdot \sum_{b \in B} c_b = (1 - 1/e) \cdot \sum_{e \in E} x^*_e \cdot w(e) \geq \mathrm{OPT}_I$ (recall that $c_b = \sum_{e \in \delta(b)} x^*_e \cdot w(e)$).

**Theorem 4.7.** *The expected weight of the output matching of Approx-QC on an input instance $I$ is at least $(1 - 1/e) \cdot \mathrm{OPT}_I$.*

**Remark.** *For the sake of completeness, we reproduce the analysis of Ehsani et al. [34] in Chapter 5 for our more general algorithm in the price of information model (see Theorem 5.8).*

# 5 Weighted Matching in the Price-of-Information Model

In this section, we present a $(1 - 1/e)$-approximation algorithm for the MWBM in the price-of-information model introduced by Singla [84]. Our strategy is essentially the same as that used in Chapter 4 for the query-commit model except for a few enhancements.

Let $G = (A \cup B, E, w)$ be a bipartite graph where each edge $e \in E$ and recall that $w(e)$ values are random. For notational convenience, we use $X_e$ to denote the random variable $w(e)$. The distributions of $X_e$ can be different for different edges and are independent. We denote the joint distribution of the weights by $\mathcal{D}$. To find the realization of $X_e$ for an edge $e$ (i.e., the actual weight of the edge $e$), we have to *query* the edge $e$ at a cost of $\pi_e$. Consider an algorithm $\mathcal{A}$ that queries a subset $Q$ of edges $E$ and outputs a valid matching $M \subseteq Q$. We call such an algorithm a price-of-information algorithm for MWBM. Recall from Chapter 2 that we define the expected utility of such an algorithm $\mathcal{A}$ on an input instance $I = (G, \mathcal{D}, \pi = (\pi_e)_{e \in E})$ as $\mathcal{A}(I) := \mathbb{E}\left[\sum_{e \in M} X_e - \sum_{e \in Q} \pi_e\right]$, where the expectation is taken over the distribution $\mathcal{D}$ and any internal randomness of the algorithm. We denote an optimal algorithm for this model by $\mathcal{A}_{\text{OPT}}$.

Fix some input instance $I$ and let $\mathcal{M}$ be the collection of all valid bipartite matchings in $G$. The following lemma is due to Singla [84].

**Lemma 5.1.** *For each edge $e \in E$, let $\tau_e$ be the solution to the equation $\mathbb{E}[\max\{(X_e - \tau_e), 0\}] = \pi_e$ and let $Y_e = \min(X_e, \tau_e)$. Then the optimal expected price-of-information utility $\mathcal{A}_{\text{OPT}}(I)$ is upper bounded by $\mathbb{E}_{\mathbf{X}}\left[\max_{M \in \mathcal{M}} \sum_{e \in M} Y_e\right]$.*

To derive our algorithm, we go through the same two stages as in Chapter 4. We first construct a linear program (LP), this time defining the constraints using the probability distributions of $Y_e$'s (that were defined in Lemma 5.1), and use its value together with Lemma 5.1 to upper bound OPT. For this, we discretize the distributions of $Y_e$'s, and in contrast to the query-commit setting, we now define variables $x_{e,v}$ for each edge-value pair $(e, v)$; We think of $x_{e,v}$ as the joint probability that $Y_e = v$ and $\mathcal{A}_{\text{OPT}}(I)$ includes edge $e$ in its output. The objective value of this LP upper bounds the quantity $\mathbb{E}_{\mathbf{X}}\left[\max_{M \in \mathcal{M}} \sum_{e \in M} Y_e\right]$, which in turn is an upper bound of the optimal expected price-of-information utility as stated in Lemma 5.1. We describe the construction of our LP in Section 5.1. Next, in Section 5.2, we analyze the structure of our new LP as we did in the previous chapter and use it to define analogous probability distributions over subsets of edge-value pairs. Finally, in Section 5.3 we put everything together to construct our $(1 - 1/e)$-approximate price-of-information algorithm for MWBM.

## 5.1 Upper-bounding the Optimal Expected Utility

Assume that the distributions of $Y_e$ are discrete[1]. For each $e \in E$, let $V_e$ denote the set of possible values of $Y_e$. For each vertex $u \in A \cup B$, let $E_u = \{(e, v) : e \in \delta(u), v \in V_e\}$ be the set of all edge-value pairs for all edges incident to $u$. Let $E_{\text{all}} = \cup_{u \in A} E_u$ be the set of all edge-value pairs. For each edge $e \in E$ and value $v \in V_e$, let $p_{e,v}$ be the probability that $Y_e = v$, and for a set $F \subseteq E_{\text{all}}$, let $f(F)$ be the probability that $Y_e = v$ for at least one edge-value pair $(e, v) \in F$.

Fix any price-of-information algorithm $\mathcal{A}$ for MWBM. For each edge-value pair $(e, v) \in E_{\text{all}}$, let $A_{e,v}$ be the event that $Y_e = v$ and $\mathcal{A}$ includes edge $e$ in its output, and let $x_{e,v} = \Pr[A_{e,v}]$. Now fix a vertex $u \in A \cup B$ and a set $F \subseteq E_u$. Then $\Pr[\cup_{(e,v)\in F} A_{e,v}] \leq \Pr[Y_e = v \text{ for some } (e, v) \in F] = f(F)$. But since all events $A_{e,v}$ for $(e, v) \in E_u$ are mutually disjoint (as with the query-commit setting, the algorithm $\mathcal{A}$ outputs a valid matching and thus the output has at most one edge incident to vertex $u$), $\Pr[\cup_{(e,v)\in F} A_{e,v}] = \sum_{(e,v)\in F} \Pr[A_{e,v}] = \sum_{(e,v)\in F} x_{e,v}$. Thus we have $\sum_{(e,v)\in F} x_{e,v} \leq f(F)$, and this must be true for all $u \in A \cup B$ and $F \subseteq E_u$.

Now consider the following LP, which we call $\text{LP}_{\text{PoI}}$.

$$\begin{aligned}
\text{Maximize} \quad & \sum_{(e,v)\in E_{\text{all}}} x_{e,v} \cdot v, \\
\text{subject to} \quad & \sum_{(e,v)\in F} x_{e,v} \leq f(F) && \text{for all } F \subseteq E_u \text{ for all } u \in A \cup B, \\
& x_{e,v} \geq 0 && \text{for all } (u, v) \in E_{\text{all}}.
\end{aligned}$$

We have the following lemma concerning $\text{LP}_{\text{PoI}}$.

**Lemma 5.2.** *The optimal expected price-of-information utility $\mathcal{A}_{\text{OPT}}(I)$ is upper bounded by the value of $\text{LP}_{\text{PoI}}$.*

*Proof.* By Lemma 5.1, we have that $\text{OPT} \leq \mathbb{E}_{\mathbf{X}}[\max_{M\in\mathcal{M}} \sum_{e\in M} Y_e]$. Now consider an algorithm $\mathcal{A}$ that queries $Y_e$ for all edges $e \in E$ and outputs a maximum weighted bipartite matching $M$ of $G$. Setting $x_{e,v}$ to be the joint probability that $Y_e = v$ and $e \in M$ for each edge-value pair $(e, v) \in E_{\text{all}}$ gives a feasible solution to $\text{LP}_{\text{PoI}}$. Hence $\mathcal{A}(I) = \mathbb{E}_{\mathbf{X}}[\max_{M\in\mathcal{M}} \sum_{e\in M} Y_e] \leq \sum_{(e,v)\in E_{\text{all}}} x^*_{e,v} \cdot v$, where $\mathbf{x}^* = (x^*_{e,v})_{(e,v)\in E_{\text{all}}}$ is an optimal solution of $\text{LP}_{\text{PoI}}$. □

## 5.2 Structure of the LP

Now we analyze the structure of $\text{LP}_{\text{PoI}}$. Our analysis closely follows that of Section 4.2.

As usual, $f$ is a coverage function and hence it is submodular. Thus, for each vertex $u \in A \cup B$, we can use submodular minimization to check whether any constraint of the form $\sum_{(e,v)\in F} x_{e,v} \leq f(F)$ is violated for any subset $F \subseteq E_u$. This yields Lemma 5.3 below.

**Lemma 5.3.** *The linear program $\text{LP}_{\text{PoI}}$ is solvable in polynomial-time.*

We proceed as follows. Consider the query strategy given in Algorithm 5.1 that queries edges

[1] We can e.g. achieve this by geometric grouping into polynomially many classes.

incident to a fixed vertex $a \in A$ in some random order. This is the price-of-information version of the Algorithm 4.1 given for the query-commit setting. Following (almost) the same procedure as in Section 4.2, we find distributions $\mathcal{D}_a^{\text{PoI}}$ that makes Algorithm 5.1 pick an edge $e$ that has value $v$ with probability $x^*_{e,v}$, and then use those to construct a price-of-information algorithm for MWBM that gives $(1 - 1/e)$ approximation guarantee. But the issue here is that Algorithm 5.1 considers the distributions of $Y_e$'s and does not pay query costs whereas our final approximate price-of-information algorithm needs to consider the distributions of $X_e$'s and has to pay query costs.

**Algorithm 5.1:** Query algorithm for selecting an edge incident to a fixed vertex $a \in A$.

Let $z_e = \text{Null}$ for all $e \in \delta(a)$.
Draw a permutation $\sigma$ from $\mathcal{D}_a^{\text{PoI}}$.
**foreach** $(e, v)$ *in the order of* $\sigma$ **do**
    If $z_e = \text{Null}$, draw $z_e$ from a distribution identical to that of $Y_e$.
    If $z_e = v$, output $e$ and terminate.

Now consider the way we defined $\tau_e$ (which we used to define $Y_e$'s), and observe that the values of $X_e$ above the threshold $\tau_e$, on expectation, covers the cost $\pi_e$ of querying it. Thus, if we can make sure that the first time we query an edge $e$ (i.e., the time where we pay the price $\pi_e$) in Algorithm 5.1 is for the value $\tau_e$, then we can still use it to construct our final price-of-information matching algorithm (where we actually query $X_e$ values, and when an edge $e$ is queried for the first time for value $\tau_e$, in expectation we actually get a net value of $\tau_e$ with probability $x^*_{e,\tau_e}$ after paying $\pi_e$). Using a careful construction, we make sure that distributions $\mathcal{D}_a^{\text{PoI}}$ only gives those permutations where for any edge $e$, the pair $(e, \tau_e)$ appears before any other pair $(e, v)$.

For such a construction, we consider a slightly different polytope $P_a^{\text{PoI}}$ (as opposed to how we defined $P_a^{\text{QC}}$) for each $a \in A$. Fix a vertex $a \in A$, and consider the family $\mathcal{E}_a$ of subsets of $E_a$ defined as follows:

$$\mathcal{E}_a := \{F \subseteq E_a : (e, v) \in F \Rightarrow (e, v') \in F \text{ for all } v' \geq v \text{ such that } (e, v') \in E_a\}.$$

I.e., $\mathcal{E}_a$ is a family of subsets of $E_a$ that satisfy the following: If a set $F$ of edge-value pairs is in $\mathcal{E}_a$ and an edge-value pair $(e, v)$ is in $F$, then $F$ also contains all edge-value pairs for the same edge $e$ having values greater than $v$. It is easy to verify that if $A, B \in \mathcal{E}_a$ then both $A \cup B \in \mathcal{E}_a$ and $A \cap B \in \mathcal{E}_a$, which makes $\mathcal{E}_a$ a lattice family. (I.e., the sets in $\mathcal{E}_a$ forms a lattice where intersection and union serve as *meet* and *join* operations respectively.)

We define below the polytope $P_a^{\text{PoI}}$ using a constraint for each set in the family $\mathcal{E}_a$.

$$\sum_{(e,v)\in F} x_{e,v} \leq f(F) \qquad \text{for all } F \in \mathcal{E}_a \tag{5.1}$$

$$x_{e,v} \geq 0 \qquad \text{for all } (e, v) \in E_a.$$

Analogous to our assumption $0 < p_e < 1$ for all $e \in E$ for the query-commit setting, we now assume that each $p_{e,v} > 0$ and for each edge $e$, $\sum_{v \in V_e} p_{e,v} < 1$. (I.e., we can assume that with some small probability $p_{e,\star}$ the edge $e$ does not exist or equivalently, we can also assume $p_{e,0} > 0$ and $0 \notin V_e$. We omit the details, but one can use the same argument of re-scaling the probabilities

to justify this assumption.) Under these assumptions on $p_{e,v}$'s, we have the following lemma. The proof resembles that of Lemma 4.4 from Section 4.2, and we defer it to Appendix A.

**Lemma 5.4.** *Fix a vertex $a \in A$. If $p_{e,v} > 0$ for all $(e,v) \in E_a$ and $\sum_{v \in V_e} p_{e,v} < 1$ for all $e \in \delta(a)$, the function $f$ is strictly submodular and strictly increasing on the lattice family $\mathcal{E}_a$. Formally,*

1. *For any $A, B \in \mathcal{E}_a$ such that $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$, $f(A) + f(B) > f(A \cap B) + f(A \cup B)$, and*
2. *For any $A \subsetneq B \subseteq E_a$, $f(B) > f(A)$.*

Similarly to the query-commit setting, we now analyze the structure of the extreme points of polytope $P_a^{\text{PoI}}$. We have the following lemma, which is a slightly different version of Lemma 4.5 from Section 4.2.

**Lemma 5.5.** *Let $\mathbf{y} = (y_{e,v})_{(e,v) \in E_a}$ be an extreme point of $P_a^{\text{PoI}}$ and let $Y = \{e \in E_a : y_{e,v} > 0\}$ be the set of non-zero coordinates of $\mathbf{y}$. Then there exist $|Y|$ subsets $S_1, \ldots, S_{|Y|}$ of $E_a$ such that $S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_{|Y|}$ with the following properties:*

1. *Constraint (5.1) is tight for all $S_1, S_2, \ldots S_{|Y|}$. That is $\sum_{(e,v) \in S_i} y_{e,v} = f(S_i)$ for all $i = 1, \ldots, |Y|$.*
2. *For each $i = 1, \ldots, |Y|$, the set $(S_i \setminus S_{i-1}) \cap Y$ contains exactly one element $(e_i, v_i)$. Moreover, for any other $(e,w) \in S_i \setminus S_{i-1}$, we have $e = e_i$ and $w \geq v_i$.*

*Proof.* Property 1 and the fact that each $(S_i \setminus S_{i-1}) \cap Y$ contains exactly one pair $(e_i, v_i)$ follows from the proof of Lemma 4.5. It remains to show that each $S_i \setminus S_{i-1}$ additionally contains only those edge-value pairs $(e_i, w)$ for which $w \geq v_i$.

Suppose to the contrary that there is some $S_i \setminus S_{i-1}$ that contains at least one other pair $(e', v')$ that violates this property. Let $S_i' = S_{i-1} \cup \{(e_i, w) : w \geq v_i\}$. Then $S_{i-1} \subsetneq S_i' \subsetneq S_i$ and $S_i'$ is also in the family $\mathcal{E}_a$. Thus we have that $f(S_i') \geq \sum_{(e,v) \in S_i'} y_{e,v} = \sum_{(e,v) \in S_i} y_{e,v} = f(S_i)$, which is a contradiction because $f$ is strictly increasing and $S_i' \subsetneq S_i$. Here the first inequality holds because $\mathbf{y}$ is in $P_a^{\text{PoI}}$ and the first equality holds because $S_i'$ contains all coordinates in $S_i$ for which $\mathbf{y}$ is non-zero. The last equality is true because $S_i$ corresponds to a tight constraint for the extreme point $\mathbf{y}$. □

As in the previous section, we are now ready to construct the distribution $\mathcal{D}_a^{\text{PoI}}$. We present this explicit construction in the proof of Lemma 5.6 stated below, which is the counterpart of Lemma 4.6.

**Lemma 5.6.** *Let $\mathbf{x}^*$ be an optimal solution of $\text{LP}_{\text{PoI}}$. For each vertex $a \in A$, there exist a distribution $\mathcal{D}_a^{\text{PoI}}$ over the permutations of (subsets of) edge-value pairs in $E_a$ that satisfies the following properties:*

1. *For each permutation $\sigma$ drawn from $\mathcal{D}_a^{\text{PoI}}$, if edge-value pair $(e,v)$ appears in $\sigma$, then the edge-value pair $(e,w)$ appears before $(e,v)$ in $\sigma$ for all $w \in V_e$ such that $w \geq v$,*

2. $\Pr[\textit{Algorithm 5.1 outputs } e] = \sum_{v \in V_e} x^*_{e,v}$ *for all* $e \in \delta(a)$, *and*
3. $\sum_{v \in V_e : v \geq w} \Pr[\textit{Algorithm 5.1 outputs } e \textit{ with value } v] \cdot v \geq \sum_{v \in V_e : v \geq w} x^*_{e,v} \cdot v$ *for all* $e \in \delta(a)$ *and* $w \in \mathbb{R}^+$.

*Moreover, a permutation of edge-value pairs from* $\mathcal{D}_a^{\text{PoI}}$ *can be sampled in polynomial time.*

*Proof.* As with the case of Lemma 4.6, we first associate a permutation of edge-value pairs with each extreme point of $P_a^{\text{PoI}}$.

For an extreme point $\mathbf{y}$, let $Y$ and $S_1, \ldots, S_{|Y|}$ be as defined in Lemma 5.5. Consider the permutation $\sigma_y$ of elements in $S_{|Y|}$ that is defined as follows: Start with $\sigma_y = [\,]$ and for each $i = 1, \ldots, |Y|$, append to it the edge-value pairs in $S_i \setminus S_{i-1}$ in the decreasing order of value. Recall that for all $i = 1, \ldots, |Y|$, all edge-value pairs in $S_i \setminus S_{i-1}$ corresponds to a single edge. Let $(e, v) \in S_i$. Then, by the definition of the family $\mathcal{E}_u$, $(e, v') \in S_i$ for all $v' \in V_e$ such that $v' \geq v$. Thus none of the sets $S_{i+1} \setminus S_i, S_{i+2} \setminus S_{i+1}, \ldots, S_{|Y|} \setminus S_{|Y|-1}$ can contain an edge-value pair $(e, v')$ such that $v' > v$. Also, since the elements in $S_i \setminus S_{i-1}$ are appended to $\sigma_y$ in decreasing order of values, $\sigma_y$ has the following property: If at any point the edge-value pair $(e, v)$ appears in $\sigma_y$, then $(e, w)$ appears in $\sigma_y$ before $(e, v)$ for all $w \in V_e$ such that $w > v$.

Let $\sigma_y = (e_1, v_1), \ldots, (e_\ell, v_\ell)$ be the permutation of edge value pairs associated with the extreme-point $\mathbf{y}$. Let $T_i := \{(e_1, v_1), \ldots, (e_i, v_i)\}$ denote the set of first $i$ edge-value pairs in $\sigma_y$. If we select permutation $\sigma_y$ in Line 2 in Algorithm 5.1, the probability of it picking edge $e_i$ with value $v_i$ is exactly $f(T_i) - f(T_{i-1})$. Now define a new vector $\mathbf{y}'$ with the same indices as $\mathbf{y}$ as follows: For each $(e_j, v_j) \in \sigma_y$, $y'_{e_j, v_j} = f(T_j) - f(T_{j-1})$, and all the other coordinates of $\mathbf{y}'$ are 0. Notice that $\mathbf{y}'$ and $\mathbf{y}$ satisfy the following:

1. $\sum_{v \in V_e} y'_{e,v} = \sum_{v \in V_e} y_{e,v}$ for all $e \in \delta(a)$, and
2. $\sum_{v \in V_e : v \geq w} y'_{e,v} \cdot v \geq \sum_{v \in V_e : v \geq w} y_{e,v} \cdot v$ for all $e \in \delta(a)$ and $w \in \mathbb{R}^+$.

To see this fix some set $S_i$ and let $(e_j, v_j), (e_{j+1}, v_{j+1}), \ldots, (e_k, v_k)$ be all the edge-value pairs in $S_i - S_{i-1}$. Then we know that $e_j = e_{j+1} = \cdots = e_k$, $v_j > v_{j+1} > \cdots > v_k$, and $y_{e_k, v_k} \neq 0$. We thus have that $\sum_{j'=j}^{k} y'_{e_k, v_j} = f(T_k) - f(T_{j-1}) = f(S_i) - f(S_{i-1}) = y_{e_k, v_k}$ (recall that $(e_k, v_k)$ is the unique element in $S_i \setminus S_{i-1}$ for which $y_{e_k, v_k}$ is non-zero). This holds for elements in $S_i \setminus S_{i-1}$ in all $i = 1, \ldots, |Y|$, and yields the Property 1 above. To see Property 2, notice that within each $S_i \setminus S_{i-1}$, the weight of the non-zero coordinate $y_{e_k, v_k}$ is re-distributed among $y_{e_j, v_j}, \ldots, y_{e_k, v_k}$, and that $v_{j'} \geq v_k$ for $j' = j, j+1, \ldots, k$.

Let $\sigma$ be the random variable that denotes the permutation picked by Algorithm 5.1. Then we have that, for all $e \in \delta(u)$,

$$\Pr[\text{Algorithm 5.1 picks } e | \sigma = \sigma_y] = \sum_{v \in V_e} y'_{e,v} = \sum_{v \in V_e} y_{e,v},$$

and for all $(e, w) \in E_u$,

$$\sum_{e, v \in V_e : v \geq w} \Pr\left[\begin{matrix}\text{Algorithm 5.1 picks } e \\ \text{with value } v\end{matrix} \middle| \sigma = \sigma_y\right] \cdot v = \sum_{e, v \in V_e : v \geq w} y'_{e,v} \cdot v \geq \sum_{e, v \in V_e : v \geq w} y_{e,v} \cdot v.$$

Now let $\mathbf{x}_a^*$ be the restriction of the optimal solution the coordinates in $E_a$. Since the constraints that define $P_a^{\text{PoI}}$ are only a subset of the constraints of $\text{LP}_{\text{PoI}}$, $\mathbf{x}_a^*$ lies in $P_a^{\text{PoI}}$. If we can optimize a linear function of $P_a^{\text{PoI}}$ in polynomial time, then we can follow the same lines of the proof of Lemma 4.6 and use the constructive version of Caratheodary's theorem to find a convex combination $\mathbf{x}_a^* = \sum_{i\in[k]} a_i \cdot \mathbf{y}^{(i)}$, where $k = \text{poly}(|E_a|)$ and for each $i \in [k]$, $\mathbf{y}^{(i)}$ is an extreme point of $P_a^{\text{PoI}}$. However, unlike in the query-commit case, the polytope $P_a^{\text{PoI}}$ only has constraints for sets of a lattice family, and as a result, we cannot use the usual submodular minimization as a separation oracle for $\text{LP}_{\text{PoI}}$. But luckily, Grötschel et al. [48] showed that we can minimize any submodular function over a lattice family in polynomial time. Thus we can efficiently find such a convex combination.

Once we have the convex combination, the rest is exactly the same as the query-commit setting. The distribution $\mathcal{D}_a^{\text{PoI}}$ returns the permutation $\sigma_{y^{(i)}}$ with probability $a_i$ for all $i \in [k]$. One can easily verify that Properties 1-3 hold for this distribution. □

## 5.3 Proposed Algorithm and Analysis

We now present a $(1 - 1/e)$-approximate price-of-information algorithm Approx-PoI for MWBM.

The algorithm closely resembles the algorithm Approx-QC we presented for the query-commit model, but has two key differences. First, we now have to pay a price for querying edges. But, as we prove later, this price is already taken care of by the way we defined $Y_e$ variables. The second difference is that for each edge $e \in E$, we now have multiple values to consider, but regardless, with respect to a fixed vertex (i.e. an item) $b \in B$, the vertices $a \in A$ can still be viewed as buyers; The appearance of multiple edge-value pairs for the same edge can be interpreted as a distribution over values that the buyer $a$ offers for the item $b$.

The outline of our algorithm is given in Algorithm 5.2. Note that, we again have the **else** clause in the conditional to make sure that we do not change the probability distributions of the appearances of edge-value pairs even if we decide not to query some edges for certain values.

The analysis of the expected price-of-information utility of Approx-PoI consists of two parts. First we use the same technique used by Singla [84] to show that we can analyze the expected utility using $Y_e$ variables and no query costs instead of using $X_e$ variables with query costs. Next we reproduce almost the same analyis by Ehsani et al. [34] to prove the approximation guarantee.

Let $Z$ be the value we get from Algorithm 5.2. Then $Z = \sum_{e\in E} \left(\mathbb{I}_e^{\text{pick}} X_e - \mathbb{I}_e^{\text{query}} \pi_e\right)$, where $\mathbb{I}_e^{\text{pick}}$ and $\mathbb{I}_e^{\text{query}}$ are the indicator variables for the events that edge $e$ is picked in Line 11 and it is queried in Line 10 respectively.

Now suppose that we run an identical copy of Algorithm 5.2 in parallel but we do not pay for querying in Line 10, but instead of gaining $X_e$, we only get $Y_e = \min(X_e, \tau_e)$. We say that this latter execution in the "free-information" world whereas the original algorithm runs in the "price-of-information" world. Let $Z'$ be the value we get in the free information world. We have the following lemma.

**Lemma 5.7.** *The expected utility $\mathbb{E}[Z]$ of Approx-PoI (which is the price-of-information world) is equal to the expected utility of its counterpart in the free information world.*

**Algorithm 5.2:** Outline of Approx-PoI.

1 Solve $LP^\star$ to get $\mathbf{x}^*$ and find the permutation distributions $\mathcal{D}_a^{\text{PoI}}$ for all $a \in A$.
2 For each vertex $a \in A$, select $t_a \in [0,1]$ (arrival time) independently and uniformly at random.
3 For each vertex $b \in B$, let $c_b = \sum_{(e,v)\in E_b} x^*_{e,v} \cdot v$.
4 Let $z_e = \text{Null}$ for all $e \in E$.
5 Let $M$ be an empty matching.
6 **foreach** *vertex* $a \in A$ *in the increasing order of* $t_a$ **do**
7  Draw a permutation $\sigma$ from $\mathcal{D}_a^{\text{PoI}}$.
8  **foreach** $(e = (a,b), v)$ *in the order of* $\sigma$ **do**
9  **if** $v \geq (1 - e^{t_a - 1}) \cdot c_b$ **and** $b$ *is not matched* **then**
10  If $z_e = \text{Null}$, pay $\pi_e$ and query edge $e$ to find its actual value. Let $z_e$ be this value.
11  If $\min(z_e, \tau_e) = v$, add $e$ to $M$ and continue to next vertex in $A$.
12  **else**
13  If $z_e = \text{Null}$, draw $z_e$ from a distribution identical to that of $X_e$.
14  If $\min(z_e, \tau_e) = v$, continue to next vertex in $A$.
15 **return** the matching $M$.

*Proof.* Consider a case where the algorithm picks an edge that is already queried before. In this case, both algorithms get the same value, so the expected increase to $Z$ and $Z'$ are the same.

Now consider the case where both algorithms query for some edge $e$. Notice that if an edge $e$ is queried at any point, it is queried for the edge-value pair $(e, \tau_e)$. This is because $\tau_e$ is the maximum possible value for an edge $e$, and as a result, $(e, \tau_e)$ appears before any other $(e, v)$ for $v < \tau_e$ (if it appears at all) in the permutations chosen in Line 7 of Algorithm 5.2. In this case, the expected increase to $Z'$ in the free information world is $\tau_e \cdot \Pr[X_e \geq \tau_e]$. The expected increase to $Z$ in the price-of-information world is

$$\begin{aligned} -\pi_e + \int_{\tau_e}^{\infty} t \cdot p_e(t) dt &= -\pi_e + \int_{\tau_e}^{\infty} (t - \tau_e) \cdot p_e(t)\, dt + \int_{\tau_e}^{\infty} \tau_e \cdot p_e(t)\, dt \\ &= \underbrace{-\pi_e + \mathbb{E}[(X_e - \tau_e)^+]}_{0} + \tau_e \cdot \Pr[X_e \geq \tau_e]. \end{aligned}$$

□

To conclude our analysis, we now present following theorem on the approximation guarantee.

**Theorem 5.8.** *The expected price-of-information utility of* Approx-PoI *is at least* $(1 - 1/e) \cdot \text{OPT}$.

*Proof.* By Lemma 5.7, we have $\text{Approx-PoI}(I) = \mathbb{E}[Z] = \mathbb{E}[Z']$. Since the optimal value of $\text{LP}_{\text{PoI}}$ is an upper bound on the optimal value OPT (by Lemma 5.2), it is sufficient to show that

$$\mathbb{E}[Z'] \geq (1 - 1/e) \cdot \sum_{(e,v)\in E_{\text{all}}} x^*_{e,v} v = (1 - 1/e) \cdot \sum_{b \in B} c_b.$$

(Recall that $c_b = \sum_{(e,v)\in E_b} x^*_{u,v} \cdot v$ as defined in Algorithm 5.2.) Let $Z'_b = \sum_{a\in\delta(b)} \mathbb{I}^{\text{pick}}_{a,b} Y_{a,b}$ so

that $Z' = \sum_{b \in B} Z'_b$. We show that for any $b \in B$, $\mathbb{E}[Z'_b] \geq (1 - 1/e) \cdot c_b$. Then the theorem follows from the linearity of expectation. The following calculations now follow the analysis in [34] and are included for completeness.

We proceed by splitting $Z'_{a,b}$ into two parts:

$$Z'_{a,b} = \underbrace{\mathbb{I}^{\text{pick}}_{a,b}(Y_{a,b} - (1 - e^{t_a - 1}) \cdot c_b)}_{M_{a,b}} + \underbrace{\mathbb{I}^{\text{pick}}_{a,b}(1 - e^{t_a - 1}) \cdot c_b}_{N_{a,b}}.$$

Define $r(t) := \Pr[\text{no edge incident to } b \text{ is picked before time } t]$ and $\alpha(t) := 1 - e^{t-1}$. Notice that $r(t)$ is decreasing, and since our $t_a$'s are from a continuous distribution, $r(t)$ is a differentiable function. Thus we have

$$\mathbb{E}\left[\sum_{a \in \delta(b)} N_{a,b}\right] = -\int_0^1 r'(t) \cdot (1 - 1/e^{t-1}) \cdot c_b \, dt = -c_b \int_0^1 r'(t) \cdot \alpha(t) \, dt.$$

By applying integration by parts,

$$\begin{aligned}
\mathbb{E}\left[\sum_{a \in \delta(b)} N_{a,b}\right] &= -c_b \left( [r(t) \cdot \alpha(t)]_0^1 - \int_0^1 r(t) \cdot \alpha'(t) \, dt \right) \\
&= c_b \left( (1 - 1/e) + \int_0^1 r(t) \cdot \alpha'(t) \, dt \right). \qquad (5.2)
\end{aligned}$$

Now we consider the expectation of $M_{a,b}$. Note that the inequality below is due to the third property of the distributions $\mathcal{D}_a^{\text{PoI}}$ of edge-value pairs we used in the algorithm (see Lemma 5.6).

$$\mathbb{E}[M_{a,b} | t_a = t] \geq \Pr\left[\begin{matrix}\text{no edge incident to} \\ b \text{ is picked before } t\end{matrix} \middle| t_a = t\right] \cdot \sum_{\substack{v \in V_{a,b} \\ v \geq \alpha(t) \cdot c_b}} x^*_{(a,b),v}(v - \alpha(t) \cdot c_b).$$

If $t_a = t$, this means that $t_a$ could not have arrived before $t$. Hence $\Pr[\text{no edge incident to } b \text{ is picked before } t \mid t_a = t] \geq \Pr[\text{no edge incident to } b \text{ is picked before } t\,]$, and thus we have

$$\begin{aligned}
\sum_{a \in \delta(b)} \mathbb{E}[M_{a,b} | t_a = t] &\geq \Pr\left[\begin{matrix}\text{no edge incident to} \\ b \text{ is picked before } t\end{matrix} \middle| t_a = t\right] \cdot \sum_{a \in \delta(b)} \sum_{\substack{v \in V_{a,b} \\ v \geq \alpha(t) \cdot c_b}} x^*_{(a,b),v}(v - \alpha(t) \cdot c_b) \\
&\geq \Pr\left[\begin{matrix}\text{no edge incident to} \\ b \text{ is picked before } t\end{matrix}\right] \cdot \sum_{a \in \delta(b)} \sum_{\substack{v \in V_{a,b} \\ v \geq \alpha(t) \cdot c_b}} x^*_{(a,b),v}(v - \alpha(t) \cdot c_b) \\
&\geq r(t) \cdot \sum_{a \in \delta(b)} \sum_{v \in V_{a,b}} x^*_{(a,b),v}(v - \alpha(t) \cdot c_b) \\
&= r(t) \cdot \left( c_b - \alpha(t) \cdot c_b \sum_{a \in \delta(b)} \sum_{v \in V_{a,b}} x^*_{(a,b),v} \right) \\
&\geq r(t) \cdot (1 - \alpha(t)) \cdot c_b.
\end{aligned}$$

Since $t_a$ is uniformly distributed over $[0,1]$ for each $a \in A$, it follows that

$$\mathbb{E}\left[\sum_{a\in\delta(b)} M_{a,b}\right] = \int_0^1 \sum_{a\in\delta(b)} \mathbb{E}[M_{a,b}|t_a = t]\,dt \geq c_b \int_0^1 r(t)\,(1-\alpha(t))\;dt. \tag{5.3}$$

Now (5.2) + (5.3) yields

$$\begin{aligned}
\mathbb{E}\left[\sum_{a\in\delta(b)} Z'_{a,b}\right] &= \mathbb{E}\left[\sum_{a\in\delta(b)} N_{a,b}\right] + \mathbb{E}\left[\sum_{a\in\delta(b)} M_{ij}\right] \\
&\geq c_b\left((1-1/e) + \int_0^1 r(t)\cdot\alpha'(t)\,dt\right) + c_b\int_0^1 r(t)\,(1-\alpha(t))\;dt \\
&= c_b(1-1/e) + c_b\int_0^1 r(t)\,\underbrace{(1-\alpha(t)+\alpha'(t))}_{0}\;dt \\
&= (1-1/e)\cdot c_b.
\end{aligned}$$

□

# Part II

# Weighted Matching in Large Graphs

## Techniques for MPC and Streaming Models

# 6 Overview of Matching in MPC and Streaming Models

In this second part of the thesis, we study the maximum weighted matching (MWM) problem in the semi-streaming and massively parallel computation (MPC) models. Both the MPC model (which encompasses many of today's most successful parallel computing paradigms such as MapReduce and Hadoop) and the semi-streaming model are motivated by the need for devising efficient algorithms for large problem instances. As data and the size of instances keep growing, this becomes ever more relevant, and a large body of recent work has been devoted to these models.

In these models of computation, prior works have left a gap between the weighted (MWM) and unweighted (MCM) versions of the matching problem, and our work attempts to develop new techniques to close this gap.

In the semi-streaming model, recall that the edges of the graph arrive one-by-one, and the algorithm is restricted to use memory that is almost linear in the number of vertices. For unweighted graphs, the very basic greedy algorithm guarantees to return $(1/2)$-approximate maximum matching. It remains a major open problem to improve upon this factor when the order of the stream is adversarial. However, in the so-called random-edge-arrival setting — where the edges of the stream are presented in random order — algorithms that are more advanced than the greedy algorithm overcome this barrier [68] for MCM. In contrast, for weighted graphs, a $(1/2-\varepsilon)$-approximation algorithm was given only recently for adversarial streams [44, 83], and here we give the first algorithm that breaks the natural "greedy" barrier of $1/2$ for random-edge-arrival streams:

**Theorem 6.1.** *There is a $(1/2+c)$-approximation algorithm for finding weighted matchings in the streaming model with random-edge-arrivals, where $c > 0$ is an absolute constant.*

As we elaborate below, the result is achieved via a general approach that reduces the task of finding weighted matchings to that of finding (short) *unweighted* augmenting paths. This allows us to incorporate some of the ideas present in the streaming algorithms for unweighted matchings to achieve our result. Our techniques, perhaps surprisingly, also simplify the previous algorithms for finding unweighted matchings and give an improved guarantee for general graphs.

The idea of reducing to the problem of finding unweighted augmenting paths is rather versatile, and we use it to obtain a general reduction from weighted matchings to unweighted matchings as our second main result. We give implementations of this reduction in the models of multi-pass streaming and MPC that incur only a constant factor overhead in the complexity. In multi-pass streaming, the algorithm is (as for single-pass) restricted to use memory that is almost linear

in the number of vertices, and the complexity is measured in terms of the number of passes that the algorithm requires over the data stream. In MPC, parallel computation is modeled by parallel machines with sublinear memory (in the input size), and data can be transferred between machines only between two rounds of computation. The complexity of an algorithm in the MPC model, also referred to as the round complexity, is then measured as the number of (communication) rounds used.

For the matching problem, McGregor [77] gave the first multi-pass semi-streaming algorithm for approximating unweighted matchings within a factor $(1-\varepsilon)$. The algorithm runs in a constant (depending only on $\varepsilon$) number of passes, and the dependency on $\varepsilon$ was later improved for bipartite graphs [3, 33]. McGregor's techniques for unweighted matchings have been very influential. In particular, his general reduction technique can be used to transform any $O(1)$-approximation unweighted matching algorithm that uses $R$ MPC rounds into a $(1-\varepsilon)$ approximation unweighted matching algorithm that uses $O_\varepsilon(R)$ rounds in the MPC model. This together with a sequence of recent papers [9, 27, 45], that give constant-factor approximation algorithms for unweighted matchings with improved round complexity, culminated in algorithms that find $(1-\varepsilon)$-approximate maximum unweighted matchings in $O_\varepsilon(\log\log n)$ rounds. However, as McGregor's techniques apply to only unweighted matchings, it was not known how to achieve an analogous result in the presence of weights. In fact, McGregor raised as an open question whether his result can be generalized to weighted graphs. Our result answers this in the affirmative and gives a reduction that is *lossless* with respect to the approximation guarantee while only increasing the complexity by a constant factor. Moreover, our reduction is to bipartite graphs. Instantiating this with the aforementioned streaming and MPC algorithms for unweighted matchings yields the following theorem. Recall that we denote the number of vertices of the input graph by $n$ and the number of edges by $m$.

**Theorem 6.2.** *There exists an algorithm that in expectation finds a $(1-\varepsilon)$-approximate weighted matching that can be implemented*

1. *in $O_\varepsilon(U_M)$ rounds, $O(m/n)$ machines per round, and $O_\varepsilon(n\,\mathrm{poly}(\log n))$ memory per machine, where $U_M$ is the number of rounds used by a $(1-\delta)$-approximation algorithm for* bipartite *unweighted matching using $O(m/n)$ machines per round, and $O_\delta(n\,\mathrm{poly}(\log n))$ memory per machine in the MPC model, and*
2. *in $O_\varepsilon(U_S)$ passes and $O_\varepsilon(n\,\mathrm{poly}(\log n))$ memory, where $U_S$ is the number of passes used by a $(1-\delta)$-approximation algorithm for* bipartite *unweighted matching using $O_\delta(n\,\mathrm{poly}(\log n))$ memory, in the multi-pass streaming model,*

*where $\delta = \varepsilon^{28+900/\varepsilon^2}$. Using the algorithm of Ghaffari et al. [45] or that of Assadi et al. [9], we get that $U_M = O_\varepsilon(\log\log n)$ and using the algorithm of Ahn and Guha [3], we get that $U_S = O_\varepsilon(1)$.*

Prior to this, the known best results for computing a $(1-\varepsilon)$-approximate *weighted* matching required super constant $\Omega(\log n)$ many passes over the stream in the streaming model [3] and $\Omega(\log n)$ rounds [4] in the MPC model. We remark that if we allow for memory $\tilde{\Theta}(n^{1+1/p})$ per machine in the MPC model, then [4] gave an algorithm that uses only a constant number of rounds (depending on $p$). Achieving a similar result with near-linear memory per machine is a major open question in the MPC literature; our results show that it is sufficient to concentrate on unweighted graphs as any progress on such graphs gives analogous progress in the weighted setting. We now give an outline of our approach.

## 6.1 Outline of Our Approach

Let $M$ be a matching in a graph $G = (V, E)$ with edge-weights $w : E \to \mathbb{R}$. Recall that an alternating path $P$ is a path in $G$ that alternates between edges in $M$ and in $E \setminus M$. If the endpoints of $P$ are unmatched vertices or incident to edges in $M \cap P$, then removing the $M$-edges in $P$ and adding the other edges of $P$ gives a new matching. In other words, $M \Delta P = (M \setminus (P \cap M)) \cup P \setminus M$ is a new matching. We say that we updated $M$ using the alternating path $P$, and we further say that $P$ is augmenting if $w(M \Delta P) > w(M)$ where we used the notation $w(F) = \sum_{e \in F} w(e)$ for a subset of edges $F \subseteq E$. Also recall that an alternating cycle $C$ is a cycle that alternates between edges in $M$ and in $E \setminus M$, and $M \Delta C$ is also a matching. We say that $C$ is augmenting if $w(M \Delta C) > w(M)$. A well-known structural result regarding approximate matchings is the following:

**Observation 6.3.** *For any $\ell \in \mathbb{N}$, if there is no augmenting path or cycle of length at most $2\ell - 1$, then $M$ is a $(1 - 1/\ell)$-approximate matching.*

In particular, this says that in order to find a $(1 - \varepsilon)$-approximate matching it is sufficient to find augmenting paths or cycles of length $O(1/\varepsilon)$. This is indeed the most common route used to design efficient algorithms for finding approximate matchings: in the streaming model with random-edge-arrivals, [68] finds augmenting paths of length $\leq 3$ and the MPC algorithms [9, 27] find augmenting paths of length $O(1/\varepsilon)$. However, those approaches work only for unweighted graphs. The high level reason being that it is easy to characterize the augmenting paths in the unweighted setting: they simply must start and end in unmatched vertices. Such a simple classification of augmenting paths is not available in the weighted setting and the techniques of those papers do not apply. Nevertheless, we propose a general framework to overcome this obstacle that allows us to tap into the results and techniques developed for unweighted matchings. Informally, we reduce the problem of finding augmenting paths in the weighted setting to the unweighted setting.

The high level idea is simple: Consider the example depicted on the left in Figure 6.1. The current matching $M$ consists of a single edge $\{c, d\}$ that is depicted by a solid line. The weights are written next to the edges and so $w(M) = 5$ (the edges $E \setminus M$ are dashed). The maximum matching consists of $\{a, c\}, \{d, f\}$ and has weight 8. Furthermore, there are several alternating paths of length 3 that are also augmenting. However, it is important to note that we cannot simply apply an algorithm for finding unweighted augmenting paths. Such an algorithm may find the alternating path $P = b, c, d, e$ which is augmenting in the unweighted sense but $w(M \Delta P) < w(M)$. To overcome this, we apply a *filtering* technique that we now explain in our simple example: First "guess" lower bounds on the weights of the edges incident to $c$ and $d$ in an augmenting path. Let $\tau_c$ and $\tau_d$ be those lower bounds. We then look for augmenting paths in the unweighted graph that keeps only those unmatched edges incident to $c$ and $d$ whose weights are above the guessed thresholds. Then to guarantee that an unweighted augmenting path that an algorithm finds is also an augmenting path in the weighted sense, we always set $\tau_c$ and $\tau_d$ such that $\tau_c + \tau_d > w(\{c, d\})$. In the center and right part of Figure 6.1 we depict two *unweighted* graphs obtained for different values of $\tau_c$ and $\tau_d$ (in the center with $\tau_c = \tau_d = 3$ and to the right with $\tau_c = 2, \tau_d = 4$). Note that in both examples any unweighted augmenting path is also augmenting with respect to the weights.

While the implementation of the basic idea is simple in the above case, there are several challenges in general. Perhaps the most obvious one is that, for weighted matchings, $M$ may be a perfect

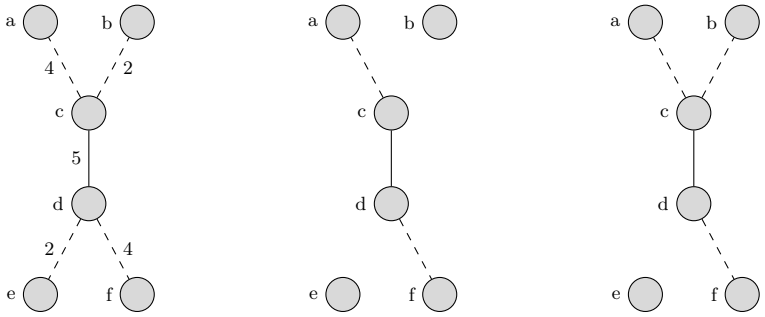



Figure 6.1 – A weighted graph with a matching (left) and two of its filtered unweighted instances (center and right).

matching but still far from optimal. And a perfect matching obviously has no unweighted augmenting paths! On a very high level, we overcome this issue by dropping edges in $M$ while making sure to set the guessed lower bounds (the $\tau$'s) so as to guarantee that any unweighted augmenting path is also a weighted augmenting path (even if the dropped edges are considered).

In what follows, we describe in more detail the implementation of the above basic idea. We start with the simpler case, single-pass streaming with random edge arrivals, where we look only for augmenting paths of length 3. We then describe the technically more involved multi-pass streaming and MPC algorithms that consider long augmenting paths and cycles.

### 6.1.1 Single-pass Streaming with Random Edge Arrivals

In contrast to unweighted graphs where the basic greedy algorithm gives a $(1/2)$-approximation, it was only very recently that a $(1/2 - \varepsilon)$-approximation streaming algorithm was given for weighted matchings [83]. The algorithm of Paz and Schwartzman is based on the local ratio technique, which we now describe. On an input graph $G = (V, E)$ with edge-weights $w : E \to \mathbb{R}$, the following simple local-ratio algorithm is known to return a $(1/2)$-approximate weighted matching: Initially, let $S = \emptyset$ and $\alpha_v = 0$ for all $v \in V$. For each $e = \{u, v\} \in E$ in an *arbitrary* order, if $\alpha_u + \alpha_v < w(e)$, add $e$ to $S$ and increase both $\alpha_u$ and $\alpha_v$ by $w(e) - \alpha_u - \alpha_v$. Finally, obtain a matching $M$ by running the basic greedy algorithm on the edges in $S$ in the reverse order (i.e., by starting with the edge last added to $S$).

Since the above algorithm returns a $(1/2)$-approximate matching irrespective of the order in which the edges are considered (in the for loop), it may appear immediate to use it in the streaming setting. The issue is that, if the edges arrive in an adversarial order, we may add *all* the edges to $S$. For dense graphs, this would lead to a memory consumption of $\Omega(n^2)$ instead of the wanted memory usage $O(n \operatorname{poly}(\log n))$ which is (roughly) linear in the output size. The main technical challenge in [83] is to limit the number of edges added to $S$; this is why that algorithm obtains a $(1/2 - \varepsilon)$-approximation, for any $\varepsilon > 0$, instead of a $(1/2)$-approximation.

McGregor and Vorotnikova observed that the technical issue in [83] disappears if we assume that edges arrive in a uniformly random order[1]. Indeed, we can then use basic probabilistic techniques

[1]Sofya Vorotnikova presented this result in the workshop "Communication Complexity and Applications, II" at the Banff International Research Station held in March 2017.

(see, e.g., the "hiring problem" in [24]) to show that the expected (over the random arrival order) number of edges added to $S$ is $O(n \log n)$. Even better, here we show that, in expectation, the following adaptation still adds only $O(n \log n)$ edges to $S$: update the vertex potentials (the $\alpha_v$'s) only for, say, 1% of the stream and then, in the remaining 99% of the stream, add *all* edges $\{u, v\}$ for which $\alpha_u + \alpha_v < w(\{u, v\})$ to $S$ (without updating the vertex potentials). This adaptation allows us to prove the following structural result:

> In a random-edge-arrival stream, either the local-ratio algorithm already obtains a (close) to (1/2)-approximate matching $M$ after seeing a small fraction of the stream (think 1%), or the set $S$ (in the adaptation that freezes vertex potentials) contains a better than (1/2)-approximation in the end of the stream.

The above allows us to concentrate on the case when we have a (close) to (1/2)-approximate matching $M_0$ after seeing only 1% of the stream. We can thus use the remaining 99% to find enough augmenting paths to improve upon the initial (1/2)-approximation. It is here that our *filtering* technique is used to reduce the task of finding weighted augmenting paths to unweighted ones. By Observation 6.3, it is sufficient to consider very short augmentations to improve upon an approximation guarantee of 1/2. Specifically, the considered augmentations are of two types:

1. Those consisting of a single edge $\{u, v\}$ to add satisfying $w(\{u, v\}) > w(M_0(u)) + w(M_0(v))$, where $w(M_0(x))$ denotes the weight of the edge of $M_0$ incident to vertex $x$ (and 0 if no such edge exists)[2].
2. Those consisting of two new edges $o_1$ and $o_2$ that form a path or a cycle $(e_1, o_1, e_2, o_2, e_3)$ with at most three edges $e_1, e_2, e_3 \in M_0$ and $w(o_1) + w(o_2) > w(e_1) + w(e_2) + w(e_3)$, i.e., adding $o_1, o_2$ and removing $e_1, e_2, e_3$ increases the weight of the matching.

For concreteness, consider the graph in Figure 6.2. The edges in $M_0$ are solid and dashed edges are yet to arrive in the stream. An example of the first type of augmentations is to add $\{e, h\}$ (and remove $\{e, f\}$ and $\{g, h\}$) which results in a gain because $w(\{e, h\}) = 2 > 1 + 0 = w(M_0(e)) + w(M_0(h))$. Two examples of the second type of augmentations are the path $(\{b, a\}, \{a, d\}, \{d, c\}, \{c, f\}, \{f, e\})$ and the cycle $(\{e, f\}, \{f, h\}, \{h, g\}, \{g, e\})$.

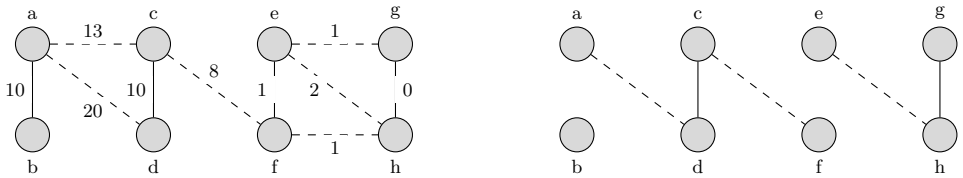



Figure 6.2 – An example of the filtering step. On the left, an example of a weighted graph with matching $M_0$ (solid edges) is shown. On the right, the unweighted graph obtained in the filtering step with $M_0' = \{\{c, d\}, \{g, h\}\}$ is shown.

The augmentations of the first type are easy to find in a greedy manner. For the second type, we now describe how to use our filtering technique to reduce the problem to that of finding

[2]To make sure that the weight of the matching increases significantly by an augmentation, the strict inequality needs to be satisfied with a slack. We avoid this technicality in the overview.

length three *unweighted* augmenting paths. Let Unw-3-Aug-Paths be a streaming algorithm for finding such unweighted augmenting paths. We first initialize Unw-3-Aug-Paths with a (random) matching $M'_0$ obtained by including each edge in $M_0$ with probability $1/2$. As we explain shortly, $M'_0$ corresponds to the edges $e_2$ from the second type of augmenting paths. Then, at the arrival of an edge $\{u, v\}$, it is forwarded as an unweighted edge to Unw-3-Aug-Paths if

$$w(\{u,v\}) > \tau_u + \tau_v, \text{where } \tau_x = \begin{cases} w(M_0(x))/2 & \text{if } x \text{ is incident to an edge in } M'_0, \\ w(M_0(x)) & \text{otherwise.} \end{cases}$$

For an example of the forwarded edges for a specific $M'_0$, see the right part of Figure 6.2.

Note that the $\tau$-values are set so that any augmenting path found by Unw-3-Aug-Paths will also improve the matching in the weighted graph[3] . Indeed, suppose that Unw-3-Aug-Paths finds the length three augmenting path $\{o_1, e_2, o_2\}$ where $e_2 \in M'_0$. Let $e_1$ and $e_3$ be the other edges in $M_0$ incident to $o_1$ and $o_2$ (if they exist). Then, by the selection of the $\tau$-values, we have

$$\begin{aligned} w(o_1) + w(o_2) &> (w(e_1) + w(e_2)/2) + (w(e_2)/2) + w(e_3)) \\ &= w(e_1) + w(e_2) + w(e_3)\,, \end{aligned}$$

as required. Hence, the $\tau$-values are set so as to guarantee that the augmenting paths will improve the weighted matching if applied.

The reason for the random selection of $M'_0$ is to make sure that any such beneficial weighted augmenting path $\{e_1, o_1, e_2, o_2, e_3\}$ is present as an unweighted augmenting path $\{o_1, e_2, o_2\}$ in the graph given to Unw-3-Aug-Paths with probability at least $1/8$. This guarantees that there will be (in expectation) many length three unweighted augmenting paths corresponding to weighted augmentations (assuming the initial matching $M_0$ is no better than $(1/2)$-approximate).

This completes the high level description of our single-pass streaming algorithm except for the following omission: all unweighted augmenting paths are equally beneficial while their weighted contributions may differ drastically. This may result in a situation where Unw-3-Aug-Paths returns a constant-fraction of the unweighted augmenting paths that have little value in the weighted graph. The solution is simple: we partition $M'_0$ into weight classes by geometric grouping, run Unw-3-Aug-Paths for each weight class in parallel, and then select vertex-disjoint augmenting paths in a greedy fashion starting with the augmenting paths in the largest weight class. This ensures that many unweighted augmenting paths also translates into a significant improvement of the weighted matching. The formal and complete description of these techniques are given in Chapter 7.

### 6.1.2 Multi-pass Streaming and MPC

In our approach for single-pass streaming, it was crucial to have an algorithm (local-ratio with frozen vertex potentials) that allowed us to reduce the problem to that of finding augmenting

[3]We remark that there may be short augmentations that are beneficial in the weighted sense that are never present in the graph forwarded to Unw-3-Aug-Paths regardless of the choice of $M'_0$. An example would be $\{e_1, o_1, e_2, o_2, e_3\}$ with $w(e_1) = w(e_2) = w(e_3) = 10$ and $w(o_1) = 20, w(o_2) = 14$. In this case, $o_2$ is not forwarded to Unw-3-Aug-Paths due to the filtering if $e_2 \in M'_0, e_1, e_3 \notin M'_0$; and, in the other choices of $M'_0$, $\{o_1, e_2, o_2\}$ is not a length three unweighted augmenting path. However, as we prove in Chapter 7, those augmentations are safe to ignore in our goal to beat the approximation guarantee of $1/2$.

paths to a matching $M_0$ that is already (close) to $1/2$-approximate. This is because, in a single-pass streaming setting, we can find a limited amount of augmenting paths leading to a limited improvement over the initial matching.

In multi-pass streaming and MPC, the setting is somewhat different. On the one hand, the above difficulty disappears because we can repeatedly find augmentations. In fact, we can even start with the empty matching. On the other hand, we now aim for the much stronger approximation guarantee of $(1-\varepsilon)$ for any fixed $\varepsilon > 0$. This results in a more complex filtering step as we now need to find augmenting paths and cycles of arbitrary length (depending on $\varepsilon$). We remark that the challenge of finding long augmenting cycles is one of the difficulties that appears in the weighted case where previous techniques do not apply [3, 77]. We overcome this and other challenges by giving a general reduction to the unweighted matching problem, which can be informally stated as follows:

> Let $M$ be the current matching and $M^*$ be an optimal matching of maximum weight. If $w(M) < (1-\varepsilon)w(M^*)$ then an $(1-\delta(\varepsilon))$-approximation algorithm for the unweighted matching problem on bipartite graphs can be used to find a collection of vertex-disjoint augmentations that in expectation increases the weight of $M$ by $\Omega_\varepsilon(w(M^*))$.

The reduction itself is efficient and can easily be implemented both in the multi-pass streaming and MPC models by incurring only a constant overhead in the complexity. Using the best-known approximation algorithms for the unweighted matching problem on bipartite graphs in these models then yields Theorem 6.2 by repeating the above $f(\varepsilon)$ times after starting with the empty matching $M = \emptyset$.

We now present the main ideas of our reduction (the formal proof is given in Chapter 8). We start with a structural statement for weighted matchings similar to Observation 6.3:

> Suppose the current matching $M$ satisfies $w(M) \leq (1-\varepsilon)w(M^*)$. Then there must exist a collection $\mathcal{C}$ of short (each consisting of $O(1/\varepsilon)$ edges) vertex-disjoint augmenting paths and cycles with total gain $\Omega(\varepsilon^2) \cdot w(M^*)$. Moreover, each augmentation $C \in \mathcal{C}$ has gain at least $\Omega(\varepsilon^2 w(C))$, i.e., proportional to its total weight.

Our goal now is to find a large fraction of these short weighted augmentations. For this, we first reduce the problem to that of finding such augmentations $C$ with $w(C) \approx W$ for some fixed $W$. This is similar to the concept of weight classes mentioned in the previous section and corresponds to the notion of augmentation classes in Chapter 8. Note that, by standard geometric grouping, we can reduce the number of choices of $W$ to be at most logarithmic. We can thus afford to run our algorithm for all choices of $W$ in parallel and then greedily select the augmentations starting with those of the highest weight augmentation class.

Now, for each augmentation class (i.e., for each choice of $W$), we give a reduction from finding weighted augmentations to finding unweighted ones by constructing a set of tailored graphs. This construction resembles some of the ideas used in the construction of [77], but they are not the same. The intuition behind our construction is as follows. Suppose that, for a fixed $W$, we aim to find augmenting paths of length $2k+1$ in the input graph $G = (V, E)$. Then, as depicted

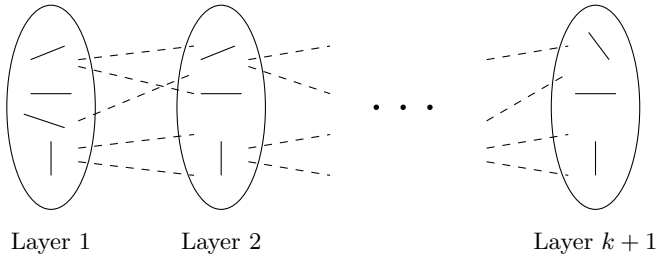



Figure 6.3 – The layered graph consisting of $k+1$ layers. The solid edges inside the layers are subsets of $M$ and the dashed edges between layers are subsets of $E \setminus M$.

in Figure 6.3, we construct a new layered graph $\mathcal{L}$ consisting of $k+1$ layers of vertices, (each layer is a copy of $V$), where the edge set of each layer consists of a subset of the edges in the current matching $M$ and the edges between layers are subsets of $E \setminus M$. The construction of $\mathcal{L}$ is so that if we consider an alternating path $C = (e_1, o_1, e_2, o_2, \ldots, e_k, o_k, e_{k+1})$ in $\mathcal{L}$ where $e_i \in M$ is an edge in layer $i$ and $o_i$ is an edge between layer $i$ and $i+1$, then, assuming they all correspond to distinct edges in $G$, we can augment $M$ with $C$ to obtain the new matching $M \Delta C$. Moreover, the augmentation improves the matching, i.e., satisfies $w(M \Delta C) > w(M)$, if

$$\sum_{i=1}^{k} w(o_i) > \sum_{i=1}^{k+1} w(e_i)\,. \tag{6.1}$$

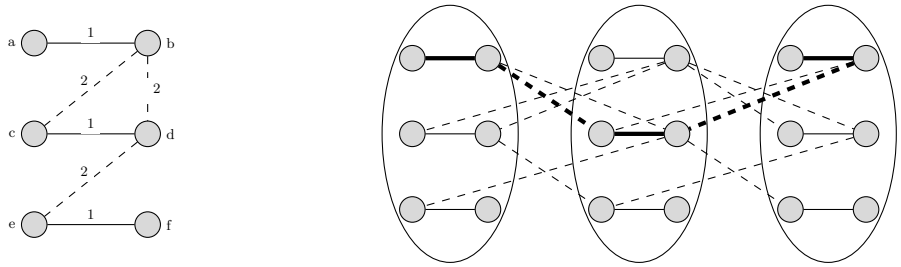
To ensure that any alternating path in the unweighted graph $\mathcal{L}$ satisfies (6.1) we use our filtering technique. For each layer $i = 1, \ldots, k+1$, we have a parameter $\tau_i^A$ that filters the edges in that layer: we keep an edge $e \in M$ in layer $i$ only if $w(e)$ rounded up to the closest multiple of $\varepsilon^{12}W$ equals $\tau_i^A W$. Similarly, we have a parameter $\tau_i^B$ for each $i = 1, \ldots, k$, and we keep an edge $e \in E \setminus M$ between layer $i$ and $i+1$ only if $w(e)$ rounded down to the closest multiple of $\varepsilon^{12}W$ equals $\tau_i^B W$. Now by considering only those $\tau$-values satisfying $\sum \tau_i^B > \sum \tau_i^A$, we ensure that any augmenting path that is found improves the matching, i.e., (6.1) holds. Moreover, the rounding of edge-weights in the filtering step still keeps large (by weight) fraction of the augmentations in the original graph as the rounding error, which is less than $\varepsilon^{12}W$ for each edge, is very small compared to the length and total gain of the structural augmentations that we are looking for. It is thus enough to find the augmentations corresponding to each fixation of $k$ and $\tau$-values. To bound the number of choices, note that we may assume that each $\tau$-value is such that $\tau \cdot W$ is a multiple of $\varepsilon^{12}W$ between 0 and $W$. Hence, as we need to consider augmentations of length $O(1/\varepsilon)$ only, we have, for a fixed $\varepsilon > 0$ and $W$, that the total number of choices of $k$ and $\tau$-values is a constant. They can thus all be considered in parallel. For each of these choices, we use the approximation algorithm for unweighted matchings to find a $(1 - \delta(\varepsilon))$-approximate maximum unweighted matching in the corresponding layered graph and take the symmetric difference with the initial matched edges to find the desired unweighted augmentations. These augmentations are then translated back to weighted augmentations in the original graph.

Note that, unlike McGregor's layered graphs, our layered graphs allow edges (both matched and unmatched) to be repeated in different layers, which is crucial in identifying weighted augmenting cycles. Furthermore, edges in each layer are filtered with respect to a given edge-weight arrangement, that ensures that the augmenting paths in our layered graphs correspond to

weighted augmentations with positive gain. These differences result from the different purposes of the two constructions: McGregor's construction aims to find unweighted augmenting paths efficiently, whereas our purpose is to reduce weighted augmentations to unweighted ones.

While, on a high level, this completes the description of our reduction, there are many interesting technical challenges to overcome. In the remaining part of this overview, we highlight two of these challenges.

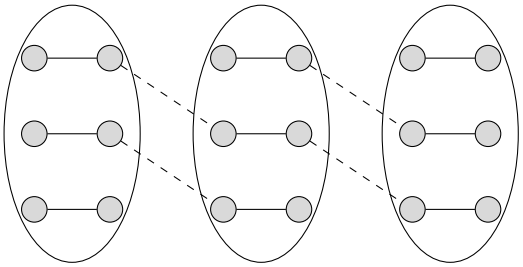
**Translating augmenting paths in layered graph to the original graph** From our high level description of the layered graph $\mathcal{L}$, there is no guarantee that an augmenting path in it corresponds to an augmentation with a positive gain in the original graph $G$. First, there is no reason that an augmenting path in $\mathcal{L}$ visits the layers from left-to-right as intended. In the formal definition of layered graphs (see Section 8.3), we take care of this and make sure[4] that any unweighted augmenting path in $\mathcal{L}$ corresponds to an alternating path of the form $(e_1, o_1, e_2, o_2, \ldots, e_k, o_k, e_{k+1})$, where $e_i \in M$ is an edge in layer $i$ and $o_i$ is an edge between layer $i$ and $i+1$. Intuitively, such an alternating path can be made an unweighted augmenting path by discarding the matching edges of the first and last layers. However, a second and more challenging issue is that such an alternating path (going from the left to the right layer) may contain repeated edges and thus do not correspond to an augmentation in $G$. An example of this phenomena is as follows:




Here, we depict the weighted graph on the left and the "incorrect" layered graph to the right with $\tau_1^A W = \tau_2^A W = \tau_3^A W = 1$ and $\tau_B^1 W = \tau_B^2 W = 2$. The weighted graph has an augmentation that adds $\{b, c\}, \{d, e\}$ and removes $\{a, b\}, \{c, d\}, \{e, f\}$ and improves the weight of the matching by one. This augmentation is also present in the layered graph. However, an equally good augmentation in that graph from an unweighted perspective corresponds to the alternating path depicted in bold. In the original graph the bold edge set corresponds to the non-simple path $a-b-c-d-b-a$. Such a non-simple path clearly does not correspond to an augmentation and, even worse, there is no augmentation with a positive gain in the support $\{a, b\}, \{b, c\}, \{c, d\}, \{d, b\}$ of the considered path.

Our main idea to overcome this issue is as follows. We first select a random bipartition $L$ and $R$ of the vertex set of $G$. Then between two layers $i$ and $i+1$, we keep only those edges that go from an $R$-vertex in layer $i$ to an $L$-vertex in layer $i+1$. We emphasize that the edges going from an L-vertex to an R-vertex between two layers are not kept. For example, if we let $L = \{a, c, e\}$ and $R = \{b, d, f\}$ in the considered example then the layered graph (with the same $\tau$-values) becomes:

[4]To be completely accurate, the edges $e_1$ and $e_{k+1}$ may not appear in the alternating path: $e_1$ does not appear if the vertex incident to $o_1$ in the first layer is not incident to a filtered edge in $M$; the case of $e_{k+1}$ is analogous.



In this example, the remaining alternating path that visits all layers (in the formal proof we further refine the layered graph to make sure that these are the only paths that are considered) corresponds to the augmentation in $G$. However, in general, an alternating path may still not correspond to a simple path and an augmentation in $G$ since it may contain repetitions. However, the bipartition and the refinement of the layered graph can be seen to introduce an "orientation" of the edges in $G$. This together with standard Eulerian techniques of directed graphs allow us to prove that any alternating path in the layered graph can be decomposed into a collection of alternating even-length cycles and an alternating path in $G$, one of which is also augmenting. Finally, let us remark that the idea to consider a bipartition $L$ and $R$ of the vertex set of $G$ and to allow only those edges that are from an $R$-vertex to an $L$-vertex between consecutive layers has the additional benefit that the layered graph becomes bipartite. This is the reason that our reduction is from weighted matchings in general graphs to unweighted matchings in bipartite graphs.

**Finding augmenting cycles** In the unweighted setting, matching algorithms do not have to consider cycles because alternating cycles cannot augment an existing matching. In contrast, algorithms for the weighted setting (at least the ones that try to iteratively improve an initial matching) have to somehow deal with augmenting cycles; weighted graphs can have perfect (unweighted) matchings whose weights are not close to the optimal and that can be improved only through augmenting cycles. For example, consider a 4-cycle with edge weights $(3, 4, 3, 4)$, where the edges of weight 3 form an initial perfect matching of weight 6, but the optimal matching consists of edges of weight 4 and has a total weight of 8. The only way to augment the weight here is to consider the whole cycle. The crucial property of our reduction is its ability to transform not only weighted augmenting paths, but also weighted augmenting *cycles* of the original graph into augmenting paths in the layered graphs.

Before explaining our solution, let us take a closer look at the above 4-cycle example. Let the edges of the 4-cycle be $(e_1, o_1, e_2, o_2)$ where $\{e_1, e_2\}$ is the current matching. Note that the cycle can be represented as an alternating path $(e_1, o_1, e_2, o_2, e_1)$ in the layered graph using three layers (consisting of the three edges of the matching with $e_1$ repeated once). However, such a representation of the augmenting cycle cannot be captured by our filtering technique due to the constraint $\sum_i \tau_i^B > \sum_i \tau_i^A$ which ensures that any alternating path in the layered graph can be translated into a weighted augmentation. The reason being that for $(e_1, o_1, e_2, o_2, e_1)$ to be present in the layered graph we would need $\tau_i^A W = 3$ for $i = 1, 2, 3$, and $\tau_i^B W = 4$ for $i = 1, 2$ which would contradict the above inequality. This approach is therefore not sufficient to find augmenting cycles and achieve a $(1 - \varepsilon)$ approximation guarantee. Specifically, the issue is due to the fact that we account for the edge weight of $e_1$ twice in the filtering process, once for $o_1$ and once more for $o_2$. To overcome this issue, consider the 4-cycle with more general weights

$2, 2+\varepsilon, 2, 2+\varepsilon$, where taking $o_1, o_2$ in place of $e_1, e_2$ gives an $\varepsilon/2$ fractional gain in weight. What we need is to make sure that, even if we account for the same edge $e_1$ (or $e_2$) twice, the alternating path we get in the layered graph ("corresponding" to the cycle) is still gainful. For this, we blow-up the cycle length by repeating the same cycle $O(1/\varepsilon)$ times. I.e., we consider the cycle

$$\underbrace{e_1, o_1, e_2, o_2}_{\text{instance 1}}, \underbrace{e_1, o_1, e_2, o_2}_{\text{instance 2}}, \ldots, \underbrace{e_1, o_1, e_2, o_2}_{\text{instance } c/\varepsilon}, e_1.$$

Since we have repeated the $o_i$ edges many times, their gains add up so that it can account for the weight of considering $e_1$ one additional time. The considered cycle of length 4 is thus present as a "repeated" alternating path in the layered graph (with the appropriate $\tau$-values and bipartition) consisting of $O(1/\varepsilon)$ layers. In general, to make sure that we can find augmenting cycles of length $O(1/\varepsilon)$ we will consider the layered graph with up to $O(1/\varepsilon^2)$ layers.

In Chapter 8, we elaborate on these techniques and prove Theorem 6.2.

## 6.2 Further Related Work

There is a large body of work devoted to (semi-)streaming algorithms for the maximum matching problem. For unweighted graphs, the basic greedy approach yields a $(1/2)$-approximation, and for weighted graphs [83] recently gave a $(1/2-\varepsilon)$-approximation based on the local ratio technique. These are the best known algorithms that take a single pass over an adversarially ordered stream. Better algorithms are known if the stream is randomly ordered or if the algorithm can take multiple passes through the stream. In the random-edge-arrival case, [68] first improved upon the approximation guarantee of $1/2$ in the unweighted case. Our results give better guarantees in that setting and also applies to the weighted setting. When considering multi-pass algorithms, [77] gave a $(1-\varepsilon)$-approximation algorithm using $(1/\varepsilon)^{O(1/\varepsilon)}$ passes. Complementing this, [3] gave a deterministic $(1-\varepsilon)$-approximation algorithm using $O(\log(n)\operatorname{poly}(1/\varepsilon))$ passes. Also, a $(1-\varepsilon)$-approximation can be obtained in $p/\varepsilon$ passes and $O(n^{1+1/p})$ space [4]; setting $p = \log n/\log\log n$ gives a (semi-)streaming algorithm that uses $\log n/(\varepsilon \log\log n)$ passes. As for hardness results, [60] showed that no algorithm can achieve a better approximation guarantee than $(1-1/e)$ in the adversarial single pass streaming setting.

A simple and general technique of reducing weighted to unweighted was given in [26] to obtain both streaming and MPC algorithms. The reduction is non-adaptive (instead of iterative) and worsens the approximation by a factor of $1/2-\varepsilon$, i.e., using an $\alpha$-approximation algorithm for the unweighted case, they give a $(1/2-\varepsilon)\alpha$-approximation for the weighted case.

The study of algorithms for matchings in models of parallel computation dates back to the eighties. A seminal work of Luby [74] shows how to construct a maximal independent set in $O(\log n)$ PRAM rounds. When this algorithm is applied to the line graph of $G$, it outputs a maximal matching of $G$. Similar results, also in the context of PRAM, were obtained in [5, 56, 57].

Perfect maximum matchings were also a subject of study in the context of PRAM. In [73] it is shown that the decision variant is in RNC. That implies that there is a PRAM algorithm that in $\operatorname{poly}\log n$ rounds decides whether a graph has a perfect matching or not. [63] were the first to prove that constructing perfect matchings is also in RNC. In [81] the same result was proved, and they also introduced the isolation lemma that had a great impact on many other problems.

In [47, 62] it was shown that it is often possible to simulate one PRAM in $O(1)$ MPC rounds with $O(n^\alpha)$ memory per machine, for any constant $\alpha > 0$. This implies that the aforementioned PRAM results lead to $O(\log n)$ MPC round complexity algorithms for computing maximal matchings. [70] developed an algorithm that computes maximal matchings in the MPC model in $O(1/\delta)$ rounds when the memory per machine is $\Omega(n^{1+\delta})$, for any constant $\delta > 0$. In the regime of $\tilde{O}(n)$ memory per machine, the algorithm given in [70] requires $\tilde{O}(\log n)$ MPC rounds of computation. Another line of work focused on improving this round complexity. Namely, [27] and [9, 45] show how to compute a constant-factor approximation of maximum unweighted matching in $O((\log \log n)^2)$ and $O(\log \log n)$ MPC rounds, respectively, when the memory per machine is $\tilde{O}(n)$. As noted in [27], any $\Theta(1)$-approximation algorithm for maximum unweighted matchings can be turned into a $(1/2 - \varepsilon)$-approximation algorithm for weighted matchings by using the approach described in Section 4 of [72]. This transformation increases the round complexity by $O(1/\varepsilon)$.

In the regime of $n^\delta$ memory per machine, for any constant $\delta \in (0, 1)$, a recent work [17] shows how to find maximal matchings in $O((\log \log n)^2)$ rounds for graphs of arboricity $\text{poly}(\log n)$. Also in this regime, [43] and [82] provide algorithms for constructing maximal matchings for general graphs in $\tilde{O}(\sqrt{\log n})$ MPC rounds. The algorithm of [43] requires $O(m)$ and the algorithm of [82] requires $O(m + n^{1+o(1)})$ total memory.

# 7 Uniformly Random Order Edge Streams

In this chapter, we present a $(1/2+c)$-approximation (semi-)streaming algorithm for the maximum weighted matching (MWM) problem in the random-edge-arrival setting, where $c > 0$ is an absolute constant, thus proving Theorem 6.1. Our result computes a large weighted matching using unweighted augmentations. In that spirit, we provide the following lemma that gives us the streaming algorithm for unweighted augmentations.

**Lemma 7.1.** *There exists an unweighted streaming algorithm* Unw-3-Aug-Paths *with the following properties:*

1. *The algorithm is initialized with a matching $M$ and a parameter $\beta > 0$. Afterwards, a set $E$ of edges is fed to the algorithm one edge at a time.*
2. *Given that $M \cup E$ contains at least $\beta|M|$ vertex disjoint 3-augmenting paths, the algorithm returns a set* Aug *of at least $(\beta^2/32)|M|$ vertex disjoint 3-augmenting paths. The algorithm uses space $O(|M|)$.*

*Proof.* Since this proof is based completely on the ideas of Kale and Tirodkar [59], we give it in the appendix for completeness. See Appendix B. □

We mentioned in the introduction that, for an effective weighted-to-unweighted reduction in the streaming model, it is important to start with a "good" approximate matching so that we can augment it using 3-augmentations afterwards. We demonstrate these ideas on unweighted matchings first (Section 7.1), and show that they lead to an improved approximation ratio for both general and bipartite graphs. Later, in Section 7.2, we study these ideas in the context of weighted matchings.

## 7.1 Demonstration of Technique via Unweighted Matching

We give an algorithm that makes one pass over a uniformly random edge stream of a graph and computes a 0.506-approximate maximum unweighted matching. For the special case of triangle-free graphs (which includes bipartite graphs), we give a better analysis to get a 0.512-approximation.

We denote the input graph by $G = (V, E)$, and use $M^*$ to indicate a matching of maximum cardinality. Assume that $M^*$ and a maximal matching $M'$ are given. For $i \in \{3, 5, 7, \ldots\}$, a connected component of $M' \cup M^*$ that is a path of length $i$ is called an $i$-augmenting path (the

component is called nonaugmenting otherwise). We say that an edge in $M'$ is 3-augmentable if it belongs to a 3-augmenting path, otherwise we say that it is non-3-augmentable. Also, for a vertex $u$, let $N(u)$ be $u$'s neighbor set, and for $S \subseteq E$, let $N_S(U)$ denote $u$'s neighbor set in the edges in the graph $(V, S)$.

In the analysis, we use the following lemma. For the sake of completeness, we reprove it in Appendix B.

**Lemma 7.2** (Lemma 1 in [68])**.** *Let $\alpha \geq 0$, $M'$ be a maximal matching in $G$, and $M^*$ be a maximum unweighted matching in $G$ such that $|M'| \leq (1/2 + \alpha)|M^*|$. Then the number of 3-augmentable edges in $M'$ is at least $(1/2 - 3\alpha)|M^*|$, and the number of non-3-augmentable edges in $M'$ is at most $4\alpha|M^*|$.*

The algorithm is as follows. Compute a maximal matching $M_0$ on initial $p$ (which we will set later) fraction of the stream. Then we run three algorithms in parallel on the remaining $(1-p)$ fraction of the stream. In the first, we store all the edges into the variable $S_1$ that are among vertices left unmatched by $M_0$. In the end, we augment $M_0$ by adding a maximum unweighted matching in $S_1$. In the second, we continue to grow $M_0$ greedily to get $M'$. In the third, to get 3-augmentations with respect to $M_0$, we invoke the Unw-3-Aug-Paths algorithm from Lemma 7.1 that accepts a matching $\tilde{M}$ and a stream of edges that contains $\beta$ augmenting paths of length 3 with respect to $\tilde{M}$. In this way we obtain a set of vertex disjoint 3-augmenting paths, which we then use to augment $M_0$. We return the best of the three algorithms.

It is clear that the second and the third algorithm use $O(n \log n)$ space. The following lemma shows that the first algorithm uses $O(n/p \cdot \log n)$ space.

**Lemma 7.3.** *With high probability it holds that $|S_1| \in O(n/p \cdot \log n)$.*

*Proof.* Fix a vertex $v$. Define $A_{v,t}$ to be the event that after processing $t$ edges from the stream it holds: $v$ is unmatched, and at least $5\frac{\log n}{p}$ neighbors of $v$ are still unmatched. We will show that $\Pr[A_{v,pm}] \leq n^{-5}$, after which the proof follows by union bound over all the vertices. We have

$$\begin{aligned}
\Pr[A_{v,t}] &= \Pr[A_{v,t}|A_{v,t-1}] \cdot \Pr[A_{v,t-1}] + \Pr[A_{v,t}|\neg A_{v,t-1}] \cdot \Pr[\neg A_{v,t-1}] \\
&= \Pr[A_{v,t}|A_{v,t-1}] \cdot \Pr[A_{v,t-1}] \\
&\leq \Pr[v \text{ is unmatched after processing } t \text{ edges}|A_{v,t-1}] \cdot \Pr[A_{v,t-1}] \\
&\leq \left(1 - \frac{5\frac{\log n}{p}}{m-t+1}\right) \cdot \Pr[A_{v,t-1}] \\
&\leq \left(1 - \frac{5\frac{\log n}{p}}{m}\right)^t \\
&\leq e^{-\frac{5t\log n}{pm}}.
\end{aligned}$$

Therefore, $\Pr[A_{v,pm}] \leq n^{-5}$ as desired.

□

We divide the analysis of approximation ratio into two cases.

**Case 1.** $|M_0| \leq (1/2 - \alpha)|M^*|$**:**

Each edge of $M_0$ can intersect with at most two edges of $M^*$, hence $S_1$ contains at least $|M^*|-2|M_0|$ edges of $M^*$ that can be added to $M_0$ to get a matching of size at least $|M^*|-|M_0| \geq (1/2+\alpha)|M^*|$.

**Case 2.** $|M_0| \geq (1/2 - \alpha)|M^*|$**:**

If $|M_0| \geq (1/2 + \alpha)|M^*|$, we are done, so assume that $|M_0| < (1/2 + \alpha)|M^*|$. In the second algorithm, $M'$ is the maximal matching at the end of the stream. If $|M'| \geq (1/2 + \alpha)|M^*|$, we are done, otherwise, by Lemma 7.2, there are at least $(1/2 - 3\alpha)|M^*|$ 3-augmentable edges in $M'$, i.e., there are at least $(1/2 - 5\alpha)|M^*|$ 3-augmentable edges in $M_0$; denote this set of edges by $E_3$. In expectation, for at least $(1 - 2p)$ fraction of $E_3$, both the $M^*$ edges incident to them appear in the latter $(1 - p)$ fraction of the stream. This can be seen by having one indicator random variable per edge in $E_3$ denoting whether two $M^*$ edges incident on that edge appear in the latter $(1 - p)$ fraction of the stream. Then we condition on the event that $uv \in E_3$, which implies that $uv$ has two $M^*$ edges, say $au$ and $vb$, incident on it. Since $uv$ was added to the greedy matching $M_0$, both $au$ and $vb$ must appear after $uv$. Any of $au$ and $vb$ appears in the latter $(1 - p)$ fraction on the stream with probability $(1 - p)$ under this conditioning. Then, by union bound, with probability at least $(1 - 2p)$ both $au$ and $vb$ appear in the latter $(1 - p)$ fraction of the stream. Then we apply linearity of expectation over the sum of the indicator random variables.

Now, by Lemma 7.1, using $\beta = (1/2 - 5\alpha)(1 - 2p)/(1/2 + \alpha) \geq (1 - 2p)(1 - 12\alpha)$, we recover at least $(1/32)(1-2p)^2(1-12\alpha)^2|M_0| \geq (1/32)(1-4p)(1-24\alpha)|M_0|$ augmenting paths in expectation. Using $|M_0| \geq (1/2 - \alpha)|M^*|$, after algebraic simplification, we get that the output size is at least $((1/2-\alpha)+(1/64)(1-4p)(1-26\alpha))|M^*|$, i.e., at least $(1/2+1/64-90\alpha/64-p)|M^*|$. Letting $\alpha = 1/154$ implies that our algorithm outputs a $(1/2 + \alpha - p)$-approximate maximum unweighted matching, i.e., 0.506-approximation for $p \leq 0.0001$.

**Theorem 7.4.** *For random-order edge-streams, there is a one-pass $O(n \operatorname{polylog} n)$-space algorithm that computes a 0.506-approximation to maximum unweighted matching in expectation.*

**Remark.** *This algorithm not only demonstrates our technique, but also improves the current best approximation ratio of 0.503 by Konrad et al. [68]. For bipartite graphs, recently, Konrad [67] gave a 0.5395-approximation algorithm.*

## 7.2 An Algorithm for Weighted Matching

Now we discuss the more general weighted case.

Let $G = (V, E, w)$ be a weighted graph with $n$ vertices and $m$ edges, and assume that the edges in $E$ are revealed to the algorithm in a uniformly random order. We further assume that the edge weights are positive integers and the maximum edge weight is $O(\text{poly}(n))$. Let $M^*$ be a fixed maximum weighted matching in $G$. For any matching $M$ of $G$ and a vertex $v \in V$, let $M(v)$ denote the edge adjacent to the vertex $v$ in the matching $M$. If some vertex $v$ is unmatched in $M$, we assume that $v$ is connected to some artificial vertex with a zero-weight edge, whenever we use the notation $M(v)$.

Similarly to the algorithm in Section 7.1, we start by computing a $(1/2)$-approximate maximum

weighted matching $M_0$ within the first $p$ fraction of the edges ($p = O(1/\log n)$) using the local-ratio technique. We recall this technique next. We consider each incoming edge $e = (u, v)$, and as long as it has a positive weight, we push it into a stack and subtract its weight from each of the remaining edges incident to any of its endpoints $u$ and $v$. To implement this approach in the streaming setting, for each vertex $v \in V$, we maintain a vertex potential $\alpha_v$. The potential $\alpha_v$ tells how much weight should be subtracted from each incoming edge that is incident to $v$. After running the local-ratio algorithm for the first $p$ fraction of the edges, computing $M_0$ greedily by popping the edges from the stack gives a $(1/2)$-approximate matching $M_0$ for that portion of the stream. This is proved using local-ratio theorem (see the work of Paz and Schwartzman [83]). We also freeze the vertex potentials $\alpha_v$ at this point.

Analogous to the unweighted case, we have three possible scenarios for $M_0$:

1. In the best case, $w(M_0) \geq (1/2 + 4c) \cdot w(M^*)$ and we are done.
2. The weight $w(M_0) \leq (1/2 - 4c) \cdot w(M^*)$, in which case we have only seen at most $(1 - 8c) \cdot w(M^*)$ worth optimal matching edges so far, and the rest of the stream contains at least $8c \cdot w(M^*)$ weight that can be added *on top of* $M_0$.

   This corresponds to having a large fraction of unmatched vertices in the unweighted case, where we could afford to store all the edges incident to those vertices and compute a maximum unweighted matching that did not conflict with $M_0$. In the weighted case, we keep all edges $e = (u, v)$ in the second part of the stream that satisfy $w(e) > \alpha_u + \alpha_v$, where $\alpha_u$ and $\alpha_v$ are the frozen vertex potentials after seeing the first $p$ fraction of the edges. Note that we continue to keep the vertex potential frozen. (Think of the unmatched vertices in the unweighted case as vertices with zero potential.) Again using the random-edge-arrival property, we show that the number of such edges that we will have to store is small with high probability. At the end of the stream, we use an (exact) maximum matching on those edges together with the edges in the local-ratio stack from the first $p$ fraction of the stream to construct a $(1/2 + 4c) \cdot w(M^*)$ matching.
3. The weight of the matching $M_0$ is between $(1/2 - 4c) \cdot w(M^*)$ and $(1/2 + 4c) \cdot w(M^*)$. In the analogous unweighted case, we did two things. We continued to maintain a greedy matching (on unmatched vertices), and we tried to find augmenting paths of length three. For the weighted case we proceed similarly: We continue to compute a constant factor approximate matching for those edges $e = (u, v)$ such that $w(e) > w(M_0(u)) + w(M_0(v))$, and akin to the unweighted 3-augmentations, we try to find the weighted 3-augmentations.

   For the latter task, we randomly choose (guess) a set of edges from $M_0$ that we consider as the *middle* edges of weighted 3-augmentations. Here, by a weighted 3-augmentation, we mean a quintuple of edges $(e_1, o_1, e_2, o_2, e_3)$ that increase the weight of the matching when the edges $e_1, e_2$, and $e_3$ are removed from $M_0$, and the edges $o_1$ and $o_2$ are added to $M_0$. (Although these are length five augmenting paths, we call them 3-augmentations because we reduce the problem of finding those to the problem of finding length three unweighted augmenting paths.) We partition the chosen *middle* edges into weight classes defined in terms of geometrically increasing weights, and for each of the weight classes we find 3-augmentations using an algorithm that finds unweighted 3-augmenting paths as a black-box.

Before we proceed to the complete algorithm, we give an algorithm to address the third case

described above. In fact, this algorithm which improves weighted matchings via unweighted augmentations is the key contribution of this chapter.

### Finding Weighted Augmenting Paths

Suppose that we have an initial matching $M_0$ such that $(1/2 - 4c) \cdot w(M^*) \leq w(M_0) \leq (1/2 + 4c) \cdot w(M^*)$. In this section, we describe how to augment $M_0$ using 3-augmentations to get an increase of weight $8c \cdot w(M^*)$ that results in a matching of weight at least $(1/2 + 4c) \cdot w(M^*)$. To achieve this, in a black-box manner we use the algorithm Unw-3-Aug-Paths whose existence is guaranteed by Lemma 7.1.

Let $W_i = \{e \in E : 2^{i-1} \leq w(e) < 2^i\}$ be the set of edges whose weight is in the range $[2^{i-1}, 2^i)$, and let $k$ be the index such that $\max_{e \in E} w(e) \in W_k$. Thus $k = O(\log n)$ (recall that the edge weights are positive integers and the maximum edge weight is $O(\text{poly}(n))$, and any edge $e \in E$ belongs to exactly one $W_i$). We refer to $W_i$'s as weight classes.

**Algorithm 7.1:** Outline of the algorithm Wgt-Aug-Paths.

**Global:** Instances $\mathcal{A}_i$ of Unw-3-Aug-Paths for $i = 1, 2, \ldots k$, a matching $M_0$, a set `Marked` of *marked* edges, and a $(1/4)$-approximate streaming algorithm for weighted matching algorithm Approx-Wgt-Matching.

**function** *Initialize*(*A matching* $M$)
- Set $M_0 = M$.
- For each $e \in M_0$, with probability $1/2$, add $e$ to `Marked`.
- **for** $i = 1$ **to** $k$ **do**
  - Initialize $\mathcal{A}_i$ with the matching in `Marked` $\cap\, W_i$.

**function** *Feed-Edge*(*An edge* $e = (u, v) \in E$)
- **if** $w(e) \geq w(M_0(u)) + w(M_0(v))$ **then**
  - Feed $e$ to Approx-Wgt-Matching with weight $w'(e) = w(e) - (w(M_0(u)) + w(M_0(v)))$.
- **if** $w(e) \leq (1 + \alpha)(w(M_0(u)) + w(M_0(v)))$ **then**
  - **if** $M_0(u) \in$ `Marked` **and** $M_0(v) \notin$ `Marked` **then**
    - **if** $w(e) \geq (1 + 2\alpha)((1/2) \cdot w(M_0(u)) + w(M_0(v))$ **then**
      - Feed $e$ into $\mathcal{A}_i$ where $i$ is such that $w(e) \in W_i$.
  - **if** $M_0(v) \in$ `Marked` **and** $M_0(u) \notin$ `Marked` **then**
    - **if** $w(e) \geq (1 + 2\alpha)(w(M_0(u)) + (1/2) \cdot w(M_0(v))$ **then**
      - Feed $e$ into $\mathcal{A}_i$ where $i$ is such that $w(e) \in W_i$.

**function** *Finalize*()
- Let $M'$ be the matching computed by Approx-Wgt-Matching.
- Let $M_1$ be the matching obtained by adding edges in $M'$ to $M_0$ and removing the conflicting edges from $M_o$.
- Let $M_2$ be the matching obtained by greedily doing the non-conflicting augmentations returned by $\mathcal{A}_i$ for $i = k, k-1, \ldots, 1$ in that order on the initial matching $M_0$.
- **return** $\arg\max_{i \in [2]} w(M_i)$

As described earlier, we would like to find both weighted 1-augmentations (i.e., single edges that could replace two incident edges in the current matching and give a significant gain in weight),

and weighted 3-augmentations. We now give the outline of our algorithm, Wgt-Aug-Paths, in Algorithm 7.1 using the object-oriented notation, and we explain its usage and intuition behind its design below.

**Initialization:** We initialize Wgt-Aug-Paths by calling the Initialize function, passing the initial matching $M_0$, which is the matching we compute after seeing the first $p$ fraction of the edges in our final algorithm. Given $M_0$, the algorithm will first independently and randomly sample a set of edges `Marked`; these are the edges that the algorithm guesses to be the *middle* edges of 3-augmentations. The algorithm will later look for pairs of edges $(o_i, o_{i+1})$ such that $(e_i, o_i, e_{i+1}, o_{i+1}, e_{i+2})$ is a weighted 3-augmentation, where $e_{i+1}$ is a guessed middle edge whereas $e_i$ and $e_{i+2}$ are not. We aim to gain at least some constant ($\alpha$ in Algorithm 7.1) fraction of the weight of the middle edge by doing the augmentations. To achieve this, we group all guessed middle edges into weight classes and use dedicated instances of Unw-3-Aug-Paths for each weight class.

**Processing the edge stream:** Next, a stream of edges (the rest of the stream) is fed to the algorithm using the function Feed-Edge. The function Feed-Edge does two things. For an edge $e = (u, v)$ that has excess weight $w'(e) = w(e) - w(M_0(u)) - w(M_0(v))$ (i.e., gain of the corresponding 1-augmentations), it tries to recover a matching with a large excess weight giving a large weight increase on top of $M_0$. On the other hand, if we do not have large matching with respect to the excess weights, then it implies that there must be a large fraction of 3-augmentations *by weight*. Thus the function Feed-Edge also looks for 3-augmentations using Unw-3-Aug-Paths as a black-box. After filtering out the edges with small excess weight, it appropriately feeds them to the Unw-3-Aug-Paths instance of the correct weight class. The filtering is needed to ensure that for each weight class, the number of 3-augmentations is large compared to the number of guessed middle edges in that weight class (which is what $\beta$ refers to in Lemma 7.1 that gives Unw-3-Aug-Paths).

**Finalizing the matching:** Finally at the end of the stream, we call the Finalize function, which uses the initial matching together with the approximate maximum matching on excess weights and the outputs of the Unw-3-Aug-Paths instances to construct the final matching.

### Analysis of the algorithm

Assume that Wgt-Aug-Paths is initialized with a matching $M_0$, and further assume that $(1/2 - 4c) \cdot w(M^*) \leq w(M_0) \leq (1/2 + 4c) \cdot w(M^*)$ for some $0 < c < 2^{-15}$ (we will set the exact value of $c$ later). Let $M^*$ be a fixed optimal weighted matching in $G$ and let $\tilde{E} \subset E$ be a subset of edges such that $w(M^* \cap \tilde{E}) \geq (1 - 0.001) \cdot w(M^*)$ (think of $\tilde{E}$ as the edges in the second part of the stream). Let $\tilde{M}^*$ be the maximum weighted matching in $\tilde{E}$. By the previous assumption, we have that $w(\tilde{M}^*) \geq (1 - 0.001) \cdot w(M^*)$. Assume that after the initialization, we feed the edges of $\tilde{E}$ one at a time to Wgt-Aug-Paths in some arbitrary order (not necessarily random).

Let $\hat{M}$ be the matching returned by the function Finalize. We show that, under the above assumptions, the expected weight $\mathbb{E}[w(\hat{M})]$ of the matching $\hat{M}$ is at least $(1/2 + 4c) \cdot w(M^*)$.

Recall that in Wgt-Aug-Paths, the output $\hat{M}$ is the maximum of two matchings $M_1$ and $M_2$. The matching $M_1$ is constructed by combining the output $M'$ of the $(1/4)$-approximate algorithm Approx-Wgt-Matching on the excess weights $w'$ with the initial matching $M_0$. That is, $M_1$ is

obtained by adding all edges of $M'$ to $M_0$ and removing the edges that conflict with those newly added edges from $M_0$. The matching $M_2$ is formed by applying the 3-augmentations given by Unw-3-Aug-Paths instances to the initial matching $M_0$.

In the construction of $M_1$, when we add an edge $e = (u, v) \in M'$ to $M_0$ and remove the two conflicting edges, the gain of weight is $w(e) - (w(M_0(u)) + w(M_0(v)) = w'(e)$. Thus we have $w(M_1) \geq w(M_0) + \sum_{e \in M'} w'(e)$, and since $\hat{M}$ is the maximum of $M_1$ and $M_2$, we have the following observation.

**Observation 7.5.** *If the weight of the matching $M'$ computed by Approx-Wgt-Matching for the excess weights $w'(e)$ is at least $2^{-12}w(M^*)$, then*

$$w(\hat{M}) \geq w(M_1) \geq w(M_0)+(2^{-12})\cdot w(M^*) \geq (1/2-4c)\cdot w(M^*)+(2^{-12})\cdot w(M^*) \geq (1/2+4c)\cdot w(M^*).$$

*For the last two inequalities we use the facts that $w(M_0) \geq (1/2 - 4c) \cdot w(M^*)$ and $c < 2^{-15}$.*

In light of Observation 7.5, we now assume that the approximate maximum matching in $\tilde{E}$ with respect to the excess weights is small. For this case, we show that the matching $M_2$ has at least $(1/2 + 4c) \cdot w(M^*)$ weight in expectation.

Let $E_1$ be the edges in $\tilde{E}$ that satisfy the criteria of Line 9, namely edges $e = (u, v) \in \tilde{E}$ such that $w(e) \leq (1 + \alpha)(w(M_0(u)) + w(M_0(v)))$. These are the edges that have small excess weight. We have the following lemma on the 3-augmentations that only use edges with small excess weight.

**Lemma 7.6.** *If the weight of the approximate maximum matching $M'$ with respect to excess weights $w'$ is at most $(2^{-12}) \cdot w(M^*)$, then there exist a set of 3-augmentations that only use edges in $E_1$ such that the total weight increase of those augmentations is at least $(0.4) \cdot w(M^*)$.*

*Proof.* Consider the symmetric difference $\tilde{M}^* \triangle M_0$ as a collection of cycles that alternate between $M^*$ and $M_0$ edges. Recall that $\tilde{M}^*$ is the maximum matching in $\tilde{E}$, and assume that both $\tilde{M}^*$ and $M_0$ are perfect matchings (with zero-weight edges between unmatched vertices).

Without loss of generality, we assume that it is a single cycle of length $2n$ (for the case of multiple cycles, the following proof can be easily modified to take the summations over all cycles and we can replace $n$ with the actual cycle length). Label the edges in the cycle as $e_1, o_1, e_2, o_2, \ldots, e_n, o_n$ (assume that the indices wrap around so that $e_{n+i} = e_i$ and $o_{n+i} = o_i$) so that the $e$-edges belong to $M_0$ and $o$-edges belong to $\tilde{M}^*$.

Let $P_i$ denote the quintuple $(e_i, o_i, e_{i+1}, o_{i+1}, e_{i+2})$ of edges, and let $g(P_i)$ denote the gain $w(o_i) + w(o_{i+1}) - w(e_i) - w(e_{i+1}) - w(e_{i+2})$ we get by augmenting $P_i$ (i.e., by removing edges $e_i, e_{i+1}, e_{i+2}$ from $M_0$ and adding edges $o_i, o_{i+1}$ to $M_0$). We have that

$$\begin{aligned}\sum_{i\in[n]} g(P_i) &= 2\sum_{i\in[n]} w(o_i) - 3\sum_{i\in[n]} w(e_i) \geq 2(1-0.001)\cdot w(M^*) - 3\cdot w(M_0)\\ &\geq (2(1-0.001) - 3(1/2+4c))\cdot w(M^*) \geq (1/2-0.003)\cdot w(M^*),\end{aligned}$$

where the inequality follows from the assumption that $w(M_0) \leq (1/2 + 4c) \cdot w(M^*)$.

Now let $L$ be the set of indices $i$ for which either $w(o_i) \geq (1 + \alpha)(w(e_i) + w(e_{i+1}))$ or $w(o_{i+1}) \geq (1 + \alpha)(w(e_{i+1}) + w(e_{i+2}))$. Thus we have that, $\sum_{i\in[n]} g(P_i) = \sum_{i\in[n]\setminus L} g(P_i) + \sum_{i\in L} g(P_i)$.

Furthermore, we have

$$\begin{aligned}
&\sum_{i\in L}(g(P_i) - w(e_{i+1})) \\
&\quad= \sum_{i\in L}((w(o_i) - w(e_i) - w(e_{i+1})) + (w(o_{i+1}) - w(e_{i+1}) - w(e_{i+2}))) \\
&\quad\le \sum_{i\in L}((w(o_i) - w(e_i) - w(e_{i+1}))^+ + (w(o_{i+1}) - w(e_{i+1}) - w(e_{i+2})))^+ \\
&\quad\le \sum_{i\in [n]}((w(o_i) - w(e_i) - w(e_{i+1}))^+ + (w(o_{i+1}) - w(e_{i+1}) - w(e_{i+2})))^+ \\
&\quad= 2\sum_{i\in[n]} \underbrace{(w(o_i) - w(e_i) - w(e_{i+1}))^+}_{w'(o_i) \text{ or } 0} \\
&\quad\le 2\cdot 4\cdot (2^{-12})\cdot w(M^*) < (0.002)\cdot w(M^*).
\end{aligned}$$

The last line above follows from the fact that $M'$ is a 4-approximation with respect to the weight function $w'$, and thus any matching has weight at most $4 \cdot w'(M')$ with respect to weights $w'$. On the other hand, for any $i \in L$, by definition, either

$$w'(o_i) = w(o_i) - w(e_i) - w(e_{i+1}) \ge \alpha(w(e_i) + w(e_{i+1})) \ge \alpha w(e_{i+1})$$

or

$$w'(o_{i+1}) = w(o_{i+1}) - w(e_{i+1}) - w(e_{i+2}) \ge \alpha(w(e_{i+1}) + w(e_{i+2})) \ge \alpha w(e_{i+1}).$$

Thus $\sum_{i\in L} w(e_{i+1}) \le (1/\alpha)\sum_{i\in[n]} w'(o_i) \le 4\cdot(2^{-12})\cdot(1/\alpha)\cdot w(M^*) \le (0.05)\cdot w(M^*)$ when $\alpha = 0.02$. Putting these together, we get

$$\begin{aligned}
\sum_{i\in[n]\setminus L} g(P_i) &\ge (1/2 - 0.003)w(M^*) - \sum_{i\in L}(g(P_i) - w(e_{i+1})) - \sum_{i\in L} w(e_{i+1}) \\
&\ge (1/2 - 0.003 - 0.002 - 0.05)\cdot w(M^*) \ge (0.4)\cdot w(M^*).
\end{aligned}$$

□

Let $O_1 = [n] \setminus L$ where $L$ is defined as in the proof of Lemma 7.6 so that the augmentations $P_i$ for $i \in O_1$ only uses edges with small excess weight. (Recall that $P_i = (e_i, o_i, e_{i+1}, o_{i+1}, e_{i+2})$ where $o_i$ and $o_{i+1}$ are edges in $\tilde{M}^*$, which is a fixed optimal matching in $\tilde{E}$.) Formally,

$$O_1 = \{i \in [n] : w(o_i) \le (1+\alpha)(w(e_i) + w(e_{i+1})) \text{ and } w(o_{i+1}) \le (1+\alpha)(w(e_{i+1}) + w(e_{i+2}))\}.$$

Let $O_2 = \{i \in O_1 : g(P_i) \ge (1/2 + 3\alpha)w(e_{i+1}) + 2\alpha w(e_i) + 2\alpha w(e_{i+2})\}$. We need the bounds we show in Lemma 7.7 below for the analysis of 3-augmentations. Note that the first two parts correspond to the conditions on Lines 11 and 14. To recover sufficient number of 3-augmentations using Unw-3-Aug-Paths as a black box, each weight class that has a large fraction of augmentations by weight should also have a large fraction of them by number. This is because the guarantee of Lemma 7.1 is conditioned on the existence of many augmenting paths. For this reason, we need the upper bound on the gain of each individual augmentation as given in the third part of the lemma.

**Lemma 7.7.** *For all $i \in O_2$, we have*

1. $w(o_i) \geq (1+2\alpha)(w(e_i) + (1/2)w(e_{i+1}))$,
2. $w(o_{i+1}) \geq (1+2\alpha)((1/2)w(e_{i+1}) + w(e_{i+2}))$, *and*
3. $g(P_i) \leq 3w(e_{i+1})$.

*Proof.* For $w(o_i)$ we have

$$\begin{aligned}
w(o_i) &= g(P_i) + (w(e_i) + w(e_{i+1}) + w(e_{i+2})) - w(o_{i+1}) \\
&\geq ((1/2 + 3\alpha)w(e_{i+1}) + 2\alpha w(e_i) + 2\alpha w(e_{i+2})) \\
&\qquad + (w(e_i) + w(e_{i+1}) + w(e_{i+2})) - w(o_{i+1}) \\
&= (1+2\alpha)w(e_i) + (1+2\alpha)w(e_{i+2}) + (3/2 + 3\alpha)w(e_{i+1}) - w(o_{i+1}) \\
&\geq (1+2\alpha)w(e_i) + (1+2\alpha)w(e_{i+2}) + (3/2 + 3\alpha)w(e_{i+1}) \\
&\qquad - ((1+\alpha)(w(e_{i+1}) + w(e_{i+2}))) \\
&\geq (1+2\alpha)(w(e_i) + (1/2)w(e_{i+1})).
\end{aligned}$$

The claim on $w(o_{i+1})$ follows similarly. This proves the first two parts of the lemma.

Now, observe that we have $(1+\alpha)(w(e_i) + w(e_{i+1})) \geq w(o_i) \geq (1+2\alpha)(w(e_i) + (1/2)w(e_{i+1}))$. This implies that

$$(1+\alpha)(w(e_i) + w(e_{i+1})) \geq (1+2\alpha)(w(e_i) + (1/2)w(e_{i+1})),$$

which simplifies to $(1/2)w(e_{i+1}) \geq \alpha w(e_i)$ or equivalently $w(e_i) \leq (1/(2\alpha))w(e_{i+1})$. Similarly we can show that $w(e_{i+2}) \leq (1/(2\alpha))w(e_{i+1})$. Thus we have that

$$\begin{aligned}
g(P_i) &= w(o_i) + w(o_{i+1}) - (w(e_i) + w(e_{i+1}) + w(e_{i+2})) \\
&\leq (1+\alpha) \cdot (w(e_i) + w(e_{i+1})) + (1+\alpha) \cdot (w(e_{i+1}) + w(e_{i+2})) \\
&\qquad - (w(e_i) + w(e_{i+1}) + w(e_{i+2})) \\
&= (1+2\alpha)w(e_{i+1}) + \alpha(w(e_i) + w(e_{i+2})) \\
&\leq (1+2\alpha)w(e_{i+1}) + 2\alpha \cdot (1/(2\alpha)) \cdot w(e_{i+1}) \\
&\leq 3 \cdot w(e_{i+1}).
\end{aligned}$$

□

The guarantee of Unw-3-Aug-Paths holds when there exist large number of vertex-disjoint 3-augmenting paths. To ensure this, we need our weighted augmentations $P_i$ to be edge-disjoint, and for this we need the following lemma.

**Lemma 7.8.** *There exists a set $Q \subseteq O_2$ of indices such that the augmenting paths $P_i$ are edge-disjoint, and $\sum_{i \in Q} g(P_i) \geq (0.02) \cdot w(M^*)$.*

*Proof.* Since $O_2 \subseteq O_1$, we show that the gain of augmentations in $O_2$ is also large by bounding the gain of augmentations in $O_1 \notin O_2$.

A single $P_i$ can be in conflict with at most two other such paths, namely $P_{i-1}$ and $P_{i+1}$. Thus by greedily picking paths $P_i$ with the maximum gain that do not share edges with the previously

picked pairs, we get at least $1/3$ fraction of the total gain of $O_2$, which is

$$\begin{aligned}
&1/3 \sum_{i \in O_2} g(P_i) \\
&\quad \geq 1/3 \left( \sum_{i \in O_1} g(P_i) - \sum_{i \in O_1 \setminus O_2} g(P_i) \right) \\
&\quad \geq 1/3 \left( 0.4w(M^*) - \sum_{i \in O_1 \setminus O_2} ((1/2 + 3\alpha)w(e_{i+1}) + 2\alpha w(e_i) + 2\alpha w(e_{i+2})) \right) \\
&\quad \geq 1/3 \left( 0.4w(M^*) - (1/2 + 7\alpha)w(M_0) \right) \\
&\quad \geq 1/3 (0.4 - (0.64)(1/2 + 4c)) \cdot w(M^*) \\
&\quad \geq (0.02) \cdot w(M^*).
\end{aligned}$$

□

Let $Q_1, Q_2, \ldots, Q_k$ be the indices $i \in Q$ partitioned into $k$ sets according to the weight class of $e_{i+1}$. That is, $i \in Q_j$ if and only if $i \in Q$ and $e_{i+1} \in W_j$. For each $j$, let $Q'_j = \{i \in Q_j : e_{i+1}$ is marked and both $e_i$ and $e_{i+2}$ are not marked$\}$. Let $N_j = M_0 \cap W_j$ be the set of edges in the initial matching $M_0$ that belong to weight class $W_j$. Let $N'_j$ denote the subset of edges in $N_j$ that are marked. Thus $Q_j$'s and $N_j$'s are fixed (given $M_0$) whereas $Q'_j$'s and $N'_j$'s are random. We assume that $|N_j| \geq (100/\beta)$ for some constant $1 > \beta > 0$. If $|N_j| < (100/\beta)$, $|N'_j| \leq |N_j|$ is also less than $(100/\beta)$. Hence we can afford to keep all the edges in the stream that are incident on any edge in $N'_j$ and run an offline algorithm at the end to find the maximum set of 3-augmenting paths that use the edges in $N'_j$ as middle edges. Such an algorithm stores at most $4 \cdot (100/\beta) \cdot n$ edges (at most $2n$ edges per one end point of an edge in $N'_j$). Thus we assume that $|N_j| \geq 100/\beta$.

Fix $j \in \{1, 2, \ldots, k\}$ and let $\text{Aug}_j$ be the set of augmentations returned by the Unw-3-Aug-Paths instance $\mathcal{A}_j$. Let $\text{Aug}'_k = \text{Aug}_k$, and for $j = k-1, k-2, \ldots, 1$, let $\text{Aug}'_j$ be the set of augmentations returned by $\mathcal{A}_j$ that are not in conflict with any of the augmentations in $\text{Aug}'_{j+1}, \ldots, \text{Aug}'_k$.

We now show that if we have large number of augmentations in some weight class, the our algorithm will pick a large fraction of them, and consequently, the unweighted algorithm will also find a large fraction of them. To be precise, we have the following lemma.

**Lemma 7.9.** *Fix some $j$ such that $|Q_j| \geq 16\beta|N_j|$. Then $\mathbb{E}[|\text{Aug}_j|] \geq 2^{-8}\beta^2 \cdot |Q_j|$.*

*Proof.* Let $B_1$ denote the event $|N'_j| \leq (1/4)|N_j|$ and $B_2$ denote the event $|Q'_j| \leq (1/16)|Q_j|$.

Each edge $e \in N_j$ appears in $N'_j$ independently with probability $1/2$. Therefore, $\mathbb{E}[|N'_j|] = (1/2)|N_j|$, and by Chernoff bounds,

$$\Pr[B_1] \leq e^{-(1/2)^2(1/2)(1/2)|N_j|} = e^{-(1/16)|N_j|} \leq 1/4.$$

Similarly, since the paths $P_i$ for $i \in Q_j$ are disjoint, each $i \in Q_j$ appears in $Q'_j$ independently with probability $(1/2)(1-1/2)^2 = 1/8$. Hence $\mathbb{E}[|Q'_j|] = (1/8)|Q_j|$, and by Chernoff bounds,

$$\Pr[B_2] \leq e^{-(1/2)^2(1/8)(1/2)|Q_j|} = e^{-(1/64)|Q_j|} \leq e^{-(1/4)\beta|N_j|} \leq 1/4.$$

The "good event" $\bar{B}_1 \cap \bar{B}_2$ implies that $(1/4)|N_j| \leq |N'_j| \leq |N_j| \leq (1/16\beta)|Q_j|$ and $|Q'_j| \geq (1/16)|Q_j|$, and consequently $|Q'_j| \geq \beta|N'_j|$. Also, $\Pr[\bar{B}_1 \cap \bar{B}_2] \geq 1 - 1/4 - 1/4 = 1/2$. Notice that $|N'_j|$ is the initial matching size of Unw-3-Aug-Paths instance $\mathcal{A}_j$ while each $i \in Q'_j$ corresponds to an unweighted 3-augmentation $(o_i, e_{i+1}, o_{i+1})$ with respect matching $N'_j$. Also notice that those 3-augmentations are vertex-disjoint since for all $i \in Q'_j$, the augmentations $(e_i, o_i, e_{i+1}, o_{i+1}, e_{i+2})$ are edge disjoint (by Lemma 7.8). Hence we have,

$$\begin{aligned}\mathbb{E}[|\operatorname{Aug}_j|] &\geq \Pr\left[\bar{B}_1 \cap \bar{B}_2\right] \cdot \underbrace{\mathbb{E}\left[|\operatorname{Aug}_j| \,\middle|\, \bar{B}_1 \cap \bar{B}_2\right]}_{\geq (\beta^2/32)|N'_j| \text{ by Lemma 7.1}} \\ &\geq \frac{1}{2}\frac{\beta^2}{32}|N'_j| \geq \frac{\beta^2}{64}\frac{1}{4}|N_j| = \frac{\beta^2}{256}|N_j| \geq \frac{\beta^2}{256}|Q_j|.\end{aligned}$$

The last inequality holds because $|Q_j| \leq |N_j|$ (each $i \in Q_j$ is associated with a unique edge in $N_j$, namely $e_{i+1}$). □

We are now ready to show that the total gain of the augmentations over all weight classes is high. Recall that this and Lemmas 7.6 to 7.9 hold under the assumption that $w'(M') \leq (2^{-12}) \cdot w(M^*)$.

**Lemma 7.10.** *The total expected gain of weight we get by doing the augmentations in Line 19 in Wgt-Aug-Paths is at least $8c \cdot w(M^*)$ for some sufficiently small constant $c > 0$.*

*Proof.* Let $\Delta' = \sum_{j\in[k]} |\operatorname{Aug}'_j| \cdot 2^{j-1}\alpha$, thus the total gain of all the augmentations is at least $\Delta'$ (for each augmentation where the middle edge belongs to $N_j$, we gain at least $2^{j-1}\alpha$). Let $\Delta = \sum_{j\in[k]} |\operatorname{Aug}_j| \cdot 2^{j-1}\alpha$. Recall that by definition of $\operatorname{Aug}'_j$'s, each augmentation in $\operatorname{Aug}_j$ can block at most 2 other augmentations of lower weight classes $\operatorname{Aug}_{j-1}, \operatorname{Aug}_{j-2}, \ldots, \operatorname{Aug}_1$. Thus if we consider a term $2^{j-1}\alpha$ in the summation $\Delta$, it can eliminate at most $2\alpha\sum_{j'=1}^{j-2} 2^{j'-1} \leq 2 \cdot 2^{j-1}\alpha$ worth of other terms in the summation $\Delta'$. Thus we have that $\Delta' \geq (1/3) \cdot \Delta$, and hence it is sufficient to show that $\mathbb{E}[\Delta] \geq 24c \cdot w(M^*)$, which would imply that $\mathbb{E}[\Delta'] \geq 8c \cdot w(M^*)$.

First notice that $\Delta = \sum_{j\in[k]} |\operatorname{Aug}'_j| \cdot 2^{j-1}\alpha \geq \sum_{j\in[k]:|Q_j|\geq 16\beta|N_j|} |\operatorname{Aug}_j| \cdot 2^{j-1}\alpha$. Thus we have

$$\mathbb{E}[\Delta] \geq \sum_{\substack{j\in[k] \\ |Q_j|\geq 16\beta|N_j|}} \mathbb{E}\left[|\operatorname{Aug}_j|\right] \cdot 2^{j-1}\alpha \tag{7.1}$$

$$\geq \sum_{\substack{j\in[k] \\ |Q_j|\geq 16\beta|N_j|}} \frac{\beta^2}{512}|Q_j|2^{j-1}\alpha = \frac{\alpha\beta^2}{1024}\left(\sum_{\substack{j\in[k] \\ |Q_j|\geq 16\beta|N_j|}} |Q_j|2^j\right)$$

$$\geq \frac{\alpha\beta^2}{1024}\left(\sum_{j=[k]} |Q_j|2^j - \sum_{j\in[k]} 16 \cdot 2 \cdot \beta|N_j|2^{j-1}\right). \tag{7.2}$$

Observe that $\sum_{j=1,\ldots,k} 16 \cdot 2 \cdot \beta|N_j|2^{j-1} = 32\beta\sum_{j\in[k]} |N_j|2^{j-1} \leq 32\beta w(M_0) \leq 32\beta \cdot w(M^*) \leq (0.001)w(M^*)$ for $\beta \leq (1/16000)$.

We now lower bound $\sum_{j=[k]} |Q_j|2^j$. We have $\sum_{j\in[k]} |Q_j|2^j \geq \sum_{j\in[k]}\sum_{i\in Q_j} w(e_{i+1})$ because for each $i \in Q_j$, the middle edge $e_{i+1}$ of $P_i$, belongs to the weight class $N_j$ and hence $w(e_{i+1}) \leq 2^j$.

But by third part of Lemma 7.7 we have $\sum_{i \in Q_j} w(e_{i+1}) \geq \sum_{i \in Q_j} (1/3) \cdot g(P_i)$, and consequently

$$\begin{aligned}\sum_{j=[k]} |Q_j| 2^j &\geq (1/3) \cdot \sum_{j \in [k]} \sum_{i \in Q_j} g(P_i) \\ &= (1/3) \sum_{i \in Q} g(P_i) \geq (1/3)(0.02) \cdot w(M^*) \\ &> (0.003) \cdot w(M^*).\end{aligned}$$

The first inequality of the last line follows from Lemma 7.8.

Combining these bounds with inequality (7.2) yields

$$\mathbb{E}[\Delta] \geq (1/1024)(\alpha\beta^2)(0.003 \cdot w(M^*) - 0.001 \cdot w(M^*)) = (1/1024)(\alpha\beta^2)(0.002) \cdot w(M^*),$$

and thus $\mathbb{E}[\Delta'] \geq (1/(3 \cdot 1024))(\alpha\beta^2)(0.002) \cdot w(M^*)$.

We earlier set $\alpha = 0.02$. To finish the proof, set $\beta = 1/16000$ and $c = (1/8) \cdot (1/(3 \cdot 1024))(\alpha\beta^2)(0.002)$. □

Lemma 7.10 implies that if the weight of $M'$ with respect to excess weights $w'$ is small, then in expectation we recover a good matching through augmentations; together with Observation 7.5, this gives the following.

**Lemma 7.11.** *There exists a constant $c > 0$ such that the following holds: if* Wgt-Aug-Paths *is initialized with a matching $M_0$ satisfying $(1/2 - 4c) \cdot w(M^*) \leq w(M_0) \leq (1/2 + 4c) \cdot w(M^*)$, and if the input edge stream $\tilde{E}$ contains a matching of weight at least $(1 - 0.001) \cdot w(M^*)$, the expected weight $\mathbb{E}[w(\hat{M})]$ of the output $\hat{M}$ of* Wgt-Aug-Paths *is at least $(1/2 + 4c) \cdot w(M^*)$.*

We finally note the following lemma on the space complexity of Wgt-Aug-Paths.

**Lemma 7.12.** *The algorithm* Wgt-Aug-Paths *uses $O(n \operatorname{poly}(\log n))$ memory.*

*Proof.* The algorithm runs at $O(\log n)$ copies of the unweighted algorithm Unw-3-Aug-Paths which in turn takes $O(n)$ memory per copy. Furthermore, the $(1/4)$-approximation algorithm for weighted matching used in Wgt-Aug-Paths can be implemented using the $(1/2 - \epsilon)$-approximation algorithm given by Paz and Schwartzman [83], which uses $O(n \operatorname{poly}(\log n))$ memory. Apart from that, Wgt-Aug-Paths only needs $O(n)$ memory to store the initial matching $M_0$. □

### Main Algorithm for $(1/2 + c)$-Approximate Matching

Now that we know how to tackle the difficult case of finding weighted 3-augmenting paths, we shift our focus back to the main algorithm. See Random-Arrival-Matching in Algorithm 7.2.

We quickly recap. Random-Arrival-Matching runs the local-ratio method for the first $p = O(1/\log n)$ fraction of the edge stream and maintains vertex potentials. Then it runs two algorithms in parallel for the rest of the stream: One is the algorithm Wgt-Aug-Paths we described in Section 7.2. The other algorithm merely stores all edges that *would* have been added to the local-ratio stack if we had continued to run it till the end.

**Algorithm 7.2:** Outline of the algorithm Rand-Arr-Matching.

**Input :** Number of vertices $2n$, number of edges $m$, a stream of edges $E$, a weight function $w : E \to \mathbb{R}^+$ where $G = (V, E, w)$ is weighted graph, and an instance `WAP` of the weighted augmenting paths algorithm Wgt-Aug-Paths.
**Output :** A matching $M$ of $G$.
**Global :** Stack $S$ of edges, set $T$ of edges, a vertex potential vector $\alpha \in \mathbb{R}^V$.

1 Let $E = (e_1 = (u_1, v_1), e_2 = (u_2, v_2), \ldots, e_m = (u_m, v_m))$. Let $S = [\,]$ and let $T = [\,]$. Let $\alpha_v \leftarrow 0$ for all $v \in V$.
2 Let $p \leftarrow 100/\log n$.
3 **for** $i \leftarrow 1$ **to** $p \cdot m$ **do**
4  Let $w'(e_i) = w(e_i) - \alpha_{u_i} - \alpha_{v_i}$.
5  **if** $w'(e_i) > 0$ **then**
6  `Push(`$S$, $e_i$`)`
7  $\alpha_{u_i} \leftarrow \alpha_{u_i} + w'(e_i)$
8  $\alpha_{v_i} \leftarrow \alpha_{v_i} + w'(e_i)$
9 Let $M_0$ be the matching computed by unwinding stack $S$.
10 Initialize(`WAP`, $M_0$`)`
11 **for** $i \leftarrow p \cdot m + 1$ **to** $m$ **do**
12  **if** $w(e_i) > \alpha_u + \alpha_v$ **then** `Add(`$T$, $e_i$`)`
13  Feed-Edge(`WAP`, $e_i$`)`
14 Let $M_1$ be the maximum matching in $T$ with respect to weights $w''(e = (u, v)) = w(e) - \alpha_u - \alpha_v$.
15 **while** $S$ *is not empty* **do**
16  $e \leftarrow$ `Pop(`$S$`)`
17  **if** *the endpoints of* $e$ *are not matched in* $M_1$ **then** `Add(`$M_1$, $e$`)`
18 Let $M_2 =$ Finalize(`WAP`)
19 **return** the better of $M_1$ and $M_2$

### Analysis of the main algorithm

We will first show that the expected weight of the matching returned by Rand-Arr-Matching is at least $(1/2 + c) \cdot w(M^*)$, where $c$ is the constant given by Lemma 7.11. We consider three cases based on the weight of $M_0$ which is computed in Line 9.

**Case 1:** The weight of $M_0$ is at least $(1/2 + 4c) \cdot w(M^*)$, in which case we have nothing to do.

**Case 2:** The weight of $M_0$ is at most $(1/2 - 4c)$. For this case, we show that the matching $M_1$ computed by Rand-Arr-Matching has a weight of at least $(1/2 + 4c) \cdot w(M^*)$ in the following lemma.

**Lemma 7.13.** *If* $w(M_0) \leq (1/2 - 4c) \cdot w(M^*)$*, then* $w(M_1) \geq (1/2 + 4c) \cdot w(M^*)$.

*Proof.* Let $A = S \cup \{e = (u, v) \in E : w(e) \leq \alpha^*_u + \alpha^*_v\}$ where $\alpha^*$ is the vertex potential vector after seeing the first $p$ fraction of the edges. Then, with respect to the graph $G' = (V, A, w)$, the local-ratio stack $S$ contains a $1/2$-approximate matching. Suppose that $w(M_0) = (1/2 - \gamma) \cdot w(M^*)$ where $\gamma \geq 4c$. Then the optimal matching of $G'$ is at most $(1 - 2\gamma) \cdot w(M^*)$, which means that the graph with the remaining edges, with respect to the weight function $w''(e = (u, v)) = w(e) - \alpha^*_u - \alpha^*_v$, has a matching of weight at least $2\gamma \cdot w(M^*)$.

Thus the matching $M_1$ computed before Line 14 has a weight of at least $2\gamma \cdot w(M^*)$ with respect to the weight function $w''$.

We now show that, by unwinding the stack $S$ in the while loop in Line 14, we can increase the weight by at least $(1/2 - \gamma) \cdot w(M^*)$. Let $M''$ be any matching in $G'$. By following the same lines of Ghaffari [44], who gave a more intuitive analysis of $(1/2 - \epsilon)$-approximate algorithm by Paz and Schwartzman [83], we show the following: There is a way to delegate the weights of $M''$-edges on to the edges of the matching $M_1$ computed by Algorithm 7.2, such that each edge $e \in M_1$ takes at most $2 \cdot w(e)$ delegated weight.

Fix some edge $e_r = (u_r, v_r)$ in $G'$ and consider the time we push it on to the stack $S$. With a slight abuse of notation, let $G$ denote the remaining graph before pushing $e_r$ on to the stack, and $w$ is the weight function at that time. Let $G_r$ be the graph after the removal of $e_r$ and let $w_r$ be the updated weight function. Let $M_r$ be the snapshot of $M_1$ just before we pop $e_r$ out of the stack, and let $M$ be the snapshot of $M_1$ after popping out $e_r$ and processing it. By induction, assume that in $G_r$, there is a way to delegate the weights $w_r$ of $M''$-edges on to the edges of $M_r$ such that each edge $e \in M_r$ takes at most $2 \cdot w_r(e)$ delegated weight. The base case is just before we start processing the stack, and the claim is trivially true as all $M''$-edges have zero weight at this point.

To conclude the inductive proof, we now show that in $G$, we can delegate the weights $w$ of $M''$-edges on to the edges of $M$ such that each edge $e \in M$ takes at most $2 \cdot w(e)$ delegated weight.

In $M''$, there can be at most two edges $e_1, e_2$ incident to the edge $e_r$. (It may happen that $e_r$ is in $M''$ so that we have exactly one such edge.) By inductive hypothesis, for each $e_i \in \{e_1, e_2\}$ we have already found a way to delegate the weight $w_r(e_i) = w(e_i) - w(e_r)$ on to $M_r$ edges. We need to find room to delegate at most $(w(e_1) - w_r(e_1)) + (w(e_2) + w_r(e_2)) = 2 \cdot w(e_r)$ more weight. When we pop $e_r$ out of the stack, we have the following two cases:

1. At least one of the endpoints $u_r$ or $v_r$ of edge $e_r$ is matched in $M_r$ with some edge $e'_r$. Thus edge $e'_r$ has taken at most $2w_r(e'_r) = 2(w(e'_r) - w(e_r))$ amount of delegated weight at the moment. But in $G$, edge $e'_r$ can take up to $2 \cdot w(e'_r)$ weight, hence we have room for $2 \cdot w(e_r)$ on $e'_r$.
2. Both endpoints $u_r$ and $v_r$ of edge $e_r$ are unmatched in $M_r$, so that we add $e_r$ to our matching as a new edge. Therefore it has its full capacity of $2 \cdot w(e_r)$ remaining for the delegated weight.

Thus we have room to delegate a weight of $2 \cdot w(e_r)$ in both the cases, and thus the step of processing the stack $S$ in the while loop in Line 14 increases the weight of $M_1$ by at least $1/2 w(M'')$ where $M''$ is any matching in $G'$. Setting $M''$ to be the maximum matching in $G'$, we get $w(M_1) \geq 1/2 \cdot (1 - 2\gamma) \cdot w(M^*) + 2\gamma \cdot w(M^*) = (1/2 + \gamma) \cdot w(M^*) \geq (1/2 + 4c) \cdot w(M^*)$. □

**Case 3:** The matching $M_0$ is such that $(1/2 - 4c) \cdot w(M^*) \leq w(M_0) \leq (1/2 + 4c) \cdot w(M^*)$. For this case, we already proved that the expected weight is at least $(1/2 + 4c) \cdot \mathrm{OPT}$ if the last $(1 - p)$ fraction of the stream contains a matching of weight at least $(1 - 0.001) \cdot w(M^*)$.

Now we put together Lemma 7.13 with Lemma 7.11 and prove the following theorem.

**Theorem 7.14.** *For sufficiently large $n$, the expected weight of the matching returned by the algorithm* Rand-Arr-Matching *is at least* $(1/2 + c) \cdot w(M^*)$.

*Proof.* Let $\tilde{M}$ be the output of Rand-Arr-Matching. Let $\mathcal{E}$ denote the event that the last $(1-p)$ fraction of the stream contains a matching of weight at least $(1 - 0.001)w(M^*)$. Let $M^*$ be a fixed optimal matching of the graph. Let $E_1$ denote the first $p$ fraction of the edges in the graph. By the random order arrival property, we have that $\mathbb{E}[w(M^* \cap E_1)] = pw(M^*) = pw(M^*)$. To see this, notice that each edge in $E_1$ has equal chance of being one of the edges of $M^*$, and then the result follows from the linearity of expectation. Thus by Markov's inequality $\Pr[w(M^* \cap E_1) \geq 0.001w(M^*)] < p/0.001 < c$ for sufficiently large $n$ as $p = O(1/\log n)$. Thus $\Pr[\mathcal{E}] \geq 1 - 4c$. Also, due to Lemma 7.13 and Lemma 7.11 $\mathbb{E}[w(\tilde{M})|\mathcal{E}] \geq (1/2 + 4c)w(M^*)$. This yields that

$$\begin{aligned}\mathbb{E}[w(\tilde{M})] &\geq \Pr[\mathcal{E}] \cdot \mathbb{E}[w(\tilde{M})|\mathcal{E}] \geq (1-c)(1/2 + 4c)w(M^*) \\ &= (1/2 + 7c/2 - 4c^2)w(M^*) \geq (1/2 + c)w(M^*).\end{aligned}$$

□

What remains now is to bound the memory requirement of Rand-Arr-Matching. We know from Lemma 7.12 that the instance `WAP` of Wgt-Aug-Paths used in Rand-Arr-Matching uses at most $O(n \operatorname{poly}(\log n))$ memory. Thus we only need to show that both the stack $S$ and the set $T$ also use $O(n \operatorname{poly}(\log n))$ memory.

**Bounding the size of $S$:** Consider a state of the local-ratio algorithm where we have added some edges to the stack and suppose that vertex potentials are $\alpha'_v$ for all $v \in V$. For an edge $e = (u, v)$, let $w'(e) = w(e) - \alpha'_u - \alpha'_v$. Let $E'$ be the set of remaining edges for which $w'(e) > 0$. The next edge added to the stack by the local ratio algorithm is equally likely to be any edge from $E'$.

For a vertex $v \in V$, let $d'_v$ be the number of edges incident to $v$ in $E'$. Consider a random edge $X$ selected as follows. First pick vertex $v \in V$ with probability proportional to $d'_v$, and then pick a uniformly random edge incident to $v$ in $E'$. It is easy to see that $X$ is a uniformly random edge of $E'$.

Now fix a vertex $v$ and order the edges in $E'$ that are incident to $v$ in increasing order of $w'$. Notice that if the local-ratio algorithm sees the $i$-th edge in ordering, then it will be added to stack and, and since its weight get subtracted from each of the other incident edges, the weights of at least $i - 1$ other edges go below zero. This means that at least $i$ gets removed from $E'$ in the perspective of the local-ratio algorithm. Let $R$ be the set of removed edges and let $E''$ be the set of remaining edges after adding the next edge to $S$. Then by the above reasoning, we have,

$$\begin{aligned}\mathbb{E}[R] &\geq \sum_{v \in V} \Pr_X[\text{pick } v] \cdot \left( \sum_{i \in [d'_v]} \Pr_X[\text{picking } i\text{-th edge incident to } v] \cdot i \right) \\ &= \sum_{v \in V} \frac{d'_v}{2|E'|} \left( \sum_{i \in [d'_v]} \frac{1}{d'_v} \cdot i \right) = \frac{1}{2|E'|} \sum_{v \in V} \frac{d'_v(d'_v + 1)}{2} \geq \frac{1}{4|E'|} \sum_{v \in V} (d'_v)^2.\end{aligned}$$

But by Cauchy-Shwartz inequality, since $\sum_{v \in V} d'_v = 2|E'|$, we have that

$$\underbrace{\left(\sum_{v \in V} 1^2\right)}_{n} \left(\sum_{v \in V} (d'_v)^2\right) \geq \underbrace{\left(\sum_{v \in V} 1 \cdot d'_v\right)^2}_{(2|E'|)^2},$$

or equivalently, $\sum_{v \in V} (d'_v)^2 \geq (4/n)|E'|^2$. Hence $\mathbb{E}[R] \geq |E'|/n$ and the expected number of remaining edges, $\mathbb{E}[E'']$, at most $|E'|(1 - 1/n)$.

This yields that after picking $100n \log n$ edges, the expected number of remaining edges is at most $|E|(1 - 1/n)^{100n \log n} \leq 1/n^3$, and thus by the Markov's inequality, size of $|S|$ is $O(n \log n)$ with high probability.

**Bounding the size of $T$:** We next show that $|T| \in O(n \operatorname{poly}(\log n))$ with high probability. To bound the size of $T$ at the end of the algorithm, we define events $B_{v,t}$ similarly to how we defined $A_{v,t}$ in the proof of Lemma 7.3.

Recall that $A_{v,t}$ are defined to capture the number of unmatched neighbors of a vertex $v$ after processing the first $t$ edges. Define $B_{v,t}$ to be the event that at least $\log^2 n$ edges $e$ incident to $v$ satisfy $w'_t(e) > 0$, where $w'_t(e)$ denote the value of $w'(e)$ just after processing the $t$-th edge of the stream. (Recall that $w'(e) = w(e) - \alpha_u - \alpha_v$ as defined in Line 4 of Algorithm 7.2.) In the rest of the proof, we show that $\Pr[B_{v,t}|B_{v,t-1}] \leq \left(1 - (\log^2 n)/(m-t+1)\right)$, after which the claim follows as in the proof of Lemma 7.3 for $p = 100/\log n$.

If $B_{v,t}$ occurs, let $C_{v,t}$ be the set of edges corresponding to the $\log^2 n$ largest *positive* values $w'_t(e)$ over all the edges $e$ incident to $v$. Then, if $B_{v,t-1}$ occurs and if the $t$-th edge from the stream is from $C_{v,t-1}$, then $B_{v,t}$ can not occur. Since each edge $e \in C_{v,t-1}$ is such that $w'_{t-1}(e) > 0$, $e$ appears in the stream after position $t - 1$. So, given that $B_{v,t-1}$ occurs, the probability that the $t$-th edge from the stream is in $C_{v,t-1}$ is $|C_{v,t-1}|/(m-t+1)$, and hence

$$\Pr[B_{v,t}|B_{v,t-1}] \leq \left(1 - \frac{\log^2 n}{m - t + 1}\right),$$

as desired. This gives the following lemma on the size of the stack $S$ and set $T$.

**Lemma 7.15.** *Given that the edges arrive in a uniformly random order, with high probability, both the local-ratio stack $S$ and the set $T$ will contain $O(n \operatorname{poly}(\log n))$ edges.*

Lemma 7.12 and Lemma 7.15 yield that the algorithm Rand-Arr-Matching uses $O(n \operatorname{poly}(\log n))$ memory with high probability.

# 8 Weighted Matchings through Unweighted Augmentations

In this chapter, we reduce the problem of finding weighted augmenting paths in general graphs to that of finding *unweighted* augmenting paths in *bipartite* graphs. Our reduction yields a $(1-\varepsilon)$-approximation maximum weighted matching algorithm that can be efficiently implemented in both the multi-pass streaming model and the MPC model. We formalize this result as Theorem 8.1 below. Throughout the section, we use $M^*$ to denote some fixed maximum weighted matching in the input graph, and we assume that edge weights are positive integers bounded by $\text{poly}(n)$.

**Theorem 8.1** (General weighted to bipartite unweighted)**.** *Let $M^*$ be a maximum weighted matching and $M$ be any weighted matching such that $w(M) < w(M^*)/(1+\varepsilon)$ for some constant $\varepsilon$. There exists an algorithm that in expectation augments the weight of $M$ by at least $\varepsilon^{O(1/\varepsilon^2)} \cdot w(M^*)$ which can be implemented*

1. *in $U_M$ rounds, $O(m/n)$ machines per round, and $O_\varepsilon(n \,\text{poly}(\log n))$ memory per machine, where $U_M$ is the number of rounds used by a $(1-\delta)$-approximation algorithm for bipartite unweighted matching that uses $O(m/n)$ machines per round and $O_\delta(n \,\text{poly}(\log n))$ memory per machine in the MPC model, and*
2. *in $U_S$ passes and $O_\varepsilon(n \,\text{poly}(\log n))$ memory, where $U_S$ is the number of passes used by a $(1-\delta)$-approximation algorithm for bipartite unweighted matching that uses $O_\delta(n \,\text{poly}(\log n))$ memory in the multi-pass streaming model,*

*where $\delta = \varepsilon^{28+900/\varepsilon^2}$. Using the algorithm of Ghaffari et al. [45] or that of Assadi et al. [9], we get that $U_M = O_\varepsilon(\log\log n)$, and using the algorithm of Ahn and Guha [3], we get that $U_S = O(\log\log(1/\delta)/\delta^2) = O((1/\varepsilon)^{56+1800/\varepsilon^2}\log(1/\varepsilon))$.*

It is easy to see that Theorem 6.2 follows directly from Theorem 8.1. If the current matching is not $(1-\varepsilon)$-approximate, after a single run of the algorithm guaranteed by Theorem 8.1, the weight of the current matching improves by at least $\varepsilon^{O(1/\varepsilon^2)} \cdot w(M^*)$ in expectation. Hence, it is sufficient to repeat the same algorithm for $(1/\varepsilon)^{O(1/\varepsilon^2)}$ iterations to get (in expectation) a $(1-\varepsilon)$-approximation. Since each iteration can reuse the memory used by the previous iteration, the space requirement of the multi-pass streaming model and the memory-per-machine requirement in the MPC model remain unchanged.

As explained in Section 6.1.2, a quick summary of our proof technique for Theorem 8.1 is as follows: First we show that, if the initial matching $M$ we have is not close to optimal, then there

exists a large-by-weight fraction of short augmentations, and these augmentations can be divided into several classes where augmentations in each class have comparable edge-weights and gains. We then show how to find weighted augmentations in each such class by reducing it to a bipartite matching problem. The reduction encompasses our layered graph construction and the filtering technique. Finally, we combine the augmentations recovered by this method in a greedy manner to significantly improve the current matching, and this yields the proof of Theorem 8.1.

In the rest of this section, we elaborate on each of these steps: In Section 8.1, we first introduce the concept of augmentation classes to capture groups of short augmentations that have similar edge-weights and gains. Then, we formally state the two intermediate results: the first on augmentation classes containing augmentations that contributes to an overall gain of $\Omega(\varepsilon^2) \cdot w(M^*)$ (Theorem 8.7) and the second on the existence of an efficient procedure to find many augmentations of those classes using a reduction to the unweighted bipartite setting (Theorem 8.8). Theorem 8.7 and Theorem 8.8 now imply a simple algorithm for proving Theorem 8.1: Run the algorithm given by Theorem 8.8 for each augmentation class (of geometrically increasing weight), and then greedily pick non-conflicting augmentations starting with the augmentation class of the highest weight. We then analyze this algorithm and prove Theorem 8.1 assuming that Theorem 8.7 and Theorem 8.8 hold (whose proofs appear later).

In Section 8.2, we prove Theorem 8.7. In fact, we prove a technical lemma that guarantees many-by-weight short augmenting paths and cycles of significant gains that also satisfy several additional constraints on edge weights (thus it is stronger than Theorem 8.7). This lemma, while implying Theorem 8.7, also assists in proving Theorem 8.8, as the additional constraints on edge weights of the augmentations make sure that many of those augmentations are captured by our reduction.

In the more involved Section 8.3, we present the precise construction of the layered graphs we introduced in Section 6.1.2, and we explain our filtering technique in detail. We then show how exactly the unweighted augmenting paths in the layered graphs relate to weighted augmenting paths and cycles of the original graph. Finally, in Section 8.4, we put together the results from Section 8.2 and Section 8.3 to prove Theorem 8.8.

Throughout the analysis, we assume that $\varepsilon < 1/16$, and also extensively use the following definitions. We begin with the definition of alternating paths and cycles.

**Definition 8.2** (Alternating paths and cycles)**.** *Let $M$ be a matching. A path $P$ is said to be* alternating *if its edges alternate between $M$ and $E \setminus M$. The first edge of $P$ can be in $M$ or $E \setminus M$. Similarly, a cycle $C$ is alternating if its edges alternate between $M$ and $E \setminus M$.*

Observe that from the definition, an alternating cycle has even length and an alternating path can be of even or odd length.

In our analysis, we sometimes consider alternating paths such that an endpoint of a path $P$ is incident to a matched edge $e$ that is not on the path $P$. For instance, let $P = v_1v_2v_3$ and $\{v_2, v_3\} \in M$, and suppose that there is another edge $\{v_0, v_1\} \in M$. Now, if we wish to add $\{v_1, v_2\}$ to the matching, we should remove both $\{v_0, v_1\}$ and $\{v_2, v_3\}$. Hence, adding some edges of a path to the matching might involve removing some edges which are not on the path. To capture this scenario, we define the following notion:

**Definition 8.3** (Matching neighborhood)**.** *Let $C$ be an alternating path or an alternating cycle*

*with respect to $M$. Then, by $C^M$, we denote all the edges of the matching $M$ incident to the vertices of $C$, including those lying on $C$ itself. Note that if $C$ is a cycle, then $C^M = C \cap M$.*

For completeness, we also define the usual notions of *applying an augmentation* and the *gain of an augmentation* below.

**Definition 8.4** (Applying augmentation)**.** *Let $C$ be an alternating path or an alternating cycle. Let $A = C^M$ and $B = C \setminus M$. Then, by* applying *$C$ we define the operation in which $A$ is removed from $M$ and $B$ is added to $M$.*

**Definition 8.5** (Gain of augmentation)**.** *Let $C$ be an alternating path or an alternating cycle. Then, the* gain *$w^+(C)$ of $C$ denotes the increase in the matching weight if $C$ is applied.*

Note that an augmentation usually means an alternating path or cycle whose unmatched edges have a larger total weight than that of the edges in its matching neighborhood. However we sometimes consider cases where each individual 'augmentation' does not satisfy this, but collectively they do. (For example consider the single edge alternating paths $v_1v_2$ and $v_3v_4$ where $v_1$ and $v_4$ are unmatched vertices and $v_2$ is matched to $v_3$. If $w(\{v_1, v_2\}) = w(\{v_3, v_4\}) = 2$ and $w(\{v_2, v_3\}) = 3$, then applying both the augmentations gives a gain of one whereas applying either one of them individually is not beneficial.)

## 8.1 The Main Algorithm

In this section, we present our main algorithm and prove Theorem 8.1 assuming the two intermediate results that we prove in the later sections. The first one claims that if the current matching is not $(1 - \varepsilon)$-approximate, then there exist many-by-weight short vertex-disjoint augmentations (Theorem 8.7) that have comparable gains and edge weights. The second one claims that, for a given weight $W$, we can efficiently find many-by-weight short augmentations whose edge weights and gains are comparable to $W$ (Theorem 8.8). We begin by defining augmentation classes, which are collections of augmentations whose gains and individual edges are similar in weight.

**Definition 8.6** (Augmentation class)**.** *Fix a weight $W$, and let $M$ be the current matching. By the* augmentation class *of $W$ we refer to the collection of all augmentations (not necessarily vertex-disjoint) such that each augmentation $C$ has the following properties:*

1. *The weight of each edge of $C$ is between $\varepsilon^{12}W$ and $2W$.*
2. *The gain $w^+(C)$ of $C$ is at most $2W$.*
3. *When the weight of each edge in $C^M$ (recall that $C^M$ is the matching neighborhood of $C$) is rounded up and the weight of each unmatched edge in $C$ (i.e., $C \setminus C^M$) is rounded down to the nearest multiple of $\varepsilon^{12}W$, the gain of such $C$ is at least $\varepsilon^{12}W$.*
4. *The augmentation $C$ consists of at most $64/\varepsilon^2 + 1$ vertices.*

By the third property, the gain of an augmentation $C$ without any rounding is also at least $\varepsilon^{12}W$. The following theorem says that if $M$ is not close to optimal, then there is a collection of vertex-disjoint augmentations, each of which belongs to some augmentation class, and collectively they have a large gain; we prove this theorem in Section 8.2.

**Theorem 8.7** (Significant weight in augmentation classes)**.** *Let $M$ be a matching such that $w(M) < (1-\varepsilon)w(M^*) \leq {}^{w(M^*)}/_{(1+\varepsilon)}$ (i.e., $M$ is not $(1-\varepsilon)$-approximate). Then, there exists a collection $\mathcal{C}$ of vertex-disjoint augmentations with the following properties:*

1. *Each augmentation $C \in \mathcal{C}$ is in the augmentation class of $(1+\varepsilon^4)^i \leq w(C)$ for some $i \leq \lceil \log_{1+\varepsilon^4}(({}^{64}/_{\varepsilon^2}+1)\cdot \max\{w(e) : e \in E\})\rceil$.*
2. *It holds that $w^+(\mathcal{C}) \geq ({}^{\varepsilon^2}/_{200})\cdot w(M^*)$.*

In the following result, we essentially claim that if a given augmentation class does not already contain many-by-weight edges of $M$, then there is an efficient procedure that finds many-by-weight vertex-disjoint augmentations in that class.

**Theorem 8.8** (Single augmentation class)**.** *Let $M$ be the current matching. Assume that $w(M) < {}^{w(M^*)}/_{(1+\varepsilon)}$. Let $\mathcal{C}_W$ denote a collection of vertex-disjoint augmentations belonging to the augmentation class of $W$. Define $w(M_W)$ to be the total weight of the edges of $M$ with weights in $[\varepsilon^{12}W, 2W]$. Then there is an algorithm that, given $W$, outputs a collection $\mathcal{A}_W$ of vertex-disjoint augmentations ($\mathcal{A}_W$ is not necessarily a subset of $\mathcal{C}_W$) having the following properties:*

*(A) $\mathcal{A}_W$ is a subset of the augmentation class of $W$.*

*(B) In expectation, $w^+(\mathcal{A}_W) \geq \varepsilon^{c/\varepsilon^2}(w^+(\mathcal{C}_W) - \varepsilon^{10}w(M_W))$, for some constant $c$.*

*This algorithm can be implemented in $U_M$ MPC rounds with $O_\varepsilon(n\log n)$ memory per machine, and $U_S$ passes and $O_\varepsilon(n\,\mathrm{poly}(\log n))$ memory in the streaming model, where $U_M$ and $U_S$ are defined in Theorem 8.1.*

Let $\mathcal{C}$ be the family of augmentations as defined in Theorem 8.7, so applying $\mathcal{C}$ increases the matching weight by $({}^{\varepsilon^2}/_{200})\cdot w(M^*)$. Consider all the weights of the form $(1+\varepsilon^4)^i$, for $i \in \mathbb{N}$. Property (B) of Theorem 8.8 implies that there is an algorithm that for those weights finds augmentations whose sum of gains, when applied *independently*, is in expectation at least $\varepsilon^{O({}^{1}/_{\varepsilon^2})}w^+(\mathcal{C})$ up to some additive loss. (This additive loss is significant only if there is already a significant weight in the matching $M_W$.) However, even if that additive loss is negligible, when those augmentations are applied simultaneously they might intersect.

But, we still manage to find a set of non-intersecting augmentations of significant total gain by following a simple greedy strategy; we consider augmentation classes in decreasing order of weight and apply only those augmentations that do not intersect with previously applied ones. This approach retains a considerable fraction of the gain since the augmentations we consider are short (thus, for a given augmentation, the number of conflicting augmentations in a given augmentation class is small), and since the weights of augmentation classes, and consequently, the maximum gains of augmentations in those classes are geometrically decreasing.

Algorithm 8.1 implements this approach, and we analyze it next to prove our main result, Theorem 8.1, assuming that we already have Theorem 8.7 and Theorem 8.8.

*Proof of Theorem 8.1.* The theorem follows from the analysis of Algorithm 8.1. Recall that $\mathcal{C}_W$ is the augmentations of $\mathcal{C}$ ($\mathcal{C}$ is defined in Theorem 8.7 and satisfies $w^+(\mathcal{C}) \geq ({}^{\varepsilon^2}/_{200})\cdot w(M^*)$)

**Algorithm 8.1:** Algorithm MAIN-ALG for improving matching weight as described by Theorem 8.1

**Input :** A weighted graph $G$, approximation parameter $\varepsilon$, the current matching $M$
**Output :** A matching of $G$

1 $i_{\max} \leftarrow \lceil \log_{1+\varepsilon^4}((64/\varepsilon^2 + 1) \cdot \max\{w(e) : e \in E\}) \rceil$.
`// One MPC round or one pass can be spent to compute` $\max\{w(e) : e \in E\}$.
2 $\mathcal{W} \leftarrow \{(1+\varepsilon^4)^i : i = 0, 1, \ldots, i_{\max}\}$
3 **for** *each* $W \in \mathcal{W}$ *in parallel* **do**
4 Let $\mathcal{A}_W$ be the set of augmentations that the algorithm of Theorem 8.8 outputs for the augmentation class $W$.
5 $\hat{\mathcal{A}} \leftarrow \emptyset$
6 **for** *each* $W \in \mathcal{W}$ *in decreasing order* **do**
7 **for** *each augmentation* $C$ *in* $\mathcal{A}_W$ **do**
8 Add $C$ to $\hat{\mathcal{A}}$ if $C$ does not conflict with any other augmentation in $\hat{\mathcal{A}}$.
9 **return** the matching obtained after applying the augmentations in $\hat{\mathcal{A}}$ to $M$.

that are also in the augmentation class $W$ and $M_W$ is the set of matching edges whose weights are between $\varepsilon^{12}W$ and $2W$.

Let $W^+_{\text{all}}$ be the total gain of all augmentations that the algorithm finds in Line 4. I.e., $W^+_{\text{all}} = \sum_{W \in \mathcal{W}} w^+(\mathcal{A}_W)$, where $\mathcal{W}$ is the set of weights of all augmentation classes considered by the algorithm. By Theorem 8.8, we have that $w^+(\mathcal{A}_W) \geq \varepsilon^{c/\varepsilon^2}\left(w^+(\mathcal{C}_W) - \varepsilon^{10} w(M_W)\right)$, which yields

$$W^+_{\text{all}} \geq \varepsilon^{c/\varepsilon^2}\left(\sum_{W \in \mathcal{W}} w^+(\mathcal{C}_W) - \varepsilon^{10} \sum_{W \in \mathcal{W}} w(M_W)\right). \tag{8.1}$$

Notice that for two weights $W_1$ and $W_2$, if $\varepsilon^{12}W_1 > 2W_2$, then $M_{W_1}$ and $M_{W_2}$ do not intersect. Since we consider weights of the form $(1+\varepsilon^4)^i$, any matching edge can be in $M_W$ for at most $\lceil \log_{1+\varepsilon^4}(2/\varepsilon^{12}) \rceil \leq 1/\varepsilon^6$ (we assumed $\varepsilon < 1/16$) different weights $W$. This yields that

$$\varepsilon^{10} \sum_{W \in \mathcal{W}} w(M_W) < \varepsilon^{10}(1/\varepsilon^6) w(M) \leq (\varepsilon^2/256) w(M),$$

where the last inequality follows from the assumption that $\varepsilon < 1/16$.

On the other hand, by Theorem 8.7, the term $\sum_{W \in \mathcal{W}} w^+(\mathcal{C}_W)$ is at least $(\varepsilon^2/200) \cdot w(M^*)$. Substituting in Eq. (8.1), we get,

$$W^+_{\text{all}} \geq \varepsilon^{c/\varepsilon^2}\left((\varepsilon^2/200) \cdot w(M^*) - (\varepsilon^2/256) w(M)\right) \geq \varepsilon^{c'/\varepsilon^2} w(M^*)$$

for some constant $c' > 0$.

Now fix some augmentation class $W_i = (1+\varepsilon^4)^i$ and an augmentation $C$ in $\mathcal{A}_{W_i}$. By definition, $w^+(C) \geq \varepsilon^{12}(1+\varepsilon^4)^i$, and for any other augmentation class $W_j = (1+\varepsilon^4)^j$, the maximum gain of any augmentation in $\mathcal{A}_{W_j}$ is at most $2W_j = 2(1+\varepsilon^4)^j$. If we apply $C$, it blocks at most $64/\varepsilon^2 + 1$ other augmentations in each of the augmentation classes below it. Thus the total gain of the

blocked augmentations if $C$ is applied is at most

$$\begin{aligned}\sum_{j<i}(64/\varepsilon^2+1)\cdot 2\cdot(1+\varepsilon^4)^j &\le (130/\varepsilon^2)(1+\varepsilon^4)^{i-1}\sum_{j=0}^{\infty}(1+\varepsilon^4)^{-j}\\ &= (130/\varepsilon^2)(1+\varepsilon^4)^{i-1}\frac{1}{1-1/(1+\varepsilon^4)}\\ &= (130/\varepsilon^6)(1+\varepsilon^4)^i \le (130/\varepsilon^{18})w^+(C),\end{aligned}$$

and this means that the gain we retain by our greedy strategy, $w^+(\hat{A})$, is at least $W^+_{\text{all}}/(1+130/\varepsilon^{18}) \ge \varepsilon^{c''/\varepsilon^2}w(M^*)$ for some constant $c'' > 0$.

**MPC implementation:** Since the maximum edge weight is $\text{poly}(n)$, the number of different augmentation classes we consider, i.e., $i_{\max}+1$, is $O(\log_{1+\varepsilon^4} n)$. Hence can implement Line 4 in $O(m/n)$ machines with $O_\varepsilon(n\,\text{poly}(\log n))$ memory by running the algorithm of Theorem 8.8 (i.e., Algorithm 8.2) in parallel for each augmentation class.

For each augmentation class $W$, the collection of augmentations $\mathcal{A}_W$ is vertex disjoint, and hence requires $O(n)$ memory. Thus all the collection $\mathcal{A}_W$ for all augmentation classes require $O_\varepsilon(n\,\text{poly}(\log n))$ memory, and hence they can be collected in a single round into a single machine, and the greedy strategy can be run in that machine.

**Streaming implementation:** The implementation is quite straightforward for the streaming setting. For each $W \in \mathcal{W}$, an instance is created, in which Algorithm 8.2 is run. Note that there are $O(\log_{1+\varepsilon^4} n)$ such instances. All the outputs ($|\mathcal{W}|$ of them) are then stored. The greedy conflict resolution that is done afterwards can be done using these stored outputs without using any pass over the stream. So the number of passes used is same as that used by Algorithm 8.2, and memory used is $O(\log_{1+\varepsilon^4} n)$ times that used by Algorithm 8.2 (see Theorem 8.8). □

## 8.2 Existence of Many-by-weight Short Augmentations

In this section we show that if the current matching is not a $(1-\varepsilon)$-approximate one, then there exists a large-by-weight number of short vertex-disjoint augmentations. Moreover, we show that many of those augmentations $C$ have the following properties: the weight of each edge of $C$ (matched or unmatched) is $\Omega(\text{poly}(\varepsilon)\cdot w(C))$ (Properties B and C of Lemma 8.9); and, $w^+(C)$ is large (Property D of Lemma 8.9). This implies that $C$ belongs to some augmentation class, e.g., to an augmentation class of $w(C)$ rounded down to $(1+\varepsilon^4)^i$ (a formal argument of this appears after the statement of the lemma). Hence, the following lemma implies that the augmentation classes all combined contain a collection of vertex-disjoint augmentations of large weight.

**Lemma 8.9.** *Let $M$ be a matching such that $w(M) \le w(M^*)/(1+\varepsilon)$ where $\varepsilon \le 1/16$. Then there exists a collection $\mathcal{C}$ of vertex-disjoint augmentations with the following properties:*

*(A) Each $C \in \mathcal{C}$ is such that $C \cup C^M$ consists of at most $4/\varepsilon$ edges.*

*(B) For every $C \in \mathcal{C}$ and every edge $e \in C \cap M^*$, $w(e) \ge (\varepsilon^2/64)\cdot w(C)$.*

*(C) For every $C \in \mathcal{C}$ and every edge $e \in C \cap M$, $w(e) \geq (\varepsilon^6/64) \cdot w(C)$.*

*(D) For every $C \in \mathcal{C}$, we have that*

$$w(C \cap M^*) \geq (1 + \varepsilon/8) \cdot w(C^M).$$

*(E) The sum of gains of the elements of $\mathcal{C}$ is at least $(\varepsilon^2/200) \cdot w(M^*)$. That is*

$$\sum_{C \in \mathcal{C}} \left( w(C \cap M^*) - w(C^M) \right) \geq (\varepsilon^2/200) \cdot w(M^*).$$

The proof of Lemma 8.9 is a simple adaptation of the proof of the known fact that a matching has many short augmentations if its value is less than $(1 - \epsilon)$ times the value of an optimal matching. It is provided in Appendix B.

We now formally argue that the above lemma implies Theorem 8.7. Recall the statement of that theorem: if $w(M) \leq w(M^*)/(1+\varepsilon)$ then there exists a collection $\mathcal{C}$ of vertex-disjoint augmentations with the following properties:

- Each augmentation $C \in \mathcal{C}$ is in the augmentation class of $(1 + \varepsilon^4)^i \leq w(C)$ for at least one $i \in \mathbb{N}$.
- It holds that $w^+(\mathcal{C}) \geq (\varepsilon^2/200) \cdot w(M^*)$.

The second item is the same as Property (E) and the first item follows because, for $C \in \mathcal{C}$, if we let $W$ be $w(C)$ rounded down to the closest power of $(1 + \varepsilon^4)$ then the following holds:

1. The weight of each edge of $C$ is between $\varepsilon^{12}W$ and $2W$ by selection of $W$ and Properties (B),(C).
2. The gain $w^+(C)$ of $C$ is at most $w(C) \leq 2W$.
3. When the weight of each matched edge (i.e., an edge in $M$) of $C^M$ (recall that $C^M$ is the matching neighborhood of $C$) is rounded up and the weight of each unmatched edge of $C$ is rounded down to the nearest multiple of $\varepsilon^{12}W$, the gain of such $C$ is at least

$$w(C \cap M^*) - w(C^M) - \varepsilon^{12}W \cdot 4/\varepsilon \gg \varepsilon^{12}W\,.$$

4. $C$ consists of at most $4/\varepsilon \leq 64/\varepsilon^2 + 1$ vertices by Property (A).

Hence, $C$ is in the augmentation class of $(1 + \varepsilon^4)^i \leq w(C)$ for at least one $i \in \mathbb{N}$.

As can be seen in the above calculations, Lemma 8.9 is more restrictive than that required by the definition of an augmentation class. The reason is as follows. Lemma 8.9 shows the existence of very structured short augmentations that have a large total gain. However, no procedure for finding those augmentations is given. In the proof of Theorem 8.8 we will give such a procedure that efficiently finds augmentations that satisfy looser guarantees than those of Lemma 8.9. These

relaxed properties of augmentations correspond to the definition of augmentation classes. The more restrictive guarantees of Lemma 8.9 are then used to show that, for each augmentation class, the efficient procedure finds in expectation a set of vertex-disjoint augmentations with a gain comparable to that promised by Lemma 8.9 (see Lemma 8.12).

## 8.3 Finding Short Augmentations

In this section, we dive in to the details of the construction of our layered graphs and the filtering technique we introduced in Section 6.1.2. For this, we first parameterize the graph in terms of a random bipartition and the current matching (Section 8.3.1). Then, in Section 8.3.2, we present the formal definition of a layered graph, and in Section 8.3.3, we explain the filtering technique. Later, in Section 8.3.4, we show that our construction captures many of the paths described by Lemma 8.9.

### 8.3.1 Graph Parametrization

As a reminder, our goal is to reduce the problem of finding weighted augmentations to the problem of finding unweighted augmenting paths. As the first step in this process, we randomly choose a bipartite subgraph of the input graph. The graph obtained in this way is referred to as *parametrized*. We now describe this step.

**Bipartiteness:** Given $V$, we construct two disjoint sets $L$ and $R$ by uniformly at random assigning each vertex of $V$ to either $L$ or $R$.

We then consider only those edges whose one endpoint is in $L$ and the other is in $R$, and define

- $A \stackrel{\text{def}}{=} M \cap (L \times R)$, i.e., $A$ consists of the matching edges that connect $L$ and $R$,
- $B \stackrel{\text{def}}{=} (E \setminus M) \cap (L \times R)$, i.e., $B$ consists of the unmatched edges that connect $L$ and $R$.

**Parametrized graph:** We say that a given graph is *parametrized* if each vertex is assigned to $L$ or $R$ as described above. Given graph $G = (V, E)$ and matching $M$, we use $G^P = (L, R, A, B)$ to denote its parametrization.

### 8.3.2 Layered Graph

We now introduce the notion of *layered graph*, that plays a key role in enabling us to turn an algorithm for finding unweighted augmenting paths into an algorithm for finding weighted augmentations. We provide an example of such graphs in Figure 8.1.

**Definition 8.10** (Weighted layered graph)**.** *Let $G^P = (L, R, A, B)$ be a parametrized graph. Recall that $A$ is a subset of matched and $B$ is a subset of unmatched edges. Let $\tau^A \in \mathbb{R}^{k+1}_{\geq 0}$ and $\tau^B \in \mathbb{R}^k_{\geq 0}$ be two sequences of non-negative multiples of $\varepsilon^{12}$. Let $w : A \cup B \to \mathbb{R}_{\geq 0}$ be a weight function, and $W$ be a positive weight. Then, we use $\mathcal{L}(\tau^A, \tau^B, W, G^P) = (V_{\mathcal{L}}, E_{\mathcal{L}})$ to denote* layered graph *which is defined in two stages. First, we define $V_{\mathcal{L}}$ and $E_{\mathcal{L}}$ as follows*

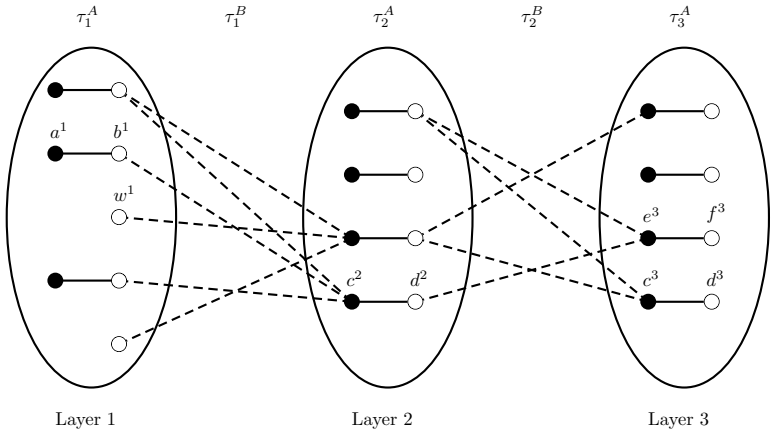



Figure 8.1 – A layered graph $\mathcal{L}$ consisting of 3 layers. In this example, we show only those vertices that have at least one edge of $\mathcal{L}$ incident to it. Full segments represent matched and dashed segments represent unmatched edges filtered in $\mathcal{L}$. The black vertices are in $L$ while the white vertices are in $R$. Pictorially, we think of a layered graph evolving from left to right. Notice that, since $(c, d)$ appears in both the 2-nd and the 3-rd layer, $\tau_2^A$ equals $\tau_3^A$.

- $V_{\mathcal{L}} = \{v^t, 1 \leq t \leq k+1 : v \in V\}$*; in other words,* $V_{\mathcal{L}}$ *represents the vertex set* $V$ *copied* $k+1$ *times. We use* Layer $t$ *to refer to the* $t$*-th copy of the vertices of* $V$*.*
- $E_{\mathcal{L}} = X \cup Y$*, where* $X$ *and* $Y$ *are defined as follows*

$$X = \{\{u^t, v^t\} : 1 \leq t \leq k+1, \{u, v\} \in A, \text{ and } w(\{u, v\}) \in ((\tau_t^A - \varepsilon^{12})\, W, \tau_t^A W]\}.$$

  *I.e., among the edges in layer* $t$ *we keep only those whose weight is relatively close from below to the threshold value* $\tau_t^A W$*.*

$$Y = \left\{ \{u^t, v^{t+1}\} : \begin{matrix} 1 \leq t \leq k, \{u, v\} \in B, u \in R, v \in L, \text{ and} \\ w(\{u, v\}) \in [\tau_t^B W, (\tau_t^B + \varepsilon^{12})\, W) \end{matrix} \right\}.$$

  *I.e., among the edges connecting layer* $t$ *and layer* $t+1$ *keep only those that are in* $B$ *(i.e., unmatched), that go from* $R$ *in layer* $t$ *to* $L$ *in layer* $t+1$*, and whose weight is relatively close from above to the threshold value* $\tau_t^B W$*.*

*In the second stage, we filter some of the vertices from* $\mathcal{L}$*.*

- ***Filtering step for intermediate layers.*** *For* $i \in \{2, 3, \ldots, k\}$ *and* $v \in V$*, remove* $v^i$ *if it is unmatched in* $X$*.*
- ***Filtering step for the first and the last layer.*** *For every vertex* $v^1$ *such that* $v \in R$ *and* $v^1$ *has no matched edge in layer* 1 *incident to it: keep* $v^1$ *only if* $v$ *is not incident to* $M$ *and* $\tau_1^A = 0$*; otherwise remove* $v^1$ *from* $V_{\mathcal{L}}$*. Analogously process every vertex* $v^{k+1}$*, i.e., if* $v \in L$ *and* $v^{k+1}$ *has no matched edge in layer* $k+1$ *incident to it: keep* $v^{k+1}$ *only if* $v$ *is not incident to* $M$ *and* $\tau_{k+1}^A = 0$*; otherwise remove* $v^{k+1}$ *from* $V_{\mathcal{L}}$*.*

When it is clear from the context, we use only $\mathcal{L}$ to denote $\mathcal{L}(\tau^A, \tau^B, W, G^P)$. We refer a reader to Figure 8.1 for an illustration of layered graphs.

Now we elaborate on why some of the vertices of $\mathcal{L}$ are filtered. First, we use layered graphs to find weighted augmentations via unweighted augmenting paths. The main idea here is to set $\tau^A$ so that sum of its elements is less than the sum of the elements of $\tau^B$. Intuitively, it guarantees that any alternating path that passes through all the layers could be used to improve the matching weight. Now, unlike in unweighted, in the weighted case a path can be weighted-augmenting even if on the path lay more matched than unmatched edges, e.g., path $a^1b^1c^2d^2e^3f^3$ in Figure 8.1 if $w(\{a^1, b^1\}) + w(\{c^2, d^2\}) + w(\{e^3, f^3\}) < w(\{b^1, c^2\} + w(\{d^2, e^3\})$. The point of the first and the last layer of $\mathcal{L}$ is exactly to capture this type of scenarios. However, sometimes there is a vertex in one of these layers, e.g., the first layer, that does not have any matched edge in $\mathcal{L}$ incident to it, as it is the case with $w^1$ in Figure 8.1. This might happen for two reasons. First, $w$ is not incident to any matched edge in $G$, in which case we keep $w^1$ only if $\tau_1^A = 0$. (This is the same as saying that $w^1$ is incident to a zero-weight matched edge.) The second case if $w$ is incident to a matched edge $e$ in $G$, but $w(e) \notin \left(\left(\tau_1^A - \varepsilon^{12}\right) W, \tau_1^A W\right]$. In this case, we should remove $w^1$ from $\mathcal{L}$ as otherwise it might not capture the case outlined above. For similar reasons vertices are removed from the last layer of $\mathcal{L}$. Furthermore, to make sure that a matching returned by the unweighted algorithm gives us augmenting paths that pass through all layers (exactly once), we remove the vertices left unmatched by $X$ in the intermediate layers. Thus we have no free vertices in the intermediate layers, therefore an augmenting path must start or end only in the first or the last layer.

### 8.3.3 Filtering – Properties of $(\tau^A, \tau^B)$ Pairs

Recall that layered graphs are defined with respect to $(\tau^A, \tau^B)$ pairs. Furthermore, such a pair determines which edges are kept in and between layers of the corresponding layered graph.

Observe that each layered graph has a property that a path passing through all the layers is an alternating path. So, it is useful to think of paths passing through all the layers as our candidates for weighted augmentations. Naturally, we would like that each candidate for weighted augmentations have a certain property, e.g., that the sum of weights of the unmatched edges is larger than the sum of weights of the matched edges. We control these properties by imposing some restrictions on the $(\tau^A, \tau^B)$ pairs that we consider. Next, we list those restrictions, and their summary is provided in Figure 8.2.

Recall that $\tau^A$ corresponds to matched and $\tau^B$ corresponds to unmatched edges. We look for short augmenting paths, so we set the length of $\tau^A$ to be $O(1/\varepsilon^2)$. The exact value is provided in Figure 8.2, property (A).

In our final algorithm, we look for augmenting paths in the graph obtained from $\mathcal{L}$ by removing all the edges in the first and the last layer. Furthermore, we require that those paths pass through all the layer. Hence, we require that $|\tau^A| = |\tau^B| + 1$ (property (B)).

As described earlier and as implied by the definition of layered graphs, we bucket the weights of edges in multiples of $\varepsilon^{12}W$. To reflect that, we set each entry of $\tau^A$ and $\tau^B$ to be of the form $\varepsilon^{12}k$, for $k \in \mathbb{N}$ (property (C)). Furthermore, we require that each unmatched edge we consider has a sufficiently large weight. To expresses that, we require that each entry of $\tau^B$ is at least $2\varepsilon^{12}$. Similarly, each matched edge which is not an end of a path is required to have non-negligible weight (property (D)).

Recall that our goal is to consider augmentations whose weight is close to $W$ (from the conditions, each augmentation has weight at least $2\varepsilon^{12}W$). Hence, we upper-bound the total sum of the weights of the edges corresponding to $\tau^B$ (property (E)).

Finally, we want to ensure that each augmentations leads to an increase in the weight. To that end, we require that the set of weights of the edges corresponding to $\tau^B$ is by at least $\varepsilon^{12}W$ larger than those corresponding to $\tau^A$ (property (F)). Observe that from this property and property (E) it implies that the sum of the weights of the edges corresponding to $\tau^A$ is upper-bounded by $(1 + \varepsilon^4 - \varepsilon^{12})W$.

A pair $(\tau^A, \tau^B)$ of sequence is called *good* if it has the following properties:

(A) The sequence $\tau^A$ consists of at most $\frac{2}{\varepsilon} \cdot \frac{16}{\varepsilon} + 1$ elements;
(B) The sequence $\tau^B$ has one element less than the sequence $\tau^A$;
(C) Each entry of $\tau^A$ and each entry of $\tau^B$ is a non-negative multiple of $\varepsilon^{12}$;
(D) Each entry of $\tau^B$ and each $\tau_i^A$, whenever $1 < i < |\tau^A|$, is at least $2\varepsilon^{12}$;
(E) $\sum_i \tau_i^B \leq 1 + \varepsilon^4$;
(F) $\sum_i \tau_i^B - \sum_i \tau_i^A \geq \varepsilon^{12}$.

Figure 8.2 – The definition of *good* $(\tau^A, \tau^B)$ pairs.

### 8.3.4 Short Augmentations in Layered Graphs

In this section, our goal is to show that each short augmentations of $\mathcal{C}$ as defined by Lemma 8.9 appears among the layered graphs our algorithm constructs.

We begin by showing that any alternating path in a layered graph could be, informally speaking, decomposed into a collection of "meaningful" augmentations in $G$. Namely, observe that an alternating path in a layered graph when translated to $G$ might contain cycles. In general, it might not be possible to augment a path intersecting itself. Nevertheless, we show that our layered graph is defined in such a way that every (not necessarily simple) path in $G$ obtained from a layered graph can be decomposed into cycles and paths each of which alone can be augmented.

**Lemma 8.11** (Decomposition on a path and even-length cycles)**.** *Let $G^P = (L, R, A, B)$ be a parametrized graph. Let $P$ be an alternating path in $\mathcal{L}(\tau^A, \tau^B, W, G^P)$. Let $S$ be the path obtained from $P$ by replacing each vertex $v^t$ by $v$. (Note that $S$ might not be a simple path.) Then, $S$ can be decomposed into a single simple path and a set of cycles. Furthermore, the edges in the path and the edges in each of the cycles alternate between $A$ and $B$.*

*Proof.* In this proof, we orient the edges of $\mathcal{L}$ as follows. Each edge $e = \{u^t, v^{t+1}\}$ connecting layer $t$ and layer $t + 1$ is oriented from $u^t$ to $v^{t+1}$. Each edge $e = \{x^s, y^s\}$ within a layer, where $x^s \in L$ and $y^s \in R$, is oriented from $x^s$ to $y^s$. Observe that in this way the head of each matched arc is in $R$ while the tail is in $L$. Also, the head of each unmatched arc is in $L$ while the tail is in $R$. Let $\vec{\mathcal{L}}$ denote the resulting oriented graph.

Observe that $P$ corresponds to a directed path $\vec{P}$ in $\vec{\mathcal{L}}$. Let $\vec{S}$ be the path obtained from $\vec{P}$ by replacing each vertex $v^t$ by $v$. Hence, disregarding the orientation in $\vec{S}$ results in $S$.

Let $\vec{C}$ be a cycle obtained by adding an arc between the last and the first vertex of $\vec{P}$. We will call that arc special. Let $\vec{S}'$ be obtained from $\vec{C}$ by replacing each vertex $v^t$ by $v$.

First, observe that each node in $\vec{S}'$ has in-degree equal to its out-degree. Hence, $\vec{S}'$ is an Eulerian graph. So, $\vec{S}'$ can be decomposed into arc-disjoint union of cycles. Let $\mathcal{C}$ be that collection of cycles excluding the cycle containing the special arc. Let $\vec{Q}$ be the path obtained by removing the special arc from the corresponding cycle of the decomposition. Note that by removing the special arc from $\vec{S}'$ we obtain $\vec{S}$. Hence, $\mathcal{C}$ and $\vec{Q}$ represent a decomposition of $\vec{S}$. Our goal is to show that each cycle of $\mathcal{C}$ and $\vec{Q}$ are alternating.

Towards a contradiction, assume that there is a vertex $v$ of a cycle of $\mathcal{C}$ or of $\vec{Q}$ such that the incoming and the outgoing arc both belong to $A$ or both belong to $B$. Then, $v$ should be both in $L$ and in $R$, which is in a contradiction with the parametrization. Hence, the lemma holds. □

We now use Lemma 8.9 to prove that specifically designed layered graphs contain many-by-weight vertex-disjoint augmentations. Specifically, we show that every augmentation considered in that lemma appears in at least one layered graph.

**Lemma 8.12.** *Let $\mathcal{C}$ be a collection of augmentations as defined by Lemma 8.9. Consider an augmentation $C \in \mathcal{C}$. Then, there exists a parametrization $G^P$, a choice a good pair $(\tau^A, \tau^B)$, and $W$ so that $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ contains a path $S$ passing through all the layers so that when Lemma 8.11 is applied on $S$ it results in a decomposition containing $C$. Furthermore, $W$ equals $(1+\varepsilon^4)^i \leq w(S)$, for some integer $i \geq 0$.*

*Proof.* We break the proof of Lemma 8.12 into two cases: $C$ is a cycle, and $C$ is a path. The proof is similar in both cases, and here we only present the proof for the case where $C$ is a cycle. For this case, we split the proof into three parts. First, we fix a parametrization of the graph, then define a layered graph based on this parametrization. And finally, we show that $C$ appears in the layered graph.

**Parametrization:** Observe that $C$ has even length, and let $C = v_1 \dots v_{2t} v_1$. Note that $t \leq 2/\varepsilon$. Without loss of generality, assume that $\{v_1, v_2\} \in M$. Consider a parametrization $G^P$ of the graph in which $v_i \in R$ for each even $i$, while $v_i \in L$ for each odd $i$. By the definition, $G^P$ contains $C$.

Let $a_1, \dots, a_t$ be the matched edges of $C$ appearing in that order, with $a_1 = \{v_1, v_2\}$. Similarly, let $b_1, \dots, b_t$ be the unmatched edges of $C$ appearing in that order, with $b_1 = \{v_2, v_3\}$.

**Layered graph:** Let $d \stackrel{\text{def}}{=} 16/\varepsilon$. We define a layered graph $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ so that it contains a (non-simple) alternating path which starts at $a_1$, goes around $C$ for $d$ times, and ends at $a_1$. More formally, $\mathcal{L}$ contains an alternating path $S$ passing through all the layers of the form

$$S = \underbrace{a_1 b_1 \dots a_t b_t}_{\text{repeated } d \text{ times}} a_1.$$

Note that $S$ consists of $2dt + 1$ many edges. Also, when Lemma 8.11 is applied to $S$ it outputs a collection of cycles in which $C$ appears $d$ times.

Define $W$ to be the largest value of the form $(1+\varepsilon^4)^i \leq w(S)$, where $i \geq 0$ is an integer. Note that, $W$ has the form as stated by lemma. Sequences $\tau^A$ and $\tau^B$ are defined as follows:

- Sequence $\tau^A$ has length $dt+1$ and $\tau^B$ has length $dt$.
- For every $a_i$ and for every integer $j$ such that $(j \equiv i \mod t)$, set $\tau^A_j$ to be the smallest $\varepsilon^{12}k$ such that $k$ is an integer and $\tau^A_j W \geq w(a_i)$.
- For every $b_i$ and for every integer $j$ such that $(j \equiv i \mod t)$, set $\tau^B_j$ to be the largest $\varepsilon^{12}k$ such that $k$ is an integer and $\tau^B_j W \leq w(b_i)$.

**Correctness:** We now show that $(\tau^A, \tau^B)$ is a good pair. To that end, show that $\tau^A$ and $\tau^B$ as defined above have all the properties stated by Figure 8.2.

Property (A)-(C) are ensured by the construction. It is easy to verify that $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ contains $S$.

We next show that property (D) holds as well. First, recall that from Lemma 8.9, for every $e \in C$ it holds $w(e) \geq (\varepsilon^6/64)w(C)$ (for the elements $C \setminus M$ we have even stronger guarantee). Then, we have

$$w(e) \geq \frac{\varepsilon^6}{64} w(C) \geq \frac{\varepsilon^6}{64} \frac{w(S)}{d+1} \geq 2\varepsilon^{12} w(S) \geq 2\varepsilon^{12} W,$$

and the property (D) follows by the definition of $\tau^A$ and $\tau^B$. In the third inequality, we use that $d = 16/\varepsilon$ and $\varepsilon \leq 1/16$.

To show Property (E), we observe that $\sum_i \tau^B_i W \leq w(S) \leq (1+\varepsilon^4)W$, implying $\sum_i \tau^B_i \leq (1+\varepsilon^4)$.

The entries of $\tau_A$ and $\tau_B$ represent discretized edge-weights of $C \cap M$ and $C \cap M^*$, respectively. Observe that $\tau^B W$ lower-bounds the edge-weights of $C \cap M^*$, while $\tau^A W$ upper-bounds the edge-weights of $C \cap M$. We will show that even when the weights are discretized, the difference between the weighted and unweighted edges of $S$ is significant. To that end, we compare $\sum_i \tau^A_i W$ and $\sum_i \tau^B_i W$. First, we have

$$\sum_i \tau^B_i W \geq d \cdot (w(C \cap M^*) - t\varepsilon^{12} W). \tag{8.2}$$

We also have

$$\begin{aligned} W &\leq w(S) \leq (d+1)w(C) = (d+1)\left(w(C \cap M) + w(C \cap M^*)\right) \\ &\leq 2(d+1)w(C \cap M^*) \leq 4dw(C \cap M^*). \end{aligned} \tag{8.3}$$

Combining (8.2) and (8.3) leads to

$$\begin{aligned} \sum_i \tau^B_i W &\geq d \cdot w(C \cap M^*)(1 - 4\varepsilon^{12} td) \geq d \cdot w(C \cap M^*)(1 - 16 \cdot 8\varepsilon^{10}) \\ &\geq d \cdot w(C \cap M^*)\left(1 - 8\varepsilon^9\right). \end{aligned} \tag{8.4}$$

The second inequality above uses that $t \leq 2/\varepsilon$ and $d = 16/\varepsilon$ while the third one uses that $\varepsilon \leq 1/16$.

Next, observe that from (8.3) and $\varepsilon \leq 1/16$ we have

$$\begin{aligned}\sum_i \tau_i^A W &\leq (d+1)(w(C \cap M) + \varepsilon^{12} tW) \leq (d+1)\left(\frac{w(C \cap M^*)}{1+\varepsilon/8} + \varepsilon^{12} tW\right) \\ &\leq \frac{(d+1)(1+8\varepsilon^{12}td)}{1+\varepsilon/8} w(C \cap M^*) \leq \frac{(d+1)(1+\varepsilon^8)}{1+\varepsilon/8} w(C \cap M^*). \end{aligned} \tag{8.5}$$

The second inequality above is due to Lemma 8.9. For the final inequality, we again use that $t \leq 2/\varepsilon$ and $d = 16/\varepsilon$.

From (8.4), (8.5) and the definition of $d$ we derive

$$\begin{aligned}\sum_i \tau_i^B W - \sum_i \tau_i^A W &\geq \left((16/\varepsilon)\left(1-8\varepsilon^9\right) - \frac{(1+16/\varepsilon)(1+\varepsilon^8)}{1+\varepsilon/8}\right) \cdot w(C \cap M^*) \\ &= \frac{(2+16/\varepsilon)\left(1-8\varepsilon^9\right) - (1+16/\varepsilon)(1+\varepsilon^8)}{1+\varepsilon/8} \cdot w(C \cap M^*) \\ &= \frac{1-16\varepsilon^7 - 129\varepsilon^8 - 16\varepsilon^9}{1+\varepsilon/8} \cdot w(C \cap M^*)\,, \end{aligned}$$

which is at least $\varepsilon^{12} W$ because $\varepsilon \leq 1/16$. The last chain of inequalities implies

$$\sum_i \tau_i^B - \sum_i \tau_i^A \geq \varepsilon^{12}, \tag{8.6}$$

hence showing that Property (F) holds as well.

For the case when $C$ is a path, we defer the proof to Appendix B as it is very similar to the previous case. □

## 8.4 Combining the Results

We are now ready to prove Theorem 8.8, and we start with the algorithm (Algorithm 8.2) that is used to prove this theorem.

**Lemma 8.13.** *Let $\mathcal{L}$ be a layered graph constructed by Algorithm 8.2. Use $w^+(\mathcal{L})$ to denote the maximum gain obtained by applying some vertex-disjoint augmenting paths of $\mathcal{L}$ where each of the paths passes through all the layers of $\mathcal{L}$. Define $\mathcal{L}'$ as the graph obtained by removing the edges in the first and the last layer of $\mathcal{L}$. Let $w(M_{\mathcal{L}'})$ be the total weight of the matching edges in $\mathcal{L}'$. Let $\mathcal{A}^{(\tau^A,\tau^B)}$ be the set of augmentations as obtained at Lines 8 to 12. Then*

$$w^+(\mathcal{A}^{(\tau^A,\tau^B)}) \geq \varepsilon^{20}\left(\frac{(1-\delta)w^+(\mathcal{L})}{2} - \frac{\delta w(M_{\mathcal{L}'})}{\varepsilon^{12}}\right).$$

*Proof.* Let $M_{\mathcal{L}'}$ denote the matching edges in $\mathcal{L}'$. Then, from the definition of $\tau^A$ and $\mathcal{L}$, we have

$$|M_{\mathcal{L}'}| \leq \frac{w(M_{\mathcal{L}'})}{\varepsilon^{12} W}. \tag{8.7}$$

**Algorithm 8.2:** Algorithm used by Theorem 8.8

**Input :** A weighted graph $G$, Approximation parameter $\varepsilon$, Weight $W$
**Output :** Augmentations corresponding to $W$

1 Partition the vertex set into $L$ and $R$ by assigning each vertex to one of the sets uniformly at random and independently. Let $G^P$ be the resulting parametrized graph.
2 Let $\mathcal{T}$ be the set of all good $(\tau^A, \tau^B)$ pairs, where good pairs are defined in Figure 8.2.
3 **for** *each* $(\tau^A, \tau^B) \in \mathcal{T}$ *in parallel* **do**
4 Define $\mathcal{L}'$ to be $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ with the edges from the first and the last layer removed.
5 Let $M' = \text{Unw-Bip-Matching}(\mathcal{L}', \delta)$ be the matching returned by a $(1-\delta)$-approximation bipartite unweighted matching algorithm (recall that $\mathcal{L}'$ is bipartite).
6 Let $M_{\mathcal{L}'}$ be the matching $M$ restricted to $\mathcal{L}'$.
7 Let $\mathcal{P}$ be the collection of augmentations in $M' \cup M_{\mathcal{L}'}$.
8 Let $\mathcal{A}^{(\tau^A, \tau^B)}$ be a set of augmentation in $G$. Initially, $\mathcal{A}^{(\tau^A, \tau^B)}$ is the empty set.
9 **for** *each* $P \in \mathcal{P}$ **do**
10 Apply Lemma 8.11 to $P$, i.e., decompose $P$ into a union of even-length cycles and a simple path in $G$. Let $\mathcal{C}$ be that decomposition.
11 Choose an augmentation $C \in \mathcal{C}$ that has the largest gain among the elements of $\mathcal{C}$.
12 If $C$ does not intersect any element of $\mathcal{A}^{(\tau^A, \tau^B)}$, add $C$ to $\mathcal{A}^{(\tau^A, \tau^B)}$.
13 Let $\mathcal{A}_W$ be a $\mathcal{A}^{(\tau^A, \tau^B)}$ set that maximizes gain over all $(\tau^A, \tau^B)$ pairs.
14 **return** $\mathcal{A}_W$

Let $\mathcal{C}$ be a collection of augmenting paths in $\mathcal{L}$ that have gain $w^+(\mathcal{L})$ and let $\mathcal{P}$ be a collection of vertex-disjoint paths found at Line 7. Observe that, as the first and the last layer of $\mathcal{L}$ consist of matched edges, each augmenting path passing through all the layers in $\mathcal{L}$ corresponds to an augmenting path in $\mathcal{L}'$ passing through all the layers as well, and vice-versa. Also, any augmenting path in $M' \cup M_{\mathcal{L}'}$ must pass through all the layers because there cannot be a free vertex with respect to $M_{\mathcal{L}'}$ except in the first and the last layer (see the filtering step in Definition 8.10). So we have that $|M'| = |M_{\mathcal{L}'}| + |\mathcal{P}|$. Since Unw-Bip-Matching returns a $(1-\delta)$-approximate matching,

$$|M_{\mathcal{L}'}| + |\mathcal{C}| \leq \frac{|M'|}{1-\delta} = \frac{|M_{\mathcal{L}'}| + |\mathcal{P}|}{1-\delta},$$

which, after simplification, gives

$$|\mathcal{P}| \geq ((1-\delta)|\mathcal{C}| - \delta|M_{\mathcal{L}'}|). \tag{8.8}$$

Next, from the definition of $\tau^A$ and $\tau^B$, each augmenting path in $\mathcal{L}$ increases the weight of the matching by at most $\sum_i (\tau_i^B + \varepsilon^{12})W \leq 2W$. So, we have $|\mathcal{C}| \geq {}^{w^+(\mathcal{L})}/_{(2W)}$, that together with (8.7) and (8.8) implies

$$|\mathcal{P}| \geq \frac{(1-\delta)w^+(\mathcal{L})}{2W} - \frac{\delta w(M_{\mathcal{L}'})}{\varepsilon^{12}W} \tag{8.9}$$

In the rest of the proof, we use the lower-bound on $|\mathcal{P}|$ to lower-bound $w^+(\mathcal{A}^{(\tau^A, \tau^B)})$.

First, consider a path $P \in \mathcal{P}$. When $P$ is translated to $G$ (Line 10), it might intersect itself and not being augmenting. From Lemma 8.11, $P$ can be decomposed into a collection of augmenting cycles and an augmenting path in $G$. Let $\mathcal{D}_P$ be the collection of components in this decomposition. From the definition of $\tau^A$ and $\tau^B$ we have that $P$ has gain at least $\varepsilon^{12}W$. Also, each $\tau_i^A$ and each $\tau_j^B$ is multiple of $\varepsilon^{12}$. This further implies that there is at least one component in $\mathcal{D}_P$ that has gain at least $\varepsilon^{12}W$. This implies that for every path of $\mathcal{P}$ there is an augmentation in $G$ that has gain at least $\varepsilon^{12}W$.

However, notice that although the paths in $\mathcal{P}$ are vertex-disjoint, when they are translated to $G$ they might share some vertices. This comes from the fact that in $\mathcal{L}$ the vertices of $G$ are copied in every layer. Now we want to account for these overlaps. First, each vertex of $G$ is copied $|\tau^A| + |\tau^B| + 1$ many times in $\mathcal{L}$. Hence, $|\tau^A| + |\tau^B| + 1$ many paths of $\mathcal{L}$ can intersect at the same vertex of $G$. Furthermore, each path in $\mathcal{L}$ consists of $|\tau^A| + |\tau^B| + 1$ vertices. Therefore, each component in $G$ obtained from a path of $\mathcal{P}$ overlaps with at most $(|\tau^A| + |\tau^B| + 1)^2 \leq 1/\varepsilon^8$ many other such components. This together with (8.9) implies that for $\mathcal{A}^{(\tau^A,\tau^B)}$ as defined at Lines 8 to 12 we have

$$\begin{aligned} w^+(\mathcal{A}^{(\tau^A,\tau^B)}) &\geq \varepsilon^{12} \cdot \varepsilon^8 W|\mathcal{P}| \\ &\geq \varepsilon^{12} \cdot \varepsilon^8 W \left( \frac{(1-\delta)w^+(\mathcal{L})}{2W} - \frac{\delta w(M_{\mathcal{L}'})}{\varepsilon^{12}W} \right) \\ &\geq \varepsilon^{20} \left( \frac{(1-\delta)w^+(\mathcal{L})}{2} - \frac{\delta w(M_{\mathcal{L}'})}{\varepsilon^{12}} \right), \end{aligned}$$

as desired. □

*Proof of Theorem 8.8.* Let $\mathcal{C}$ be the family of augmentations as defined by Lemma 8.9. From Lemma 8.12, for every $C \in \mathcal{C}$ there exists a parametrization of $G$, weight $W$, and a layered graph defined with respect to $W$ and considered by Algorithm 8.2 in which $C$ appears and passes through all its layers.[1] Let $\mathcal{C}_W \subseteq \mathcal{C}$ be the subcollection of $\mathcal{C}$ appearing in layered graphs defined with respect to $W$. Algorithm 8.2 fixes a parametrization of $G$ and then constructs layered graphs with respect to that parametrization. $C$ appears in a layered graph if its vertices are properly assigned to $L$ and $R$. Recall that each vertex gets assigned to one of the two sets with probability $1/2$ and independently of other vertices. Hence, the probability that $C \in \mathcal{C}$ remains in a random parametrization is at least $2^{-|C|} \geq 2^{-65/\varepsilon^2}$. This, implies that the expected gain obtained by applying all the augmentations of $\mathcal{C}_W$ that remain in one parametrization is at least $2^{-65/\varepsilon^2} w^+(\mathcal{C}_W)$. Our goal now is to show that Algorithm 8.2 finds augmentations whose gain is "close" to this remained gain.

Algorithm 8.2 finds augmentations in all the layered graphs independently (Line 7) and, hence, those augmentations might overlap. Furthermore, even a single augmentation from a layered graph when translated to $G$ might intersect itself. In both of these cases, our aim is to resolve overlap-conflicts while retaining large gain.

Note that the number of layered graphs for a constant $\varepsilon$ is $O(1)$. Hence, to show that Algorithm 8.2 retains large gain, it suffices to show that for a fixed $(\tau^A, \tau^B)$ pair the following is achieved:

[1] In our analysis, given a layered graph we only consider paths that pass through all the layers, i.e., only those paths that have at least one vertex in each of the layers. For the sake of brevity, we will omit specifying that a path passes through all the layers and, instead, only say that a path appears in a layered graph.

- Algorithm 8.2 finds many-by-weight augmentations of the corresponding layered graph.
- Algorithm 8.2 translates those augmentations to $G$ so to retain most of their gain (Lines 10 to 12).

Notice that these properties are essentially guaranteed by Lemma 8.13. So, it remains to count the number of layered graphs and apply Lemma 8.13 to conclude the proof. To that end, for a fixed $W$, let $\mathcal{L}$ be a layered graph that maximizes the gain. Let $w^+(\mathcal{L})$ be the maximum gain that can be obtained by applying vertex-disjoint augmenting paths of $\mathcal{L}$. We next lower-bound $w^+(\mathcal{L})$.

Observe that there are at most $(2/\varepsilon^{12}+2)^{65/\varepsilon^2}$ distinct $(\tau^A,\tau^B)$ pairs. (In this bound, the term "+2" comes from the fact that $\tau_i^A$ can be zero, and from the fact that a layer might not exist in which case we think that it has value $-1$.) Hence, in expectation over all parametrization, we have

$$w^+(\mathcal{L}) \geq 2^{-65/\varepsilon^2}(2/\varepsilon^{12}+2)^{-65/\varepsilon^2} w^+(\mathcal{C}_W) \geq \varepsilon^{900/\varepsilon^2} w^+(\mathcal{C}_W). \tag{8.10}$$

**Proving Properties A and B:** As in the statement of Lemma 8.13, $\mathcal{L}'$ is obtained by removing the edges from the first and the last layer of $\mathcal{L}$, and $\mathcal{A}^{(\tau^A,\tau^B)}$ is obtained at Lines 8 to 12. We will show that $\mathcal{A}^{(\tau^A,\tau^B)}$ satisfies the required properties. From it will follow that $\mathcal{A}_W$ returned at Line 13 satisfies those properties as well. Property A follows by the definition of layered graphs and our discussion above. So it remains to show that Property B holds as well.

As a reminder, $\mathcal{A}^{(\tau^A,\tau^B)}$ corresponds to $\mathcal{L}$ that maximizes the gain among all the layered graphs for $W$. From Lemma 8.13 and (8.10) we have that in expectation

$$w^+(\mathcal{A}^{(\tau^A,\tau^B)}) \geq (1-\delta)\varepsilon^{21+900/\varepsilon^2} w^+(\mathcal{C}_W) - \varepsilon^8 \delta w(M_{\mathcal{L}'}). \tag{8.11}$$

Let $w(M_W)$ be the weight of the matched edges of $G$ such that each edge has weight between $\varepsilon^{12}W$ and $2W$. Notice that a matched edge of $G$ appears at most $32/\varepsilon^2+1 \leq 1/\varepsilon^4$ many times in $\mathcal{L}$. Recall that each matching edge in $\mathcal{L}'$ has weight at least $\varepsilon^{12}W$. Hence,

$$w(M_{\mathcal{L}'}) \leq w(M_W)/\varepsilon^4. \tag{8.12}$$

Letting $\delta \stackrel{\text{def}}{=} \varepsilon^{28+900/\varepsilon^2}$, from (8.11) and (8.12) we obtain that $w^+(\mathcal{A}_W) \geq w^+(\mathcal{A}^{(\tau^A,\tau^B)}) \geq \varepsilon^{22+900/\varepsilon^2} \cdot w^+(\mathcal{C}_W) - \varepsilon^{32+900/\varepsilon^2} w(M_W)$. This proves that Property B holds as well.

**MPC implementation:** Algorithm 8.2 can be implemented in $U_M$ MPC rounds in the following way. Line 1 is implemented by collecting all the vertices to one machine, call that machine $\mu$, and randomly assigning them to $L$ and $R$ (in the way as described in Section 8.3.1). Then, the edge-set of $G$ is distributed across the machines, while the vertex sets $L$ and $R$ are sent to each of those machines. Notice that $\mu$ cannot send directly $L$ and $R$ to each of the machines, as it would result in outgoing communication of $\mu$ being at least $n\Gamma$ bits (recall that $\Gamma$ denotes the number of machines) which could be much larger than the memory of $\mu$. So, distributing $L$ and $R$ to each of the machines is performed in two steps as follows. First, $\mu$ locally splits $L \cup R$ into $\Gamma$ sets, so that each set has $\lceil n/\Gamma \rceil$ or $\lfloor n/\Gamma \rfloor$ vertices. Notice that in our case, $n \geq \Gamma$ and hence each of the sets is non-empty. Then, these sets are sent to the $\Gamma$ machines – one set per machine. In

the second step, each machine sends its set to each of the other machines. Since we assumed that the memory per machine is at least $n$, the total incoming and outgoing communication of a machine in this step does not exceed its memory. In the similar way, we can make sure that each machine knows the current matching $M$.

Then all $(\tau^A, \tau^B)$ pairs are generated by each machine. For constant $\varepsilon$, there are at most $O(1)$ many such pairs. For each $(\tau^A, \tau^B)$, each machine can then generate its part of $\mathcal{L}'$ as follows. Each vertex is replicated many times, where copy $v^{W,(\tau^A,\tau^B),t}$ corresponds to the parameters: weight $W$, a good pair $(\tau^A, \tau^B)$, and the layer $t$ it belongs to. Let $e = (u, v)$ be a parametrized edge of $G^P$. The edge $e$ is replicated *locally* to each layer for which it satisfies the weight requirements. If $e = \{u^i, v^{i+1}$ is not a matching edge, then we need to check if one of $u^i$ and $v^{i+1}$ is removed in the filtering step (see the description of layered graphs in Section 8.3.1). These checks are straightforward because each machine knows $M$.

After that Unw-Bip-Matching is called for each $(\tau^A, \tau^B)$, which uses $U_M$ MPC rounds and $O_\varepsilon(n)$ memory per machine, because $\delta$ is a function of only $\varepsilon$. Irrespective of how Unw-Bip-Matching stores its output, $\mathcal{P}$ can be collected on a fixed machine, which then does the remaining processing, and redistributes the output $\mathcal{A}_W$.

**Streaming implementation:** Algorithm 8.2 can be implemented in $U_S$ passes as follows. Random assignment to $L$ and $R$ can be done initially and stored. Then $O_\varepsilon(1)$ pairs $(\tau^A, \tau^B)$ are generated, and for each pair, Unw-Bip-Matching is then called, which uses $U_S$ passes and $O_\varepsilon(n \operatorname{poly}(\log(n)))$ memory. When an edge $e$ arrives in the stream, it is fed to those instances of Unw-Bip-Matching for which it appears in some layer. This happens if the edge $e$ and neighboring matching edges $e_1$ and $e_2$ satisfy weight and orientation (with respect to $L$ and $R$) requirements (see Section 8.3.1). Outputs of all the instances are then collected together after which the further processing is straightforward. □

# Part III

# Online Maximum Matching

## Beyond One-sided Vertex Arrivals

# 9 Overview of Online Matching

In this final part of the thesis, we study the maximum cardinality matching (MCM) problem in the online vertex-arrival and edge-arrival models.

Given the prominence of matching theory in combinatorial optimization, it is not surprising that the maximum matching problem was one of the first problems studied from the perspective of online algorithms and competitive analysis. In 1990, Karp, Vazirani, and Vazirani [65] introduced the online matching problem and studied it under one-sided bipartite arrivals. For such arrivals, Karp et al. noted that the trivial $1/2$-competitive greedy algorithm (which matches any arriving vertex to an arbitrary unmatched neighbor, if one exists) is optimal among deterministic algorithms for this problem. More interestingly, they provided an elegant randomized online algorithm for this problem, called RANKING, which achieves an optimal $(1 - 1/e)$ competitive ratio. (This bound has been re-proven many times over the years [15, 30, 31, 36, 46].) Online matching and many extensions of this problem under one-sided bipartite vertex arrivals were widely studied over the years, both under adversarial and stochastic arrival models. See recent work [23, 53, 54, 55] and the excellent survey of Mehta [78] for further references on this rich literature.

Despite our increasingly better understanding of one-sided online bipartite matching and its extensions, the problem of online matching under more general arrival models, including edge arrivals and general vertex arrivals, has remained staunchly defiant, resisting attacks. In particular, the basic questions of whether the trivial $1/2$ competitive ratio is optimal for the adversarial edge-arrival and general vertex-arrival models have remained tantalizing open questions in the online algorithms literature. In the following two chapters, we answer both of these questions.

## 9.1 Prior Work and Our Results

Here we outline the most relevant prior work and our contributions. Recall that an algorithm is *$\alpha$-competitive* if, for all input graphs and arrival orders, the ratio of the expected cardinality of the algorithm's output to the cardinality of a maximum matching in the input graph is at least $\alpha$. For deterministic algorithms, we drop the adjective "expected". For fractional algorithms, the ratio is between the size of the fractional matching output by the algorithm and the size of the maximum cardinality matching. As is standard in the online algorithms literature on maximization problems, we use upper bounds on $\alpha$ to refer to hardness results and lower bounds on $\alpha$ to refer to positive results.

**Edge Arrivals.** Arguably the most natural and the least restricted arrival model for online matching is the edge arrival model. In this model, edges are revealed one by one, and an online matching algorithm must decide immediately and irrevocably whether to match the edge on arrival or whether to leave both endpoints free to be possibly matched later.

On the hardness front, the problem is known to be strictly harder than the one-sided vertex arrival model of Karp et al. [65], which admits a competitive ratio of $1-1/e \approx 0.632$. In particular, Epstein et al. [35] gave an upper bound of $\frac{1}{1+\ln 2} \approx 0.591$ for this problem, recently improved by Huang et al. [55] to $2-\sqrt{2} \approx 0.585$. (Both bounds apply even to online algorithms with preemption; i.e., allowing edges to be removed from the matching in favor of a newly-arrived edge.) On the positive side, as pointed out by Buchbinder et al. [19], the edge arrival model has proven challenging, and results beating the $1/2$ competitive ratio were only achieved under various relaxations, including: random order edge arrival [50], bounded number of arrival batches [71], on trees, either with or without preemption [19, 85], and for bounded-degree graphs [19]. The above papers all asked whether there exists a randomized $(1/2+\Omega(1))$-competitive algorithm for adversarial edge arrivals (see also Open Question 17 in Mehta's survey [78]).

In this work, we answer this open question, providing it with a strong negative answer. In particular, we show that no online algorithm for *fractional* matching (i.e., an algorithm which immediately and irrevocably assigns values $x_e$ to edge $e$ upon arrival such that $\vec{x}$ is in the fractional matching polytope $\mathcal{P} = \{\vec{x} \geq \vec{0} \mid \sum_{e \ni v} x_e \leq 1 \ \forall v \in V\}$) is better than $1/2$ competitive. As any randomized algorithm induces a fractional algorithm with the same competitive ratio, this rules out any randomized online matching algorithm better than deterministic algorithms.

**Theorem 9.1.** *No fractional online algorithm is $1/2+\Omega(1)$ competitive for online matching under adversarial edge arrivals, even in bipartite graphs.*

This result shows that the study of relaxed variants of online matching under edge arrivals is not only justified by the difficulty of beating the trivial bound for this problem, but rather by its *impossibility*.

**General Vertex Arrivals.** In the online matching problem under vertex arrivals, vertices are revealed one at a time, together with their edges to their previously-revealed neighbors. An online matching algorithm must decide immediately and irrevocably upon arrival of a vertex whether to match it (or keep it free for later), and if so, who to match it to. The one-sided bipartite problem studied by Karp et al. [65] is precisely this problem when all vertices of one side of a bipartite graph arrive first. As discussed above, for this one-sided arrival model, the problem is thoroughly understood (even down to lower-order error terms [36]). Wang and Wong [86] proved that general vertex arrivals are strictly harder than one-sided bipartite arrivals, providing an upper bound of $0.625 < 1-1/e$ for the more general problem, later improved by Buchbinder et al. [19] to $\frac{2}{3+\phi^2} \approx 0.593$. Clearly, the general vertex arrival model is no harder than the online edge arrival model but is it *easier*? The answer is "yes" for *fractional* algorithms, as shown by combining our Theorem 9.1 with the 0.526-competitive fractional online matching algorithm under general vertex arrivals of Wang and Wong [86]. For *integral* online matching, however, the problem has proven challenging, and the only positive results for this problem, too, are for various relaxations, such as restriction to trees, either with or without preemption [19, 22, 85], for bounded-degree graphs [19], or (recently) allowing vertices to be matched during some known time interval [53, 55].

We elaborate on the last relaxation above. In the model recently studied by Huang et al. [53, 55] vertices have both arrival and departure times, and edges can be matched whenever both their endpoints are present. (One-sided vertex arrivals is a special case of this model with all online vertices departing immediately after arrival and offline vertices departing at $\infty$.) We note that any $\alpha$-competitive online matching under general vertex arrivals is $\alpha$-competitive in the less restrictive model of Huang et al. As observed by Huang et al., for their model an optimal approach might as well be greedy; i.e., an unmatched vertex $v$ should always be matched at its departure time if possible. In particular, Huang et al. [53, 55], showed that the RANKING algorithm of Karp et al. is optimal in this model, giving a competitive ratio of $\approx 0.567$. For general vertex arrivals, however, RANKING (and indeed any maximal matching algorithm) is no better than $1/2$ competitive, as is readily shown by a path on three edges with the internal vertices arriving first. Consequently, new ideas and algorithms are needed.

The natural open question for general vertex arrivals is whether a competitive ratio of $(1/2 + \Omega(1))$ is achievable by an *integral* randomized algorithm, without any assumptions (see e.g., [86]). In this work, we answer this question in the affirmative:

**Theorem 9.2.** *There exists a $(1/2 + \Omega(1))$-competitive randomized online matching algorithm for general adversarial vertex arrivals.*

## 9.2 Techniques

**Edge Arrivals.** All prior upper bounds in the online literature [19, 35, 36, 55, 65] can be rephrased as upper bounds for *fractional* algorithms; i.e., algorithms which immediately and irrevocably assign each edge $e$ a value $x_e$ on arrival, so that $\vec{x}$ is contained in the fractional matching polytope, $\mathcal{P} = \{\vec{x} \geq \vec{0} \mid \sum_{e \ni v} x_e \leq 1 \ \forall v \in V\}$. With the exception of [19], the core difficulty of these hard instances is uncertainty about "identity" of vertices (in particular, which vertices will neighbor which vertices in the following arrivals). Our hardness instances rely on uncertainty about the "time horizon". In particular, the underlying graph, vertex identifiers, and even arrival order are known to the algorithm, but the number of edges of the graph to be revealed (to arrive) is uncertain. Consequently, an $\alpha$-competitive algorithm must accrue high enough value up to each arrival time to guarantee a high competitive ratio at all points in time. As we shall show, for competitive ratio $1/2 + \Omega(1)$, this goal is at odds with the fractional matching constraints, and so such a competitive ratio is impossible. In particular, we provide a family of hard instances and formulate their prefix-competitiveness and matching constraints as linear constraints to obtain a linear program whose objective value bounds the optimal competitive ratio. Solving the obtained LP's dual, we obtain by weak duality the claimed upper bound on the optimal competitive ratio.

**General Vertex Arrivals.** Our high-level approach here will be to round online a fractional online matching algorithm's output, specifically that of Wang and Wong [86]. While this approach sounds simple, there are several obstacles to overcome. First, the fractional matching polytope is not integral in general graphs, where a fractional matching may have value, $\sum_e x_e$, some $3/2$ times larger than the optimal matching size. (For example, in a triangle graph with value $x_e = 1/2$ for each edge $e$.) Therefore, any general rounding scheme must lose a factor of $3/2$ on the competitive ratio compared to the fractional algorithm's value, and so to beat a competitive ratio of $1/2$ would require an online fractional matching with competitive ratio $> 3/4 > 1 - 1/e$, which is impossible.

To make matters worse, even in bipartite graphs, for which the fractional matching polytope is integral and offline lossless rounding is possible [2, 42], *online* lossless rounding of fractional matchings is impossible, even under one-sided vertex arrivals [23].

Despite these challenges, we show that a slightly better than $1/2$-competitive fractional matching computed by the algorithm of [86] can be rounded online without incurring too high a loss, yielding $(1/2 + \Omega(1))$-competitive randomized algorithm for online matching under general vertex arrivals.

To outline our approach, we first consider a simple method to round matchings online. When vertex $v$ arrives, we pick an edge $\{u, v\}$ with probability

$$z_u = \frac{x_{uv}}{\Pr[u \text{ free when } v \text{ arrives}]}$$

and add it to our matching if $u$ is free.

If $\sum_u z_u \leq 1$, this allows us to pick at most one edge per vertex and have each edge $e = \{u, v\}$ be in our matching with the right marginal probability, $x_e$, resulting in a lossless rounding. Unfortunately, we know of no better-than-$1/2$-competitive fractional algorithm for which this rounding guarantees $\sum_u z_u \leq 1$.

However, we observe that, for the correct set of parameters, the fractional matching algorithm of Wang and Wong [86] makes $\sum_u z_u$ close to one, while still ensuring a better-than-$1/2$-competitive fractional solution. Namely, as we elaborate later in Section 11.3, we set the parameters of their algorithm so that $\sum_u z_u \leq 1 + O(\epsilon)$, while retaining a competitive ratio of $1/2 + O(\epsilon)$. Now consider the same rounding algorithm with normalized probabilities: I.e., on $v$'s arrival, sample a neighbor $u$ with probability $z'_u = z_u / \max\{1, \sum_u z_u\}$ and match if $u$ is free. As the sum of $z_u$'s is slightly above one in the worst case, this approach does not drastically reduce the competitive ratio. But the normalization factor is still too significant compared to the competitive ratio of the fractional solution, driving the competitive ratio of the rounding algorithm slightly below $1/2$.

To account for this minor yet significant loss, we therefore augment the simple algorithm by allowing it, with small probability (e.g., say $\sqrt{\epsilon}$), to sample a second neighbor $u_2$ for each arriving vertex $v$, again with probabilities proportional to $z'_{u_2}$: If the first sampled choice, $u_1$, is free, we match $v$ to $u_1$. Otherwise, if the second choice, $u_2$, is free, we match $v$ to $u_2$. What is the marginal probability that such an approach matches an incoming vertex $v$ to a given neighbor $u$? Letting $F_u$ denote the event that $u$ is free when $v$ arrives, this probability is precisely

$$\Pr[F_u] \cdot \left( z'_u + z'_u \cdot \sqrt{\epsilon} \cdot \sum_w z'_w \cdot (1 - \Pr[F_w \mid F_u]) \right). \tag{9.1}$$

Here the first term in the parentheses corresponds to the probability that $v$ matches to $u$ via the first choice, and the second term corresponds to the same happening via the second choice (which is only taken when the first choice fails).

Ideally, we would like (9.1) to be at least $x_{uv}$ for all edges, which would imply a lossless rounding. However, as mentioned earlier, this is difficult and in general impossible to do, even in much more restricted settings including one-sided bipartite vertex arrivals. We therefore settle for showing that (9.1) is at least $x_{uv} = \Pr[F_u] \cdot z_u$ for *most* edges (weighted by $x_{uv}$). Even this goal, however,

is challenging and requires a nontrivial understanding of the correlation structure of the random events $F_u$. To see this, note that for example if the $F_w$ events are perfectly positively correlated, i.e., $\Pr[F_w \mid F_u] = 1$, then the possibility of picking $e$ as a second edge does not increase this edge's probability of being matched *at all* compared to if we only picked a single edge per vertex. This results in $e$ being matched with probability $\Pr[F_u] \cdot z'_u = \Pr[F_u] \cdot {}^{z_u}/_{\sum_w z_w} = {}^{x_{uv}}/_{\sum_w z_w}$, which does not lead to any gain over the $1/2$ competitive ratio of greedy. Such problems are easily shown not to arise if all $F_u$ variables are independent or negatively correlated. Unfortunately, positive correlation does arise from this process, and so we the need to control these positive correlations.

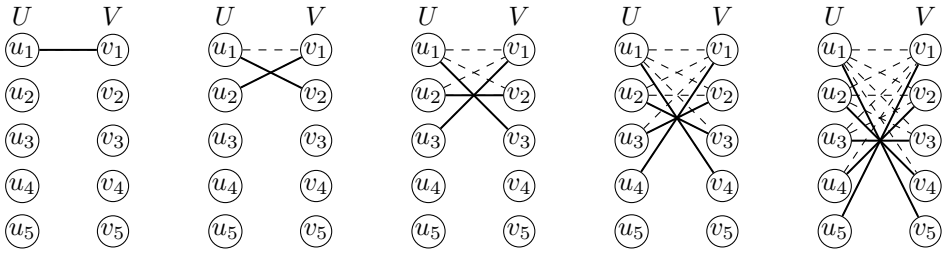
The core of our analysis is therefore dedicated to showing that even though positive correlations do arise, they are by and large rather weak. Our main technical contribution consists of developing techniques for bounding such positive correlations. The idea behind the analysis is to consider the primary choices and secondary choices of vertices as defining a graph, and showing that after a natural pruning operation that reflects the structure of dependencies, most vertices are most often part of a very small connected component in the graph. The fact that connected components are typically very small is exactly what makes positive correlations weak and results in the required lower bound on (9.1) for most edges (in terms of $x$-value), which in turn yields our $1/2 + \Omega(1)$ competitive ratio.

# 10 Hardness of Online Edge Arrivals

In this short chapter we prove the asymptotic optimality of the greedy algorithm for online matching under adversarial edge arrivals. As discussed briefly in Chapter 9, our main idea is to provide a "prefix hardness" instance, where an underlying input and the arrival order is known to the online matching algorithm, but the prefix of the input to arrive (or "termination time") is not. Consequently, the algorithm must accrue high enough value up to each arrival time, to guarantee a high competitive ratio at all points in time. As we show, the fractional matching constraints rule out a competitive ratio of $1/2 + \Omega(1)$ even in this model where the underlying graph is known.

**Theorem 10.1.** *There exists an infinite family of bipartite graphs with maximum degree $n$ and edge arrival order for which any online matching algorithm is at best $\left(\frac{1}{2} + \frac{1}{2n+2}\right)$-competitive.*

*Proof.* We will provide a family of graphs for which no fractional online matching algorithm has better competitive ratio. Since any randomized algorithm induces a fractional matching algorithm, this immediately implies our claim. The $n^{th}$ graph of the family, $G_n = (U \cup V, E)$, consists of a bipartite graph with $|U| = |V| = n$ vertices on either side. We denote by $u_i \in U$ and $v_i \in V$ the $i^{th}$ node on the left and right side of $G_n$, respectively. Edges are revealed in $n$ discrete rounds. In round $i = 1, 2, \ldots, n$, the edges of a perfect matching between the first $i$ left and right vertices arrive in some order. I.e., a matching of $u_1, u_2, \ldots, u_i$ and $v_1, v_2, \ldots, v_i$ is revealed. Specifically, edges $(u_j, v_{i-j+1})$ for all $i \geq j$ arrive. (See Figure 10.1 for example.) Let $M_i^*$ denote the unique maximum cardinality matching at the end of the $i$-th round. Intuitively, the difficulty for an algorithm attempting to assign much value to edges of $M_i^*$ is that the unique maximum matching $M_i^*$ changes every round, and no edge ever re-enters $M_i^*$ after getting removed from some $M_{i'}^*$ where $i' < i$.




(a) round 1 (b) round 2 (c) round 3 (d) round 4 (e) round 5

Figure 10.1 – $G_5$ together with arrival order. Edges of current (prior) round are solid (dashed).

Consider some $\alpha$-competitive fractional algorithm $\mathcal{A}$. We call the edge of a vertex $w$ in the (unique) maximum matching of the subgraph of $G_n$ following round $i$ the $i^{th}$ edge of $w$. For $i \geq j$, denote by $x_{i,j}$ the value $\mathcal{A}$ assigns to the $i^{th}$ edge of vertex $u_j$ (and of $v_{i-j+1}$); i.e., to $(u_j, v_{i-j+1})$. By feasibility of the fractional matching output by $\mathcal{A}$, we immediately have that $x_{i,j} \geq 0$ for all $i, j$, as well as the following matching constraints for $u_j$ and $v_j$. (For the latter, note that the $i^{th}$ edge of $v_{i-j+1}$ is assigned value $x_{i,j} = x_{i,i-(i-j+1)+1}$ and so the $i^{th}$ edge of $v_j$ is assigned value $x_{i,i-j+1}$).

$$\sum_{i=j}^{n} x_{i,j} \leq 1. \qquad (u_j \text{ matching constraint}) \tag{10.1}$$

$$\sum_{i=j}^{n} x_{i,i-j+1} \leq 1. \qquad (v_j \text{ matching constraint}) \tag{10.2}$$

On the other hand, as $\mathcal{A}$ is $\alpha$-competitive, we have that after some $k^{th}$ round – when the maximum matching has cardinality $k$ – algorithm $\mathcal{A}$'s fractional matching must have value at least $\alpha \cdot k$. (Else an adversary can stop the input after this round, leaving $\mathcal{A}$ with a worse than $\alpha$-competitive matching.) Consequently, we have the following competitiveness constraints.

$$\sum_{i=1}^{k} \sum_{j=1}^{i} x_{i,j} \geq \alpha \cdot k \qquad \forall k \in [n]. \tag{10.3}$$

Combining constraints (10.1), (10.2) and (10.3) together with the non-negativity of the $x_{i,k}$ yields the following linear program, LP($n$), whose optimal value upper bounds any fractional online matching algorithm's competitiveness on $G_n$, by the above.

$$\begin{array}{lrll} \text{Maximize } \alpha, & & & \\ \text{subject to} & \sum_{i=j}^{n} x_{i,j} & \leq 1 & \text{for all } j \in [n], \\ & \sum_{i=j}^{n} x_{i,i-j+1} & \leq 1 & \text{for all } j \in [n], \\ & \sum_{i=1}^{k} \sum_{j=1}^{i} x_{i,j} & \geq \alpha \cdot k & \text{for all } k \in [n], \\ & x_{i,j} & \geq 0 & \text{for all } i, j \in [n]. \end{array}$$

To bound the optimal value of LP($n$), we provide a feasible solution its LP dual, which we denote by Dual($n$). By weak duality, any dual feasible solution's value upper bounds the optimal value of LP($n$), which in turn upper bounds the optimal competitive ratio. Using the dual variables $\ell_j, r_j$ for the degree constraints of the $j^{th}$ left and right vertices respectively ($u_j$ and $v_j$) and dual variable $c_k$ for the competitiveness constraint of the $k^{th}$ round, we get the following dual linear program. Recall here again that $x_{i,i-j+1}$ appears in the matching constraint of $v_j$, with dual variable $r_j$, and so $x_{i,j} = x_{i,i-(i-j+1)+1}$ appears in the same constraint for $v_{i-j+1}$.)

$$\begin{array}{lrll} \text{Minimize} & \sum_{j=1}^{n} (\ell_j + r_j), & & \\ \text{subject to} & \sum_{k=1}^{n} k \cdot c_k & \geq 1, & \\ & \ell_j + r_{i-j+1} - \sum_{k=i}^{n} c_k & \geq 0 & \text{for all } i \in [n], j \in [i], \\ & \ell_j, r_j, c_k & \geq 0 & \text{for all } j, k \in [n]. \end{array}$$

Consider the following dual solution:

$$c_k = \frac{2}{n(n+1)} \qquad \text{for all } k \in [n] \text{ and ,}$$
$$\ell_j = r_j = \begin{cases} \frac{n-2(j-1)}{n(n+1)} & \text{if } j \leq n/2 + 1 \\ 0 & \text{if } n/2 + 1 < j \leq n. \end{cases}$$

We start by proving feasibility of this solution. The first constraint is satisfied with equality. For the second constraint, as $\sum_{k=i}^{n} c_k = \frac{2(n-i+1)}{n(n+1)}$ it suffices to show that $\ell_j + r_{i-j+1} \geq \frac{2(n-i+1)}{n(n+1)}$ for all $i \in [n], j \in [i]$. Note that if $j > n/2 + 1$, then $\ell_j = r_j = 0 > \frac{n-2(j-1)}{n(n+1)}$. So, for all $j$ we have $\ell_j = r_j \geq \frac{n-2(j-1)}{n(n+1)}$. Consequently, $\ell_j + r_{i-j+1} \geq \frac{n-2(j-1)}{n(n+1)} + \frac{n-2(i-j+1-1)}{n(n+1)} = \frac{2(n-i+1)}{n(n+1)}$ for all $i \in [n], j \in [i]$. Non-negativity of the $\ell_j, r_j, c_k$ variables is trivial, and so we conclude that the above is a feasible dual solution.

It remains to calculate this dual feasible solution's value. We do so for even $n$ for which

$$\sum_{j=1}^{n} (\ell_j + r_j) = 2 \cdot \sum_{j=1}^{n} \ell_j = 2 \cdot \sum_{j=1}^{n/2+1} \frac{n - 2(j-1)}{n(n+1)} = \frac{1}{2} + \frac{1}{2n+2},$$

completing the proof. The case of odd $n$ is similar, but it is unnecessary to establish the result of this theorem. □

**Remark 1.** Recall that Buchbinder et al. [19] and Lee and Singla [71] presented better-than-1/2-competitive algorithms for bounded-degree graphs and bounded number of arrival batches. Our upper bound above shows that a deterioration of the competitive guarantees as the maximum degree and number of arrival batches increase (as in the algorithms of [19, 71]) is inevitable.

**Remark 2.** Recall that the *asymptotic* competitive ratio of an algorithm is the maximum $c$ such that the algorithm always guarantees value at least $\mathcal{A}(G) \geq c \cdot \text{OPT}_G - b$ for some fixed $b > 0$. Our proof extends to this weaker notion of competitiveness easily, by revealing multiple copies of the hard family of Theorem 10.1 and letting $x_{ik}$ denote the average of its counterparts over all copies.

# 11 An Algorithm for General Vertex Arrivals

In this chapter we present a $(1/2 + \Omega(1))$-competitive randomized algorithm for online matching under general arrivals. As discussed in Chapter 9, our approach will be to round (online) a *fractional* online matching algorithm's output. Specifically, this will be an algorithm from the family of fractional algorithms introduced by Wang and Wong [86]. In Section 11.1 we describe this family of algorithms. To motivate our rounding approach, in Section 11.2 we first present a simple lossless rounding method for a $1/2$-competitive algorithm in this family. In Section 11.3 we then describe our rounding algorithm for a better-than-$1/2$-competitive algorithm in this family. Finally, in Section 11.4 we analyze this rounding scheme, and show that it yields a $(1/2 + \Omega(1))$-competitive algorithm.

## 11.1 Finding a Fractional Solution

In this section we revisit the algorithm of Wang and Wong [86], which beats the $1/2$ competitiveness barrier for online fractional matching under general vertex arrivals. Their algorithm (technically, family of algorithms) applies the primal-dual method to compute both a fractional matching and a fractional vertex cover – the dual of the fractional matching relaxation. The LPs defining these dual problems are as follows.

Primal-Matching:

$$\begin{array}{lrll} \text{Maximize} & \sum_{e \in E} x_e, & & \\ \text{subject to} & \sum_{u \in N(v)} x_{uv} & \leq 1 & \text{for all } u \in V, \\ & x_e & \geq 0 & \text{for all } e \in E. \end{array}$$

Dual-Vertex Cover:

$$\begin{array}{lrll} \text{Minimize} & \sum_{u \in V} y_u, & & \\ \text{subject to} & y_u + y_v & \geq 1 & \text{for all } e = \{u, v\} \in E, \\ & y_u & \geq 0 & \text{for all } u \in V. \end{array}$$

Before introducing the algorithm of [86], we begin by defining the fractional online vertex cover problem for vertex arrivals. When a vertex $v$ arrives, if $N_v(v)$ denotes the previously-arrived neighbors of $v$, then for each $u \in N_v(v)$, a new constraint $y_u + y_v \geq 1$ is revealed, which an online algorithm should satisfy by possibly increasing $y_u$ or $y_v$. Suppose $v$ has its dual value set to $y_v = 1 - \theta$. Then all of its neighbors should have their dual increased to at least $\theta$. Indeed, an algorithm may as well increase $y_u$ to $\max\{y_u, \theta\}$. The choice of $\theta$ therefore determines an

online fractional vertex cover algorithm. The increase of potential due to the newly-arrived vertex $v$ is thus $1 - \theta + \sum_{u \in N_v(v)} (\theta - y_u)^+$.[1] In [86] $\theta$ is chosen to upper bound this term by $1 - \theta + f(\theta)$ for some function $f(\cdot)$. The primal solution (fractional matching) assigns values $x_{uv}$ so as to guarantee feasibility of $\vec{x}$ and a ratio of $\beta$ between the primal and dual values of $\vec{x}$ and $\vec{y}$, implying $\frac{1}{\beta}$-competitiveness of this online fractional matching algorithm, by feasibility of $\vec{y}$ and weak duality. The algorithm, parameterized by a function $f(\cdot)$ and parameter $\beta$ to be discussed below, is given formally in Algorithm 11.1. In the subsequent discussion, $N_v(u)$ denotes the set of neighbors of $u$ that arrive before $v$.

**Algorithm 11.1:** Online general vertex arrival fractional matching and vertex cover

**Input :** A stream of vertices $v_1, v_2, \ldots v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v_i)$ are revealed.
**Output :** A fractional vertex cover solution $\vec{y}$ and a fractional matching $\vec{x}$.

Let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$.
**foreach** *$v$ in the stream* **do**
  Maximize $\theta$,
  subject to $\theta \leq 1$,
  $\sum_{u \in N_v(v)} (\theta - y_u)^+ \leq f(\theta)$.
  **foreach** $u \in N_v(v)$ **do**
    $x_{uv} \leftarrow \frac{(\theta - y_u)^+}{\beta} \left(1 + \frac{1-\theta}{f(\theta)}\right)$.
    $y_u \leftarrow \max\{y_u, \theta\}$.
  $y_v \leftarrow 1 - \theta$.

Algorithm 11.1 is parameterized by a function $f$ and a constant $\beta$. The family of functions considered by [86] are as follows.

**Definition 11.1.** *Let* $f_\kappa(\theta) := \left(\frac{1+\kappa}{2} - \theta\right)^{\frac{1+\kappa}{2\kappa}} \left(\theta + \frac{\kappa-1}{2}\right)^{\frac{\kappa-1}{2\kappa}}$. *We define* $W := \{f_\kappa \mid \kappa \geq 1\}$.

As we will see, choices of $\beta$ guaranteeing feasibility of $\vec{x}$ are related to the following quantity.

**Definition 11.2.** *For a given* $f : [0,1] \to \mathbb{R}_+$ *let*

$$\beta^*(f) := \max_{\theta \in [0,1]} 1 + f(1-\theta) + \int_\theta^1 \frac{1-t}{f(\theta)}\, d\theta.$$

For functions $f \in W$ this definition of $\beta^*(f)$ can be simplified to $\beta^*(f) = 1 + f(0)$, due to the observation (see [86, Lemmas 4,5]) that all functions $f \in W$ satisfy

$$\beta^*(f) = 1 + f(1-\theta) + \int_\theta^1 \frac{1-\theta}{f(\theta)}\, d\theta \qquad \forall \theta \in [0,1]. \tag{11.1}$$

As mentioned above, the competitiveness of Algorithm 11.1 for appropriate choices of $f$ and $\beta$ is obtained by relating the overall primal and dual values, $\sum_e x_e$ and $\sum_v y_v$. As we show (and rely on later), one can even bound individual vertices' contributions to these sums. In particular, for any vertex $v$'s arrival time, each vertex $u$'s contribution to $\sum_e x_e$, which we refer to as its *fractional degree*, $x_u := \sum_{w \in N_v(u)} x_{uw}$, can be bounded in terms of its dual value by this point, $y_u$, as follows.

[1] Here and throughout this chapter, we let $x^+ := \max\{0, x\}$ for all $x \in \mathbb{R}$.

**Lemma 11.3.** *For any vertex $u, v \in V$, let $y_u$ be the potential of $u$ prior to arrival of $v$. Then the fractional degree just before $v$ arrives, $x_u := \sum_{w \in N_v(u)} x_{uw}$, is bounded as follows:*

$$\frac{y_u}{\beta} \leq x_u \leq \frac{y_u + f(1 - y_u)}{\beta}.$$

Broadly, the lower bound on $x_u$ is obtained by lower bounding the increase $x_u$ by the increase to $y_u/\beta$ after each vertex arrival, while the upper bound follows from a simplification of a bound given in [86, Invariant 1] (implying feasibility of the primal solution), which we simplify using (11.1). See Appendix C for a full proof.

Another observation we will need regarding the functions $f \in W$ is that they are decreasing.

**Observation 11.4.** *Every function $f \in W$ is non-increasing in its argument in the range $[0, 1]$.*

*Proof.* As observed in [86], differentiating (11.1) with respect to $z$ yields $-f'(1-z) - \frac{1-z}{f(z)} = 0$, from which we obtain $f(z) \cdot f'(1-z) = z - 1$. Replacing $z$ by $1-z$, we get $f(1-z) \cdot f'(z) = -z$, or $f'(z) = -\frac{z}{f(1-z)}$. As $f(z)$ is positive for all $z \in [0, 1]$, we have that $f'(z) < 0$ for all $z \in [0, 1]$. □

The next lemma of [86] characterizes the achievable competitiveness of Algorithm 11.1.

**Lemma 11.5** ([86])**.** *Algorithm 11.1 with function $f \in W$ and $\beta \geq \beta^*(f) = 1 + f(0)$ is $\frac{1}{\beta}$ competitive.*

Wang and Wong [86] showed that taking $\kappa \approx 1.1997$ and $\beta = \beta^*(f_\kappa)$, Algorithm 11.1 is $\approx 0.526$ competitive. In later sections we show how to round the output of Algorithm 11.1 with $f_\kappa$ with $\kappa = 1 + 2\epsilon$ for some small constant $\epsilon$ and $\beta = 2 - \epsilon$ to obtain a $(1/2 + \Omega(1))$-competitive algorithm. But first, as a warm up, we show how to round this algorithm with $\kappa = 1$ and $\beta = \beta^*(f_1) = 2$.

## 11.2 Warmup: a 1/2-Competitive Randomized Algorithm

In this section we will round the 1/2-competitive fractional algorithm obtained by running Algorithm 11.1 with function $f(\theta) = f_1(\theta) = 1 - \theta$ and $\beta = \beta^*(f) = 2$. We will devise a lossless rounding of this fractional matching algorithm, by including each edge $e$ in the final matching with a probability equal to the fractional value $x_e$ assigned to it by Algorithm 11.1. Note that if $v$ arrives after $u$, then if $F_u$ denotes the event that $u$ is free when $v$ arrives, then edge $\{u, v\}$ is matched by an online algorithm with probability $\Pr[\{u, v\} \in M] = \Pr[\{u, v\} \in M \mid F_u] \cdot \Pr[F_u]$. Therefore, to match each edge $\{u, v\}$ with probability $x_{uv}$, we need $\Pr[\{u, v\} \in M \mid F_u] = x_{uv}/\Pr[F_u]$. That is, we must match $\{u, v\}$ with probability $z_u = x_{uv}/\Pr[F_u]$ conditioned on $u$ being free. The simplest way of doing so (if possible) is to pick an edge $\{u, v\}$ with the above probability $z_u$ always, and to match it only if $u$ is free. Algorithm 11.2 below does just this, achieving a lossless rounding of this fractional algorithm. As before, $N_v(u)$ denotes the set of neighbors of $u$ that arrive before $v$.

Algorithm 11.2 is well defined if for each vertex $v$'s arrival, $z$ is a probability distribution; i.e., $\sum_{u \in N_v(v)} z_u \leq 1$. The following lemma asserts precisely that. Moreover, it asserts that Algorithm 11.2 matches each edge with the desired probability.

**Algorithm 11.2:** Online vertex arrival warmup randomized fractional matching

**Input :** A stream of vertices $v_1, v_2, \ldots, v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v_i)$ are revealed.
**Output :** A matching $M$.

Let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$, let $M \leftarrow \emptyset$.
**foreach** *$v$ in the stream* **do**
  Update $y_u$'s and $x_{uv}$'s using Algorithm 11.1 with $\beta = 2$ and $f = f_1$.
  **foreach** $u \in N_v(v)$ **do**
    $z_u \leftarrow \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]}$. `// z_u is x_uv/(1−y_u) as shown later`
  Sample (at most) one neighbor $u \in N_v(v)$ according to $z_u$.
  **if** *a free neighbor $u$ is sampled* **then**
    Add $\{u, v\}$ to $M$.

**Lemma 11.6.** *Algorithm 11.2 is well defined, since for every vertex $v$ on arrival, $z$ is a valid probability distribution. Moreover, for each $v$ and $u \in N_v(v)$, it matches edge $\{u, v\}$ with probability $x_e$.*

*Proof.* We prove both claims in tandem for each $v$, by induction on the number of arrivals. For the base case ($v$ is the first arrival), the set $N_v(v)$ is empty and thus both claims are trivial. Consider the arrival of a later vertex $v$. By the inductive hypothesis we have that each vertex $u \in N_v(v)$ is previously matched with probability $\sum_{w \in N_v(u)} x_{wu}$. But by our choice of $f(\theta) = f_1(\theta) = 1 - \theta$ and $\beta = 2$, if $w$ arrives after $u$, then $y_u$ and $\theta$ at arrival of $w$ satisfy $x_{uw} = \frac{(\theta - y_u)^+}{\beta} \cdot \left(1 + \frac{1-\theta}{f(\theta)}\right) = (\theta - y_u)^+$. That is, $x_{uw}$ is precisely the increase in $y_u$ following arrival of $w$. On the other hand, when $u$ arrived we have that its dual value $y_u$ increased by $1 - \theta = \sum_{v' \in N_u(u)} (\theta - y_{v'})^+ = \sum_{v' \in N_u(u)} x_{uv'}$. To see this last step, we recall first that by definition of Algorithm 11.1 and our choice of $f(\theta) = 1 - \theta$, the value $\theta$ on arrival of $v$ is chosen to be the largest $\theta \leq 1$ satisfying

$$\sum_{\forall u \in N_v(v)} (\theta - y_u)^+ \leq 1 - \theta. \tag{11.2}$$

But the inequality (11.2) is an equality whether or not $\theta = 1$ (if $\theta = 1$, both sides are zero). We conclude that $y_u = \sum_{v' \in N_v(u)} x_{uv'}$ just prior to arrival of $v$. But then, by the inductive hypothesis, this implies that $\Pr[u \text{ free when } v \text{ arrives}] = 1 - y_u$ (yielding an easily-computable formula for $z_u$). Consequently, by (11.2) we have that when $v$ arrives $z$ is a probability distribution, as

$$\sum_{u \in N_v(v)} z_u = \sum_{u \in N_v(v)} \frac{(\theta - y_u)^+}{1 - y_u} \leq \sum_{u \in N_v(v):\, y_u \leq \theta} \frac{(\theta - y_u)^+}{1 - \theta} = \sum_{u \in N_v(v)} \frac{(\theta - y_u)^+}{1 - \theta} \leq 1.$$

Finally, for $u$ to be matched to a latter-arriving neighbor $v$, it must be picked and free when $v$ arrives, and so $\{u, v\}$ is indeed matched with probability

$$\Pr[\{u, v\} \in M] = \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]} \cdot \Pr[u \text{ is free when } v \text{ arrives}] = x_{uv}. \qquad \square$$

In the next section we present an algorithm which allows to round better-than-1/2-competitive algorithms derived from Algorithm 11.1.

## 11.3 An Improved Algorithm

In this section, we build on Algorithm 11.2 and show how to improve it to get a $(1/2 + \Omega(1))$ competitive ratio.

There are two concerns when modifying Algorithm 11.2 to work for a general function from the family $W$. The first is how to compute the probability that a vertex $u$ is free when vertex $v$ arrives, in Line 5. In the simpler version, we inductively showed that this probability is simply $1 - y_u$, where $y_u$ is the dual value of $u$ as of $v$'s arrival (see the proof of Lemma 11.6). With a general function $f$, this probability is no longer given by a simple formula. Nevertheless, it is easily fixable: We can either use Monte Carlo sampling to estimate the probability of $u$ being free at $v$'s arrival to a given inverse polynomial accuracy, or we can in fact exactly compute these probabilities by maintaining their marginal values as the algorithm progresses. In what follows, we therefore assume that our algorithm can compute these probabilities exactly.

The second and more important issue is with the sampling step in Line 6. In the simpler algorithm, this step is well-defined as the sampling probabilities indeed form a valid distribution: I.e., $\sum_{u \in N_v(v)} z_u \leq 1$ for all vertices $v$. However, with a general function $f$, this sum can exceed one, rendering the sampling step in Line 6 impossible. Intuitively, we can normalize the probabilities to make it a proper distribution, but by doing so, we end up losing some amount from the approximation guarantee. We hope to recover this loss using a second sampling step, as we mentioned in Section 9.2 and elaborate below.

Suppose that, instead of $\beta = 2$ and $f = f_1$ (i.e., the function $f(\theta) = 1 - \theta$), we use $f = f_{1+2\epsilon}$ and $\beta = 2 - \epsilon$ to define $x_{uv}$ and $y_u$ values. As we show later in this section, for an $\epsilon$ sufficiently small, we then have $\sum_{u \in N_v(v)} z_u \leq 1 + O(\epsilon)$, implying that the normalization factor is at most $1 + O(\epsilon)$. However, since the approximation factor of the fractional solution is only $1/2 + O(\epsilon)$ for such a solution, (i.e., $\sum_{\{u,v\} \in E} x_{uv} \geq (1/\beta) \cdot \sum_{u \in V} y_u$), the loss due to normalization is too significant to ignore.

Now suppose that we allow arriving vertices to sample a second edge with a small (i.e., $\sqrt{\epsilon}$) probability and match that second edge if the endpoint of the first sampled edge is already matched. Consider the arrival of a fixed vertex $v$ such that $\sum_{u \in N_v(v)} z_u > 1$, and let $z'_u$ denote the normalized $z_u$ values. Further let $F_w$ denote the event that vertex $w$ is free (i.e, unmatched) at the arrival of $v$. Then the probability that $v$ matches $u$ for some $u \in N_v(v)$ using either of the two sampled edges is

$$\Pr[F_u] \cdot \left( z'_u + z'_u \sqrt{\epsilon} \cdot \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \right), \tag{11.3}$$

which is the same expression from (9.1) from Section 9.2, restated here for quick reference. Recall that the first term inside the parentheses accounts for the probability that $v$ matches $u$ via the first sampled edges, and the second term accounts for the probability that the same happens via the second sampled edge. Note that the second sampled edge is used only when the first one is incident to an already matched vertex and the other endpoint of the second edge is free. Hence we have the summation of conditional probabilities in the second term, where the events are conditioned on the other endpoint, $u$, being free. If the probability given in (11.3) is $x_{uv}$ for all $\{u, v\} \in E$, we would have the same guarantee as the fractional solution $x_{uv}$, and the rounding

would be lossless. This seems unlikely, yet we can show that the quantity in (11.3) is at least $(1-\epsilon^2)\cdot x_{uv}$ for most (not by number, but by the total fractional value of $x_{uv}$'s) of the edges in the graph, showing that our rounding is *almost* lossless. We postpone further discussion of the analysis to Section 11.4 where we highlight the main ideas and proceed with the formal proof.

**Algorithm 11.3:** A randomized online matching algorithm under general vertex arrivals.

**Input** : A stream of vertices $v_1, v_2, \ldots, v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v)$ are revealed.
**Output** : A matching $M$.

1 Let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$, let $M \leftarrow \emptyset$.
2 **foreach** *$v$ in the stream* **do**
3  Update $y_u$'s and $x_{uv}$'s using Algorithm 11.1 with $\beta = 2-\epsilon$ and $f = f_{1+2\epsilon}$.
4  **foreach** $u \in N_v(v)$ **do**
   `// Compute` $\Pr[u$ `is free when` $v$ `arrives]` `as explained in Section 11.3`
5  $z_u \leftarrow \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]}$.
6  **foreach** $u \in N_v(v)$ **do**
7  $z'_u \leftarrow z_u / \max\left\{1, \sum_{u \in N_v(v)} z_u\right\}$.
8  Pick (at most) one $u_1 \in N_v(v)$ with probability $z'_{u_1}$.
9  **if** $\sum_{u \in N_v(v)} z_u > 1$ **then**
10  With probability $\sqrt{\epsilon}$, pick (at most) one $u_2 \in N_v(v)$ with probability $z'_{u_2}$.
   `// Probability of dropping edge` $\{u,v\}$ `is computed using Eq. (11.3).`
11  Drop $u_2$ with minimal probability ensuring $\{u_2, v\}$ is matched with probability at most $x_{u_2 v}$.
12  **if** *a free neighbor $u_1$ is sampled* **then**
13  Add $\{u_1, v\}$ to $M$.
14  **else if** *a free neighbor $u_2$ is sampled* **then**
15  Add $\{u_2, v\}$ to $M$.

Our improved algorithm is outlined in Algorithm 11.3. Up until Line 5, it is similar to Algorithm 11.2 except that it uses $\beta = 2-\epsilon$ and $f = f_{1+2\epsilon}$ where we choose $\epsilon > 0$ to be any constant small enough such that the results in the analysis hold. In Line 7, if the sum of $z_u$'s exceeds one we normalize the $z_u$ to obtain a valid probability distribution $z'_u$. In Line 8, we sample the first edge incident to an arriving vertex $v$. In Line 10, we sample a second edge incident to the same vertex with probability $\sqrt{\epsilon}$ if we had to scale down $z_u$'s in Line 7. Then in Line 11, we drop the sampled second edge with the minimal probability to ensure that no edge $\{u, v\}$ is matched with probability more than $x_{uv}$. Since (11.3) gives the exact probability of $\{u, v\}$ being matched, this probability of dropping an edge $\{u, v\}$ can be computed by the algorithm. However, to compute this, we need the conditional probabilities $\Pr[F_w \mid F_u]$, which again can be estimated using Monte Carlo sampling[2]. In the subsequent lines, we match $v$ to a chosen free neighbor (if any) among its chosen neighbors, prioritizing its first choice.

For the purpose of analysis we view Algorithm 11.3 as constructing a greedy matching on a directed acyclic graph (DAG) $H_\tau$ defined in the following two definitions.

**Definition 11.7** (Non-adaptive selection graph $G_\tau$)**.** *Let $\tau$ denote the random choices made by the vertices of $G$. Let $G_\tau$ be the DAG defined by all the arcs $(v, u_1)$, $(v, u_2)$ for all vertices $v \in V$. We call the arcs $(v, u_1)$* primary *arcs, and the arcs $(v, u_2)$ the* secondary *arcs.*

[2]It is also possible to compute them exactly if we allow the algorithm to take exponential time.

**Definition 11.8** (Pruned selection graph $H_\tau$)**.** *Construct $H_\tau$ from $G_\tau$ by removing all arcs $(v,u)$ (primary or secondary) such that there exists a* primary *arc $(v',u)$ with $v'$ arriving before $v$. We further remove a secondary arc $(v,u)$ if there is a primary arc $(v,u)$; i.e., if a vertex $u$ has at least one incoming primary arc, remove all incoming primary arcs that came after the first primary arc and all secondary arcs that came after or from the same vertex as the first primary arc.*

It is easy to see that the matching constructed by Algorithm 11.3 is a greedy matching constructed on $H_\tau$ based on order of arrival and prioritizing primary arcs. The following lemma shows that the set of matched vertices obtained by this greedy matching does not change much for any change in the random choices of a single vertex $v$, which will prove useful later on. It can be proven rather directly by an inductive argument showing the size of the symmetric difference in matched vertices in $G_\tau$ and $G_{\tau'}$ does not increase after each arrival besides the arrival of $v$, whose arrival clearly increases this symmetric difference by at most two. See Appendix C for details.

**Lemma 11.9.** *Let $G_\tau$ and $G_{\tau'}$ be two realizations of the random digraph where all the vertices in the two graphs make the same choices except for* one *vertex $v$. Then the number of vertices that have different* matched status *(free/matched) in the matchings computed in $H_\tau$ and $H_{\tau'}$ at any point of time is at most two.*

## 11.4 Analysis of the Improved Algorithm

In this section, we analyze the competitive ratio of Algorithm 11.3. We start with an outline of the analysis where we highlight the main ideas.

### 11.4.1 Outline of the Analysis

As described in Section 11.3, the main difference compared to the simpler 1/2-competitive algorithm is the change of the construction of the fractional solution, which in turn makes the rounding more complex. In particular, we may have at the arrival of a vertex $v$ that $\sum_{u\in N_v(v)} z_u > 1$. The majority of the analysis is therefore devoted to such "problematic" vertices since otherwise, if $\sum_{u\in N_v(v)} z_u \leq 1$, the rounding is lossless due to the same reasons as described in the simpler setting of Section 11.2. We now outline the main ideas in analyzing a vertex $v$ with $\sum_{u\in N_v(v)} z_u > 1$. Let $F_w$ be the event that vertex $w$ is free (i.e., unmatched) at the arrival of $v$. Then, as described in Section 11.3, the probability that we select edge $\{u,v\}$ in our matching is the minimum of $x_{uv}$ (because of the pruning in Line 11), and

$$\Pr[F_u]\cdot\left(z'_u + z'_u\sqrt{\varepsilon}\cdot\sum_{w\in N_v(v)} z'_w\cdot(1-\Pr[F_w \mid F_u])\right).$$

By definition, $\Pr[F_u]\cdot z_u = x_{uv}$, and the expression inside the parentheses is at least $z_u$ (implying $\Pr[\{u,v\}\in M] = x_{uv}$) if

$$1+\sqrt{\varepsilon}\cdot\sum_{w\in N_v(v)} z'_w\cdot(1-\Pr[F_w \mid F_u]) \geq \frac{z_u}{z'_u}. \tag{11.4}$$

To analyze this inequality, we first use the structure of the selected function $f = f_{1+2\epsilon}$ and the selection of $\beta = 2-\epsilon$ to show that if $\sum_{u\in N_v(v)} z_u > 1$ then several structural properties hold

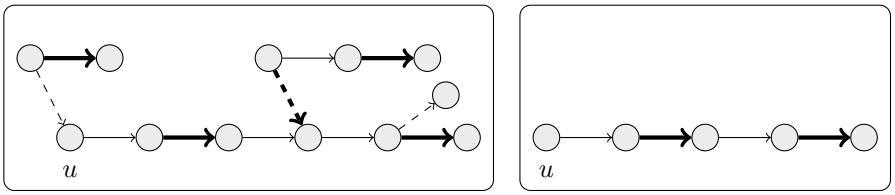



Figure 11.1 – Two examples of the component of $H_\tau$ containing $u$. Vertices are depicted from right to left in the arrival order. Primary and secondary arcs are solid and dashed, respectively. The edges that take part in the matching are thick.

(see Lemma 11.10 and Section 11.4.2 in Section 11.4.2). In particular, there are absolute constants $0 < c < 1$ and $C > 1$ (both independent of $\epsilon$) such that

1. $\sum_{u \in N_v(v)} z_u \leq 1 + C\epsilon$;
2. $z_u \leq C\sqrt{\epsilon}$ for every $u \in N_v(v)$; and
3. $c \leq \Pr[F_w] \leq 1 - c$ for every $w \in N_v(v)$.

The first property implies that the right-hand-side of (11.4) is at most $1 + C\epsilon$; and the second property implies that $v$ has at least $\Omega(1/\sqrt{\epsilon})$ neighbors and that each neighbor $u$ satisfies $z'_u \leq z_u \leq C\sqrt{\epsilon}$.

For simplicity of notation, we assume further in the high-level overview that $v$ has exactly $1/\sqrt{\epsilon}$ neighbors and each $u \in N_v(v)$ satisfies $z'_u = \sqrt{\epsilon}$. Inequality (11.4) would then be implied by

$$\sum_{w \in N_v(v)} (1 - \Pr[F_w \mid F_u]) \geq C\,. \tag{11.5}$$

To get an intuition why we would expect the above inequality to hold, it is instructive to consider the unconditional version:

$$\sum_{w \in N_v(v)} (1 - \Pr[F_w]) \geq c|N_v(v)| = c/\sqrt{\varepsilon} \gg C\,,$$

where the first inequality is from the fact that $\Pr[F_w] \leq 1 - c$ for any neighbor $w \in N_v(v)$. The large slack in the last inequality, obtained by selecting $\epsilon > 0$ to be a sufficiently small constant, is used to bound the impact of conditioning on the event $F_u$. Indeed, due to the large slack, we have that (11.5) is satisfied if the quantity $\sum_{w \in N_v(v)} \Pr[F_w|F_u]$ is not too far away from the same summation with unconditional probabilities, i.e., $\sum_{w \in N_v(v)} \Pr[F_w]$. Specifically, it is sufficient to show

$$\sum_{w \in N_v(v)} (\Pr[F_w|F_u] - \Pr[F_w]) \leq c/\sqrt{\varepsilon} - C\,. \tag{11.6}$$

We do so by bounding the correlation between the events $F_u$ and $F_w$ in a highly non-trivial manner, which constitutes the heart of our analysis. The main challenges are that events $F_u$ and $F_w$ can be positively correlated and that, by conditioning on $F_u$, the primary and secondary choices of different vertices are no longer independent.

We overcome the last difficulty by replacing the conditioning on $F_u$ by a conditioning on the component in $H_\tau$ (at the time of $v$'s arrival) that includes $u$. As explained in Section 11.3, the matching output by our algorithm is equivalent to the greedy matching constructed in $H_\tau$ and so the component containing $u$ (at the time of $v$'s arrival) determines $F_u$. But how can this component look like, assuming the event $F_u$? First, $u$ cannot have any incoming primary arc since then $u$ would be matched (and so the event $F_u$ would be false). However, $u$ could have incoming secondary arcs, assuming that the tails of those arcs are matched using their primary arcs. Furthermore, $u$ can have an outgoing primary and possibly a secondary arc if the selected neighbors are already matched. These neighbors can in turn have incoming secondary arcs, at most one incoming primary arc (due to the pruning in the definition of $H_\tau$), and outgoing primary and secondary arcs; and so on. In Figure 11.1, we give two examples of the possible structure, when conditioning on $F_u$, of $u$'s component in $H_\tau$ (at the time of $v$'s arrival). The left example contains secondary arcs, whereas the component on the right is arguably simpler and only contains primary arcs.

An important step in our proof is to prove that, for most vertices $u$, the component is of the simple form depicted to the right with probability almost one. That is, it is a path $P$ consisting of primary arcs, referred to as a primary path (see Definition 11.11) that further satisfies:

(i) it has length $O(\ln(1/\epsilon))$; and

(ii) the total $z$-value of the arcs in the blocking set of $P$ is $O(\ln(1/\epsilon))$. The blocking set is defined in Definition 11.12. Informally, it contains those arcs that if appearing as primary arcs in $G_\tau$ would cause arcs of $P$ to be pruned (or blocked) from $H_\tau$.

Let $\mathcal{P}$ be the primary paths of above type that appear with positive probability as $u$'s component in $H_\tau$. Further let $\mathrm{EQ}_P$ be the event that $u$'s component equals $P$. Then we show (for most vertices) that $\sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u]$ is almost one. For simplicity, let us assume here that the sum is equal to one. Then by the law of total probability and since $\sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] = 1$,

$$\begin{aligned}
&\sum_{w\in N_v(v)} (\Pr[F_w \mid F_u] - \Pr[F_w]) \\
&\qquad = \sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] \left( \sum_{w\in N_v(v)} (\Pr[F_w \mid F_u, \mathrm{EQ}_P] - \Pr[F_w]) \right) \\
&\qquad = \sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] \left( \sum_{w\in N_v(v)} (\Pr[F_w \mid \mathrm{EQ}_P] - \Pr[F_w]) \right),
\end{aligned}$$

where the last equality is because the component $P$ determines $F_u$. The proof is then completed by analyzing the term inside the parentheses for each primary path $P \in \mathcal{P}$ separately. As we prove in Lemma 11.13, the independence of primary and secondary arc choices of vertices is maintained after conditioning on $\mathrm{EQ}_P$.[3] Furthermore, we show that there is a bijection between the outcomes of the unconditional and the conditional distributions, so that the expected number of vertices that make different choices under this pairing can be upper bounded by roughly the length of the path plus the $z$-value of the edges in the blocking set. So, for a path $P$ as above, we

[3] To be precise, conditioning on a primary path $P$ with a so-called termination certificate $T$, see Definition 11.11. In the overview, we omit this detail and consider the event $\mathrm{EQ}_{P,T}$ (instead of $\mathrm{EQ}_P$) in the formal proof.

have that the expected number of vertices that make different choices in the paired outcomes is $O(\ln(1/\epsilon))$ which, by Lemma 11.9, implies that the expected number of vertices that change matched status is also upper bounded by $O(\ln(1/\epsilon))$. In other words, we have for every $P \in \mathcal{P}$ that

$$\sum_{w \in N_v(v)} (\Pr[F_w|\mathrm{EQ}_P] - \Pr[F_w]) \leq \sum_{w \in V} (\Pr[F_w|\mathrm{EQ}_P] - \Pr[F_w]) = O(\ln(1/\epsilon)),$$

which implies (11.6) for a small enough choice of $\varepsilon$. This completes the overview of the main steps in the analysis. The main difference in the formal proof is that not all vertices satisfy that their component is a short primary path with probability close to 1. To that end, we define the notion of *good* vertices in Section 11.4.4, which are the vertices that are very unlikely to have long directed paths of primary arcs rooted at them. These are exactly the vertices $v$ for which we can perform the above analysis for most neighbors $u$ (in the proof of the "key lemma") implying that the rounding is almost lossless for $v$. Then, in Section 11.4.5, we show using a rather simple charging scheme that most of the vertices in the graph are good. Finally, in Section 11.4.6, we put everything together and prove Theorem 9.2.

### 11.4.2 Useful Properties of $W$-Functions and Algorithm 11.3

For the choice of $f = f_{1+2\epsilon} \in W$ as we choose, we have $f(\theta) = (1+\varepsilon-\theta) \cdot \left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$. In Appendix C we give a more manageable upper bound for $f(\theta)$ which holds for sufficiently small $\epsilon$. Based on this simple upper bound on $f$ and some basic calculus, we obtain the following useful structural properties for the conditional probabilities, $z_u$, of Algorithm 11.3. See Appendix C.

**Lemma 11.10.** *(Basic bounds on conditional probabilities $z_u$) There exist absolute constants $c \in (0,1)$ and $C > 1/c > 1$ and $\epsilon_0 \in (0,1)$ such that for every $\epsilon \in (0,\epsilon_0)$ the following holds: for every vertex $v \in V$, if $y_u$ is the dual variable of a neighbor $u \in N_v(v)$ before $v$'s arrival and $\theta$ is the value chosen by Algorithm 11.1 on $v$'s arrival, then for $z_u$ as defined in Algorithm 11.3, we have:*

**(1)** *If $\theta \notin (c, 1-c)$, then $\sum_{u \in N_v(v)} z_u \leq 1$,*

**(2)** *If $\theta \in [0,1]$, then $\sum_{u \in N_v(v)} z_u \leq 1 + C\epsilon$,*

**(3)** *If $\sum_{u \in N_v(v)} z_u > 1$, then $z_u \leq C\sqrt{\epsilon}$ for every $u \in N_v(v)$,*

**(4)** *If $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ such that $z_u > 0$, one has $y_u \in [c/2, 1-c/2]$, and*

**(5)** *For all $u \in N_v(v)$, one has $z_u \leq 1/2 + O(\sqrt{\epsilon})$.*

The following corollary will be critical to our analysis: There exist absolute constants $c > 0$ and $\epsilon_0 > 0$ such that for all $\epsilon \in (0,\varepsilon)$, on arrival of any vertex $v \in V$, if $z$ as defined in Algorithm 11.3 satisfies $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ we have

$$c \leq \Pr[u \text{ is free when } v \text{ arrives}] \leq 1 - c.$$

*Proof.* By Lemma 11.10, **(1)** and **(4)** we have that if $\sum_{u \in N_v(v)} z_u > 1$, then $\theta \in (c, 1-c)$ ($c$ is the constant from Lemma 11.10), and for every $u \in N_v(v)$ one has

$$y_u \in [c/2, 1 - c/2]. \tag{11.7}$$

On the other hand, by Lemma 11.3 one has

$$\frac{y_u}{\beta} \leq x_u \leq \frac{y_u + f(1-y_u)}{\beta}, \tag{11.8}$$

where $x_u$ is the fractional degree of $u$ when $v$ arrives.

We now note that by Lemma 11.10, **(2)**, we have that Algorithm 11.3 matches every vertex $u$ with probability at least $x_u/(1+C\epsilon)$ (due to choices of primary arcs), and thus

$$\begin{aligned}
\Pr[u \text{ is free when } v \text{ arrives}] &\leq 1 - \frac{x_u}{1+C\epsilon} \\
&\leq 1 - \frac{y_u}{\beta(1+C\epsilon)} && \text{(by (11.8))} \\
&\leq 1 - \frac{c/2}{2(1+C\epsilon)} && \text{(by (11.7) and the setting } \beta = 2-\epsilon \leq 2) \\
&\leq 1 - c/5,
\end{aligned}$$

as long as $\epsilon$ is sufficiently small.

For the other bound we will use two facts. The first is that the since $f(y)$ is monotone decreasing by Observation 11.4 and since we picked $\beta > \beta^*(f) = 1 + f(0)$, we have that for any $y \leq 1 - c/2 \leq 1$,

$$y + f(1-y) \leq 1 - c/2 + f(0) < \beta - c/2. \tag{11.9}$$

Then, using the fact that by Line 11, Algorithm 11.3 matches every vertex $u$ with probability at most $x_u$, we obtain the second bound, as follows.

$$\begin{aligned}
\Pr[u \text{ is free when } v \text{ arrives}] &\geq 1 - x_u \\
&\geq 1 - \frac{y_u + f(1-y_u)}{\beta} && \text{(by (11.8))} \\
&\geq 1 - \frac{\beta - c/2}{\beta} && \text{(by (11.7) and (11.9))} \\
&\geq c/5. && (\beta = 2 - \epsilon < 2.5)
\end{aligned}$$

Choosing $c/5$ as the constant in the statement of the lemma, we obtain the result. □

Finally, for our analysis we will rely on the competitive ratio of the fractional solution maintained in Line 3 being $1/\beta$. This follows by Lemma 11.5 and the fact that for our choices of $\beta = 2 - \epsilon$ and $f = f_{1+2\epsilon}$ we have that $\beta \geq \beta^*(f)$. See Appendix C for a proof of this fact.

**Fact 11.1.** *For all sufficiently small $\epsilon > 0$, we have that $2 - \epsilon \geq \beta^*(f_{1+2\epsilon})$.*

### 11.4.3 Structural Properties of $G_\tau$ and $H_\tau$

In our analysis later, we focus on maximal primary paths (directed paths made of primary arcs) in $H_\tau$, in the sense that the last vertex along the primary path has no outgoing primary arc in $H_\tau$. The following definition captures termination certificates of such primary paths.

**Definition 11.11** (Certified Primary Path)**.** *A tuple $(P, T)$ is a certified primary path in $H_\tau$ if $P$ is a directed path of primary arcs in $H_\tau$ and either*

- **(a)** *the last vertex of $P$ does not have an outgoing primary arc in $G_\tau$ and $T = \emptyset$, or*
- **(b)** *the last vertex $u$ of $P$ has an outgoing primary arc $(u, w)$ in $G_\tau$ and $T = (u', w)$ is a primary arc in $H_\tau$ such that $u'$ precedes $u$ in the arrival order.*

To elaborate, a certified primary path $(P, T)$ is made of a (directed) path $P$ of primary arcs in $H_\tau$ and $T$ is a certificate of $P$'s termination in $H_\tau$ that ensures the last vertex $u$ in $P$ has no outgoing primary arc in $H_\tau$, either due to $u$ not picking a primary arc with $T = \emptyset$, or due to the picked primary arc $(u, w)$ being blocked by another primary arc $T = (u', w)$ which appears in $H_\tau$.

As described, $G_\tau$ and $H_\tau$ differ in arcs $(u, w)$ that are *blocked* by previous primary arcs to their target vertex $w$. We generally define sets of arcs which can block an edge, or a path, or a certified path from appearing in $H_\tau$ as in the following definition:

**Definition 11.12** (Blocking sets)**.** *For an arc $(u, w)$, define its* blocking set

$$B(u, w) := \{(u', w) \mid \{u', w\} \text{ is an edge and } u' \text{ arrived before } u\}$$

*to be those arcs, the appearance of any of which as primary arc in $G_\tau$ blocks $(u, v)$ from being in $H_\tau$. In other words, an arc $(u, v)$ is in $H_\tau$ as primary or secondary arc if and only if $(u, v)$ is in $G_\tau$ and none of the arcs in its blocking set $B(u, v)$ is in $G_\tau$ as a primary arc.*

*The blocking set of a path $P$ is simply the union of its arcs' blocking sets,*

$$B(P) := \bigcup_{(u,v)\in P} B(u, v)\,.$$

*The blocking set of a certified primary path $(P, T)$ is the union of blocking sets of $P$ and $T$,*

$$B(P, T) := B(P \cup T).$$

The probability of an edge, or path, or certified primary path appearing in $H_\tau$ is governed in part by the probability of arcs in their blocking sets appearing as primary arcs in $G_\tau$. As an arc $(v, u)$ is picked as primary arc by when $v$ arrives with probability roughly $z_u$ (more precisely, $z'_u \in [z_{vu}/(1+C\epsilon), z_{vu}]$, by Lemma 11.10), it will be convenient to denote by $z(v, u)$ and $z'(v, u)$ the values $z_u$ and $z'_u$ when $v$ arrives, and by $z(S) = \sum_{s\in S} z(s)$ and $z'(S) = \sum_{s\in S} z(s)$ the sum of $z$- and $z'$-values of arcs in a set of arcs $S$.

**Product distributions.** Note that by definition the distribution over primary and secondary arc choices of vertices are product distributions (they are independent). As such, their joint

distribution is defined by their marginals. Let $p_w$ and $s_w$ denote the distribution on primary and secondary arc choices of $w$, respectively. That is, for every $u \in N_w(w)$, $p_w(u)$ is the marginal probability that $w$ selects $(w,u)$ as its primary arc, and $s_w(u)$ is the marginal probability that $w$ selects $(w,u)$ as its secondary arc. Given our target bound (11.4), it would be useful to show that conditioning on $F_u$ preserves the independence of these arc choices. Unfortunately, conditioning on $F_u$ does not preserve this independence. We will therefore refine our conditioning later on the existence of primary paths in $H_\tau$, which as we show below maintains independence of the arc choices.

**Lemma 11.13.** *For a certified primary path $(P,T)$ let $\mathrm{EQ}_{(P,T)}$ be the event that the path $P$ equals a maximal connected component in $H_\tau$ and the termination of $P$ is certified by $T$. Then the conditional distributions of primary and secondary choices conditioned on $\mathrm{EQ}_{(P,T)}$ are product distributions; i.e., these conditional choices are independent. Moreover, if we let $\tilde{p}_w$ and $\tilde{s}_w$ denote the conditional distribution on primary and secondary choices of $w$, respectively, then*

$$\mathrm{TV}(p_w, \tilde{p}_w) \le z(R(w)) \qquad \text{and} \qquad \mathrm{TV}(s_w, \tilde{s}_w) \le z(R(w)),$$

*where $R(w) \subseteq \{w\} \times N_w(w)$ is the set of arcs leaving $w$ whose existence as primary arcs in $G_\tau$ is ruled out by conditioning on $\mathrm{EQ}_{(P,T)}$, and the union of these $R(w)$, denoted by $R(P,T)$, satisfies*

$$R(P,T) := \bigcup_w R(w) \subseteq B(P,T) \cup \{(w,r) \mid r \text{ is root of } P\} \cup \bigcup_{w \in P \cup \{w:\, T=(w,w')\}} \{w\} \times N_w(w). \quad (11.10)$$

*Proof.* We first bound the total variation distance between the conditional and unconditional distributions. For primary choices, conditioning on $\mathrm{EQ}_{(P,T)}$ rules out the following sets of primary arc choices. For vertex $w \notin P$ arriving before the root $r$ of $P$ this conditioning rules out $w$ picking any edge in $B(P,T)$ as primary arc. For vertices $w \notin P$ with $w$ arriving after the root $r$ of $P$ this conditioning rules out picking arcs $(w,r)$. Finally, this conditioning rules out some subset of arcs leaving vertices in $P \cup \{w : T = (w,w')\}$. Taking the union over these supersets of $R(w)$, we obtain (11.10). Now, the probability of each ruled out primary choice $(w,u) \in R(w)$ is zero under $\tilde{p}_w$ and $z'(w,u)$ under $p_w$, and all other primary choices have their probability increase, with a total increase of $\sum_{(w,u)\in R(w)} z'(w,u)$, from which we conclude that

$$\mathrm{TV}(p_w, \tilde{p}_w) = \frac{1}{2} \sum_{u \in N_w(w)} |p_w(u) - \tilde{p}_w(u)| = z'(R(w)) \le z(R(w)).$$

The proof for secondary arcs is nearly identical, the only differences being that the sets of ruled out secondary arcs can be smaller (specifically, secondary arcs to $w'$ such that $T = (u,w')$ are not ruled out by this conditioning), and the probability of any arc $(w,u)$ being picked as secondary arc of $w$ is at most $\sqrt{\epsilon} \cdot z'(w,u) \le z(w,u)$.

Finally, we note that primary and secondary choices for different vertices are independent. Therefore, conditioning on each vertex $w$ not picking a primary arc in its ruled out set $R(w)$ still yields a product distribution, and similarly for the distributions over secondary choices. □

It is easy to show that a particular certified primary path $(P,T)$ with high value of $z(B(P,T))$ is unlikely to appear in $H_\tau$, due to the high likelihood of arcs in its breaking set being picked as primary arcs. The following lemma asserts that the probability of a vertex $u$ being the root of *any* primary certified path $(P,T)$ with high $z(B(P,T))$ value is low.

**Lemma 11.14.** *For any $k \geq 0$ and any vertex $u$, we have*

$$\Pr\begin{bmatrix} H_\tau \text{ contains any certified primary path } (P,T) \\ \text{with } P \text{ rooted at } u \text{ and } z(B(P,T)) \geq k \end{bmatrix} \leq e^{-k/2}$$

*and*

$$\Pr\begin{bmatrix} H_\tau \text{ contains any primary path } P \text{ rooted at } u \\ \text{with } z(B(P)) \geq k \end{bmatrix} \leq e^{-k/2}.$$

*Proof.* We first prove the bound for certified primary paths. For a certified primary path $(P,T)$ where the last vertex of $P$ is $w$, define $P^*$ as follows:

$$P^* = \begin{cases} P & \text{if } T = \emptyset \\ P \cup \{(w,w'')\} & \text{if } T = (w',w''). \end{cases}$$

Observe that $z(B(P^*)) \geq k$ whenever $z(B(P,T)) \geq k$. This is trivial when $T = \emptyset$. To see this for the case $T = (w',w'')$, let $w$ be the last vertex of $P$, and note that $B(w',w'') \subseteq B(w,w'')$, as $w$ arrives after $w'$. Also note that for $(P,T)$ to be in $H_\tau$, we have that $P^*$ must be in $G_\tau$.

We say a directed primary path $P' = u \to u_1 \to \cdots \to u_{\ell-1} \to u_\ell$ is *$k$-minimal* if $z(B(P')) \geq k$ and $z(B(P' \setminus \{(u_{\ell-1},u_\ell)\})) < k$. For such a path $P'$, define $B^*(P')$ as follows: Initially set $B^*(P') = B(P \setminus \{(u_{\ell-1},u_\ell)\})$. Then from $B(u_{\ell-1},u_\ell)$, the breaking set of the last arc of $P'$, add arcs to $B^*(P')$ in reverse order of their sources' arrival until $z(B^*(P')) \geq k$.

Consider a certified primary path $(P,T)$ with $P$ rooted at $u$. If a $k$-minimal path rooted at $u$ which is not a prefix of $P^*$ is contained in $G_\tau$, then $(P,T)$ does not appear in $G_\tau$, and therefore it does not appear in $H_\tau$. On the other hand, if $z(B(P,T)) \geq k$ then for $(P,T)$ to appear in $H_\tau$, we must have that the (unique) $k$-minimal prefix $P'$ of $P^*$ must appear in $G_\tau$, and that none of the edges of $B^*(P')$ appear in $G_\tau$. Moreover, for any certified primary path with $z(B(P,T))$, conditioning on the existence of $P'$ in $G_\tau$ does not affect random choices of vertices with outgoing arcs in $B^*(P')$, as these vertices are not in $P'$. Since by Lemma 11.10 each arc $(w,w')$ appears in $G_\tau$ with probability $z'(v,u) \geq {}^{z(v,u)}/_{(1+C\epsilon)} \geq {}^{z(v,u)}/_2$, we conclude that for any $k$-minimal primary path $P'$ rooted at $u$, we have

$$\begin{aligned}
&\Pr[H_\tau \text{ contains any certified primary path } (P,T) \text{ with } z(B(P,T)) \geq k \mid P' \text{ is in } G_\tau] \\
&\quad \leq \Pr[\text{No edge in } B^*(P') \text{ is in } G_\tau \mid P' \text{ is in } G_\tau] \\
&\quad = \prod_{w \notin P'} (1 - \Pr[\text{Some primary edge in } B^*(P') \cap (\{w\} \times N_w(w)) \text{ is in } G_\tau]) \\
&\quad \leq \prod_{w \notin P'} \exp\left( - \sum_{(w,w') \in B(P,T) \times N_w(w)} {}^{z(w,w')}/_2 \right) \\
&\quad \leq \exp({}^{-z(B^*(P'))}/_2) \leq e^{-k/2}.
\end{aligned}$$

Taking total probability $\mathcal{P}_u$, the set of all $k$-minimal primary paths $P'$ rooted at $u$, we get that

indeed, since $u$ is the root of at most one $k$-minimal primary path in any realization of $G_\tau$,

$$\begin{aligned}&\Pr[H_\tau \text{ contains a certified primary path } (P,T) \text{ rooted at } u \text{ with } z(B(P,T)) \geq k] \\ &\quad \leq \sum_{P' \in \mathcal{P}_u} \underbrace{\Pr[H_\tau \text{ contains a } (P,T) \text{ with } z(B(P,T)) \geq k \mid P' \text{ is in } G_\tau]}_{\leq e^{-k/2}} \cdot \Pr[P' \text{ is in } G_\tau] \\ &\quad \leq e^{-k/2}.\end{aligned}$$

The proof for primary path is essentially the same as the above, taking $P^* = P$. □

### 11.4.4 Analyzing Good Vertices

Consider the set of vertices that are unlikely to be roots of long directed paths of primary arcs in $H_\tau$. In this section, we show that Algorithm 11.3 achieves almost lossless rounding for such vertices, and hence we call them *good* vertices. We start with a formal definition:

**Definition 11.15** (Good vertices)**.** *We say that a vertex $v$ is* good *if*

$$\Pr_\tau[H_\tau \textit{ has a primary path rooted at } v \textit{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \leq \varepsilon^6.$$

*Otherwise, we say $v$ is* bad*.*

As the main result of this section, for good vertices, we prove the following:

**Theorem 11.16.** *Let $v$ be a good vertex. Then*

$$\Pr[v \textit{ is matched on arrival}] \geq (1-\varepsilon^2) \cdot \sum_{u \in N_v(v)} x_{uv}.$$

**Notational conventions.** Throughout this section, we fix $v$ and let $z, z'$ be as in Algorithm 11.3. Moreover, for simplicity of notation, we suppose that the stream of vertices ends just before $v$'s arrival and so quantities, such as $G_\tau$ and $H_\tau$, refer to their values when $v$ arrives. For a vertex $u$, we let $F_u$ denote the event that $u$ is free (i.e., unmatched) when $v$ arrives. In other words, $F_u$ is the event that $u$ is free in the stream that ends just before $v$'s arrival.

To prove the theorem, first note that it is immediate if $\sum_{u \in N_v(v)} z_u \leq 1$: in that case, we have $z' = z$ and so the probability to match $v$ by a primary edge, by definition of $z_u$, is simply

$$\sum_{u \in N_v(v)} z_u \cdot \Pr[F_u] = \sum_{u \in N_v(v)} x_{uv}.$$

From now on we therefore assume $\sum_{u \in N_v(v)} z_u > 1$, which implies

(I) $\sum_{u \in N_v(v)} z'_u = 1$,

and moreover, by Lemma 11.10 and Section 11.4.2, for every $u \in N_v(v)$:

(II) $z_u \leq C\sqrt{\varepsilon}$,

(III) $z_u \leq (1 + C\varepsilon) \cdot z'_u$, and

(IV) $c \leq \Pr[F_u] \leq 1 - c$,

where $c$ is the constant of Section 11.4.2 and $C$ is the constant of Lemma 11.10.

We now state the key technical lemma in the proof of Theorem 11.16:

**Lemma 11.17.** *Consider a neighbor $u \in N_v(v)$ such that*

$$\Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length at least } 2000 \cdot \ln(1/\varepsilon) \mid F_u] \leq \varepsilon^2 . \quad (11.11)$$

*Then,*

$$\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \leq \varepsilon^{1/3} . \quad (11.12)$$

Note that the above lemma bounds the quantity $\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u]$, which will allow us to show that (11.4) holds and thus the edge $\{u, v\}$ is picked in the matching with probability very close to $x_{uv}$. Before giving the proof of the lemma, we give the formal argument why the lemma implies the theorem.

*Proof of Theorem 11.16.* Define $S$ to be the neighbors $u$ in $N_v(v)$ satisfying

$$\Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length at least } 2000 \cdot \ln(1/\varepsilon) \mid F_u] > \varepsilon^2 .$$

In other words, $S$ is the set of neighbors of $v$ that violate (11.11). As $v$ is good, we have

$$\begin{aligned}
\varepsilon^6 &\geq \Pr_\tau[H_\tau \text{ has a primary path rooted at } v \text{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \\
&\geq \sum_{u \in N_v(v)} z'_u \cdot \Pr[F_u] \cdot \Pr_\tau \left[ \begin{matrix} H_\tau \text{ has a primary path rooted at } u \\ \text{of length at least } 2000 \cdot \ln(1/\varepsilon) - 1 \end{matrix} \middle| F_u \right] \\
&\geq \sum_{u \in N_v(v)} z'_u \cdot \Pr[F_u] \cdot \Pr_\tau \left[ \begin{matrix} H_\tau \text{ has a primary path rooted at } u \\ \text{of length at least } 2000 \cdot \ln(1/\varepsilon) \end{matrix} \middle| F_u \right] \\
&\geq \sum_{u \in S} z'_u \cdot \Pr[F_u] \cdot \varepsilon^2.
\end{aligned}$$

The second inequality holds because $v$ selects the primary arc $(u, v)$ with probability $z'_u$ and, conditioned on $F_u$, $u$ cannot already have an incoming primary arc, which implies that $(u, v)$ is present in $H_\tau$. The last inequality follows from the choice of $S$.

By Property (III), $z_u \leq (1 + C\varepsilon) \cdot z'_u$ and so by rewriting we get

$$\sum_{u \in S} x_{uv} = \sum_{u \in S} z_u \cdot \Pr[F_u] \leq (1 + C\varepsilon) \cdot \sum_{u \in S} z'_u \cdot \Pr[F_u] \leq (1 + C\varepsilon) \cdot \varepsilon^4 \leq \varepsilon^3.$$

In other words, the contribution of the neighbors of $v$ in $S$ to $\sum_{u \in N_v(v)} x_{uv}$ is insignificant

compared to the contribution of all neighbors,

$$\sum_{u \in N_v(v)} x_{uv} = \sum_{u \in N_v(v)} z_u \cdot \Pr[F_u] \geq c, \tag{11.13}$$

where the inequality follows by the assumption $\sum_{u \in N_v(v)} z_u \geq 1$ and $\Pr[F_u] \geq c$ by Property (IV).

We proceed to analyze a neighbor $u \in N_v(v) \setminus S$. Recall that it is enough to verify (11.4) to conclude that edge $\{u, v\}$ is picked in the matching with probability $x_{uv}$. We have that

$$\begin{aligned}
&1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \\
&\geq 1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w]) - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && \text{(by Lemma 11.17)} \\
&\geq 1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot c - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && (\Pr[F_w] \leq 1 - c \text{ by (IV)}) \\
&= 1 + \sqrt{\varepsilon} c - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && \left( \sum_{w \in N_v(v)} z'_w = 1 \text{ by (I)} \right) \\
&\geq 1 + C\varepsilon && \text{(for } \varepsilon \text{ small enough)} \\
&\geq z_u / z'_u. && \text{(by (III))}
\end{aligned}$$

Therefore, by definition of $S$ and Lemma 11.17, we thus have that for every $u \in N_v(v) \setminus S$, the edge $\{u, v\}$ is taken in the matching with probability $x_{uv}$. Thus, the probability that $v$ is matched on arrival is, as claimed, at least

$$\sum_{u \in N_v(v) \setminus S} x_{uv} = \sum_{u \in N_v(v)} x_{uv} - \sum_{u \in S} x_{uv} \geq \sum_{u \in N_v(v)} x_{uv} - \varepsilon^3 \geq (1 - \varepsilon^2) \sum_{u \in N_v(v)} x_{uv} \,,$$

where the last inequality holds because we have $\sum_{u \in N_v(v)} x_{uv} \geq c$, as calculated in (11.13). □

It remains to prove the key lemma, Lemma 11.17, which we do here.

*Proof of Lemma 11.17.* For a certified primary path $(P, T)$ let $\mathrm{EQ}_{(P,T)}$ be the event as defined in Lemma 11.13, and let $\mathrm{IN}_{(P,T)}$ be the event that $P$ is a maximal *primary* path in $H_\tau$ and the termination of $P$ is certified by $T$. Further, let

$$\mathcal{C} = \{(P, T) : (P, T) \text{ is a certified primary path rooted at } u \text{ with } \Pr[\mathrm{IN}_{(P,T)}] > 0\}$$

be the set of certified primary paths rooted at $u$ that have a nonzero probability of being maximal in $H_\tau$. Then, by the law of total probability and since $\sum_{(P,T) \in \mathcal{C}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] = 1$ (since conditioning on $F_u$ implies in particular that $u$ has no incoming primary arc), we can rewrite the expression to bound, $\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w]$, as

$$\sum_{(P,T) \in \mathcal{C}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u, \mathrm{IN}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right). \tag{11.14}$$

We analyze this expression in two steps. First, in the next claim, we show that we can focus on

the case when the certified path $(P,T)$ is very structured and equals the component of $u$ in $H_\tau$. We then analyze the sum in that structured case.

**Claim 11.2.** *Let $\mathcal{P} \subseteq \mathcal{C}$ contain those certified primary paths $(P,T)$ of $\mathcal{C}$ that satisfy: $P$ has length less than $2000 \cdot \ln(1/\varepsilon)$ and $z(B(P,T)) \leq 2\ln(1/\varepsilon)$. Then, we have*

$$(11.14) \leq \sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left( \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right) + \varepsilon^{1/3}/2.$$

*Proof.* Define the following subsets of certified primary paths rooted at $u$:

$$\mathcal{C}_1 = \{(P,T) \in \mathcal{C} \mid P \text{ is of length at least } 2000 \cdot \ln(1/\varepsilon)\}$$
$$\mathcal{C}_2 = \{(P,T) \in \mathcal{C} \setminus \mathcal{C}_1 \mid z(B(P,T)) > 2\ln(1/\varepsilon)\}$$

Note that $\mathcal{P} = \mathcal{C} \setminus (\mathcal{C}_1 \cup \mathcal{C}_2)$. Since $u$ satisfies (11.11), we have that

$$\sum_{(P,T)\in\mathcal{C}_1} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \leq \varepsilon^2 \leq \varepsilon^{1/3}/6.$$

On the other hand, by Lemma 11.14 and $\Pr[F_u] \geq c$ (by Property (IV)), we have that

$$\sum_{(P,T)\in\mathcal{C}_2} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \leq c^{-1} \cdot \sum_{(P,T)\in\mathcal{C}_2} \Pr[\mathrm{IN}_{(P,T)}] \leq c^{-1} \cdot \varepsilon \leq \varepsilon^{1/3}/6.$$

In other words, almost all probability mass lies in those outcomes where one of the certified paths $(P,T) \in \mathcal{P}$ is in $H_\tau$. It remains to prove that, in those cases, we almost always have that the component of $u$ in $H_\tau$ equals the path $P$ (whose termination is certified by $T$). Specifically, let $\overline{\mathrm{EQ}_{(P,T)}}$ denote the complement of $\mathrm{EQ}_{(P,T)}$. We show

$$\Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right] \leq \varepsilon^{1/3}/7\,. \tag{11.15}$$

To see this, note that by the definition of the event $\mathrm{IN}_{(P,T)}$, if we restrict ourselves to primary edges then the component of $u$ in $H_\tau$ equals $P$. We thus have that for the event $\overline{\mathrm{EQ}_{(P,T)}}$ to be true at least one of the vertices in $P$ must have an incoming or outgoing *secondary* edge. Hence the expression $\Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right]$ can be upper bounded by

$$\Pr[\text{a vertex in } P \text{ has an incoming or outgoing secondary arc in } G_\tau \mid \mathrm{IN}_{(P,T)}] \tag{11.16}$$

Note that event $\mathrm{IN}_{(P,T)}$ is determined solely by choices of primary arcs. By independence of these choices and choices of secondary arcs, conditioning on $\mathrm{IN}_{(P,T)}$ does not affect the distribution of secondary arcs. So the probability that any of the nodes in $P$ selects a secondary edge is at most $\sqrt{\epsilon}$. Thus, by union bound, the probability that any of the $|P| \leq 2000 \cdot \ln(1/\epsilon)$ vertices in $P$ pick a secondary arc is at most $\sqrt{\epsilon} \cdot 2000 \cdot \ln(1/\epsilon)$. We now turn our attention to incoming secondary arcs. First, considering the secondary arcs that go into $u$, we have

$$c \leq \Pr[F_u] \leq \prod_{(w,u)\in B(v,u)} (1 - z(w,u)/2) \leq \exp(-z(B(v,u))/2),$$

because any arc $(w,u) \in B(v,u)$ appears as a primary arc in $G_\tau$ independently with probability at least $z(w,u)/2$ and the appearance of such an arc implies that $u$ has an incoming primary arc in $H_\tau$ and is therefore matched; i.e., the event $F_u$ is false in this case. We thus have $z(B(v,u)) \leq 2\ln(1/c)$. Further, since $(P,T) \notin \mathcal{C}_2$, we have $z(B(P)) \leq z(B(P,T)) \leq 2\ln(1/\varepsilon)$. Again using that the conditioning on $\text{IN}_{(P,T)}$ does not affect the distribution of secondary edges, we have that the probability of an incoming secondary arc to any vertex in $P$ is at most $\sqrt{\varepsilon} \cdot (2\ln(1/c) + 2\ln(1/\varepsilon))$. Thus, by union bound, the probability that any vertex in $P$ has an incoming or outgoing secondary arc conditioned on $\text{IN}_{(P,T)}$ is at most

$$\sqrt{\epsilon} \cdot 2000 \cdot \ln(1/\epsilon) + \sqrt{\varepsilon} \cdot (2\ln(1/c) + 2\ln(1/\varepsilon)) \leq \varepsilon^{1/3}/7,$$

for sufficiently small $\epsilon$, which implies (11.15) via (11.16).

We now show how the above concludes the proof of the claim. We have shown that each one of the two sets $\mathcal{C}_1, \mathcal{C}_2$ contributes at most $\varepsilon^{1/3}/6$ to (11.14) (where we use that $\sum_{w \in N_v(v)} z'_w = 1$). Hence,

$$(11.14) \leq \sum_{(P,T) \in \mathcal{P}} \Pr[\text{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \text{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) + 2\varepsilon^{1/3}/6.$$

This intuitively concludes the proof of the claim as (11.15) says that $\Pr[\text{EQ}_{(P,T)} | \text{IN}_{(P,T)}]$ is almost 1. The formal calculations are as follows. Since the event $\text{EQ}_{(P,T)}$ implies the event $\text{IN}_{(P,T)}$, we have that

$$\Pr[\text{EQ}_{(P,T)}] = \Pr[\text{EQ}_{(P,T)} \wedge \text{IN}_{(P,T)}] = \Pr[\text{IN}_{(P,T)}] - \Pr[\overline{\text{EQ}_{(P,T)}} \wedge \text{IN}_{(P,T)}],$$

which by (11.15) implies

$$\Pr[\text{EQ}_{(P,T)}] = \Pr[\text{IN}_{(P,T)}] \left(1 - \Pr\left[\overline{\text{EQ}_{(P,T)}} \mid \text{IN}_{(P,T)}\right]\right) \geq \Pr[\text{IN}_{(P,T)}] \left(1 - \varepsilon^{1/3}/7\right). \quad (11.17)$$

We use this to rewrite

$$\Pr[\text{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \text{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right).$$

Specifically, by law of total probability, it can be rewritten as the sum of the expressions (11.18) and (11.19) below:

$$\begin{aligned} &\Pr[\text{EQ}_{(P,T)} \wedge \text{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \text{EQ}_{(P,T)}, \text{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \\ &\quad = \Pr[\text{EQ}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \text{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \end{aligned} \quad (11.18)$$

and

$$\Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \overline{\mathrm{EQ}_{(P,T)}}, \mathrm{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right), \tag{11.19}$$

where (11.19) can be upper bounded as follows:

$$\begin{aligned}
(11.19) &\leq \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] && (\text{by } \sum_{w \in N_v(v)} z'_w \leq 1) \\
&\leq c^{-1} \cdot \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)}] && (\text{by } c \leq \Pr[F_u]) \\
&= c^{-1} \cdot \Pr[\mathrm{IN}_{(P,T)}] \cdot \Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right] \\
&\leq c^{-1} \cdot \frac{\Pr[\mathrm{EQ}_{(P,T)}]}{1 - \varepsilon^{1/3}/7} \cdot (\varepsilon^{1/3}/7) && (\text{by (11.15) and (11.17)}) \\
&\leq \Pr[\mathrm{EQ}_{(P,T)}] \cdot \varepsilon^{1/3}/6. && (\text{for } \epsilon \text{ small enough})
\end{aligned}$$

As at most one of the events $\{\mathrm{EQ}_{(P,T)}\}_{(P,T)\in\mathcal{P}}$ is true in any realization of $G_\tau$, we have that $\sum_{(P,T)\in\mathcal{P}} \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] \leq \sum_{(P,T)\in\mathcal{P}} \left(\Pr[\mathrm{EQ}_{(P,T)}] \cdot \varepsilon^{1/3}/6\right) \leq \varepsilon^{1/3}/6$. Thus, again using that $\sum_{w\in N_v(v)} z_w \leq 1$, we have that

$$\begin{aligned}
(11.14) &\leq \sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \\
&\qquad + 2\varepsilon^{1/3}/6 \\
&\leq \sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \\
&\qquad + 3\varepsilon^{1/3}/6,
\end{aligned}$$

as claimed. □

The previous claim bounded the contribution of certified primary paths in $\mathcal{C} \setminus \mathcal{P}$ to (11.14). The following claim bounds the contribution of paths in $\mathcal{P}$.

**Claim 11.3.** *Let $\mathcal{P} \subseteq \mathcal{C}$ contain those certified primary paths $(P,T)$ of $\mathcal{C}$ that satisfy: $P$ has length less than $2000 \cdot \ln(1/\varepsilon)$ and $z(B(P,T)) \leq 2\ln(1/\varepsilon)$. Then, we have*

$$\sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)}] \left( \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \leq \epsilon^{1/3}/2.$$

*Proof.* We prove the claim in two steps: first we construct a chain of distributions that interpolates between the unconditional distribution of $H_\tau$ and its conditional distribution, and then bound the expected number of vertices that change their matched status along that chain. For the remainder of the proof we fix the certified primary path $(P,T)$.

**Constructing a chain of distributions.** Let $H_\tau^{(0)}$ denote the unconditional distribution of $H_\tau$ when $v$ arrives, and let $H_\tau^{(n)}$ denote the distribution of $H_\tau$ conditioned on $\mathrm{EQ}_{(P,T)}$ when $v$ arrives. Here $n = |V|$ is the number of vertices in the input graph. For every $w \in V$ let $F_w^{(0)}$ denote the indicator of $w$ being free when $v$ arrives (unconditionally) and let $F_w^{(n)}$ denote the indicator variables of $w$ being free when $v$ arrives conditioned on $\mathrm{EQ}_{(P,T)}$. Note that $F^{(0)}$ is determined by $H_\tau^{(0)}$ and $F^{(n)}$ is determined by $H_\tau^{(n)}$. For $t = 0, \ldots, n$, we define distributions $H_\tau^{(t)}$ that interpolate between $H_\tau^{(0)}$ and $H_\tau^{(n+1)}$ as follows.

As in Lemma 11.13, for every $w \in V$ we denote the unconditional distribution of its primary choice by $p_w$, and the unconditional distribution of its secondary choice by $s_w$. Similarly, we denote the conditional distribution given $\mathrm{EQ}_{(P,T)}$ of the primary choice by $\widetilde{p}_w$ and the conditional distribution of the secondary choice by $\widetilde{s}_w$. For every $t = 0, \ldots, n$ the primary choice of vertices $w_j, j = 1, \ldots, t$ are sampled independently from $\widetilde{p}_{w_j}$, and the primary choices of vertices $w_j, j = t+1, \ldots, n$ are sampled independently from the unconditional distribution $p_{w_t}$. Similarly, secondary choices of vertices $w_j, j = 1, \ldots, t$ are sampled independently from $\widetilde{s}_{w_j}$ and secondary choices of vertices $w_j, j = t+1, \ldots, n$ are sampled independently from $s_{w_j}$. Note that $H_\tau^{(0)}$ is sampled from the unconditional distribution of $H_\tau$, and $H_\tau^{(n)}$ is sampled from the conditional distribution (conditioned on $\mathrm{EQ}_{(P,T)}$), as required, due to the independence of the conditional probabilities $\tilde{p}_{w_j}$ and $\tilde{s}_{w_j}$, by Lemma 11.13. For $t = 0, \ldots, n$ let $M_t$ denote the matching constructed by our algorithm on $H_\tau^{(t)}$, and let $F_w^{(t)}$ be the indicator variable for $w$ being free when $v$ arrives in the DAG sampled from $H_\tau^{(t)}$.

**Coupling the distributions of $H_\tau^{(t)}$.** We now exhibit a coupling between the $H_\tau^{(t)}, t = 0, \ldots, n$. Specifically, we will show that for every such $t$ the following holds.

$$\mathbb{E}\left[\sum_{q \in V} |F_q^{(t+1)} - F_q^{(t)}|\right] \leq 4z(R(w_{t+1})), \tag{11.20}$$

where $R(w_{t+1})$ is as defined in Lemma 11.13 with regard to the certified primary path $R(P,T)$. Recall that $z(R(w_{t+1}))$ is the total probability assigned to arcs leaving $w_{t+1}$ which are ruled out from being primary arcs in $G_\tau$ by conditioning on $\mathrm{EQ}_{(P,T)}$.

We construct the coupling by induction. The base case corresponds to $t = 0$ and is trivial. We now give the inductive step ($t \to t+1$). We write $w := w_{t+1}$ to simplify notation. Let $Z^p \in N_w(w)$ denote the primary choice of $w$ in $H_\tau^{(t)}$, and let $Z^s \in N_w(w)$ denote the secondary choice of $w$ in $N_w(w)$ (they are sampled according to the unconditional distributions $p_w$ and $s_w$ respectively). Let $\widetilde{Z}^p \in N_w(w)$ and $\widetilde{Z}^s \in N_w(w)$ be sampled from the conditional distributions $\widetilde{p}_w$ and $\widetilde{s}_w$ respectively, such that that the joint distributions $(Z^p, \widetilde{Z}^p)$ and $(Z^s, \widetilde{Z}^s)$ satisfy

$$\Pr[Z^p \neq \widetilde{Z}^p] = \mathrm{TV}(p_w, \widetilde{p}_w) \quad \text{and} \quad \Pr[Z^s \neq \widetilde{Z}^s] = \mathrm{TV}(s_w, \widetilde{s}_w). \tag{11.21}$$

First, we note that if $Z^p = \widetilde{Z}^p$ and $Z^s = \widetilde{Z}^s$, then $w = w_{t+1}$ is matched to the same neighbor under $H_\tau^{(t)}$ and $H_\tau^{(t+1)}$, and so $M_t = M_{t+1}$, due to the greedy nature of the matching constructed. Otherwise, by Lemma 11.9, at most two vertices have different matched status in $M_t$ and $M_{t+1}$ in the latter case (in the former case every vertex has the same matched status). To summarize,

we have, for $R(w)$ determined by $(P,T)$ as in Lemma 11.13, that

$$\begin{aligned}\mathbb{E}\left[\sum_{q\in V}|F_q^{(t+1)}-F_q^{(t)}|\right] &\le 2\cdot\Pr[Z^p\neq\widetilde{Z}^p \text{ or } Z^s\neq\widetilde{Z}^s]\\ &\le 2(\mathrm{TV}(p_w,\widetilde{p}_w)+\mathrm{TV}(s_w,\widetilde{s}_w)) \quad \text{(by (11.21) and union bound)}\\ &\le 4z(R(w)). \quad \text{(by Lemma 11.13)}\end{aligned} \tag{11.22}$$

This concludes the proof of the inductive step, and establishes (11.20). In particular, we get

$$\begin{aligned}\mathbb{E}\left[\sum_{q\in V}|F_q^{(n)}-F_q^{(0)}|\right] &\le \sum_{t=0}^{n-1}\mathbb{E}\left[\sum_{q\in V}|F_q^{(t+1)}-F_q^{(t)}|\right]\\ &\le \sum_{t=0}^{n-1}4z(R(w_{t+1})) \qquad \text{(by (11.22))}\\ &= 4z(R(P,T)),\end{aligned} \tag{11.23}$$

by the definition of $R(P,T)=\bigcup_w R(w)$ in Lemma 11.13.

We now finish the claim. First note that for any $(P,T)$ such that $P$ has length at most $2000\cdot\ln(1/\varepsilon)$ and $z(B(P,T))\le 2\ln(1/\varepsilon)$ one has $\sum_w z(R(w))=z(R(P,T))=O(\ln(1/\varepsilon))$. Indeed, by Lemma 11.13 and linearity of $z$, recalling that $u$ is the root of $P$ and that no vertex appears after $v$ (and thus $B(v,u)=\{(w,u)\mid w \text{ arrives between } u \text{ and } v\}$), we have

$$z(R(P,T))\le z(B(P,T))+z(B(v,u))+\sum_{w\in P\cup\{w:\,T=(w,w')\}} z\left(\{w\}\times N_w(w)\right). \tag{11.24}$$

We now bound the contribution to the above upper bound on $\sum_w z(R(w))=z(R(P,T))$ in (11.24). First, we have that $z(B(P,T))\le 2\ln(1/\varepsilon)$ by assumption of the lemma. To bound the contribution of $z(B(v,u))$, we note that by Property IV, we have

$$\begin{aligned}c\le\Pr[F_u]&=\prod_{e\in B(v,u)}(1-z_e)\le\exp\left(-\sum_{(w,u)\in B(v,u)}z(w,u)/2\right)\\ &\le\exp(-z(B(v,u))/2),\end{aligned}$$

because any arc $e=(w,u)$ appears as a primary arc in $G_\tau$ with probability $z'(w,u)\ge z(w,u)/2$, independently of other such arcs, and the appearance of any such an edge implies that $u$ has an incoming primary edge in $H_\tau$ when $v$ arrives and is therefore matched; i.e., the event $F_u$ is false in this case. We thus have $z(B(v,u))\le 2\ln(1/c)$. Finally, it remains to note that for every one of the at most $2000\cdot\ln(1/\epsilon)+1$ vertices $w\in P\cup\{w:\,T=(w,w')\}$ the contribution of $z(\{w\}\times N_w(w))$ to the right hand side of (11.24) is at most $1+C\epsilon\le 2$, by Lemma 11.10, **(2)**. Putting these bounds together, we get that for sufficiently small $\epsilon$,

$$z(R(P,T))\le 2\ln(1/\varepsilon)+2\ln(1/c)+2\cdot 2000\cdot\ln(1/\epsilon)+2=O(\ln(1/\epsilon)). \tag{11.25}$$

The term we wish to upper bound is at most

$$\begin{aligned}
&\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \\
\leq &\left(\max_{w\in N_v(v)} z'_w\right) \cdot \sum_{w\in N_v(v)} \left|\Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \Pr[F_w]\right| \\
\leq &C\sqrt{\epsilon} \cdot \sum_{w\in N_v(v)} \left|\Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \Pr[F_w]\right| && \text{(by Lemma 11.10, } \mathbf{(3)}\text{)} \\
= &C\sqrt{\epsilon} \cdot \mathbb{E}\left[\sum_{w\in N_v(v)} |F_w^{(n)} - F_w^{(0)}|\right] && \text{(by definition of } F^{(0)} \text{ and } F^{(n)}\text{)}
\end{aligned}$$

then, using (11.23) and (11.25), we find that the term we wish to upper bound is at most

$$\begin{aligned}
C\sqrt{\epsilon} \cdot \mathbb{E}\left[\sum_{w\in V} |F_w^{(n)} - F_w^{(0)}|\right] &\leq C\sqrt{\epsilon} \cdot z(R(P,T)) = O(\sqrt{\epsilon} \cdot \log(1/\epsilon)) \\
&\leq \epsilon^{1/3}/2,
\end{aligned}$$

completing the proof. Here, the first inequality is by (11.23) and the next equality is by (11.25). □

Finally, we obtain Lemma 11.17 by combining Claim 11.2 and Claim 11.3, to find that

$$\begin{aligned}
(11.14) \leq &\sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w]\right) \\
&+ \varepsilon^{1/3}/2 \\
\leq &\epsilon^{1/3}/2 + \epsilon^{1/3}/2 = \epsilon^{1/3}
\end{aligned}$$

as claimed.

□

### 11.4.5 Bounding the Impact of Bad Vertices

In this section, we show that we can completely ignore the bad vertices without losing too much. From the definition of good vertices, for a bad vertex $v$, we have that

$$\Pr_\tau[H_\tau \text{ has a primary path rooted at } v \text{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \geq \varepsilon^6.$$

As the main result of this section, we prove the following theorem:

**Theorem 11.18.** *The number of bad vertices is at most* $\varepsilon^3 \cdot \sum_{e\in E} x_e$.

To prove this, we first describe a charging mechanism in which, for each bad vertex, a charge of one is distributed among a subset of other vertices. Then, using the following supplementary lemma, we show that the total distributed charge over all vertices in the graph is at most $\varepsilon^3 \cdot \sum_{(u,v)\in E} x_{uv}$.

**Lemma 11.19.** *We call a primary path $P$ a* primary predecessor path *of $v$ if it ends at $v$. That is, $P = v_\ell \to v_{\ell-1} \to \cdots \to v_1 = v$. We have*

$$\Pr_\tau \begin{bmatrix} v \text{ has any primary predecessor path } P \text{ with} \\ z(B(P)) \leq 20 \cdot \ln(1/\epsilon) \text{ and } |P| \geq 1000 \cdot \ln(1/\epsilon) \end{bmatrix} \leq \epsilon^{10}.$$

*Proof.* We use the principle of deferred decisions and traverse the path backwards. Let $b$ be the current vertex, which is initially set to $v$. Consider all incoming arcs to $b$, say $(a_1, b), \ldots, (a_k, b)$ where we index $a$'s by time of arrival; i.e., $a_i$ arrives before $a_j$ if $i < j$ (and $b$ arrived before any $a_i$).

First consider the random choice of $a_1$ and see if it selected the arc $(a_1, b)$.

- If it does, then the path including $b$ in $H_\tau$ will use the arc $(a_1, b)$.
- Otherwise, if $a_1$ does not select the arc $(a_1, b)$, then go on to consider $a_2$ and so on.

If no $a_1, \ldots, a_k$ selects $b$, then the process stops; i.e., the primary path starts at this vertex since $b$ has no incoming primary arc. Otherwise let $i$ be the first index so that $(a_i, b)$ was selected. Then $(a_i, b)$ is in the primary path ending at $v$ in $H_\tau$. Now, observe that *no* $a_1, \ldots, a_{i-1}$ may be in the path in this case, because these vertices arrived before $a_i$ and after $b$. Moreover, we have not revealed any randomness regarding $a_{i+1}, \ldots, a_k$ that may appear later in the path. We can therefore repeat the above process with $b$ now set to $a_i$ and "fresh" randomness for all vertices we consider, as the random choices of arcs of all vertices are independent. We now show that this process, with good probability, does not result in a long predecessor path $P$ of low $z(B(P))$ value.

Recall from Lemma 11.10, **(5)**, that $z(u, v) \leq 3/5$ for all $(u, v) \in V \times V$. Suppose that $\sum_{i=1}^{k} z(a_i, b) \geq 4/5$. Let $j$ be the first index such that $\sum_{i=1}^{j} z(a_i, b) \geq 1/5$. Thus $\sum_{i=1}^{j} z(a_i, b) \leq 4/5$, and hence the probability that none of the first $j$ vertices select $b$ is at least $\prod_{i=1}^{j}(1 - z(a_i, b)) \geq 1 - \sum_{i=1}^{j} z(a_i, b) \geq 1/5$. Consequently, with probability at least $1/5$, vertex $b$ either has no predecessor or the increase to $z(B(P))$ is at least $1/5$.

In the other case, we have $\sum_{i=1}^{k} z(a_i, b) \leq 4/5$. Then the probability that $b$ has *no* predecessor is $\prod_{i=1}^{k}(1 - z(a_i, b)) \geq 1 - \sum_{i=1}^{k} z(a_i, b) \geq 1/5$.

Therefore, at any step in the above random process, with probability at least $1/5$, we either stop or increase $z(B(P))$ by $1/5$. Let $Z_i$ be an indicator variable for the random process either stopping or increasing $z(B(P))$ by at least $1/5$ at step $i$, and notice that according to the above random process, each $Z_i$ is lower bounded by an independent Bernoulli variable with probability $1/5$. Thus if we define $Z = \sum_{i \in [1000 \cdot \ln(1/\epsilon)]} Z_i$, we have $\mathbb{E}[Z] \geq 200 \cdot \ln(1/\epsilon)$, and thus by standard coupling arguments and Chernoff bounds, we have that

$$\Pr[Z \leq 100 \cdot \ln(1/\epsilon)] \leq \Pr\left[Z \leq (1 - 1/2) \cdot \mathbb{E}[Z]\right] \leq e^{-(1/2)\cdot(1/2)^2 \cdot 200 \cdot \ln(1/\epsilon)} \leq \epsilon^{10}.$$

But if the path does not terminate within $1000 \cdot \ln(1/\epsilon)$ steps and $Z \geq 100 \cdot \ln(1/\varepsilon)$, then $z(B(P)) \geq 20 \cdot \ln(1/\epsilon)$. □

We now prove Theorem 11.18.

*Proof of Theorem 11.18.* By Lemma 11.14, the probability that $H_\tau$ has a primary path $P$ with $z(B(P)) \geq 20 \cdot \ln(1/varepsilon)$ starting at $v$ is at most $\varepsilon^{10}$. Thus, for a bad vertex $u$, the probability that $H_\tau$ has some primary path $P$ rooted at $u$ with $|P| \geq 2000 \cdot \ln(1/\varepsilon)$ and $z(B(P)) \leq 20 \cdot \ln(1/\epsilon)$ is at least $\varepsilon^6 - \varepsilon^{10} \geq \varepsilon^6/2$.

Let $k = 20 \cdot \ln(1/\varepsilon)$ and $\ell = 2000 \cdot \ln(1/\varepsilon)$. Let $\mathcal{P}_u$ be the set of all primary paths $P$ rooted at $u$ such that $z(B(P)) \leq k$ and $|P| = \ell$ starting at $u$. Since all such primary paths with length more than $\ell$ are extensions of those with length exactly $\ell$, we have $\sum_{P \in \mathcal{P}_u} \Pr[P \text{ is in } H_\tau] \geq \epsilon^6/2$. For each such path $P \in \mathcal{P}_u$, consider the two vertices $w_\ell^P$ and $w_{\ell-1}^P$ at distances $\ell$ and $\ell - 1$ respectively from $u$. For each such vertex $w_j^P$ ($j \in \{\ell - 1, \ell\}$), charge $(2/\varepsilon^6) \cdot \Pr[P \text{ is in } H_\tau] \cdot y_{w_j^P}$. Then the sum of these charges is

$$\sum_{P \in \mathcal{P}_u} (2/\varepsilon^6) \cdot \Pr[P \text{ is in } H_\tau] \cdot \underbrace{(y_{w_\ell^P} + y_{w_{\ell-1}^P})}_{\geq 1} \geq (2/\epsilon^6) \cdot \sum_{P \in \mathcal{P}_u} \Pr[P \text{ is in } H_\tau]$$
$$\geq 1.$$

Notice that the fact $(y_{w_\ell^P} + y_{w_{\ell-1}^P}) \geq 1$ follows because $y_w$'s form a feasible dual solution (to the vertex cover problem).

On the other hand, consider how many times each vertex is charged. For this, for every vertex $w$, let $\mathcal{Q}_w$ be the set of primary predecessor paths $Q$ of $u$ such that $|Q| = \ell - 1$ and $z(B(P)) \leq k$. As $|Q| = \ell - 1 \geq 1000 \cdot \ln(1/\epsilon)$ for all $Q \in \mathcal{Q}_w$, by Lemma 11.19, $\sum_{Q \in \mathcal{Q}_w} \Pr[Q \text{ is in } H_\tau] \leq \epsilon^{10}$ . For a primary predecessor path $Q \in \mathcal{Q}_w$ (or one of its extensions), the vertex $w$ can be charged at most twice according to the above charging mechanism. Since any predecessor path of $w$ with length more than $\ell - 1$ must be an extension of one with length exactly $\ell - 1$, we have that the amount $w$ is charged is at most

$$\sum_{Q \in \mathcal{Q}_w} 2 \cdot 2 \cdot \Pr[Q \text{ is in } H_\tau] \cdot y_w/\epsilon^6 \leq 4 \cdot (\epsilon^{10}/\epsilon^6) \cdot y_w$$
$$\leq 4 \cdot \epsilon^4 \cdot y_w.$$

Summing over all $w \in V$ and using Lemma 11.3, the total charge is at most

$$\sum_{w \in v} 4 \cdot \epsilon^4 \cdot y_w \leq 4 \cdot \epsilon^4 \cdot \beta \cdot \sum_{e \in E} x_e \leq \epsilon^3 \sum_{e \in E} x_e.$$ □

### 11.4.6 Calculating the Competitive Ratio of Algorithm 11.3

We now show that the competitive ratio of Algorithm 11.3 is indeed $(1/2 + \alpha)$ competitive for some sufficiently small absolute constant $\alpha > 0$, thus proving Theorem 9.2. This essentially combines the facts that for good vertices, the matching probability is very close to the fractional values of incident edges, and that the number of bad vertices is very small compared to the total value of the fractional algorithm (over the entire graph).

*Proof of Theorem 9.2.* Let OPT denote the size of the maximum cardinality matching in the input graph $G$. Then, by Lemma 11.5 and our choice of $f = f_{1+2\epsilon}$ and $\beta = 2 - \varepsilon \geq \beta^*(f_{1+2\epsilon})$, we have that $\sum_e x_e \geq (1/\beta) \cdot \text{OPT} \geq (1/2 + \epsilon/4) \cdot \text{OPT}$, where the $x_e$'s are the fractional values we

compute in Algorithm 11.3.

Now let $M$ be the matching output by Algorithm 11.3. We have

$$\begin{aligned}
\mathbb{E}[|M|] &= \sum_{e\in E} \Pr[e \text{ is matched}] \\
&\geq \sum_{\text{good } v\in V} (1-\epsilon^2) \cdot \sum_{u\in N_v(v)} x_{uv} && \text{(By Theorem 11.16)} \\
&\geq (1-\epsilon^2) \cdot \left( \sum_{e\in E} x_e - \sum_{\text{bad } v\in V} \sum_{u\in N_v(v)} x_{uv} \right) \\
&\geq (1-\epsilon^2) \cdot \left( \sum_{e\in E} x_e - \sum_{\text{bad } v\in V} 1 \right) && \left( \sum_{u\in N_v(v)} x_{uv} \leq 1 \right) \\
&\geq (1-\epsilon^2) \cdot \left( \sum_{e\in E} x_e - \epsilon^3 \sum_{e\in E} x_e \right) && \text{(By Theorem 11.18)} \\
&\geq (1-2\epsilon^2) \cdot \sum_{e\in E} x_e \\
&\geq (1-2\epsilon^2) \cdot (1/2 + \epsilon/4) \cdot \text{OPT} \\
&\geq (1/2 + \epsilon/5) \cdot \text{OPT},
\end{aligned}$$

where the last line holds for a sufficiently small constant $\epsilon > 0$. □

# 12 Conclusions

The maximum matching problem is a classic combinatorial optimization problem considered in several influential works in theoretical computer science. In this thesis, we have studied several variants of the matching problem (i.e., the cardinality/weighted versions for bipartite/general graphs) in different computational models where the algorithms have to make decisions without complete information about the input graphs. The computational settings we covered include

1. the stochastic settings where the input is sampled from a known distribution with implicit or explicit costs for knowing the parts of the realized graphs,

2. streaming and MPC settings where the algorithm instances do not have sufficient space to store the input graph completely, and

3. the online settings where the input is revealed one edge/vertex at a time.

In the stochastic setting of query-commit, we showed a $(1 - 1/e)$-approximation algorithm for the weighted matching in bipartite graphs. However, we do not know any better hardness result than 0.898, which is the known best upper bound for the approximation ratio in the unweighted setting. Thus proving tight approximation ratios for either unweighted or weighted version of maximum matching remains open in this model.

We also presented a $(1 - 1/e)$-approximation algorithm for MWBM in the price-of-information model. Price-of-information is a relatively new model, and proving any non-trivial upper bound on the approximation ratio for MWM in this model remains open. Moreover, it is also interesting to explore how to beat the approximation ratio $1/2$ given by the greedy algorithm for weighted general (i.e., non-bipartite) graphs in this setting.

In the semi-streaming setting, we presented the first weighted matching algorithm that is better-than-$1/2$-approximate under edge arrivals in uniformly random order. However, under such random order edge arrivals, no $1 - \Omega(1)$ upper bound for the approximation ratio is known even for the MWM problem. On the other hand, there exists a $2/3$-approximate algorithm for MCM in this setting [14]. For MCM in *arbitrary order* streams, the approximation ratio is upper bounded by $\frac{1}{1+\ln 2}$ unless the algorithm is only allowed $O(n^{1+1/\log\log n})$ bits of space [61]. I.e., MCM is provably easier in random order arrivals compared to the adversarial streams. Thus it is worth investigating whether the same is true for MWM as well. We also note that perhaps the most well-known open question related to matching in the semi-streaming setting is whether we can beat the greedy competitive ratio of $1/2$ for MCBM under adversarial streams.

In the multi-pass streaming and MPC settings, recall that we have reduced the problem of finding a $(1-\varepsilon)$-approximate MWM to that of solving $g(\varepsilon)$ many instances of $(1-f(\varepsilon))$-approximate MCBM in the respective settings. Here, both $f$ and $g$ are exponential functions of $1/\varepsilon$. Thus, while any improvements to MCBM in these models automatically yields improved results for MWM, it is also worth considering possible improvements to the reduction itself in terms of the required precision of the MCBM algorithm (i.e., $f(\varepsilon)$) or the required number of MCBM instances (i.e., $g(\varepsilon)$).

In the regime of online algorithms for maximum matching, we proved that no online algorithm is $(1/2+\Omega(1))$-competitive for MCBM under edge-arrivals. Hence, the greedy algorithm is tight for this case. Consequently, a natural next step is to consider the online preemptive matching problem in the same model. In the preemptive version, upon arrival of an edge, it is allowed to evict the edges from the already constructed part of the matching. The currently known best upper bound for the competitive ratio of preempting online matching is $2-\sqrt{2} \simeq 0.585$ due to Huang et al. [55]. (This is, in fact, the known best upper bound for the general vertex arrival model. This construction of hard instances also works for online edge arrivals with preemption as preemption cannot help improve the matching in the considered family of graphs.)

Considering MCM in the online vertex arrival setting, we presented the first randomized integral matching that achieves $(1/2+c)$ competitive ratio for some constant $c > 0$, but the exact advantage over $1/2$, i.e., $c$, achieved by our algorithm is likely very small. The main open question related to this problem thus is to prove a tight competitive ratio.

# A Deffered Proofs of Part I

In this appendix, we restate and Lemma 5.4 and prove it.

**Lemma 5.4.** *Fix a vertex $a \in A$. If $p_{e,v} > 0$ for all $(e,v) \in E_a$ and $\sum_{v \in V_e} p_{e,v} < 1$ for all $e \in \delta(a)$, the function $f$ is strictly submodular and strictly increasing on the lattice family $\mathcal{E}_a$. Formally,*

1. *For any $A, B \in \mathcal{E}_a$ such that $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$, $f(A) + f(B) > f(A \cap B) + f(A \cup B)$, and*

2. *For any $A \subsetneq B \subseteq E_a$, $f(B) > f(A)$.*

*Proof.* It is easy to see that $f(F) = 1 - \prod_{e \in E}(1 - \sum_{v \in V_e:(e,v) \in F} p_{e,v})$. Consider two sets $A, B \in \mathcal{E}_a$ such that $A \setminus B \neq \emptyset$ and $B \setminus A \neq \emptyset$. Fore each $e \in E$, let $a_e = 1 - \sum_{v \in V_e:(e,v) \in A} p_{e,v}$ and $b_e = 1 - \sum_{v \in V_e:(e,v) \in B} p_{e,v}$. Thus we have that $f(A) = 1 - \prod_{e \in E} a_e$, $f(B) = 1 - \prod_{e \in E} b_e$, $f(A \cup B) = 1 - \prod_{e \in E} \min(a_e, b_e)$, and $f(A \cap B) = 1 - \prod_{e \in E} \max(a_e, b_e)$. The last two equations follow from the definition of the family $\mathcal{E}_a$.

Now we have

$$\begin{aligned} &f(A) + f(B) - f(A \cup B) - f(A \cap B) \\ &\qquad = \prod_{e \in E} \min(a_e, b_e) + \prod_{e \in E} \max(a_e, b_e) - \prod_{e \in E} a_e - \prod_{e \in E} b_e. \end{aligned} \tag{A.1}$$

Thus to prove Property 1, it is sufficient to prove that the right hand side of (A.1) is strictly greater than zero, which is equivalent to showing that $\prod_{e \in E} \min(a_e, b_e) + \prod_{e \in E} \max(a_e, b_e) > \prod_{e \in E} a_e + \prod_{e \in E} b_e$.

Since $A \setminus B \neq \emptyset$, we have $a_e < b_e$ for at least one edge $e_1 \in E$. To see this, let $(e,v) \in A \setminus B$. Then for such edge $e$, it follows from the definition of set family $\mathcal{E}_a$ that the set $A_e = A \cap \{(e,w) : w \in V_e\}$ contains all the elements in the set $B_e = B \cap \{(e,w) : w \in V_e\}$, and in addition, it also contains at least one more element, namely $(e,v)$. Since $p_{e,v} > 0$, $a_e < b_e$. Similarly, we have $a_e > b_e$ for at least one edge $e_2 \in E$ (since we assumed that $p_{e,v} > 0$ for all $(e,v) \in E_a$). Let $E_1$ be the (nonempty) set of edges for which $a_e < b_e$, and let $E_2$ be the (nonempty) set of edges for which $a_e > b_e$. Without loss of generality we assume that $E_1 \cup E_2 = E$ (if $a_e = b_e$ for some $e$, then we

can divide (A.1) by $a_e$ since $\sum_{v \in V_e} p_{e,v} < 1$, $a_e > 0$). We then have

$$\begin{aligned}\prod_{e \in E} \min(a_e, b_e) + \prod_{e \in E} \max(a_e, b_e) &= \prod_{e \in E_2} b_e \prod_{e \in E_1} a_e + \prod_{e \in E_2} a_e \prod_{e \in E_1} b_e \\ &> \prod_{e \in E_2} a_e \prod_{e \in E_1} a_e + \prod_{e \in E_2} b_e \prod_{e \in E_1} b_e \\ &= \prod_{e \in E} a_e + \prod_{e \in E} b_e\end{aligned}$$

as required. The inequality above follows from the rearrangement inequality as $\prod_{e \in E_1} b_e > \prod_{e \in E_1} a_e$ and $\prod_{e \in E_2} a_e > \prod_{e \in E_2} b_e$.

As for Property 2, suppose that $A \subsetneq B$. Then for all $e \in E$, $a_e \geq b_e$, and for at least one $e \in E$, $a_e > b_e$. Hence $f(B) - f(A) = \prod_{e \in E} a_e - \prod_{e \in E} b_e > 0$. □

# B Deffered Proofs of Part II

In this section, we present the proofs omitted in Part II.

**Lemma 7.1.** *There exists an unweighted streaming algorithm* Unw-3-Aug-Paths *with the following properties:*

1. *The algorithm is initialized with a matching $M$ and a parameter $\beta > 0$. Afterwards, a set $E$ of edges is fed to the algorithm one edge at a time.*
2. *Given that $M \cup E$ contains at least $\beta|M|$ vertex disjoint 3-augmenting paths, the algorithm returns a set* Aug *of at least $(\beta^2/32)|M|$ vertex disjoint 3-augmenting paths. The algorithm uses space $O(|M|)$.*

*Proof.* The algorithm maintains a support set $S$ greedily. We use a parameter $\lambda$ that depends on $\beta$. Whenever we see an edge $uv$ such that $u$ is an unmatched vertex and $v$ is a matched vertex, we add it to $S$ if degree of $u$ in $S$ is less than $\lambda$ and degree of $v$ in $S$ is less than 2. In the end, we greedily find vertex disjoint 3-augmentations and return them.

Let $E_3 \subseteq M$ be the set of 3-augmentable edges, so $|E_3| \geq \beta|M|$. We call an edge $vw$ in $E_3$ a *bad* edge, if one of the following happens in $S$:

- There is no edge incident to $v$ or $w$.
- There is exactly one edge incident to each of $v$ and $w$, but it is to the same vertex (which, gives us a triangle, not an augmentation).

We can individually augment all edges in $E_3 \setminus E_B$, which we call *good* edges, and we denote this set of good edges by $E_G$. The crucial observation is that a for a bad edge $vw$, one of the edges on its 3-augmenting path $avwb$ was not added to $S$ by the algorithm. Which means that one of $a$ and $b$ already had $\lambda$ edges incident to it. Hence, $\lambda|E_B| \leq |S| \leq 4|M|$, because each edge in $M$ can have at most 4 support edges incident to it. This gives $\lambda(|E_3| - |E_G|) \leq 4|M|$. Using $|E_3| \geq \beta|M|$ and algebraic simplification, we get that $|E_G| \geq (\beta - 4/\lambda)|M|$. When we greedily augment using $S$, for each augmentation $avwb$, we may potentially lose up to $2\lambda$ augmentations, because we cannot use the support edges incident to $a$ or $b$ any more, otherwise we lose the vertex-disjointness property of the 3-augmenting paths that we return. Therefore, the number of 3-augmentations that we return is at least

$$\frac{|E_G|}{2\lambda} \geq \left(\frac{\beta}{2\lambda} - \frac{2}{\lambda^2}\right)|M|\,,$$

which finishes the proof if we use $\lambda = 8/\beta$. □

**Lemma 7.2** (Lemma 1 in [68]). *Let $\alpha \geq 0$, $M'$ be a maximal matching in $G$, and $M^*$ be a maximum unweighted matching in $G$ such that $|M'| \leq (1/2 + \alpha)|M^*|$. Then the number of 3-augmentable edges in $M'$ is at least $(1/2 - 3\alpha)|M^*|$, and the number of non-3-augmentable edges in $M'$ is at most $4\alpha|M^*|$.*

*Proof.* Let the number of 3-augmentable edges in $M'$ be $k$. For each 3-augmentable edge in $M'$, there are two edges in $M^*$ incident on it. Also, each non-3-augmentable edge in $M'$ lies in a connected component of $M' \cup M^*$ in which the ratio of the number of $M^*$-edges to the number of $M'$-edges is at most $3/2$. Hence,

$$\begin{aligned}
|M^*| &\leq 2k + \frac{3}{2}(|M'| - k) && \text{since there are } |M'| - k \text{ non-3-augmentable edges}, \\
&\leq 2k + \frac{3}{2}\left(\left(\frac{1}{2} + \alpha\right)|M^*| - k\right) && \text{because } |M'| \leq (1/2 + \alpha)|M^*|, \\
&= \frac{1}{2}k + \left(\frac{3}{4} + \frac{3}{2}\alpha\right)|M^*|,
\end{aligned}$$

which, after simplification, gives $k \geq (1/2 - 3\alpha)|M^*|$. And the number of non-3-augmentable edges in $M'$ is $|M'| - k \leq |M'| - (1/2 - 3\alpha)|M^*| \leq (1/2 + \alpha - 1/2 + 3\alpha)|M^*| = 4\alpha|M^*|$. □

**Lemma 8.9.** *Let $M$ be a matching such that $w(M) \leq w(M^*)/(1+\varepsilon)$ where $\varepsilon \leq 1/16$. Then there exists a collection $\mathcal{C}$ of vertex-disjoint augmentations with the following properties:*

*(A) Each $C \in \mathcal{C}$ is such that $C \cup C^M$ consists of at most $4/\varepsilon$ edges.*

*(B) For every $C \in \mathcal{C}$ and every edge $e \in C \cap M^*$, $w(e) \geq (\varepsilon^2/64) \cdot w(C)$.*

*(C) For every $C \in \mathcal{C}$ and every edge $e \in C \cap M$, $w(e) \geq (\varepsilon^6/64) \cdot w(C)$.*

*(D) For every $C \in \mathcal{C}$, we have that*

$$w(C \cap M^*) \geq (1 + \varepsilon/8) \cdot w(C^M).$$

*(E) The sum of gains of the elements of $\mathcal{C}$ is at least $(\varepsilon^2/200) \cdot w(M^*)$. That is*

$$\sum_{C \in \mathcal{C}} \left(w(C \cap M^*) - w(C^M)\right) \geq (\varepsilon^2/200) \cdot w(M^*).$$

*Proof.* We first provide a proof in which Property C is ignored, and Property D is replaced by a more strict property

$$\text{For every } C \text{ in the collection, we have that } w(C \cap M^*) \geq (1 + \varepsilon/4) \cdot w(C^M). \tag{B.1}$$

We construct $\mathcal{C}'$ having this modified set of properties. After that, we show how to obtain $\mathcal{C}$ from $\mathcal{C}'$.

**Constructing $\mathcal{C}'$: (Property C ignored. Property D replaced by Property** (B.1)**)** Without loss of generality, assume that $M \cup M^*$ is a union of cycles[1], say $C_1, C_2, \ldots$. Label edges of $M^*$ using the set $[|M^*|]$, i.e., $\{1, 2, \ldots, |M^*|\}$, in such a way that $M^*$-edges in a cycle $C_i$ get labels that "respect" the cycle order. To elaborate, first number the $M^*$-edges in $C_1$ starting at an arbitrary edge, in the cyclical order, as $1, 2, \ldots, |C_1|/2$. Then continue on to $C_2$, and start with $|C_1|/2 + 1$, and so on.

Now, for $i \in [4/\varepsilon]$, let $M^*_{-i}$ be the matching obtained by removing edges $M^*_i = \{i, i+4/\varepsilon, i+8/\varepsilon, \ldots\}$ from $M^*$. For any $i$, the set of edges $M \cup M^*_{-i}$ is a union of vertex disjoint paths or cycles, each of which has length at most $4/\varepsilon$, and each path starts and ends in an $M$-edge. If we pick $i$ uniformly at random from $\{1, \ldots, 4/\varepsilon\}$, then $\mathbb{E}[M^*_{-i}] = (1 - \varepsilon/4)w(M^*)$. Thus, by the probabilistic method, there is some $i$ for which $w(M^*_{-i}) \geq (1 - \varepsilon/4)w(M^*)$. We show the existence of the desired set $\mathcal{C}$ using $M^*_{-i}$.

Let $\tilde{\mathcal{C}} := \{H_1, H_2, \ldots, H_k\}$ be the collection of paths and cycles in $M \cup M^*_{-i}$. Construct $\mathcal{C}_A$ as follows: Start $\mathcal{C}_A$ being empty. For each $H \in \tilde{\mathcal{C}}$, split $H$ into pieces by removing each edge $e \in H \cap M^*$ such that $w(e) < (\varepsilon^2/64)w(H)$, and add the pieces to $\mathcal{C}_A$. Notice that if $C'$ is path obtained from a path or cycle $C \in \tilde{\mathcal{C}}$ after removing some $M^*$-edges, $C'$ must start and end in $M$-edges, and the removal of such edges can only decrease the path length. Furthermore, if $e \in C' \cap M^*$ is a remaining edge, then $w(e) \geq (\varepsilon^2/64)w(C) \geq (\varepsilon^2/64)w(C')$. Thus $\mathcal{C}_A$ satisfies Property A and Property B.

First, note that from the way we constructed path $C \in \mathcal{C}_A$ it holds $C^M = C \cap M$. Now, let $\mathcal{C}_{\text{bad}} \stackrel{\text{def}}{=} \{C \in \mathcal{C}_A : w(C \cap M^*) < (1 + \varepsilon/4)w(C \cap M)$ and let $\mathcal{C}' \stackrel{\text{def}}{=} \mathcal{C}_A \setminus \mathcal{C}_{\text{bad}}$. Hence, $\mathcal{C}'$ satisfies Property (B.1). Also, since $\mathcal{C}'$ is a sub-collection of $\mathcal{C}_A$, $\mathcal{C}'$ satisfies Property A and Property B.

**Proving Property E for $\mathcal{C}'$:** What remains is to show that Property E holds for $\mathcal{C}'$ as well.

For an element $C \in \mathcal{C}'$, we first show that the following holds

$$w(C \cap M^*) - w(C \cap M) \geq \varepsilon w(C)/16.$$

For $C$ satifying Property (B.1), we have

$$\begin{aligned} w(C \cap M^*) - w(C \cap M) &\geq w(C \cap M^*) - \frac{w(C \cap M^*)}{1 + \varepsilon/4} \\ &= \frac{\varepsilon/4}{1 + \varepsilon/4} w(C \cap M^*) \\ &\geq \frac{\varepsilon}{8} \cdot w(C \cap M^*). \end{aligned} \tag{B.2}$$

This further implies

$$w(C \cap M^*) - w(C \cap M) \geq \varepsilon w(C \cap M^*)/8 \geq \varepsilon w(C)/16. \tag{B.3}$$

The first inequality in the above expression follows by (B.2) and the second one follows by

[1] By adding zero-weight edges, one can assume that $M$ and $M^*$ are two perfect matchings. Then the edges in $M \cap M^*$ can be considered as pair of different edges that form a 2-cycles.

Property (B.1)

Next, we upper-bound the total weight of the edges that were removed when constructing $\mathcal{C}_A$ from $\tilde{\mathcal{C}}$. For any path or cycle $C \in \tilde{\mathcal{C}}$, the total removed weight from $C$ is at most $(\varepsilon^2/64)(4/\varepsilon)w(C) \leq (\varepsilon/16)w(C)$ (recall that $|C| \leq 4/\varepsilon$). Let $R$ be the set of all such removed edges. Then

$$w(R) \leq \sum_{C \in \tilde{\mathcal{C}}} (\varepsilon/16)w(C) \leq (\varepsilon/16)w(M \cup M^*_{-i}) \leq (\varepsilon/8)w(M^*).$$

Let $w(\mathcal{X} \cap M^*)$ denote $\sum_{C \in \mathcal{X}} w(C \cap M^*)$; therefore $w(\tilde{\mathcal{C}} \cap M^*) \geq (1 - \varepsilon/4)w(M^*)$. Notice that this implies

$$\begin{aligned} w(\mathcal{C}_A \cap M^*) &= w(\tilde{\mathcal{C}} \cap M^*) - w(R) \\ &\geq (1 - \varepsilon/4 - \varepsilon/8)\, w(M^*) \\ &\geq (1 - 3\varepsilon/8)\, w(M^*). \end{aligned} \tag{B.4}$$

Now, we claim that

$$w(\mathcal{C}' \cap M^*) \geq \varepsilon w(M^*)/4. \tag{B.5}$$

Towards a contradiction, assume that $w(\mathcal{C}' \cap M^*) < \varepsilon w(M^*)/4$. This implies

$$\begin{aligned} w(\mathcal{C}_{\text{bad}} \cap M^*) &= w(\mathcal{C}_A \cap M^*) - w(\mathcal{C}' \cap M^*) \\ &> (1 - 3\varepsilon/8)w(M^*) - \varepsilon w(M^*)/4 \\ &= (1 - 5\varepsilon/8)w(M^*)\,. \end{aligned}$$

From this we derive

$$\begin{aligned} w(\mathcal{C}_{\text{bad}} \cap M) &> \frac{w(\mathcal{C}_{\text{bad}} \cap M^*)}{(1 + \varepsilon/4)} \\ &> (1 - \varepsilon/4)(1 - 5\varepsilon/8)w(M^*) \\ &> (1 - 7\varepsilon/8)w(M^*). \end{aligned}$$

The last chain of inequalities implies that

$$w(M) \geq w(\mathcal{C}_{\text{bad}} \cap M) > (1 - 7\varepsilon/8)w(M^*) \geq \underbrace{(1 - 7\varepsilon/8)(1 + \varepsilon)}_{\geq 1 \text{ for } \varepsilon \leq 1/8} w(M) \geq w(M),$$

which is a contradiction. Therefore, (B.5) holds.

Now we can prove that Property E holds for $\mathcal{C}'$.

$$\sum_{C \in \mathcal{C}'} \big(w(C \cap M^*) - w(C^M)\big) \geq \sum_{C \in \mathcal{C}'} \varepsilon w(C)/16 \geq \varepsilon^2 w(M^*)/64. \tag{B.6}$$

The first inequality here follows by (B.5) while the second one follows by (B.3)

**Constructing $\mathcal{C}$:** By exhibiting $\mathcal{C}'$, we showed that the lemma holds if Property D is replaced by (B.1) and also when Property C is ignored. Now we prove that the lemma holds even if Property D is not replaced and Property C is taken into account. To that end, we obtain $\mathcal{C}$ from $\mathcal{C}'$ in the following way.

Initially, $\mathcal{C}$ is empty. We consider each element $C \in \mathcal{C}'$ separately and apply the following procedure:

- Step 1: Remove all the edges $C \cap M$ violating Property C. Let $\mathcal{P}$ be the obtained collection.
- Step 2: Add to $\mathcal{C}$ all the elements of $\mathcal{P}$ that satisfy Property D.

In the procedure above, the removed edge $e$ is never from $M^*$, and removing an edge $e \in M$ from $C$ only increases the contribution of an edge from $M^*$ to the remaining path. This implies that Property B holds for the elements of $\mathcal{C}$. So, for $\mathcal{C}$ hold all the Properties A-D. It remains to show that Property E holds as well.

**Proving Property E for $\mathcal{C}$:** Let $C \in \mathcal{C}'$ be an element decomposed into $\mathcal{P}$ in Step 1. First, observe that it does not necessarily hold that the gain of $C$ equals to the sum of gains of the elements of $\mathcal{P}$. The reason is that those edges from $C \cap M$ that are removed in Step 1 could be deducted twice when calculating the sum of gains of the elements of $\mathcal{P}$. However, it is easy to upper-bound their negative contribution as follows. Recall that each $C$ has length at most $4/\varepsilon$, so we can remove at most $4/\varepsilon$ edges from each $C$. So, the gain-loss in $\mathcal{P}$ compared to $C$ is at most $\frac{4}{\varepsilon} \cdot \frac{\varepsilon^6}{64} w(C)$.

Let $X \stackrel{\text{def}}{=} w^+(C)$. By Property (B.1), $X \geq \varepsilon w(C^M)/4$. Recall also that by (B.3) we showed that $X \geq \varepsilon w(C)/16$, hence

$$X \geq \max\{\varepsilon w(C^M)/4, \varepsilon w(C)/16\}. \tag{B.7}$$

When, in Step 1, $C$ is decomposed into $\mathcal{P}$, by our discussion above, the sum of gains of the elements of $\mathcal{P}$ is at least $X - \varepsilon^5 w(C)/8$. On the other hand, in Step 2, the algorithm removes all the elements of $C' \in \mathcal{P}$ that have gain less than $\varepsilon w(C'^M)/8$. So, the total gain loss of $\mathcal{P}$ due to Step 2 is

$$\sum_{C' \text{ is removed from } \mathcal{P}} \varepsilon w(C'^M)/8 \leq \varepsilon w(C^M)/8 + \varepsilon^5 w(C)/8.$$

This implies that the elements of $\mathcal{P}$ that are added to $\mathcal{C}$ have gain at least

$$X - \varepsilon^5 w(C)/8 - \left(\varepsilon w(C^M)/8 + \varepsilon^5 w(C)/8\right) \geq X/3$$

where the inequaliy follows from (B.7) and the fact that $\varepsilon < 1/16$.

We now conclude that after applying the above steps the sum of gains of the elements of $\mathcal{C}$ are at least $1/3$ of that in $\mathcal{C}'$. Therefore, Property E for $\mathcal{C}$ follows from (B.6). □

**Lemma 8.12.** *Let $\mathcal{C}$ be a collection of augmentations as defined by Lemma 8.9. Consider an augmentation $C \in \mathcal{C}$. Then, there exists a parametrization $G^P$, a choice a good pair $(\tau^A, \tau^B)$,*

*and $W$ so that $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ contains a path $S$ passing through all the layers so that when Lemma 8.11 is applied on $S$ it results in a decomposition containing $C$. Furthermore, $W$ equals $(1+\varepsilon^4)^i \leq w(S)$, for some integer $i \geq 0$.*

*Proof for the case when $C$ is a path.* This is a simpler version of the proof for the case of cycles we presented in Section 8.3. For the sake of completeness we provide its proof as well.

**Transformation of $Q$:** If $Q$ does not start by a matched edge, attach to the beginning of $Q$ an edge of weight 0 and add that edge to the current matching. In a similar way alter $Q$ if it does not end by a matched edge. Notice that this does not change the gain of $Q$. After this transformation, we conveniently have that $Q \cap M$ equals $Q^M \cap M$.

**Parametrization:** Let $Q = v_1 \dots v_{2t}$. Taking into account the properties of $\mathcal{C}$ and the transformation of $Q$, $Q$ has at most $4/\varepsilon + 2$ edges and $Q$ has odd length. So, $t \leq 2/\varepsilon + 2 \leq 4/\varepsilon$. Consider a parametrization $G^P$ of the graph in which $v_i \in R$ for each even $i$, while $v_i \in L$ for each odd $i$. By the definition, $G^P$ contains $Q$.

Define $W$ to be the largest value such that $W = (1+\varepsilon^4)^i \leq w(Q)$, for some integer $i \geq 0$. Let $a_1, \dots, a_t$ be the matched edges of $Q$ appearing in that order, with $a_1 = \{v_1, v_2\}$. Similarly, let $b_1, \dots, b_{t-1}$ be the unmatched edges of $Q$ appearing in that order, with $b_1 = \{v_2, v_3\}$.

**Layered graph:** Now, we define a layered graph $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ that contains $Q$ passing through all the layers.

- Sequence $\tau^A$ has length $t$ and $\tau^B$ has length $t-1$.
- For every $a_i$, set $\tau_i^A$ to be the smallest $k\varepsilon^{12}$ such that $k$ is an integer and $\tau_j^A W \geq w(a_i)$.
- For every $b_i$, set $\tau_i^B$ to be the largest $k\varepsilon^{12}$ such that $k$ is an integer and $\tau_j^B W \leq w(b_i)$.

It is easy to verify that $\mathcal{L}(\tau^A, \tau^B, W, G^P)$ contains $Q$.

**Correctness:** All the properties (A)-(E) given in Figure 8.2 follow directly by the properties of $Q$ (see Lemma 8.9) and the definition of $(\tau^A, \tau^B)$. (Note that property (D) is not affected by appending zero-weight matched edges in the transformation of $Q$.) So, it remains to show that property (F) holds as well.

As in the cycle case, we compare $\sum_i \tau_i^A W$ and $\sum_i \tau_i^B W$. We have

$$\sum_i \tau_i^B W \geq w(Q \cap M^*) - t\varepsilon^{12} W, \tag{B.8}$$

and

$$W \leq w(Q) = w(Q \cap M) + w(Q \cap M^*) \leq 2w(Q \cap M^*). \tag{B.9}$$

Using that $t \leq 4/\varepsilon$ and combining (B.8) and (B.9) implies

$$\sum_i \tau_i^B W \geq w(Q \cap M^*) \left(1 - 2\varepsilon^{10}\right). \tag{B.10}$$

Next, observe that from (B.9) and $t \leq 4/\varepsilon$ we have

$$\begin{aligned} \sum_i \tau_i^A W &\leq w(Q \cap M) + t\varepsilon^{12} W \\ &\leq \frac{(1 + 4\varepsilon^{10})}{1 + \varepsilon/8} w(Q \cap M^*) \\ &\leq \frac{(1 + \varepsilon^9)}{1 + \varepsilon/8} w(Q \cap M^*). \end{aligned} \tag{B.11}$$

The second inequality is due to Lemma 8.9 and the third one holds because $\varepsilon \leq 1/16$.

From (B.10) and (B.11) we derive

$$\begin{aligned} \sum_i \tau_i^B W - \sum_i \tau_i^A W &\geq \left( \left(1 - 2\varepsilon^{10}\right) - \frac{(1 + \varepsilon^9)}{1 + \varepsilon/8} \right) w(Q \cap M^*) \\ &= \frac{(1 + \varepsilon/8)\left(1 - 2\varepsilon^{10}\right) - (1 + \varepsilon^9)}{1 + \varepsilon/8} w(Q \cap M^*) \\ &= \frac{\varepsilon/8 - \varepsilon^9 - 2\varepsilon^{10} - \varepsilon^{11}/4}{1 + \varepsilon/8} w(C \cap M^*), \end{aligned}$$

which is $\geq \varepsilon^{12} W$ because $\varepsilon \leq 1/16$. The last chain of inequalities implies

$$\sum_i \tau_i^B - \sum_i \tau_i^A \geq \varepsilon^{12}. \tag{B.12}$$

□

# C Deferred Proofs of Part III

In this section, we present the proofs omitted in Part III.

We start by proving that a change of the realized arc choices of any vertex does not change the matched status of more than two vertices (at any point in time). This is Lemma 11.9, restated below.

**Lemma 11.9.** *Let $G_\tau$ and $G_{\tau'}$ be two realizations of the random digraph where all the vertices in the two graphs make the same choices except for* one *vertex $v$. Then the number of vertices that have different* matched status *(free/matched) in the matchings computed in $H_\tau$ and $H_{\tau'}$ at any point of time is at most two.*

*Proof.* We consider the evolution, following each vertex arrival, of the matchings $M_\tau$ and $M_{\tau'}$ computed in $H_\tau$ and $H_{\tau'}$, respectively, as well as the set of vertices with different matched status in these matchings, denoted by $D := (M_\tau \setminus M_{\tau'}) \cup (M_{\tau'} \setminus M_\tau)$. The set $D$ is empty before the first arrival and remains empty until the arrival of $v$, as all earlier vertices than $v$ have the same primary and secondary arcs and have the same set of free neighbors in $H_\tau$ and $H_{\tau'}$ (as $D = \emptyset$, by induction). Now, if immediately after $v$ arrives it remains free in both $M_\tau$ and $M_{\tau'}$, or it is matched to the same neighbor in both matchings, then clearly $D$ remains empty. Otherwise, either $v$ is matched to different neighbors in $M_\tau$ and $M_{\tau'}$, or $v$ is matched in one of these matchings but not in the other. Both these cases result in $|D| = 2$. We now show by induction that the cardinality of $D$ does not increase following subsequent arrivals, implying the lemma.

Let $u$ be some vertex which arrives after $v$. If when $u$ arrives $u$ is matched to the same neighbor $w$ in $M_\tau$ and $M_{\tau'}$ or if $u$ remains free in both matchings, then $D$ is unchanged. If $u$ is matched to some $w$ on arrival in $M_\tau$, but not in $M_{\tau'}$, then since the arcs of $u$ are the same in $G_\tau$ and $G_{\tau'}$, this implies that $w$ must have been free in $M_\tau$ but not in $M_{\tau'}$, and so $D \ni w$. Therefore, after $u$ arrives, we have $D \leftarrow (D \setminus \{w\}) \cup \{u\}$, and so $D$'s cardinality is unchanged. Finally, if $u$ is matched to two distinct neighbors, denoted by $w$ and $w'$, respectively, then one of $(u, w)$ and $(u, w')$ must be the primary arc of $u$ in both $G_\tau$ and $G_{\tau'}$. Without loss of generality, say $(u, w)$ is this primary arc. Since $u$ is matched to $w$ in $M_\tau$ but not in $M_{\tau'}$, then $w$ must be free in $M_\tau$ when $u$ arrives, but not in $M_{\tau'}$, and so $D \ni w$. Consequently, we have that after $u$ arrives we have $D \leftarrow S$ for some set $S \subseteq (D \setminus \{w\}) \cup \{w'\}$, and so $D$'s cardinality does not increase. □

We now prove the bound on the fractional degree $x_u$ in terms of its dual value, restated below.

**Lemma 11.3.** *For any vertex $u, v \in V$, let $y_u$ be the potential of $u$ prior to arrival of $v$. Then*

*the fractional degree just before $v$ arrives, $x_u := \sum_{w \in N_v(u)} x_{uw}$, is bounded as follows:*

$$\frac{y_u}{\beta} \leq x_u \leq \frac{y_u + f(1-y_u)}{\beta}.$$

*Proof.* Let $y_0$ be $u$'s potential after $u$'s arrival. For the lower bound, note that it suffices to prove that every increase in the fractional degree is bounded below by the increase in the potential divided by $\beta$. When vertex $u$ first arrived, we consider two cases.

1. $y_0 > 0$ (thus $y_0 = 1 - \theta > 0$, and so $\theta < 1$), then the increase in $u$'s fractional degree was:

$$\sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \left(1 + \frac{1-\theta}{f(\theta)}\right) = \frac{f(\theta) + 1 - \theta}{\beta} = \frac{f(1-y_0) + y_0}{\beta} \geq \frac{y_0}{\beta}.$$

2. $y_0 = 0$ (thus $\theta = 1$), then the increase in $u$'s fractional degree was:

$$\sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \left(1 + \frac{1-\theta}{f(\theta)}\right) = \sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \geq 0 = \frac{y_0}{\beta}.$$

For every subsequent increase of the fractional degree due to a newly-arrived vertex we have that:

$$\frac{(\theta - y_u^{old})^+}{\beta} \left(1 + \frac{1-\theta}{f(\theta)}\right) \geq \frac{(\theta - y_u^{old})^+}{\beta},$$

Which concludes the proof for the lower bound.

For the upper bound, by [86, Invariant 1], we have that

$$\beta \cdot x_u \leq y_c + f(1-y_0) + \int_{y_0}^{y_c} \frac{1-x}{f(x)}\, dx. \tag{C.1}$$

This upper bound can be simplified by using Equation (11.1), as follows. Taking (C.1), adding and subtracting $1 + f(1-y_u)$ and writing the integral $\int_{y_0}^{y_u} \frac{1-x}{f(x)}\, dx$ as the difference of two integrals $\int_{y_0}^{1} \frac{1-x}{f(x)}\, dx$ - $\int_{y_u}^{1} \frac{1-x}{f(x)}\, dx$, and relying on Equation (11.1), we find that

$$\begin{aligned}
\beta \cdot x_u &\leq y_c + f(1-y_0) + \int_{y_0}^{y_c} \frac{1-x}{f(x)}\, dx \\
&= \left(1 + f(1-y_0) + \int_{y_0}^{1} \frac{1-x}{f(x)}\, dx\right) - 1 + y_c + \int_{1}^{y_c} \frac{1-x}{f(x)}\, dx \\
&= \beta^*(f) + y_c - \left(1 + f(1-y_c) + \int_{y_c}^{1} \frac{1-x}{f(x)}\, dx\right) + f(1-y_c) \\
&= \beta^*(f) + y_c - \beta^*(f) + f(1-y_c) \\
&= y_c + f(1-y_c),
\end{aligned}$$

from which the lemma follows. □

We now present the proofs deferred from Section 11.4.2. We start by presenting a more manageable form for the function $f = f_{1+2\epsilon}$ which we use.

A function in the WW family is determined by a parameter $k \geq 1$ and takes the following form

$$f_\kappa(\theta) = \left(\frac{1+\kappa}{2} - \theta\right)^{\frac{1+\kappa}{2\kappa}} \left(\theta + \frac{\kappa-1}{2}\right)^{\frac{\kappa-1}{2\kappa}}.$$

Letting $\kappa = 1 + 2\varepsilon$, we get that $f := f_\kappa$ is of the form

$$f(\theta) = (1+\varepsilon-\theta)^{\frac{1+\varepsilon}{1+2\varepsilon}} \cdot (\theta+\varepsilon)^{\frac{\varepsilon}{1+2\varepsilon}} = (1+\varepsilon-\theta) \cdot \left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}.$$

Clearly this is water filling when $\varepsilon = 0$ and otherwise we have that the first term is like water filling and then the second term is less than 1 for $z \leq 1/2$ and greater than 1 if $z > 1/2$.

By Taylor expansion, we obtain the following more manageable form for $f$.

**Lemma C.1.** *There exists $\epsilon_0 \in (0,1)$ such that for every $\epsilon \in (0, \epsilon_0)$ and every $\theta \in [0,1]$, we have*

$$f(\theta) \leq (1-\theta)\left(1 + \varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right) + 1.01\epsilon.$$

*Proof.* Taking the Taylor expansion of $e^x$, we find that

$$\begin{aligned}
f(\theta) &= (1+\varepsilon-\theta) \cdot \left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}} \\
&= (1+\varepsilon-\theta) \cdot \sum_{i=0}^{\infty} \frac{\left(\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) \cdot \frac{\varepsilon}{1+2\varepsilon}\right)^i}{i!} \\
&= (1+\varepsilon-\theta)\left(1 + \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) \cdot \frac{\varepsilon}{1+2\varepsilon}\right) + o(\varepsilon) \\
&= (1+\varepsilon-\theta) + (1-\theta)\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) \cdot \frac{\varepsilon}{1+2\varepsilon} + o(\varepsilon) \\
&= (1+\varepsilon-\theta) + (1-\theta)\varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) + o(\varepsilon) \\
&= (1-\theta)\left(1 + \varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right) + \epsilon + o(\varepsilon).
\end{aligned}$$

To be precise, for $\theta \in [0,1]$ and $0 < \epsilon \leq \epsilon_0 \leq 1$ (implying for example $\frac{\theta+\epsilon}{1+\epsilon-\theta} \leq \frac{2}{\epsilon}$), we will show that terms dropped in the third, fourth and fifth lines are all at most some $O((\ln(\frac{1}{\epsilon}) \cdot \epsilon)^2) = o(\epsilon)$, from which the lemma follows as the sum of these terms is at most $0.01\epsilon$ for $\epsilon \leq \epsilon_0$ and $\epsilon_0$ sufficiently small.

Indeed, in the third line, we dropped

$$\begin{aligned}
(1+\varepsilon-\theta) \cdot \sum_{i=2}^{\infty} \frac{\left(\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) \cdot \frac{\varepsilon}{1+2\varepsilon}\right)^i}{i!} &\leq 2 \cdot \sum_{i=2}^{\infty} \frac{(\ln(\frac{2}{\epsilon}) \cdot \varepsilon)^i}{i!} \leq \cdot \sum_{i=2}^{\infty} \frac{\left(\ln\left(\frac{2}{\epsilon}\right) \cdot \varepsilon\right)^i}{i^2} \\
&= O((\ln(1/\epsilon) \cdot \varepsilon)^2),
\end{aligned}$$

where the last step used that $\ln(1/\varepsilon) \cdot \epsilon \leq 1$ holds for all $\epsilon \geq 0$. In the fourth line, we dropped

$$\epsilon \cdot \ln\left(\frac{\theta+\epsilon}{1+\epsilon-\theta}\right) \cdot \frac{\epsilon}{1+2\epsilon} \leq \epsilon^2 \cdot \ln(2/\epsilon) = O((\ln(1/\epsilon) \cdot \varepsilon)^2).$$

Finally, in the fifth line, we dropped

$$(1-z) \cdot \left(\epsilon - \frac{\epsilon}{1+2\epsilon}\right) \cdot \ln\left(\frac{\theta+\epsilon}{1+\epsilon-\theta}\right) \leq 1 \cdot (\epsilon^2/(1+2\varepsilon)) \cdot \ln(2/\epsilon) = O((\ln(1/\epsilon) \cdot \varepsilon)^2). \qquad \square$$

Given this more manageable form for $f$, we can now turn to prove Lemma 11.10, restated below.

**Lemma 11.10.** *(Basic bounds on conditional probabilities $z_u$) There exist absolute constants $c \in (0,1)$ and $C > 1/c > 1$ and $\epsilon_0 \in (0,1)$ such that for every $\epsilon \in (0,\epsilon_0)$ the following holds: for every vertex $v \in V$, if $y_u$ is the dual variable of a neighbor $u \in N_v(v)$ before $v$'s arrival and $\theta$ is the value chosen by Algorithm 11.1 on $v$'s arrival, then for $z_u$ as defined in Algorithm 11.3, we have:*

**(1)** *If $\theta \notin (c, 1-c)$, then $\sum_{u \in N_v(v)} z_u \leq 1$,*

**(2)** *If $\theta \in [0,1]$, then $\sum_{u \in N_v(v)} z_u \leq 1 + C\epsilon$,*

**(3)** *If $\sum_{u \in N_v(v)} z_u > 1$, then $z_u \leq C\sqrt{\epsilon}$ for every $u \in N_v(v)$,*

**(4)** *If $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ such that $z_u > 0$, one has $y_u \in [c/2, 1-c/2]$, and*

**(5)** *For all $u \in N_v(v)$, one has $z_u \leq 1/2 + O(\sqrt{\epsilon})$.*

*Proof.* We begin by getting a generic upper bound for $z_u$. We note that each edge $e$ is matched by Algorithm 11.3 with probability at most $x_e$ by Line 11. Therefore, $u$ is matched before $v$ arrives with probability at most $x_u := \sum_{w \in N_v(u) \setminus \{v\}} x_{wu}$, the fractional degree of $u$ before $v$ arrives. Therefore, by Lemma 11.3, the probability that $u$ is free is at least

$$\Pr[u \text{ free when } v \text{ arrives}] \geq 1 - x_u \geq 1 - \frac{y_u + f(1-y_u)}{\beta}, \tag{C.2}$$

from which, together with the definition of $x_{uv} = \frac{1}{\beta}(\theta - y_u)^+ \left(1 + \frac{1-\theta}{f(\theta)}\right)$, we obtain the following upper bound on $z_u$:

$$z_u = \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrive}]} \leq \frac{\frac{1}{\beta}(\theta - y_u)^+ \left(1 + \frac{1-\theta}{f(\theta)}\right)}{1 - \frac{y_u + f(1-y_u)}{\beta}} = \frac{(\theta - y_u)\left(1 + \frac{1-\theta}{f(\theta)}\right)}{\beta - (y_u + f(1-y_u))}. \tag{C.3}$$

We start by upper bounding $\sum_{u \in N_v(v)} z_u$, giving a bound which will prove useful in the proofs of both **(1)** and **(2)**. Recall that $\theta$ is defined as the largest $\theta \leq 1$ such that

$$\sum_{u \in N_v(v)} (\theta - y_u)^+ \leq f(\theta). \tag{C.4}$$

Summing (C.3) over all $u \in N_v(v)$, we find that

$$\begin{aligned}\sum_{u\in N_v(v)} z_u &\le \sum_{u\in N_v(v)} \frac{(\theta - y_u)^+ \cdot (1+\frac{1-\theta}{f(\theta)})}{\beta - (\theta + f(1-\theta))} && (f(\cdot) \text{ is non-increasing, by Observation 11.4})\\ &\le \frac{f(\theta)+1-\theta}{\beta-(\theta+f(1-\theta))} && \left(\begin{matrix}\text{by (C.4) and } \beta \ge \beta^*(f) = 1+f(0) \ge \\ \theta + f(1-\theta))\end{matrix}\right)\end{aligned}$$

We therefore wish to upper bound $\frac{f(\theta)+1-\theta}{\beta-\theta-f(1-\theta)}$. To this end let $\gamma(\theta,\epsilon) := \epsilon \ln\left(\frac{\theta+\epsilon}{1+\epsilon-\theta}\right)$. Before proceeding to the proof, it would be useful to summarize some properties of the function $\gamma(\theta,\epsilon)$.

1. $\gamma(\theta,\epsilon) = -\gamma(1-\theta,\epsilon)$ for all $\theta \in [0,1]$ .
2. For $c$, $\epsilon_0$ sufficiently small we have for all $\theta \in [0,c)$ that $\gamma(\theta,\varepsilon) \le \epsilon \ln\left(\frac{c+\epsilon}{1+\epsilon-c}\right) \le -20\cdot\varepsilon$, and for all $\theta \in (1-c,1]$ that $\gamma(\theta,\epsilon) \ge \epsilon \ln\left(\frac{1-c+\epsilon}{1+\epsilon-(1-c)}\right) \ge 20\cdot\varepsilon$.
3. $\gamma(\theta,\epsilon)\cdot(1-2\theta) \le 0$ for $\theta \in [0,1]$, since $\gamma(\theta,\epsilon) \le 0$ for $\theta \le 1/2$ and $\gamma(\theta,\epsilon) \ge 0$ for $\theta \ge 1/2$.
4. $\theta\cdot\gamma(\theta,\epsilon) \ge -\epsilon$ for all $\theta \in [0,1]$.

   The last property follows from $\ln\left(\frac{1+\epsilon-\theta}{\theta+\epsilon}\right) \le \ln\left(\frac{1+\epsilon+\theta}{\theta+\epsilon}\right) \le \ln\left(1+\frac{1}{\theta+\epsilon}\right) \le \frac{1}{\theta+\epsilon} \le \frac{1}{\theta}$, which implies in particular that $\theta\cdot\gamma(\theta,\epsilon) = \theta\cdot\epsilon\cdot\left(-\ln\left(\frac{1+\epsilon-\theta}{\theta+\epsilon}\right)\right) \ge -\epsilon$.

We will use $\gamma$ as shorthand for $\gamma(\theta,\epsilon)$. Recalling that $\beta = 2-\epsilon$ and using Lemma C.1, we have:

$$\begin{aligned}\frac{f(\theta)+1-\theta}{\beta-(\theta+f(1-\theta))} &\le \frac{(1-\theta)\left(1+\epsilon\ln\left(\frac{\theta+\epsilon}{1+\epsilon-\theta}\right)\right) - \theta + 1 + 1.01\epsilon}{2-\epsilon-\theta-\theta\left(1+\epsilon\ln\left(\frac{1-\theta+\epsilon}{\theta+\epsilon}\right)\right) - 1.01\epsilon}\\ &\le \frac{(1-\theta)(2+\gamma)+2\epsilon}{2-2\theta+\theta\gamma-3\epsilon}\\ &= 1 + \frac{\gamma(1-2\theta)+5\epsilon}{2-2\theta+\theta\gamma-3\epsilon}.\end{aligned} \tag{C.5}$$

We will continue by proving that the second term is negative. First we prove that the denominator is positive. To this end, first consider the case when $\theta \in [0,c)$. In this case for $\epsilon_0, c$ sufficiently small one has that: $2-2\theta+\theta\gamma-2\epsilon > 2-2\theta-\epsilon-2\epsilon > 0$ from Item 4. Moreover, when $\theta \in (1-c,1]$ one has that $\theta > \frac{1}{2}$ (since $c$ is small) and $\gamma \ge 20\epsilon$ from Item 2. Thus $2-2\theta+\theta\gamma-2\epsilon \ge \theta\gamma - 2\epsilon \ge \frac{1}{2}\cdot 20\epsilon - 3\epsilon = 7\epsilon > 0$. Now, it remains to prove that the numerator is always negative. When $\theta \in [0,c)$ we have that $1-2\theta \ge 3/4$(since $c$ is small) and $\gamma \le -20\epsilon$ from Item 2, therefore $\gamma(1-2\theta)+5\epsilon \le \gamma\cdot\frac{3}{4} + 5\cdot(-\frac{\gamma}{20}) = \frac{\gamma}{2} < 0$. In the case where $\theta \in (1-c,1]$, we have that $1-2\theta < -3/4$, and $\theta > 1/2$ (since $c$ is small), and $\gamma \ge 20\epsilon$ from Item 2, thus $\gamma(1-2\theta)+5\epsilon \le -\frac{3}{4}\cdot 20\epsilon + 5\epsilon = -10\epsilon < 0$.

We now turn to **(2)**. We assume that $\theta \in (c,1-c)$, since otherwise the claim is trivial, by **(1)**. We have by (C.5) that $\frac{f(\theta)+1-\theta}{\beta-(\theta-f(1-\theta))} \le 1 + \frac{\gamma(1-2\theta)+5\epsilon}{2-2\theta+\theta\gamma-3\epsilon}$. We have that $\gamma(1-2\theta)+5\epsilon \le 5\epsilon$ from Item 3. Furthermore, using Item 4 we have that $2-2\theta+\theta\gamma-3\epsilon \ge 2c + -4\epsilon > c$ for a sufficiently small $\epsilon_0$. Overall, the second term is bounded above by $\frac{5}{c}\cdot\epsilon < C\cdot\epsilon$, for $C > \frac{5}{c} > \frac{1}{c}$ as required.

We now prove **(3)**. Note that by **(1)**, $\sum_{u\in N_v(v)} z_u > 1$ implies that $\theta \in (c, 1-c)$. Now, for every $u \in N_v(v)$, let $\alpha_u := \frac{(\theta-y_u)^+}{f(\theta)}$, so that $y_u = \theta - f(\theta)\cdot\alpha_u$ if $y_u \le \theta$. We also note that by definition of $\alpha_u$ and our choice of $\theta$, we have $\sum_{u\in N_v(v)} \alpha_u = \sum_{u\in N_v(v)} \frac{(\theta-y_u)^+}{f(\theta)} \le 1$. In the proof of **(3)** and **(4)** we will assume for notational simplicity that all $u \in N_v(v)$ have $y_u \le \theta$, implying $z_u \ge 0$. Summing up (C.3) over all $u \in N_v(v)$ and substituting in $\alpha_u$, we thus find that

$$\begin{aligned}
\sum_{u\in N_v(v)} z_u &\le \sum_{u\in N_v(v)} \frac{(\theta-y_u)^+(1+\frac{1-\theta}{f(\theta)})}{\beta-(y_u+f(1-y_u))} \\
&= \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{\beta-(y_u+f(1-y_u))} \\
&\le \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-y_u-f(1-y_u)-2.01\epsilon} \qquad \text{(by Lemma C.1 and } \beta = 2-\varepsilon) \\
&\le \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\epsilon-2y_u},
\end{aligned}$$

In the last transition we used again (as in Item 4) that $y_u \cdot \epsilon \ln\left(\frac{1-y_u+\epsilon}{y_u+\epsilon}\right) \le \epsilon$, which implies $f(\theta) \le 1-\theta+\epsilon$ for all $\theta \in [0,1]$. Substituting $y_u = \theta - f(\theta)\cdot\alpha_u$ into the above upper bound on $\sum_{u\in N_v(v)} z_u$, we get

$$\begin{aligned}
\sum_{u\in N_v(v)} z_u &\le \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\varepsilon-2\theta+2f(\theta)\cdot\alpha_u} \\
&= \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\epsilon-2\theta} \\
&\qquad - \sum_{u\in N_v(v)} \frac{(f(\theta)+1-\theta)\cdot 2f(\theta)\cdot\alpha_u^2}{(2-4\epsilon-2\theta)\cdot(2-4\epsilon-2\theta+2f(\theta)\cdot\alpha_u)},
\end{aligned} \tag{C.6}$$

using the elementary identity $\frac{1}{a+b} = \frac{1}{a} - \frac{b}{a(a+b)}$ for appropriate $a$ and $b$. Now, both terms in the last line of (C.6) can be significantly simplified, as follows. For the former term, again using that $f(\theta) \le 1-\theta+\epsilon$, together with $\sum_{u\in N_v(v)} \alpha_u \le 1$ noted above, we find that

$$\begin{aligned}
\sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\epsilon-2\theta} &\le \sum_{u\in N_v(v)} \alpha_u \cdot \frac{2+\epsilon-2\theta}{2-4\epsilon-2\theta} \\
&= \sum_{u\in N_v(v)} \alpha_u \cdot \left(1+\frac{5\varepsilon}{2-4\epsilon-2\theta}\right) \le 1+O(\varepsilon),
\end{aligned} \tag{C.7}$$

where in the last step we used that $\theta \le 1-c$ and $c$ is some fixed constant. For the second term in the last line of (C.6), we note that

$$\sum_{u\in N_v(v)} \frac{(f(\theta)+1-\theta)\cdot 2f(\theta)\cdot\alpha_u^2}{(2-4\epsilon-2\theta)\cdot(2-4\epsilon-2\theta+2f(\theta)\cdot\alpha_u)} = \Omega(1)\cdot\left(\sum_{u\in N_v(v)} \alpha_u^2\right). \tag{C.8}$$

To see this, first note that for $\theta \in (c, 1-c)$, the numerator of each summand of the LHS is at least $2f(c)^2\cdot\alpha_u^2 \ge \Omega(\alpha_u^2)$, since $f$ is decreasing by Observation 11.4 and $f(c) \ge \frac{1}{2}\cdot(1+\varepsilon-c) \ge \Omega(1)$

for $c$ and $\varepsilon$ sufficiently small. To verify the first inequality of this lower bound for $f(c)$, recall that $f(c) = (1+\varepsilon-c)\cdot\left(\frac{c+\varepsilon}{1+\varepsilon-c}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$. Now, for $\epsilon$ tending to zero and $c < 1/2$, the term $\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$ tends to one as $\epsilon$ tends to zero. Therefore for $\epsilon$ sufficiently small we have $f(c) \geq \frac{1}{2}\cdot(1+\epsilon-c)$ for all $c < 1/2$. We now turn to upper bounding the denominator of each summand in the LHS of Equation (C.8). Indeed, substituting $y_u = \theta - f(\theta)\cdot\alpha_u$, we find that each such denominator is at most $(2-4\epsilon-2\theta)\cdot(2-4\epsilon-2\theta+2f(\theta)\cdot\alpha_u) \leq (1/2)\cdot(2-4\varepsilon-2y_u) \leq (1/2)\cdot(2-4\varepsilon-2c) \leq O(1)$ for $c$ and $\epsilon$ sufficiently small. Note that both numerator and denominator are positive for sufficiently small $c$ and $\epsilon_0$. Substituting the bounds of (C.7) and (C.8) into (C.6), we obtain

$$\sum_{u\in N_v(v)} z_u \leq 1 + O(\epsilon) - \Omega(1)\cdot\left(\sum_{u\in N_v(v)} \alpha_u^2\right). \tag{C.9}$$

From Eq. (C.9) and $\sum_{u\in N_v(v)} z_u > 1$ by assumption of **(3)**, we get that

$$\sum_{u\in N_v(v)} \alpha_u^2 \leq C'\epsilon \tag{C.10}$$

for an absolute constant $C' > 1$, since otherwise $\sum_{u\in N_v(v)} z_u \leq 1$. Finally, it remains to note that

$$\begin{aligned}
&\sum_{u\in N_v(v)} z_u^2 \\
&\quad= \sum_{u\in N_v(v)} \left(\frac{\alpha_u\cdot(f(\theta)+1-\theta)}{\beta-(y_u+f(1-y_u))}\right)^2 \\
&\quad\leq \left(\sum_{u\in N_v(v)} \alpha_u^2\right)\cdot\left(\frac{f(\theta)+1-\theta}{\beta-(\theta+f(1-\theta))}\right)^2 && \text{(Observation 11.4 and } y_u \leq \theta) \\
&\quad\leq \left(\sum_{u\in N_v(v)} \alpha_u^2\right)\cdot\left(\frac{f(\theta)+1-\theta}{\beta-(1-c+f(c))}\right)^2 && \text{(Observation 11.4 and } \theta \leq 1-c) \\
&\quad\leq \left(\sum_{u\in N_v(v)} \alpha_u^2\right)\cdot\left(\frac{1-\theta+\varepsilon+1-\theta}{\beta-(1-c+1-c+\varepsilon)}\right)^2 && (f(c) \leq 1-c+\varepsilon) \\
&\quad\leq \left(\sum_{u\in N_v(v)} \alpha_u^2\right)\cdot\frac{2}{2c-2\epsilon} \leq C\epsilon,
\end{aligned}$$

for some constant $C \geq \frac{2}{2c-2\epsilon}$. Thus $z_u^2 \leq \sum_{u\in N_v(v)} z_u \leq C\epsilon$ and so $z_u \leq \sqrt{C\cdot\epsilon} \leq C\sqrt{\epsilon}$, as claimed.

We now prove **(4)**. Since $\sum_{u\in N_v(v)} z_u > 1$ implies $\theta \in (c, 1-c)$ by **(1)**, using the definition of $\alpha_u$'s from the proof of **(3)** together with the fact that $\alpha_u \leq C'\sqrt{\epsilon}$ for every $u \in N_v(v)$ by (C.10)

and the fact that $f(\theta) \leq 2$ for all $\theta \in [0,1]$ (by Lemma C.1), we get that

$$y_u = \theta - f(\theta) \cdot \alpha_u \in [c - O(\sqrt{\epsilon}), 1-c] \subseteq [c/2, 1 - c/2],$$

for sufficiently small $\epsilon_0 > 0$, as required.

As for **(5)**, simplifying (C.3) and using the fact that $\theta - y_u \leq f(\theta)$, we get

$$z_u \leq \frac{\theta - y_u + 1 - \theta}{\beta - y_u - f(1-y_u)} = \frac{1-y_u}{\beta - y_u - f(1-y_u)}.$$

Recall from Lemma C.1 that for all $\theta \in [0,1]$, we have $f(\theta) \leq (1-\theta)\left(1 + \varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right) + 1.01\epsilon$, which implies the following:

1. For all $\theta \in [0,1]$, we have $f(\theta) \leq 1 - \theta + \sqrt{\epsilon}$, and
2. For $\theta < e^{-10}$, we have $f(\theta) \leq (1-\theta)\left(1 + \epsilon \ln\left(\frac{e^{-10}+\epsilon}{1-e^{-10}+\epsilon}\right)\right) + 1.01\epsilon \leq 1 - \theta - 2\epsilon$.

Suppose that $y_u \leq 1 - e^{-10}$. Then using Item 1, we have

$$\begin{aligned} z_u &\leq \frac{1-y_u}{\beta - y_u - f(1-y_u)} \leq \frac{1-y_u}{2 - \epsilon - y_u - y_u - \sqrt{\epsilon}} \\ &\leq \frac{1-y_u}{2(1-y_u) - 2\sqrt{\epsilon}} \leq 1/2 + \frac{2\sqrt{\epsilon}}{2e^{-10} - 2\sqrt{\epsilon}} \leq 1/2 + O(\sqrt{\epsilon}). \end{aligned}$$

Now suppose that $y_u > 1 - e^{-10}$. Then $1 - y_u < e^{-10}$, and so by Item 2, $f(1-y_u) \leq 1 - y_u - 2\epsilon$. Thus we have

$$z_u \leq \frac{1-y}{\beta - y_u - f(1-y_u)} \leq \frac{1-y_u}{2 - \epsilon - y_u - (y_u - 2\epsilon)} = \frac{1-y_u}{2(1-y_u) + \epsilon} \leq 1/2,$$

completing the proof. □

Finally, we rely on Lemma C.1 to prove that the fractional solution maintained by Line 3 is $1/\beta$ competitive, as implied by Lemma 11.5 and the following restated fact.

**Fact 11.1.** *For all sufficiently small $\epsilon > 0$, we have that $2 - \epsilon \geq \beta^*(f_{1+2\epsilon})$.*

*Proof.* Let us denote as before $f = f_{1+2\epsilon}$. Recall that $\beta^*(f) = 1 + f(0)$. By Lemma C.1, this is at most $1 + f(0) \leq 1 + \left(1 + \varepsilon \ln\left(\frac{\varepsilon}{1+\varepsilon}\right)\right) + 1.01\epsilon$. But for small enough $\epsilon$, we have that $\ln\left(\frac{\varepsilon}{1+\varepsilon}\right) \leq -2.01$, implying that $1 + f(0) \leq 2 - \epsilon$, as claimed. □

# Bibliography


[1] Marek Adamczyk, Fabrizio Grandoni, and Joydeep Mukherjee. Improved approximation algorithms for stochastic matching. In *Algorithms - ESA 2015*, pages 1–12, 2015.

[2] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8 (3):307–328, 2004.

[3] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, January 2013. ISSN 0890-5401. doi: 10.1016/j.ic.2012.10.006. URL http://dx.doi.org/10.1016/j.ic.2012.10.006.

[4] Kook Jin Ahn and Sudipto Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. *ACM Transactions on Parallel Computing (TOPC)*, 4(4):17, 2018.

[5] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *Journal of Algorithms*, 7(4):567–583, 1986. doi: 10.1016/0196-6774(86)90019-2. URL http://dx.doi.org/10.1016/0196-6774(86)90019-2.

[6] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th ACM Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31–June 3, 2014*, pages 574–583, 2014. doi: 10.1145/2591796.2591805. URL http://doi.acm.org/10.1145/2591796.2591805.

[7] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 43–60, 2016.

[8] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem: Beating half with a non-adaptive algorithm. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, pages 99–116, 2017.

[9] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets Meet EDCS: Algorithms for Matching and Vertex Cover on Massive Graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019.

[10] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, Aug 2012. URL https://doi.org/10.1007/s00453-011-9511-8.

[11] Alok Baveja, Amit Chavan, Andrei Nikiforov, Aravind Srinivasan, and Pan Xu. Improved bounds in stochastic matching and optimization. *Algorithmica*, Oct 2017. ISSN 1432-0541. doi: 10.1007/s00453-017-0383-4. URL https://doi.org/10.1007/s00453-017-0383-4.

[12] Paul Beame, Paraschos Koutris, and Dan Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA, June 22–27, 2013*, pages 273–284, 2013. doi: 10.1145/2463664.2465224. URL http://doi.acm.org/10.1145/2463664.2465224.

[13] Soheil Behnezhad and Nima Reyhani. Almost optimal stochastic weighted matching with few queries. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, EC '18, pages 235–249, 2018.

[14] Aaron Bernstein. Improved bound for matching in random-order streams. *arXiv preprint arXiv:2005.00417*, 2020.

[15] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM SIGACT News*, 39(1):80–87, 2008.

[16] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 325–342, 2015.

[17] Sebastian Brandt, Manuela Fischer, and Jara Uitto. Matching and MIS for Uniformly Sparse Graphs in the Low-Memory MPC Model. *arXiv preprint arXiv:1807.05374*, 2018.

[18] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New Algorithms, Better Bounds, and a Novel Model for Online Stochastic Matching. In *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57, pages 24:1–24:16, 2016.

[19] Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, pages 1–19, Aug 2018.

[20] Moses Charikar, Chandra Chekuri, and Martin Pál. Sampling bounds for stochastic optimization. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 257–269, 2005. ISBN 978-3-540-31874-3.

[21] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Proc. 36th International Colloquium on Automata, Languages and Programming*, pages 266–278, 2009.

[22] Ashish Chiplunkar, Sumedh Tirodkar, and Sundar Vishwanathan. On randomized algorithms for matching in the online preemptive model. In *Algorithms-ESA 2015*, pages 325–336. Springer, 2015.

[23] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979, 2018.

[24] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[25] Kevin P. Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *Proc. 39th International Colloquium on Automata, Languages and Programming*, pages 822–833, 2012.

[26] Michael Crouch and Daniel M. Stubbs. Improved Streaming Algorithms for Weighted Matching, via Unweighted Matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2014)*, volume 28 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 96–104, 2014.

[27] Artur Czumaj, Jakub Łącki, Aleksander Mądry, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 471–484. ACM, 2018.

[28] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: the benefit of adaptivity. In *Proc. 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 208–217, 2004.

[29] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 395–404, 2005. URL http://dl.acm.org/citation.cfm?id=1070432.1070487.

[30] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 101–107, 2013.

[31] Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economic-based analysis of ranking for online bipartite matching. *arXiv preprint arXiv:1804.06637*, 2018.

[32] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[33] Sebastian Eggert, Lasse Kliemann, Peter Munstermann, and Anand Srivastav. Bipartite matching in the semi-streaming model. *Algorithmica*, 63(1):490–508, 2012.

[34] Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim, and Sahil Singla. Prophet secretary for combinatorial auctions and matroids. In *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 700–714, 2018. URL http://dl.acm.org/citation.cfm?id=3174304.3175316.

[35] Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for online preemptive matching. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, page 389, 2013.

[36] Uriel Feige. Tighter bounds for online bipartite matching. *arXiv preprint arXiv:1812.11774*, 2018.

[37] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, December 2005.

[38] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S. Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Proc. 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 117–126, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3850-1. doi: 10.1109/FOCS.2009.72. URL https://doi.org/10.1109/FOCS.2009.72.

[39] Buddhima Gamlath, Sagar Kale, Slobodan Mitrovic, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, page 491–500, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362177. doi: 10.1145/3293611.3331603. URL https://doi.org/10.1145/3293611.3331603.

[40] Buddhima Gamlath, Sagar Kale, and Ola Svensson. Beating greedy for stochastic bipartite matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2841–2854. SIAM, 2019.

[41] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 26–37, 2019. doi: 10.1109/FOCS.2019.00011.

[42] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006.

[43] Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2019.

[44] Mohsen Ghaffari and David Wajc. Simplified and Space-Optimal Semi-Streaming $(2+\varepsilon)$-Approximate Matching. In *2nd Symposium on Simplicity in Algorithms*, 2019.

[45] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 129–138. ACM, 2018.

[46] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008.

[47] Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the MapReduce framework. In *International Symposium on Algorithms and Computation*, pages 374–383. Springer, 2011.

[48] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, jun 1981.

[49] M Grötschel, L Lovász, and A Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, 1988.

[50] Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 241–253, 2017.

[51] Bernhard Haeupler, Vahab S. Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *Internet and Network Economics*, WINE '11, pages 170–181, 2011.

[52] John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 122–125, 1971.

[53] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. How to match when all vertices arrive online. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 17–29, 2018.

[54] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1070–1081, 2018.

[55] Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2875–2886, 2019.

[56] Amos Israeli and Alon Itai. A fast and simple randomized parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):77–80, 1986. doi: 10.1016/0020-0190(86)90144-4. URL http://dx.doi.org/10.1016/0020-0190(86)90144-4.

[57] Amos Israeli and Yossi Shiloach. An improved parallel algorithm for maximal matching. *Information Processing Letters*, 22(2):57–60, 1986. doi: 10.1016/0020-0190(86)90141-9. URL http://dx.doi.org/10.1016/0020-0190(86)90141-9.

[58] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 39(3):624–646, Aug 2014.

[59] Sagar Kale and Sumedh Tirodkar. Maximum Matching in Two, Three, and a Few More Passes Over Graph Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, volume 81 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 15:1–15:21, 2017.

[60] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 1679–1697, 2013.

[61] Michael Kapralov. Space lower bounds for approximating maximum matching in the edge arrival model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1874–1893. SIAM, 2021.

[62] Howard J. Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17–19, 2010*, pages 938–948, 2010. doi: 10.1137/1.9781611973075.76. URL http://dx.doi.org/10.1137/1.9781611973075.76.

[63] Richard M Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random nc. *Combinatorica*, 6(1):35–48, 1986.

[64] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 352–358, 1990.

[65] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990.

[66] Irit Katriel, Claire Kenyon-Mathieu, and Eli Upfal. Commitment under uncertainty: Two-stage stochastic matching problems. In *Automata, Languages and Programming*, pages 171–182. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73420-8.

[67] Christian Konrad. A Simple Augmentation Method for Matchings with Applications to Streaming Algorithms. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117, pages 74:1–74:16, 2018.

[68] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012, Proceedings*, volume 7408, page 231. Springer, 2012. URL http://arxiv.org/abs/1112.0184.

[69] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[70] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in MapReduce. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2011, San Jose, CA, USA, June 4–6, 2011*, pages 85–94, 2011. doi: 10.1145/1989493.1989505. URL http://doi.acm.org/10.1145/1989493.1989505.

[71] Euiwoong Lee and Sahil Singla. Maximum matching in the online batch-arrival model. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 355–367, 2017.

[72] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *Journal of the ACM*, 62(5):38:1–38:17, November 2015. ISSN 0004-5411. doi: 10.1145/2786753. URL http://doi.acm.org/10.1145/2786753.

[73] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, volume 79, pages 565–574, 1979.

[74] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15(4):1036–1053, 1986. doi: 10.1137/0215074. URL http://dx.doi.org/10.1137/0215074.

[75] Takanori Maehara and Yutaro Yamaguchi. Stochastic packing integer programs with few queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '18, pages 293–310, 2018.

[76] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1285–1294, 2011.

[77] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 170–181. Springer, 2005.

[78] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013.

[79] Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V|}|E|)$ algoithm for finding maximum matching in general graphs. In *Proceedings of the 21st Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980.

[80] Marco Molinaro and R. Ravi. The query-commit problem. *CoRR*, abs/1110.0990, 2011. URL https://arxiv.org/abs/1110.0990.

[81] Ketan Mulmuley, Umesh V Vazirani, and Vijay V Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354. ACM, 1987.

[82] Krzysztof Onak. Round Compression for Parallel Graph Algorithms in Strongly Sublinear Space. *arXiv preprint arXiv:1807.08745*, 2018.

[83] Ami Paz and Gregory Schwartzman. A $(2+\varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2153–2161, 2017. doi: 10.1137/1.9781611974782.140. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611974782.140.

[84] Sahil Singla. The Price of Information in Combinatorial Optimization. *Proc. 29th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2523–2532, nov 2018.

[85] Sumedh Tirodkar and Sundar Vishwanathan. Maximum matching on trees in the online preemptive and the incremental dynamic graph models. In *Proceedings of the 23rd International Computing and Combinatorics Conference (COCOON)*, pages 504–515, 2017.

[86] Yajun Wang and Sam Chiu-wai Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1070–1081, 2015.

# Gamlath Ralalage Buddhima Ruwanmini Gamlath

bgamlath@gmail.com +41 78 673 65 28


## Education

Sept 2016 - Present **École polytechnique fédérale de Lausanne (EPFL)** **Switzerland**
*Ph.D. in Computer Science (Expected graduation date: September 2021)*

- Designed and analyzed algorithms for matching and clustering:
  - Algorithms for matching in modern computational models.
  - Algorithms for clustering problems.
- Cumulative GPA: 5.78/6.00.

Oct 2010 - Mar 2015 **University of Moratuwa** **Sri Lanka**
*B.Sc. Engineering (Hons)*

- Specialized in Electronic and Telecommunication Engineering.
- Cumulative GPA: 3.77/4.20 (First Class).

## Publications

2020 Gamlath B., Grinberg V., "Approximating Star Cover Problems", In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM) 2020, Available at: https://arxiv.org/abs/1912.01195.*

2019 Gamlath B., Kapralov M., Maggiori A., Svensson O., Wajc D., "Online Matching with General Arrivals", In *IEEE Annual Symposium on Foundations of Computer Science (FOCS) 2019, Available at: https://arxiv.org/abs/1904.08255.*

2019 Gamlath B., Kale S., Mitrović S., Svensson O., "Weighted Matchings via Unweighted Augmentation", In *ACM Symposium on Principles of Distributed Computing (PODC) 2019, Available at: https://arxiv.org/abs/1811.02760.*

2019 Gamlath B., Kale S., Svensson O., "Beating Greedy for Stochastic Bipartite Matching", In *ACM-SIAM Symposium on Discrete Algorithms (SODA) 2019, Available at: https://arxiv.org/abs/1909.12760.*

2018 Gamlath B., Huang S., Svensson O., "Semi-Supervised Algorithms for Approximately Optimal and Accurate Clustering", In *International Colloquium on Automata, Languages and Programming (ICALP) 2018, Available at: https://arxiv.org/abs/1803.00926.*

Google Scholar citations: `https://scholar.google.com/citations?user=g2160uEAAAAJ`.

## Experience

Feb 2017 - Jan 2021 **Teaching Assistant** **EPFL, Switzerland**

- Algorithms (Fall 2017, 2019, and 2020), Advanced Algorithms (Spring 2018, 2019, and 2020), Computational Complexity (Fall 2018), and Theory of Computation (Spring 2017).
- Designed algorithmic problems and prepared online contests in Codeforces for course assignments with sample solutions and testcase generators written in **C++**.

Jan 2016 - Jul 2016 **Lecturer** **University of Moratuwa, Sri Lanka**

- Prepared and conducted laboratory experiments and tutorials for several undergraduate courses.
- Supervised student projects in electronics-related courses.

Nov 2015 - Dec 2015 **Research Intern** **National University of Singapore, Singapore**

- Conducted research on distributed message-passing protocol for networks with crash failures.

Nov 2013 - May 2014 **Software Engineer** **Zone24x7 Private Limited, Sri Lanka**

- Integrated an in-house secure communication protocol into mobile clients developed in **C++** and **Java** and a sever developed in **C#**.
- Discovered and debugged a rarely occurring bug in closing and reusing pooled connection objects in the server.

## Mathematics and Programming Contests

- Silver medal at International Mathematics Competition for University Students (Bulgaria 2014).
- Bronze medal at International Mathematics Olympiad (Germany 2009).
- Honorable mention at International Mathematics Olympiad (Spain 2008).
- Champions of the SARRC Coding Competition 2013 organized by IIT Delhi, India.
- Ranked in the top 50 in IEEEXtreme programming contest for five years from 2010 to 2014.
- Participated in algorithm contests in HackerRank (Rating: 2400+) and Codeforces (Rating: 2100+).
- Have solved 400+ problems in ProjectEuler.net.

## Skills

| | |
|---|---|
| **Programming** | Fluent in C++, C, Java, Python.<br>Familiar with Scala, OCaml, Go, and Mathematica. |
| **Miscellaneous** | Excellent analytic and problem solving skills.<br>Fluent in Unix command-line tools and version controlling with Git.<br>Experienced with OSX, Linux, and Windows platforms. |
| **Languages** | English (Fluent), French (Elementary), and Sinhalese (Native). |

## Voluntary Work

- Contributed as a sub-reviewer in conferences SODA 2021, FOCS 2020, SOSA 2020, ESA 2019, ICALP 2019, and SODA 2019.
- Setup and tested algorithmic problems for the Helvitic Coding Contest organized by the PolyProg association of EPFL, Switzerland (2018-2019).
- Worked as a voluntary instructor of the Sri Lankan team for the International Mathematics Olympiad (2010-2016).