Contract Type Sequencing for Reallocative Negotiation

Martin Andersson and **Tuomas Sandholm**[*]
{mra, sandholm}@cs.wustl.edu
Department of Computer Science
Washington University
St. Louis, MO 63130-4899

Abstract

*The capability to reallocate items—*e.g. *tasks, securities, bandwidth slices, Mega Watt hours of electricity, and collectibles—is a key feature in automated negotiation. Especially when agents have preferences over combinations of items, this is highly nontrivial. Marginal cost based reallocation leads to an anytime algorithm where every agent's utility increases monotonically over time. Different contract types head toward different locally optimal task allocations, and contracts from a recently introduced comprehensive contract type, OCSM-contracts, head toward the global optimum. Reaching it can take impractically long, so it is important to trade off solution quality against negotiation time. To construct negotiation protocols that lead to the best achievable allocations in a bounded amount of time, we compared sequences of four contract types: original, cluster, swap, and multiagent contracts. The experiments show that it is profitable to use multiple contract types in the sequence: significantly better solutions are reached, and faster, than if only one contract type is used. However, the best sequences only include original and cluster contracts. Swap and multiagent contracts lead to bad local optima quickly. Interestingly, the number of contracts using any given contract type does not always decrease over time: contracts play the role of enabling further contracts.*

1 Introduction

The importance of automated negotiation systems is increasing as a consequence of the development of technology as well as increased application pull, *e.g.*, electronic commerce [5], electricity markets [17], and transportation exchanges [9]. A central part of such systems is the ability to (re)allocate tasks (or analogously, other types of items, *e.g.* securities, bandwidth slices, Mega Watt hours of electricity, or collectibles) among the agents. Generally, the tasks have a dependency upon each other, as well as upon the agents. That is, some of the tasks are synergistic and preferably handled by the same agent, whereas others interact negatively and are better handled by different agents. The agents can also have different resources that lead to different costs for handling the various tasks.[1] Furthermore, an agent may not be capable of handling all combinations of the tasks.

Definition 1.1 *Our* task allocation problem *[12] is defined by a set of tasks T, a set of agents A, a cost function $c_i : 2^T \rightarrow \Re \cup \{\infty\}$ (which states the cost that agent i incurs by handling a particular subset of tasks), and the initial allocation of tasks among agents $\langle T_1^{init}, ..., T_{|A|}^{init} \rangle$, where $\bigcup_{i \in A} T_i^{init} = T$, and $T_i^{init} \cap T_j^{init} = \emptyset$ for all $i \neq j$.*[2]

In our example problem the agents incur different costs for handling the tasks, however all the agents have the capability to handle any task. The agents in the example problem are *self-interested* and *myopically individually rational*. This means that an agent agrees to a contract if and only if the contract increases the agent's immediate payoff which consists of the side payments received from other agents (for handling their tasks) minus the cost c_i of handling tasks. Recently, new types of contracts were introduced [9, 11, 12] to be used in contract nets [16]. In earlier research we have applied each of these contracts to a task allocation problem and found the local optima the different protocols reach [1, 2]. In order to improve the achievable social welfare, these contract types can be sequenced

[*]This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IRI-9610122, and Grant IIS-9800994. An early version of this paper appeared at the Agent-Mediated Electronic Trading (AMET) Workshop, Minneapolis, MN, May, 1998.

[1]The dependencies between tasks in human negotiations are discussed in [7]. The concepts of linkage and log-rolling are also presented, which are similar to swapping tasks and clustering tasks.

[2]This definition generalizes the "Task Oriented Domain" presented by [8] by allowing different cost functions among agents, and the possibility of some agent not being able to handle some task sets (corresponds to infinite costs).

in a number of different ways. In this paper the entire negotiation is divided into five *intervals*, and in each interval only one of the contract types is used. In this manner performance profiles for all possible sequences of contract types are created.

The rest of the paper is organized as follows. The example problem is defined in Section 2. A summary of the contract types and how they are sequenced is presented in Section 3, followed by the experimental setup in Section 4. Section 5 covers the results. Section 6 presents conclusions and potential future research directions.

2 Example Problem: multiagent TSP

The multiagent TSP is defined as follows [1, 2]: Several salesmen will visit several cities in a world that consists of a unit square, Figure 1. Each city must

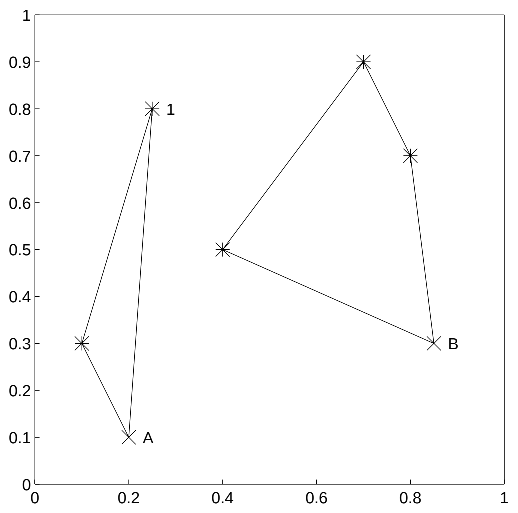


Figure 1. *An example problem instance of a multiagent TSP consisting of five cities (*) and two salesmen (X). If salesman A contracts out city 1 to salesman B, the social welfare will increase due to less travel, i.e., lower costs.*

be visited by exactly one salesman, and each salesman must return to his starting location after visiting the cities assigned to him. A salesman can visit the cities assigned to him in any order.

Solution quality is measured in social welfare, *i.e.*, the negative of the sum of the costs (distances traveled) of the salesmen. Initially the location of the cities and starting points of the salesmen are randomly chosen as is each salesman's initial assignment of cities to visit. After this initial assignment, the salesmen can exchange cities with each other. The payoff of a salesman consists of the side payments received from other salesmen (as compensation for handling their tasks) minus the side payments paid to other salesmen (to compensate them for handling some of this agent's tasks) minus the cost of travel. Each salesman is individually rational and myopic, which means that he agrees to a contract if and only if the contract increases his immediate payoff. Therefore, contracts are made only when they improve the utility of all contract parties. It follows that social welfare increases monotonically, *i.e.*, total travel distance decreases.

3 Contract Types and Their Sequencing

The contract most commonly used in multiagent contracting systems only allows for one task to move from one agent to another at a time [4, 15, 16]. We will refer to this type as an *original (O) contract*. For more efficient contracting, new types of contracts were recently introduced [9, 11, 12]: *cluster (C)*, *swap (S)*, and *multiagent (M) contracts*, as well as all the above, including the original contracts, combined (*OCSM-contracts*). These new contract types allow more than one task to be transferred between the agents participating in the contract. C-contracts transfer at least two tasks from one agent to another, while the S-contracts let two agents swap tasks with each other (one task is transferred from each agent to the other). In the M-contracts exactly three tasks are being transferred between exactly three agents. These limitations were introduced so that the length of the intervals would be of the same order.[3] Each agent gives away only one task, but can receive more than one task.

In this paper different sequences of the elementary contract types, *i.e.*, original, cluster, swap, and multiagent contracts are studied. The total time of contracting is divided into five intervals. In each interval, one contract type is applied. All possible sequences of contract types were investigated, that is, all $4 \times 4 \times 4 \times 4 \times 4 = 1024$ sequences of contract types were applied to each TSP instance.

In each interval, all contracts that are possible to construct with the contract type in question were tried.[4] Because of this, the length of the intervals varied for different contract types and task allocations.

[3] If all possible M-contracts including more than three agents and three tasks were to be checked in an interval, that interval would be much longer than the intervals for the other contract types [2]. A simpler version of multiagent contracts where bids were grouped into cascades were studied by [14].

[4] The possible contracts depend on the current allocation of

This does not mean that a local optimum is necessarily reached in each interval because one contract can enable another contract. If this other contract was tried earlier in the interval, it will not be retried in the same interval. A contract was only performed if it was individually rational to all the agents participating in it.

3.1 Sequencing of Contracts within an Interval

When searching for a good task allocation, the contract types used in the negotiation are applied repeatedly for all possible combinations of agents and tasks that suit the contract type. Although the algorithm heads toward a local optimum, it does not always reach one after trying just five contract types sequentially (and all possible contracts within each type). This is because some contracts enable others that may not have been profitable initially. The algorithm knows that a local optimum with respect to a particular contract type has been reached when no contracts of the given type have been made during one interval, that is, all possible contracts of the type have been tried but none have been performed.

The next subsections discuss the order of trying different contracts within each contract type, *i.e.*, within each interval. The agents are numbered from 1 to $|A|$, and each agent's tasks from 1 to $|T_i|$.

3.1.1 Sequencing of Original Contracts

An O-contract allows one agent to move one task to one other agent. The former agent pays the latter for accepting the contract at least as much as it costs the latter agent to handle the task, and at most as much as it costs the former agent to handle it. In our experiments, O-contracts were sequenced as follows. First, agent 1's tasks are attempted to be moved, one at a time, to agent 2. If any contract (move of a task) is profitable, it is performed and the next contract is tried. After having tried to move all tasks one at a time from agent 1 to 2, agent 1 tries to move its tasks to agent 3. This continues until agent 1 has attempted to move all its tasks to all the other agents. Then the procedure continues with agent 2, which tries to move its tasks to agent 1, followed by all the other agents in increasing order. When agent $|A|$ has attempted to move all its tasks to all the other agents this interval is finished and the negotiation process continues with the next contract type.

tasks among the agents.

3.1.2 Sequencing of Cluster Contracts

In a C-contract one agent moves at least two tasks to one other agent, and a side payment is used as with O-contracts. C-contracts were sequenced as follows. We start by trying out all combinations of two tasks followed by all combinations of three tasks, and so on. The order in which the tasks are tried to be moved is: (1,2), (1,3), ..., (1, $|T_1|$), (2,3), (2,4), ..., ($|T_1|$-1, $|T_1|$), (1,2,3), (1,2,4), ... If any contract is profitable, it is performed and the next contract is tried. After having tried to move all tasks (one at a time) from agent 1 to 2, agent 1 tries to move its tasks to agent 3. This continues until agent 1 has attempted to move all its tasks to all the other agents. Then the procedure continues with agent 2, which tries to move its tasks to agent 1, followed by all the other agents in increasing order. When agent $|A|$ has attempted to move its tasks to all other agents, this interval is finished and the negotiation process continues with the next contract type.

3.1.3 Sequencing of Swap Contracts

In an S-contract, one agent transfers one task to another agent and it also receives one task from that agent. If the S-contract is acceptable, *i.e.*, social welfare improving, a side payment can be used so that each one of the two agents is better off than before the contract. S-contracts were sequenced as follows. One at a time, agent 1 tries to move its tasks to agent 2, and in exchange agent 2 tries to move one task to agent 1. For every task agent 1 tries to move, agent 2 tries to move all its tasks to agent 1 one at a time before agent 1 continues with its next task. If any contract is profitable, it is performed and the next contract is tried. When all contracts that include agent 1 and agent 2 have been attempted, all possible contracts including agent 1 and agent 3 are tried according to the procedure above. When agent 1 has attempted all contracts with all the other agents, agent 2 tries all contracts, according to the procedure above, with agent 1 followed by the other agents in increasing order. When agent $|A|$ has attempted to exchange tasks with all other agents, this interval is finished and the negotiation process continues with the next contract type.

3.1.4 Sequencing of Multiagent Contracts

In an M-contract three tasks are moved between three agents, and each agent can only move one task to one other agent. If the M-contract is acceptable, *i.e.*, social welfare increasing, side payments can be used so that each contract party is better off than before the contract. M-contracts were sequenced as follows. First, all combinations which include agent 1 are tried in the following order: (1,2,3), (1,2,4), ..., (1,2,$|A|$), (1,3,2), (1,3,4), ..., (1,$|A|-1$,$|A|$). Then, all combinations of agents including agent 2 are tried: (2,1,3), (2,1,4), ..., (2,1,$|A|$), (2,3,1), (2,3,4), ..., (2,$|A|-1$,$|A|$), *etc.*, until all combinations have been tried successively for all agents. At that time the interval ends.

For each combination of agents, different task transfers are tried. The order in which the tasks are tried are (from the first agent to agent no., from the second agent to agent no., from the third agent to agent no.): (2,1,1), (2,1,2), (2,3,1), (2,3,2), (3,1,1), (3,1,2), (3,3,1), (3,3,2). If one of the agents does not have the task needed, that combination is skipped.

4 Experimental Setup

In principle our contracting system implementation can be used to solve task reallocation problems with any number of agents and tasks. The simulations of this paper focus on the multiagent TSP domain with 8 agents and 8 tasks per problem instance. 1000 instances were generated. The initial locations of the cities and the start locations of the salesmen in each instance were randomly chosen in the unit square. Each instance was solved with each of the *protocols*, *i.e.*, sequences of contract types. In addition, an exhaustive enumeration of task allocations was conducted in order to find the globally optimal allocation.

In the experiments, each problem instance was tackled in two phases: first all possible TSPs (*i.e.*, TSPs with any of the salesmen getting any combination of cities to visit[5]) were solved to determine the cost functions, c_i, and then simulations using different contracting protocols were conducted to solve the task allocation problem. The IDA* search algorithm [6] was used to solve the TSPs. To ensure that the optimal solution was reached, an admissible $\hat{h}$-function was used. It was constructed by under-estimating the cost function of the remaining nodes by the minimum spanning tree of those nodes, *i.e.*, nodes not yet on that path of the search tree, the last city of that path of the search tree, and the finish (=start) location of the salesman [3].

4.1 Evaluation Criteria

To compare the solution quality obtained by different protocols, the *ratio bound* was used. Let x_j^l denote the social welfare of the task allocation achieved by protocol l on problem instance j, $j \in \{1, \ldots, 1000\}$. Let x_j^G denote the social welfare of the global optimum (or equivalently OCSM-contracts). The ratio bound, r_j^l, is the optimal welfare divided by the welfare obtained by a given protocol: $r_j^l = \frac{x_j^G}{x_j^l}$. The average ratio bound is $\overline{r}^l = \frac{1}{1000}\sum_{j=1}^{1000} r_j^l$. CPU-time usage, the number of contracts tried, and the number of contracts performed were also calculated as an average over the 1000 problem instances.

[5]Salesman 1 visits city 1, salesman 1 visits city 2,..., salesman 1 visits cities 1 and 2,..., salesman 2 visits city 1,..., salesman 8 visits all eight cities.

5 Results

To study which sequences of contract types performed best, all 1024 sequences were ordered according to ascending average ratio bounds. The 15 best protocols are shown in Table 1. Different protocols led to very different solution qualities. For example, the best protocol found task allocations that were 3.1% from optimum on average while the worst protocol led to solutions 89.3% off optimum.

Rank	Sequence	Average ratio bound
1	OCOCO	1.03113
2	OOCCO	1.03268
3	OCCOC	1.03276
4	OOCOC	1.03279
5	OCOOC	1.03413
6	SOCOC	1.03488
7	SOCCO	1.03536
8	COCOC	1.03755
9	OCOCC	1.03857
10	MCOCO	1.03945
11	OCCCO	1.03954
12	MOCCO	1.03988
13	MOCOC	1.04001
14	MCCOC	1.04304
15	COCCO	1.04407

Table 1. *The 15 best sequences of contract types.*

The best sequences are close to each other (Table 1 and Figure 2): the best protocol is 3.1% from optimum, and the 12 best protocols are all between 3% and 4% from optimum. The best 24 protocols are within 5% from optimum, and the best 181 are within 10% from optimum.

If the same contract type is applied in all five intervals, the solutions are significantly worse on average than if different contract types are used, Table 2. The two worst sequences both consisted of only one contract type: S-, and M-contracts, respectively. The sequence with only O-contracts performed better, but

Rank	Sequence	Average ratio bound
1	OCOCO	1.03113
2	OOCCO	1.03268
3	OCCOC	1.03276
4	OOCOC	1.03279
375	*C-local*	*1.13557*
565	*O-local*	*1.2025*
579	OOOOO	1.21298
696	CCCCC	1.23515
1021	CSSSS	1.61181
1022	CMMMM	1.65965
1023	MMMMM	1.76634
1024	SSSSS	1.89321

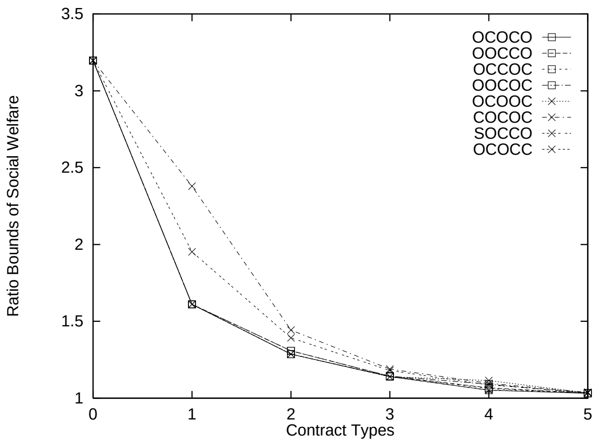
Table 2. *The four best and the four worst sequences, the cases where only one contract type was used in all intervals, and the results when O- and C-contracts are allowed to reach a local optimum. The local optimum for S- and M-contracts was reached in the bottom two cases.*

is still in the bottom half. The same is true for C-contracts (which are still, after the five intervals, doing worse than O-contracts). O- and C-contracts did not reach their respective local optima in the five intervals, so this result says that it is better to switch to another contract type even before reaching the local optimum using one contract type.

When O- and C-contracts were allowed to continue their computation until a local optimum was reached, C-contracts were better than O-contracts—with average ratio bounds 1.14 and 1.20, respectively. This can also be seen if the curves for the two protocols are extrapolated in Figure 3. Even if the local optimum is computed for C-contracts, a sequence of mixed contract types achieves better allocations, and in a shorter amount of time. Figure 3 shows that M-contracts get stuck in a local optimum in the first interval, and S-contracts in the second.

Figure 4 shows the number of contracts tried and performed. For C-contracts, more contracts are tried and performed in the second interval than in the first, *i.e.*, the curve is convex. This is because after the first interval of C-contracts the resulting task allocation makes more C-contracts possible to perform, because C-contracts concentrate the tasks among a smaller number of agents. In the subsequent intervals the number of individually rational C-contracts decreases, resulting in fewer contracts performed, *i.e.*, the curve is concave.

Figure 5 shows how social welfare increases as more contracts are tried. As is desirable in an anytime algorithm, most of the savings are achieved quickly, and diminishing returns to computation follow. The length of



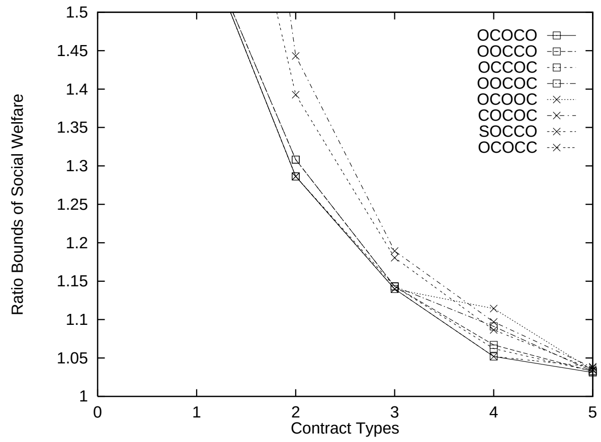


Figure 2. *Performance profiles: Decrease of the average ratio bound. The bottom graph is a zoom of the top graph.*

the intervals differ both within and between the curves in Figure 5 since the number of contracts tried within each interval might differ. This is because of the varying task allocations among the agents and the different number of contracts possible to form with the different contract types.

If two contract types are mixed in a sequence, the social welfare is greater than with either contract type alone in the sequence. Also, more mixing—*i.e.*, not having the same contract type applied several times in a row—led to higher social welfare. In those cases where the social welfare of a mixed sequence was worse than that of one of its contract types applied alone, the mixed sequence consisted mostly of one contract type. In no case were a mixed sequence between two contract types worse than both the sequences where the two contract types were applied alone.

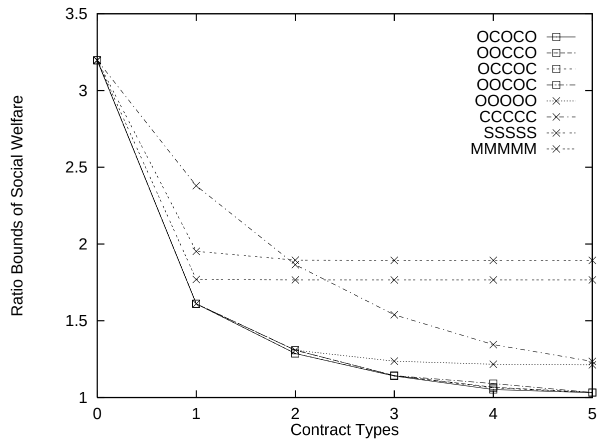


Figure 3. *Performance profiles for the four best sequences and ones in which only one contract type is used.*

6 Conclusions

The capability of reallocating tasks is a key feature in automated negotiation systems. The use of marginal cost based task reallocation leads to an anytime algorithm where every agent's utility increases monotonically over time. Different contract types head toward different locally optimal task allocations, and OCSM-contracts head toward the global optimum. Reaching the global optimum can take an impractically long time, so in real world applications it is important to be able to trade off solution quality against negotiation time.

In order to construct negotiation protocols that lead to the best achievable task allocations in a bounded amount of time, we compared sequences of four contract types: original, cluster, swap, and multiagent contracts. The results regarding solution quality achieved by the different sequences provide guidelines for system builders regarding what contract types to use and how to sequence them when negotiation time is limited.

It is clearly profitable to mix different contract types in the sequence: significantly better solutions are reached, and in a shorter amount of time than if only one contract type is used. Interleaving contract types more often was better than clustering the same contract types together in the sequence. The best sequences alternated between intervals of original and cluster contracts. Swap and multiagent contracts led to bad local optima quickly.

Interestingly, the number of contracts performed using a given contract type does not always decrease over time. When a contract type is applied, the number

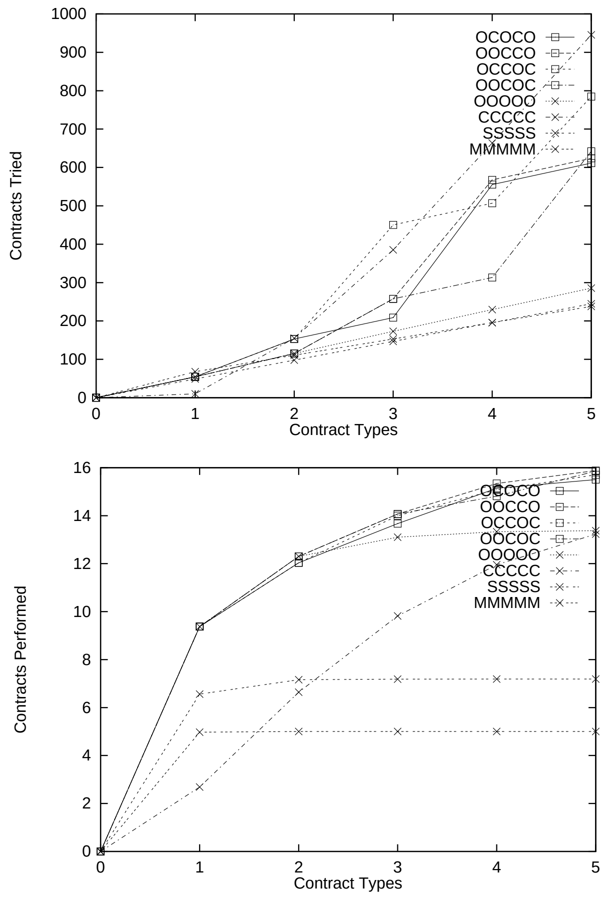


Figure 4. *Contracts tried (top) and contracts performed (bottom) for the four best sequences and ones in which only one contract type was used.*

of contracts performed (and tried) can increase after a while because the task allocation arising from contracts of that type can make more contracts of the same type possible.

The results apply directly to the allocation of other items besides tasks once one interprets the cost functions c_i as value functions that the agents want to maximize instead of minimize. Such applications include most many-to-many (after)markets where agents have preferences over bundles instead of simply over individual items (in the latter case original contracts can be trivially used to reach the optimal allocation). Many significant markets are of this type, *e.g.* markets for securities, bandwidth, Mega Watt hours of electricity, and collectibles.

Future research includes formally comparing the results of sequences that mix contract types to sequences that do not. We would also like to study changing the contract type for each contract as opposed to keeping

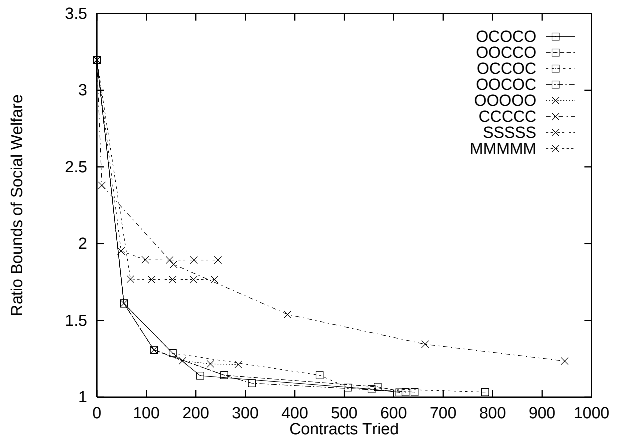


Figure 5. *Solution quality as a function of contracts tried.*

the type fixed within each interval. Additionally, we would like to apply different numbers of contracts of one contract type before changing the type, or sequencing the contract types in a way where a local optimum is found with one type before switching to another.

Yet another interesting area for future work is combining the different contract types, thus forming atomic contracts having characteristics of more than one of the O-, C-, S-, and M-contracts, but not all of them (unlike OCSM-contracts). These composite contract types would not guarantee that individually rational agents will reach the globally optimal task allocation, but they would lead to a local optimum faster then OCSM-contracts, and most likely to higher average social welfare than O-, C-, S-, or M-contracts.

Also, we would like to study agents that may lie about their marginal costs of handling the task sets under negotiation [8, 10]. Finally, we plan to study the use of backtracking in contracting, which is nontrivial to implement among self-interested agents [13].

References

[1] M. R. Andersson and T. W. Sandholm. Leveled commitment contracting among myopic individually rational agents. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS)*, pages 26–33, Paris, France, July 1998.

[2] M. R. Andersson and T. W. Sandholm. Time-quality tradeoffs in reallocative negotiation with combinatorial contract types. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 3–10, Orlando, FL, 1999.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

[4] C. Gu and T. Ishida. Analyzing the social behavior of contract net protocol. In W. V. de Velde and J. W. Perram, editors, *Agents Breaking Away; MAAMAW'96, Lecture Notes in Artificial Intelligence 1038, Springer-Verlag*, pages 116–127, 1996.

[5] R. Kalakota and A. B. Whinston. *Frontiers of Electronic Commerce.* Addison-Wesley Publishing Company, Inc, 1996.

[6] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1):97–109, 1985.

[7] H. Raiffa. *The Art and Science of Negotiation.* Harvard Univ. Press, Cambridge, Mass., 1982.

[8] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, 1994.

[9] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 256–262, Washington, D.C., July 1993.

[10] T. W. Sandholm. Limitations of the Vickrey auction in computational multiagent systems. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, pages 299–306, Keihanna Plaza, Kyoto, Japan, Dec. 1996.

[11] T. W. Sandholm. *Negotiation among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, Amherst, 1996. Available at http:// www.cs.wustl.edu/ ~sandholm/ dissertation.ps.

[12] T. W. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium Series: Satisficing Models*, pages 68–75, Stanford University, CA, Mar. 1998.

[13] T. W. Sandholm and V. R. Lesser. Advantages of a leveled commitment contracting protocol. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 126–133, Portland, OR, Aug. 1996.

[14] A. Sathi and M. Fox. Constraint-directed negotiation of resource reallocations. In M. N. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence*, volume 2 of *Research Notes in Artificial Intelligence*, chapter 8, pages 163–193. Pitman, 1989.

[15] S. Sen. *Tradeoffs in Contract-Based Distributed Scheduling.* PhD thesis, Univ. of Michigan, 1993.

[16] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, Dec. 1980.

[17] F. Ygge and J. M. Akkermans. Power load management as a computational market. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS)*, pages 393–400, Keihanna Plaza, Kyoto, Japan, Dec. 1996.