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. It involves constraint optimization problems that often can only be solved by local search algorithms.

Using local search poses problems of incentive-compatibility and individual rationality. We thus define a weaker notion of bounded-rational incentive-compatibility where manipulation is made impossible with high probability through computational complexity. We observe that in real life, manipulation of complex situations is often impossible because the effect of the manipulation cannot be predicted with sufficient accuracy. We show how randomization schemes in local search can make predicting its outcome hard and thus form a bounded-rational incentive-compatible coordination algorithm.

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 that can be formulated as constraint optimization problems. We thus extend the standard definition of constraint optimization 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.

An example of an 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. Constraints would enforce for example that successive links and times are assigned to the same agent.

Agents serve customers' transportation demands with different efficiency by using different combinations of links. Thus, each agent has utilities for being able to use certain combinations of links, and reports these as relations. Agents want to find a combined assignment that maximizes the sum of their utilities. 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: individual rationality and incentive-compatibility. 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.

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. It is often achieved through tax or auction mechanisms such as the VCG mechanism ([Vickrey, 1961; Clarke, 1971; Groves, 1973]).

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 in MCOP 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 rendering 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.

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
```

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 fixes the otherwise open 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 $\underline{v^{old}}$ but assign to x_i different values in the domain of x_i that are consistent with the rest of $\underline{v_{old}}$ and the constraints in C.

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 ([Kirkpatrick *et al.*, 1983; Selman *et al.*, 1994]). This means that occasionally *LocalChoice* chooses a value that may even decrease agent utility.

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 a provably optimal solution. 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:

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.*

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.*

Earlier work, such as ([Conitzer and Sandholm, 2003]), has proposed using NP-hardness as a protection against manipulation. However, our definition goes further as it requires *in almost all cases*, manipulation requires an amount of computation that is beyond the means of a bounded-rational agent. Any real computational agent is bounded rational for a sufficiently high C.

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.

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.

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 a 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. We call such agents *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. More details on the mechanism can be found in [Faltings, 2004].

In certain cases the sum of the taxes could be less than the utility loss of the excluded 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 every 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. No agent is systematically disadvantaged by the randomization, and so in expectation the scheme is individually rational for all agents.

This is confirmed in our simulations, where individual rationality was always satisfied for all agents in all runs.

4 Sequences of local choices

A local search algorithm is in general incomplete and not guaranteed to find a particular optimal solution. Thus, as pointed out by [Nisan and Ronen, 2000], non-truthful declarations can drive the local search algorithm to a solution that gives a manipulating agent (MA) a better utility than the truthful declaration. However, effectively using such manipulation requires that the MA is capable of correctly predicting the effect of a non-truthful utility declaration on the outcome, and compare it against the utility loss it incurs by carrying out the manipulation in one or several local choice steps. We now show that in a randomized local search algorithm and a sufficiently large problem, with high probability (arbitrarily close to 1), such prediction would require an amount of computation that is beyond the capabilities of a bounded-rational agent.

To obtain a worst-case result, we assume that a manipulating agent (MA) has complete and accurate knowledge of the relations declared by all other agents. Furthermore, we assume that is has access to an oracle that provides it with the most promising manipulation.

The remaining task of the MA is then to show that the manipulation actually produces a better utility than truthful behavior. As the local search algorithm is randomized, the MA can only predict an *expected* utility, obtained by considering the probability of certain states and the utilities that the agent would obtain in each of them. The key idea of our argument is that with high probability, the number of states that need to be considered in this calculation will grow exponentially with the size of the problem. Thus, for a certain problem size it will exceed the computational capacity of a bounded rational agent.

To show this result, we first argue that the MA has to examine a significant fraction of the probability mass of the states to ensure success of the manipulation. This fraction depends on two factors:

- the utility distribution of the problem: if only few states give a significant utility, or if there are strong symmetries so that the state space can be factored, it could be sufficient to sample only a small subset of the states, and
- the desired confidence of the prediction: since a manipulation will mean a certain utility loss to the agent in the search step where it is applied, the manipulation needs to succeed with a certain minimal probability in order to give an increase in expected utility. Depending on the utility distribution, this translates to a certain fraction of the state probability mass that will need to be examined.

Note that both parameters are independent of the size of the search space. Thus, we can assume that the MA will have to examine a minimal number of states that corresponds to some fraction α of the probability mass of the entire search space.

Next, we show that with high probability, this probability mass is distributed over a number of states that grows exponentially with problem size.

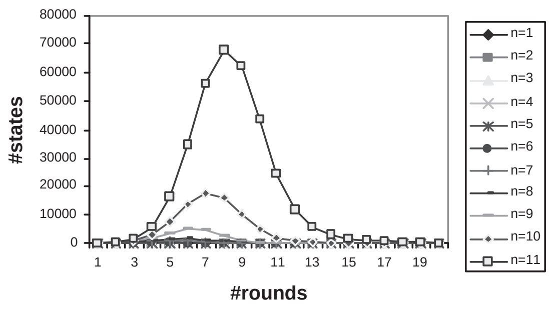

Figure 1: *New states discovered in successive cycles of a simulation of local search, for several problem sizes.*

The local search algorithm contains several possibilities for randomization that can make its outcome hard to predict:

- random choice of neighbourhood,
- random choice of excluded agent a_e,
- random choice of 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. Furthermore, a manipulating agent could make declarations to create symmetries that makes the randomization ineffective.

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. Furthermore, since the manipulating agent's relations may be excluded, it cannot render the randomization invalid through its own declarations.

To find the global optimum, a local search algorithm has to be able to reach the entire search space. However, eventually it will come close to the global optimum and then remain within a much smaller subspace of nearly optimal states. While it is possible to give a theoretical analysis that shows that with arbitrarily high probability, the probability of reaching any given state is bounded by an exponentially decreasing value, such an analysis requires many independence assumptions that may not hold in practice. Here, we present the following experimental measurements, obtained on the experimental scenario given in the next section.

Figure 1 shows the number of new states discovered in successive cycles of a simulated randomized local search. It initially grows exponentially, but eventually search stabilizes on certain optimal outcomes and thus fails to discover new states. Importantly, however, the total number of states discovered, shown in Figure 2, still grows exponentially with problem size: in this example, it muliplies with a factor of about 3 whenever the size increases by 1. Thus, the total

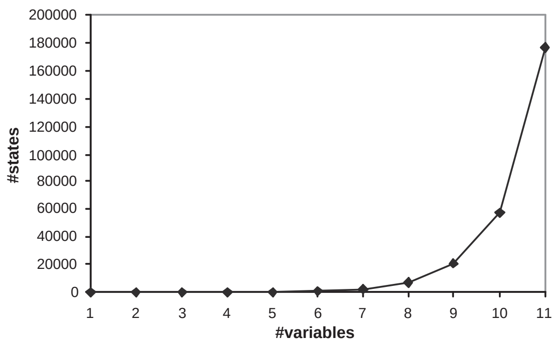

Figure 2: *Growth of the total number of states involved in a local search simulation as a function of the problem size.*

number of states has exponential growth with problem size, even though it does not reach the total state space because of the convergence of the algorithm.

Another aspect that needs to be shown is that the MA cannot limit its consideration to only certain states in this space, i.e. that the probability mass is distributed over a large subset of the states. We show this by considering the probabilities of the resulting states at each randomized step. Let p_m denote the probability that at a random branch, each of the branches is taken with probability at most $1/m$. We have measured p_m experimentally (see later section) and have obtained for example for $p_4 \simeq 0.908$, showing that the search process has a significant branching factor. Thus, with high probability the probability mass is distributed among a large number of states.

A theoretical analysis with independence assumptions on this basis gives for example that in a local search with 1000 variables, a stopping probability of 0.0001 (expected number of cycles = 10'000) and $\alpha = 0.1$, the probability that an agent would have to examine less than 10^{54} states is bounded by $p_t < 10^{-9}$. This is certainly well beyond the capability of any computational agent today.

While we have so far only analyzed relatively simple models, it seems clear that in general the probability mass is very likely to be spread among a large set of states as the size of the problem increases, and thus the method will be bounded-rational incentive-compatible with the parameter being the problem size.

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 3 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

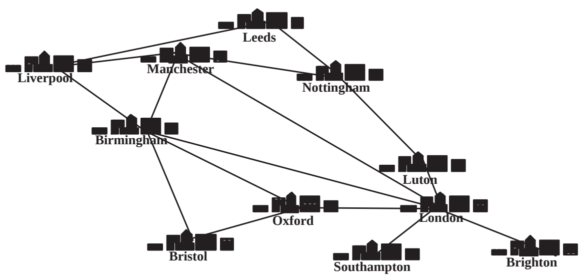

Figure 3: *The transportation network used in the experiments*

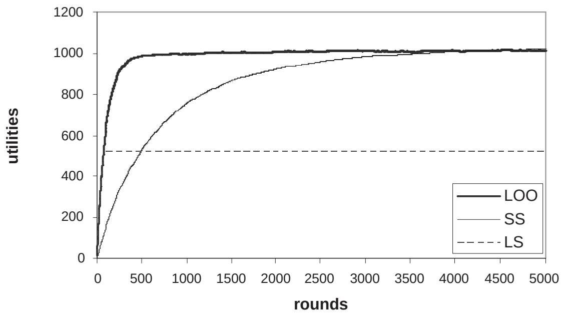

Figure 4: *Average utility gain of different local search algorithms as a function of the number of steps.*

$(London \rightarrow Birmingham \rightarrow Manchester)$, 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 4 compares the performance of local search with randomization (LOO, shown by the thick line) with stochastic search (SS, thin line) and strict hill-climbing (LS, dashed line) on 100 randomly generated problem instances. 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.

Second, we are interested in the average probability p_m

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

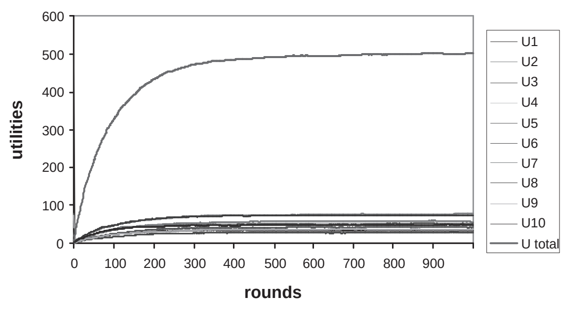

Figure 5: *Agents' utilities during search*

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 1 shows the result for $m \leq 10$. From the table, we can estimate for example $p_4 \simeq 0.908$.

Third, we are interested in the actual utilities for each agent, and in particular whether we can guarantee individual rationality. Figure 5 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 in sufficiently large 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 the techniques in this paper all rely on local choices and payments and are thus very suitable for asynchronous, distributed implementation.

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.

While no randomized local search algorithm can guarantee individual rationality, we found that it seems to be satisfied with high probability. It would be interesting to analyze the individual rationality properties of local search schemes to obtain probabilistic guarantees similar to those for non-manipulability.

The most important weakness of the current scheme is that the parameter that needs to be varied to guaranteed bounded-rational incentive-compatibility is the size of the problem. It would be much better if we had a mechanism that could guarantee high manipulation complexity even for small problems through suitable randomization of this choice, similarly to certain cryptographic hash functions.

Acknowledgements

This work has been funded by the Swiss National Science Foundation under contract No. 200020-103421/1.

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. *Proceedings of IJCAI-03*, pp. 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*, pp. 173–178, San Jose, California, July 1991.

[Faltings, 2004] Boi Faltings. A Budget-balanced, Incentive-compatible Scheme for Social Choice, *Agent-Mediated Electronic Commerce VI*, Springer LNAI 3435, 2004.

[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 AMEC-2000*, pp. 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*, pp. 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*, pp. 337–343, 1994.

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