Multi-agent Coordination using Local Search

Boi Faltings and Quang-Huy Nguyen
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Artificial Intelligence Laboratory (LIA)
CH-1015 Lausanne, Switzerland
{boi.faltings, quanghuy.nguyen}@epfl.ch

Abstract

We consider the problem of coordinating the behavior of multiple self-interested agents. This problem often translates into complex constraint optimization problems. When they are large, they can often only be solved by local search algorithms.

Using local search poses problems of incentive-compatibility and individual rationality of the agents. We define a weaker notion of bounded-rational incentive-compatibility where manipulation is made impossible with high probability through computational complexity. We show how it can apply to local search both through theoretical analysis and practical experiments that show the practicality of the method.

1 Introduction

There are many practical settings where multiple self-interested agents have to coordinate their actions. This coordination often involves joint decisions about resource allocation, scheduling and planning. These problems usually lead to constraint optimization problems. We thus extend the standard definition of such problems to the multi-agent setting as follows:

Definition 1.1 *A discrete* multi-agent constraint optimization problem *(MCOP) is a tuple* $< A, X, D, C, R >$ *where:*

- $A = \{A_1, .., A_m\}$ *is a set of agents.*
- $X = \{x_1, .., x_n\}$ *is a set of variables.*
- $D = \{d_1, .., d_n\}$ *is a set of domains of the variables, each given as a finite set of possible values.*
- $C = \{c_1, .., c_p\}$ *is a set of constraints, where a constraint* c_i *is a function* $d_{i1} \times .. \times d_{il} \rightarrow \{0, 1\}$ *that returns* 1 *if the value combination is allowed and* 0 *if it is not.*
- $R = \{r_1, .., r_o\}$ *is a set of relations, where a relation* r_i *is a function* $d_{i1} \times .. \times d_{il} \rightarrow \Re$ *giving the utility of choosing each combination of values.*
- R_i *is the subset of* R *that gives the relations associated with agent* A_i.

The solution of an MCOP is an assignment of values to all variables that satisfies all constraints and maximizes the sum of agent utilities as expressed by their relations. Note that variables, domains and constraints are common and agreed upon knowledge among the agents. On the other hand, relations are specified by the individual agents, and they do not necessarily have to report them correctly.

Many multi-agent coordination problems can be formulated as a MCOP. For example, meeting scheduling can be formulated by a variable for the starting time of each meeting, constraints between any two meetings involving the same participants ruling out start times that would make them overlap, and relations that express each agent's preferences for meeting times. Additional constraints can be used to express precedence constraints between meetings or external constraints on their times.

Another example of a MCOP problem is allocating capacity in a public network, for example a train or pipeline network. The network is a graph of connections, and only one agent can use any one connection at a given time. This can be represented by having one variable per link and time interval whose domain ranges over the set of agents. No additional constraints are needed.

Agents have the opportunity to serve customers with varying profit margins. Each customer requires linking different combinations of nodes, which in turn translates into combinations of links in the network itself. Thus, each agent has utilities for being able to use certain combinations of links. To optimize their profits, agents would want to find a combined assignment of network capacity to agents so that the sum of their profits can be maximized. Such combinatorial optimization is NP-complete and thus can be solved exactly only for small problems. For large instances, in many cases only *local search* methods can be implemented. They can provide no optimality guarantees, but with high probability will find a solution that is very close to optimal.

Two additional considerations need to be addressed due to the multi-agent setting: incentive-compatibility and individual rationality. We say that an optimization mechanism is *incentive-compatible* if and only if each agent maximizes its expected utility when the protocol finds the truly optimal solution. Depending on the protocol, incentive-compatibility often means that each agent is reporting its relations truthfully, thus one also speaks of *truthful* mechanisms. Clearly, incentive-compatibility is important to obtain a meaningful solution to the MCOP. Incentive-compatibility is often achieved through tax or auction mechanisms such as the VCG mechanism ([Vickrey, 1961; Clarke, 1971; Groves, 1973]).

We say that a mechanism is *individually rational* if and only if it is in the best interest of each agent to participate in the mechanism, i.e if the expected utility that each agent gets when it participates in the mechanism is at least as high as if it did not. This is important because otherwise, agents may choose not to participate in the mechanism.

The seminal work of Ephrati and Rosenschein [1991] was the first to propose applying VCG mechanisms to agent coordination. For constraint optimization, game theory has shown that the only practical mechanism for incentive-compatibility is of the form of a VCG mechanism ([Green and Laffont, 1979]). However, it has also been shown that VCG mechanisms require finding the provably optimal solution ([Nisan and Ronen, 2000]). Many practical settings of optimization problems are too large for complete optimization algorithms. We thus introduce a weaker concept of *bounded-rational incentive-compatibility* where manipulation is hard through computational complexity. The uncertainty created by randomized local search makes it computationally intractable to evaluate the outcome of an untruthful behavior, thus making it uninteresting to agents.

This paper is structured as follows: we first present the local search framework for solving MCOP and define bounded-rational incentive-compatibility. We show how incentive-compatibility can be achieved in each local search step, and then address the key issue of avoiding speculation through the succession of local search steps. We report on experimental results on randomly generated network resource allocation problems that show their performance.

2 Assumptions and Definitions

2.1 Local search framework

Since finding the optimal solution for MCOP is computationally infeasible (NP-complete), in this paper we use local search algorithms (also called *neighborhood search*) to find good, but not necessarily optimal solutions at a reasonable computational cost. Local search is widely used for solving large optimization problems. It has been particularly well studied for satisfiability problems, based on the GSAT procedure ([Selman *et al.*, 1992]). The local search framework we assume in this paper is given in algorithm 1.

The algorithm manipulates a complete assignment of values to all variables, represented as a vector $\underline{v}$. It is initially set by function *SelectInitialSolution* to an assignment that satisfies all constraints and could be random. The algorithm then asks each agent to state its set of relations R_i, where the utilities should be relative to the initial assignment $\underline{v}$ such that the utility of all relations for the assignment $\underline{v}$ is zero. This eliminates the otherwise open parameter of the utility of the initial assignment which has no influence on the optimization result.

Search then proceeds iteratively by local improvements. Function *ChooseNeighbours* provides a set of candidate assignments that are close to the current one and could possibly improve it. In our implementation, they are generated by randomly selecting a variable $x_i \in X$ and generating all assignments that are equal to $\underline{v}$ but assign to x_i different values in the domain of x_i that are consistent with the rest of $\underline{v}$.

Algorithm 1 Local search algorithm for MCOP

procedure *LocalSearch(A,X,D,C)*
  $\underline{v} \leftarrow SelectInitialSolution(X, D, C)$
  for $i \leftarrow 1 : m$ **do**
    $R_i \leftarrow AskRelations(a_i, \underline{v})$
  end for
  repeat
    $\underline{v^{old}} \leftarrow \underline{v}$
    $N \leftarrow ChooseNeighbours(\underline{v^{old}}, X, D, C)$
    $(\underline{v}, \underline{pay}) \leftarrow LocalChoice(N, R)$
    agents make/receive payments according to $\underline{pay}$
  until termination condition met
  return $\underline{v}$
end procedure

In the second step of the iteration, the assignment $\underline{v}$ is updated using the function *LocalChoice*. It chooses a new assignment to optimize the combined utility according to the relations in R. It also computes a vector of payments $\underline{pay}$ that agents must make or receive in the third step of the iteration. The payments sum up to zero and the way they are derived is described in detail in Section 3.

The iteration continues until a termination condition is met, for example when there is no further improvement in the utility of all agents for some number of steps. To avoid getting stuck in local optima, the performance of a local search procedure is significantly improved by randomization. This means that occasionally *LocalChoice* chooses a value that may even decrease decrease agent utility ([Kirkpatrick *et al.*, 1983; Selman *et al.*, 1994]).

2.2 Bounded-rational Incentive-compatibility

The local search procedure can only work correctly if agents accurately report their utilities R. Using side payments, we can create an *incentive-compatible* mechanism where agents are motivated to truthfully report these valuations. Well-known results in game theory ([Green and Laffont, 1979]) have shown that all mechanisms for MCOP that are incentive-compatible, individually rational and select the optimal solution must be a kind of VCG mechanism. Furthermore, Nisan and Ronen [2000] have shown that a VCG mechanism requires computation of a provably optimal solution to the optimization problem. Thus, there is no mechanism that makes local search incentive-compatible while maintaining individual rationality.

We thus replace incentive-compatibility with a weaker notion, called *bounded-rational incentive-compatibility* that uses computational complexity to rule out manipulation. Such a protection by computational complexity is similar to what has been proposed for other coordination mechanisms such as voting ([Conitzer and Sandholm, 2003]).

Definition 2.1 *(Bounded-rational agent): an agent is called bounded rational if it can examine at most C states of the* *local search before declaring its utilities R_i.*

Any real computational agent is bounded rational for a sufficiently high C.

Definition 2.2 *Let p_t be a bound on the probability that a bounded rational agent can predict whether it has an expected utility gain from an untruthful utility declaration. A mechanism is* bounded rational incentive compatible *if by varying a parameter, p_t can be made arbitrarily close to 0.*

Thus, we we require that *in almost all cases*, manipulation requires an amount of computation that is beyond reach of a bounded-rational agent.

3 Local choice step

We now consider incentive-compatibility and randomization of a single step in the local search, carried out by the function *LocalChoice*. We consider that local changes are made to an individual variable x only, but it is straightforward to generalize the mechanism to other neighbourhoods.

We assume that the number of possible alternatives is sufficiently small so that *LocalChoice* can apply a systematic and complete optimization procedure. We call v_c the current assignment to x, and let v_S^* be the value that would be optimal for the set of agents S, i.e. would most improve the sum of their utilities.

3.1 Incentive-compatibility

As the local choice depends on the relations declared by the individual agents, agents would normally have an incentive to report excessive utility gains for their preferred changes so that they impose them over those preferred by others. Thus, the optimal solution according to the declarations of a set of agents A, which we call $\overline{v_A}$, could be different from v_A^*. We counter this tendency by imposing side payments depending on the utility declarations and the change chosen by the mechanism.

As already mentioned earlier, the only side payments that achieve incentive-compatibility, individual rationality and choose the optimal solution are VCG payments. For an agent a_i, the VCG payment is the "damage" it does the others, i.e. the decrease in utility gain its presence causes to the remaining agents:

$$\begin{aligned}
& VCGtax(a_i) \\
= & \sum_{r \in R \setminus r_i} \left[(r(\overline{v_{A \setminus a_i}}) - r(v_c)) - (r(\overline{v_A}) - r(v_c)) \right] \\
= & \sum_{r \in R \setminus r_i} \left[r(\overline{v_{A \setminus a_i}}) - r(\overline{v_A}) \right]
\end{aligned}$$

Note that since $\overline{v_{A \setminus a_i}}$ is optimized for $A \setminus a_i$, the sum of its utilities for these agents will always be at least as large as that for $\overline{v_A}$ and thus the $VCGtax$ is never negative. Thus, the payments of all agents together leave a positive budget surplus.

This can change when a_i can control the values of certain variables, such as in a trading scenario. It can then actually enable the other agents to produce a higher utility. In this case, the payment will be negative and in fact the sum of all payments may leave a deficit. This is not the case in an MCOP, and thus the payments always produce a surplus.

3.2 Randomization

Randomization has first been proposed as simulated annealing ([Kirkpatrick *et al.*, 1983]), a technique inspired from the cooling of spin glasses. More recently, it has been studied more systematically, particularly in the context of satisfiability problems. For example, the GSAT algorithm ([Selman *et al.*, 1992]) has been turned into the GWSAT algorithm ([Selman *et al.*, 1994]) by adding a random walk strategy: with some probability, the strategy forces an improvement in a clause by ignoring the other clauses that would also be affected. It has been shown that this randomization has the effect that the algorithm eventually finds the optimal solution with high probability.

The random walk steps in GWSAT leave certain randomly chosen constraints out of the local choice steps. We adopt a similar scheme where we randomly select a set of relations to be left out of the optimization at a local choice step. It turns out that a good way to select these relations is to take all the relations belonging to a randomly selected agent a_e. As we show below, this way of randomizing allows us to simultaneously guarantee budget-balance of the VCG tax mechanism.

The GWSAT algorithm has been further extended into the popular Walksat algorithm ([Selman *et al.*, 1994]) by adding further heuristics on when to carry out random walk steps. In our case, this would complicate the incentive-compatibility properties so we have not followed this extension. As it stands, the performance of this randomization scheme is already very good. In the experiments we report later in the paper, we show that it significantly improves search performance over strict hill-climbing.

3.3 Payment budget balance and individual rationality

One problem with the VCG mechanism is that agents generate a surplus of taxes that cannot be returned to them without violating the incentive-compatibility properties. This not only reduces their net utility gain, but also creates incentives for whatever third party receives this gain.

The randomization allows us to make the VCG tax scheme budget balanced by simply paying the payment surplus to the agent a_e that was excluded from the optimization step. Each agent a_i other than a_e pays to a_e the following tax:

$$VCGtax_{-e}(a_i) = \sum_{r \in R \setminus (r_i \cup r_e)} \left[r(\overline{v_{A \setminus (a_i \cup a_e)}}) - r(\overline{v_{A \setminus a_e}}) \right]$$

and the mechanism chooses $\overline{v_{A \setminus a_e}}$ to implement. This can be seen as compensating the agent for the loss of utility it is likely to incur as a consequence of having been left out of the optimization, and does not affect the incentive-compatibility properties:

- for agents other than a_e, it is still best to report their utilities truthfully since they follow a VCG mechanism in world where a_e does not exist.
- for a_e, its declarations have no effect on the outcome or payments so any declaration is equally good. However,

it does not know in advance that it will be excluded, so it still has an interest to make a truthful declaration.

This mechanism is similar to the proposal in [Ephrati and Rosenschein, 1991], who proposed giving the surplus to agents that have no interest in the variable being considered, called uninterested agents. The mechanism proposed here applies even when no uninterested agent exists. When there are uninterested agents, optimization can be improved by selecting these to be chosen as excluded agents.

In certain cases this compensation could be less than the utility loss of the agent, and thus it would not be individually rational for the agent to participate. In fact, no matter what payment scheme is used, whenever the local search step leads to a reduction in total agent utility, there must be at least one agent for which individual rationality is violated. Any randomized local search algorithm will occasionally make such moves, for otherwise it would be susceptible to getting stuck in local optima. Thus, no scheme can guarantee individual rationality at each randomized local choice step.

As the algorithm on the whole improves utility for the community of agents, this does not mean that the local search process as a whole is not individually rational. In particular, if all agents are symmetrical no agent is systematically disadvantaged by the randomization, and so in expectation the scheme should be individually rational for all agents. This is confirmed in the simulations we report later in this paper, where individual rationality was always satisfied for all agents in all runs. However, no hard guarantee can be given for this.

4 Sequences of Local Choices

In the LocalChoice process (Algorithm 1), the utilities associated with each choice of a variable x_i depend on the current assignment $\underline{v^{old}}$ to the remaining variables. Certain assignments $\underline{v^{old}}$ could be more favorable to an agent a_j in that it can obtain its favored value with a lower payment. a_j thus has an incentive to manipulate this assignment by making non-truthful utility declarations in earlier search steps.

In order to carry out such manipulation, a_j has to predict the outcome of the local search computation given a manipulation and compare it with the outcome that would be obtained otherwise. We assume that it has perfect knowledge of the utility declarations of all other agents. Obviously, this is the best possible situation for a manipulator so it gives us worst-case guarantees.

To determine whether a certain manipulation is profitable, an agent needs to determine the expected utility it obtains from the entire local search process when it applies the manipulation. To do this, it can either adopt a steady-state view, where it considers the search as a stationary Markov process and determines its optimal policy, or it can explicitly simulate the behavior from the current step forward. We assume that the optimization problem is sufficiently complex that the number of reachable states and transitions is well beyond the capacity C of a bounded rational agent, so that only the simulation of the solving process remains as an option. This is made difficult by the fact that for each random decision, every branch should be considered.

We assume that the utility that an agent can get from any one state is bounded to an interval $[0..D]$. We further assume that a manipulator can bound the utility of the states that have not been simulated to the interval $[0..\beta D]$, with $0 \leq \beta \leq 1$. Let α be the total probability mass of the states that the agent has simulated, i.e. with probability α the actual course of the search is part of simulation. Then the manipulator can can guarantee success of the manipulation in the best case if all simulated states have utility D for the manipulated case and 0 for the non-manipulated case, and $\alpha D > (1-\alpha)\beta D \Rightarrow \alpha > \frac{\beta}{1+\beta}$ so that $\alpha > \alpha_0 = \frac{\beta}{1+\beta}$. In well-behaved domains without extreme utility variations, we can assume that β does not have an extremely small value. For example, $\beta = 0.1$ would lead to a threshold of $\alpha_0 = 0.1$.

The local search algorithm contains several possibilities for randomization that make manipulation hard:

- random choice of neighbourhood,
- random choice of excluded agent a_e,
- random choice of one among several equivalent local choices.

To make the effect of randomization easy to analyze, we consider only randomizations whose outcomes can be regarded as independent. This is not the case of the random choice of neighbourhood, as it will often be the case that choosing n_1 and then n_2 will lead to the same states as choosing n_2 and then n_1, thus cancelling the randomization effect. Also, local search has to ensure that all neighbourhoods are considered, placing limits on the amount of randomization that can be allowed.

Fortunately, for the choice of excluded agent as well as the choice among equivalent solutions, it does appear reasonable to assume independence of subsequent random choices. We thus make the simplifying assumption that the state space is a tree, i.e. we do not reach the same state through several paths in the simulation. A good local search algorithm that discovers the global optimum with significant probability must be able to access a large fraction of the possible state space. When the problem is sufficiently large so that this total space is big, the algorithm thus can only rarely revisit the same states.

Consider now a simulation of a sequence of states $s_1, ..., s_i$, where s_1 is the starting state of the search. Because of the random decisions each local choice step are independent, the probability of reaching s_i is equal to:

$$p(s_i) = p(s_i|s_{i-1}) \cdots p(s_2|s_1)$$

Let e be the event that in a local choice step, the most common outcome will be chosen at most $1/m$ of the time, i.e. $p(s_j|s_{j-1}) \leq 1/m$. Let p_m be the probability that this event happens at a local search step, and let k be the number of times e happens in a local search of i steps. Then we can give the following bound k_i as a function of the depth i such that $k < k_i$ with probability at most p_t:

$$p(k < k_i) \quad = \quad \sum_{j=0}^{k_i-1} \left(\begin{array}{c} i \\ j \end{array} \right) p_m^j (1-p_m)^{i-j}$$

$$\leq \quad (1-p_m)^i \sum_{j=0}^{k_i-1} i^j \left(\frac{p_m}{1-p_m}\right)^j$$

$$= \quad (1-p_m)^i \sum_{j=0}^{k_i-1} \left(\frac{ip_m}{1-p_m}\right)^j$$

Assuming that i is sufficiently large so that $\frac{ip_m}{1-p_m} > 2$:

$$p(k < k_i) \leq (1-p_m)^i \left(\frac{ip_m}{1-p_m}\right)^{k_i} \overset{!}{<} p_t$$

so that

$$k_i \leq \frac{\log p_t - i \log(1-p_m)}{\log i + \log p_m - \log(1-p_m)}$$

Thus, k_i is $O(i/logi)$ and we can make it arbitrarily large for a sufficiently large i. For example, for $p_m = 0.5$, $i = 1000$ and $p_t = 10^{-9}$, we have

$$k_{1000} \leq \frac{-30+i}{log_2 i} = \frac{970}{10} = 97$$

In our experiments, we have observed $p_4 \simeq 0.8$ (see below) so that 0.5 is a conservative estimate for this example. Note that the size of the problem must be sufficiently large to allow a sufficiently high value for i.

Then we can bound $p(s_i)$ by replacing all k factors that satisfy this bound by $1/m$, and all others by 1:

$$p(s_i) \leq m^{-k}$$

Assume now that we can guarantee that for any state s_i at depth i in the simulation, $k \geq k_i$, the number of states that would have to be generated to obtain at least a probability mass of α would be:

$$n(i) \geq \alpha m^{k_i}$$

Note that even if the simulation runs beyond depth i, it needs to examine at least this number of states at depth i in order to generate the states corresponding to at least α probability mass at the next time instant.

Assuming that at each step, the probability of satisfying the termination condition and stopping is p_s, the probability of a run that reaches depth at least i is $(1-p_s)^{i-1}$. We assume, again as a simplification to obtain a bound, that for all searches that stop in less than i steps, the outcome can be examined without computational cost. Then, in a local search with timeout T, a manipulator would have to examine at least $t(T)$ states:

$$t(T) = max_{i=0}^{T} \left[max(\alpha - (1-p_s)^{i-1}, 0) m^{k_i}\right]$$

For example, if $m = 4$, $p_s = 0.0001$ and $\alpha = 0.1$ and $T = 1000$, we have $k_i = 97$ and $t(1000) \geq 10^{54}$ which is certainly out of reach of any computational agent today. Thus, a problem with these parameters would be bounded-rational incentive-compatible with probability at least $p_t = (1-10^{-9})$.

The key condition to make the manipulation hard is that the problem is sufficiently large. Only in a large problem can we expect the stopping probability p_s to be sufficiently small and the timeout T to be sufficiently large to allow a large i.

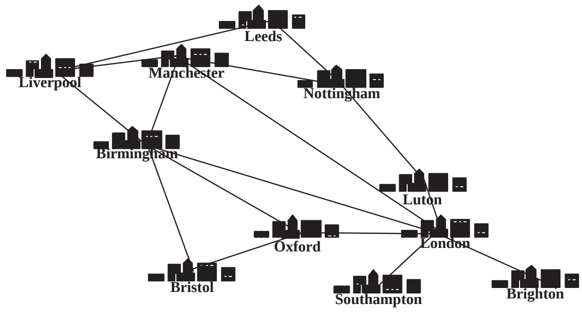

Figure 1: The transportation network used in the experiments

5 Experimental results

We have implemented the mechanism we described for a network resource allocation problem. It consists of allocating tracks in the train network shown in Figure 1 to different operators. To avoid collisions, each arc in the graph can only be allocated to one operator who can then run trains on it. At the same time, there are certain demands for transporting goods. For each demand, 3 feasible routes are computed and it can take any of these routes. This is modelled as an MCOP having a variable for each task whose domain is the agent and route assigned to it. For example, if task 3 $(London, Manchester)$ is assigned to agent a_1 on the path $(London \rightarrow Birmingham \rightarrow Manchester)$, the corresponding variable x_3 is assigned the value $(a_1, London \rightarrow Birmingham \rightarrow Manchester)$.

The network capacity is modelled by binary constraints between any pair of tasks whose routes share at least one arc that rule out assigning it to such overlapping routes but different agents. Each operator has a different and randomly generated profit margin for being assigned a demand/route combination, and declares these through its relations. We randomly generated tasks and routes and simulated the problem starting from a situation where no task is assigned to any agent.

We used the experiments to observe three properties: efficiency, branching probabilities and individual rationality. First, we want to show the efficiency of the randomized protocol with respect to straight hill-climbing to show that it indeed escapes from local minima. Figure 2 we compares the performance of local search with randomization (LOO, shown by the thick line) with strict hill-climbing (LS, dashed line) on the same problem and starting configuration. We see that local search gets stuck in a local optimum and only reaches about half the total utility that the randomized search gets. Thus, the scheme seems to be effective at avoiding local minima.

m	1	2	3	4	5	6	7	≥ 8
$r(m)$	2	21	69	101	194	120	160	333

Table 1. Computational results for p_m

Second, we are interested in the average probability p_m that a Localchoice generates no branch with probability mass larger than $1/m$. We thus took a histogram over 1000 iterations of the number $r(m)$ that this condition is satisfied for m. Table 5 shows the result for $m \leq 10$. From the table, we can estimate for example $p_4 \simeq 0.807$.

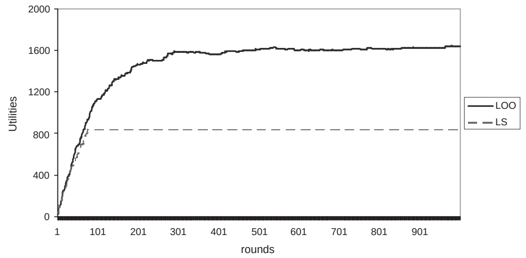

Figure 2: Performance of the local search algorithm with leave-one-out scheme

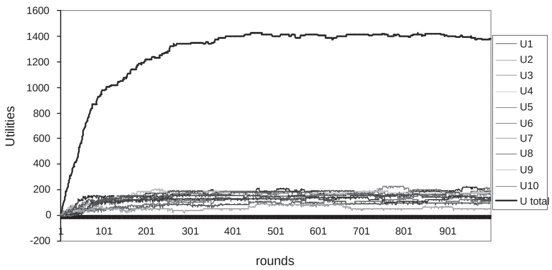

Figure 3: Agents' utilities during search

Third, we are interested in the actual utilities for each agent, and in particular whether we can guarantee individual rationality. Figure 3 shows the utilities of agents during the local search process. In this experiment, we run the local search on random problems with 10 agents and 100 tasks for 1000 rounds. It can be seen that the agents' net utilities are positive and stable when the number of rounds increases.

6 Conclusions

Finding an optimal coordination between multiple self-interested agents is a problem that occurs frequently in practice. Incentive-compatibility is an essential property to ensure meaningful results of such an optimization. Previous work ([Ephrati and Rosenschein, 1991]) has shown the applicability of VCG mechanisms to such problems. However, it requires provably optimal solutions to the NP-hard optimization problem and thus cannot be applied to large problems.

Our work is based on the observation that in real life, the potential for manipulation is limited by uncertainty and risk. This uncertainty makes it difficult for a manipulator to predict the consequences of his manipulation and thus makes attempts at manipulating it uninteresting. Similar uncertainty exists in local search algorithms where randomization is necessary to escape local optima. We have analyzed a scheme for randomization and shown that even in relatively small problems, it creates a large amount of uncertainty so that simulating a sufficient part of the possible outcomes quickly surpasses the computational capacity of any real computational agent. Problems that are too small for this result to apply can likely be addressed by VCG mechanisms with complete optimization.

Note that an important limitation of the MCOP formulation is that agents cannot claim private constraints. Thus, it is impossible to model trading scenarios where an agent has control over certain variables, for example the ownership of a good. This limitation is important because it ensures that the VCG payments never leave a deficit that would have to be covered by the excluded agent.

Note that while no randomized local search algorithm can guarantee individual rationality, we found that it seems to be satisfied with probability. In the future, we would like to analyze the individual rationality properties of local search schemes to obtain probabilistic guarantees similar to those for non-manipulability. Furthermore, we would also like to consider distributed implementation of the methods. As all decisions and payments are computed locally, it is possible implement local search in a distributed algorithm such as the distributed SAT algorithm presented in ([Walsh *et al.*, 2001]).

References

[Clarke, 1971] E.H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[Conitzer and Sandholm, 2003] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of IJCAI-03*, pages 781–788, 2003.

[Ephrati and Rosenschein, 1991] E. Ephrati and J. S. Rosenschein. The clarke tax as a consensus mechanism among automated agents. In *Proceedings of AAAI-91*, pages 173–178, San Jose, California, July 1991.

[Green and Laffont, 1979] J. Green and J.J. Laffont. Incentives in public decision making. *Studies in Public Economics*, 1, 1979.

[Groves, 1973] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–31, 1973.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[Nisan and Ronen, 2000] Noam Nisan and Amir Ronen. Computationally feasible VCG mechanisms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 242–252, 2000.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of AAAI-92*, pages 440–446, 1992.

[Selman *et al.*, 1994] B. Selman, H. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of AAAI-94*, pages 337–343, 1994.

[Vickrey, 1961] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16(2):8–37, 1961.

[Walsh *et al.*, 2001] W. Walsh, M. Yokoo, K. Hirayama, and M. Wellman. On market-inspired approaches to propositional satisfiability. In *Proceedings of IJCAI-01*, pages 1152–1158, 2001.