Time-Quality Tradeoffs in Reallocative Negotiation with Combinatorial Contract Types

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 payoff increases monotonically over time. Different contract types head toward different locally optimal allocations of items, and 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 good allocations quickly, we evaluated original (O), cluster (C), swap (S), and multiagent (M) contracts experimentally. O-contracts led to the highest social welfare when the ratio of agents to tasks was large, and C-contract were best when that ratio was small. O-contracts led to the largest number of contracts made. M-contracts were slower per contract, and required a significantly larger number of contracts to be tried to verify that a local optimum had been reached. S-contracts were not competitive because they restrict the search space by keeping the number of items per agent invariant. O-contracts spread the items across agents while C-contracts and M-contracts concentrated them on a few agents.

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 (Kalakota & Whinston 1996), electricity markets (Sandholm & Ygge 1997), and transportation exchanges (Sandholm 1993). 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. We analyze task allocation (or, analogously, the allocation of other types of items) under the following model which captures these considerations.

Definition. 1 *Our* task allocation problem *is 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 the case where agent i cannot handle a specific set of tasks, T_i, the cost function $c_i(T_i) = \infty$. In our example problem domain the agents incur different costs for handling tasks but each agent has the capability to handle any tasks.

In task allocation, the agents try to minimize the cost functions. The same would hold for reallocation of any types of undesirable items. To model settings where the items are desirable, such as bandwidth slices, the cost functions can be interpreted as value functions which the agents attempt to maximize.

Combinatorial auctions, i.e. auctions where the bidders can bid on combinations of items, have recently received a lot of interest because they often lead to better allocations than traditional auctions in settings where the items' valuations are not additive (Rothkopf, Pekeč, & Harstad 1998; Rassenti, Smith, & Bulfin 1982). Combinatorial auctions are a special case of our setting, where one agent has all the tasks initially, and allocates them to the other agents. Unlike combinatorial auctions which are one-to-many, our setting allows many-to-many markets. This is needed in settings where there are inherently many buyers and many sellers. Our approach can also be used as a reallocative market to

*Copyright ©, American Association of Artificial Intelligence (www.aaai.org). All rights reserved. This material is based upon work supported by the National Science Foundation under CAREER Award IRI-9703122, Grant IRI-9610122, and Grant IIS-9800994.

[1]Dependencies between tasks in human negotiations are discussed e.g. in (Raiffa 1982). 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 (Rosenschein & Zlotkin 1994). Particularly asymmetric cost functions among agents are allowed, as well as the possibility that some agents may be unable to handle some tasks. In that case the cost of handling the task will be infinite.

correct inefficient allocations after some one-to-many auction has been held. In other words, some of the bidders may not have received the (combinations of) items that they want, and may have received (combinations of) items that they do not want. The bidders can then participate in our reallocative many-to-many aftermarket to improve the overall allocation of items.

The agents can change the task allocation by reallocating tasks among themselves by contracting. The agents can also recontract out tasks that they contracted in earlier. We study agents that 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. An agent's payoff consists of the payments received from others for handling their tasks minus the current value of the cost function, c_i, minus payments sent to others for them to handle some of the former agent's tasks.

This paper experimentally studies task reallocation among such agents using combinatorial contract types that were recently introduced (Sandholm 1993; 1996; 1998) to be used in contract nets (Smith 1980). The next section presents the application domain of the experiments. The different contract types and their use is described in the following section. Then evaluation criteria are discussed and the results are presented. The final section concludes the paper.

Example problem: Multiagent TSP

We study contracting in a particular task allocation problem, the *multiagent Traveling Salesman Problem (TSP).* This domain is used as an example because it is structurally simple—providing repeatability and easy presentability—yet it captures the essence of the difficulties in reallocative negotiation. The TSP is NP-complete and the space of task allocations contains many local optima when using hill-climbing-based contracting algorithms (Sandholm 1993; 1998).

The multiagent TSP is defined as follows (Andersson & Sandholm 1998a; 1998b). Several salesmen are going to visit several cities in a world that consists of a unit square, see Figure 1. Each city must 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. The locations of the cities and the starting points of the salesmen are randomly chosen as is each salesman's initial assignment of cities to visit.

Agents' objectives

From this initial assignment, the salesmen can exchange cities, *i.e.* tasks, with each other. Each salesman, i.e. agent, tries to maximize his immediate payoff. The payoff of salesman i consists of the payments received from others for handling their cities, minus his distance traveled, c_i, minus payments sent to others for them to handle some of the former agent's cities.

The cost of travel between any two locations, q and r, is the Euclidean distance: $c_{qr} = \sqrt{(x_q - x_r)^2 + (y_q - y_r)^2}$. The cost that salesman i incurs when visiting his cities is the sum of the costs along his tour:

$$c_i = \sum_{q,r \in \text{ tour of salesman } i} c_{qr}$$

Solution quality, social welfare, is the sum of the distances traveled by the salesmen:

$$\textit{Social welfare} = -\sum_{i=1}^{|A|} c_i$$

The side payments do not affect social welfare because they just redistribute wealth among the agents. Since the salesmen are individually rational in giving and taking tasks and side payments, each agent's payoff improves monotonically during the negotiation. It follows that social welfare improves monotonically.

In our experiments the negotiation occurs before the salesmen are dispatched. The contracting scheme can also directly be used in dynamic settings where new tasks arrive and are handled during the negotiation. Each new domain event simply changes the cost function, c_i, of the corresponding agent.

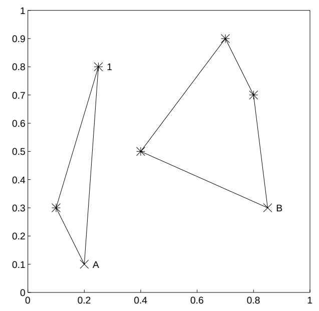

Figure 1: *An example problem instance of a multiagent TSP consisting of five cities (*) and two salesmen (X). If salesman A contracts out city 1 to salesman B, the social welfare will increase due to less travel, i.e., lower costs.*

Summary of contract types

The contracts most commonly used in multiagent contracting systems only allow for one task to move from one agent to another at a time (Smith 1980; Sen 1993). We will refer to this type as an *original (O) contract.* In settings where an agent's cost (or even feasibility) of handling a task depends on what other tasks the agent has, O-contracts often reach local optima where no agreeable contract exists—*i.e.* a contract that improves social welfare so that it could be made profitable for both parties using a side payment—but a globally optimal allocation has not been reached (Sandholm 1998). To address this problem, new types of contracts were recently introduced (Sandholm 1993; Sandholm & Lesser 1995; Sandholm 1996; 1998): *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 two or more tasks from one agent to another. S-contracts let two agents swap tasks with each other: one task is transferred from each agent to the other. M-contracts allow atomic transfers of tasks among more than two agents. We now present these contract types in more detail.

Original contracts (O-contracts)

The most common contracts used in contract net implementations and analysis of contracting games are ones in which one task is transferred from one agent to another. A side payment can also be transferred between the agents to compensate the party that is worse off after the transfer of the tasks, *i.e.*, the agent taking on the task will be paid to do so. Due to this, any social welfare improving contract can be made beneficial to both parties. Formally:

Definition. 2 *An* O-contract *is a pair* $\langle T_{i,j}, \rho_{i,j} \rangle$*, where* $|T_{i,j}| = 1$*.* $T_{i,j}$ *is the task set (including one task) that agent* i *gives to agent* j*, and* $\rho_{i,j}$ *is the contract price that* i *pays to* j *for handling the task set.*

If the agents carried out a full lookahead, *i.e.*, completely searched the tree of all possible future contracts, O-contracts would suffice to reach the globally optimal task allocation (Sandholm 1998). However, this generally cannot be accomplished, except for small problem instances, due to the complexity of searching the tree. The global optimum is not necessarily reached if the agents are myopically individually rational when contracting. Such agents may get stuck in a local optimum since they will not accept a temporary decrease of payoff, which may be necessary to reach the global optimum. Individually rational contracting can be seen as hill-climbing in the task allocation space, where there is a risk of being trapped in a local optimum instead of reaching the globally optimal task allocation.

Cluster contracts (C-contracts)

Cluster contracts allow the agents to exchange more than one task in each contract, together with a side payment (Sandholm 1993; 1998):

Definition. 3 *(Sandholm 1997; 1998) A* cluster contract (C-contract) *is a pair* $\langle T_{i,j}, \rho_{i,j} \rangle$*, where* $|T_{i,j}| > 1$*.* $T_{i,j}$ *is the task set that agent* i *gives to agent* j*, and* $\rho_{i,j}$ *is the contract price that* i *pays to* j *for handling the task set.*

Swap contracts (S-contracts)

Even when both O-contracts and C-contracts are used, local optima exist. For example, even if there is no profitable O- or C-contract, there may be beneficial swaps of tasks to be made between two agents. In a swap contract (Sandholm 1998), one agent gives a task to another agent and receives another task from the latter agent. A side payment may also be paid between the agents to compensate for any value difference between the tasks. Formally:

Definition. 4 *(Sandholm 1997; 1998) A* swap contract (S-contract) *is a 4-tuple* $\langle T_{i,j}, T_{j,i}, \rho_{i,j}, \rho_{j,i} \rangle$*, where* $|T_{i,j}| = |T_{j,i}| = 1$*.* $T_{i,j}$ *is the task set (including one task) that agent* i *gives to agent* j*.* $T_{j,i}$ *is the task set (including one task) that agent* j *gives to agent* i*.* $\rho_{i,j}$ *is the amount that* i *pays to* j*, and* $\rho_{j,i}$ *is the amount that* j *pays to* i*.*

Multiagent contracts (M-contracts)

Even when all three of the contracts described above (O, C, and S) are used, the global optimum may still not be reached if the agents are myopically individually rational when contracting. To avoid some of the remaining local optima, M-contracts were introduced (Sandholm 1998):[3]

Definition. 5 *(Sandholm 1997; 1998) A* multiagent contract (M-contract) *is a pair* $\langle \mathbf{T}, \boldsymbol{\rho} \rangle$ *of* $|A| \times |A|$ *matrices, where at least three elements of* $\mathbf{T}$ *are non-empty (otherwise this would be just a 2-agent contract), and for all* i *and* j*,* $|T_{i,j}| \leq 1$*. An element* $T_{i,j}$ *is the set of tasks that agent* i *gives to agent* j*, and an element* $\rho_{i,j}$ *is the amount that* i *pays to* j*.*

OCSM-contracts

OCSM-contracts are defined as contracts that merge the characteristic of all the contract types discussed so far. That is, any number of tasks can be transferred from and to any agent or between many agents in one single contract:

Definition. 6 *An* OCSM-contract *is a pair* $\langle \mathbf{T}, \boldsymbol{\rho} \rangle$ *of* $|A| \times |A|$ *matrices. An element* $T_{i,j}$ *is the set of tasks that agent* i *gives to agent* j*, and an element* $\rho_{i,j}$ *is the amount that* i *pays to* j*.*

OCSM-contracts are necessary and sufficient for reaching a globally optimal allocation (Sandholm 1996; 1997; 1998). A global optimum will be reached in a finite number of contracts using any hill-climbing algorithm, *i.e.* via any sequence of individually rational contracts. So, from a social welfare perspective, the agents need not look ahead in the tree of possible future contracts. They do not have to take unprofitable contracts in anticipation of synergic ones later on that would make the combination profitable. Furthermore, when accepting a profitable contract, an agent does not need to worry that it will preclude other profitable contracts later on. This is a powerful result for small problem instances, but for large ones the number of contracts made before the global optimum is reached may be prohibitively large. Also, identifying profitable OCSM-contracts can be difficult in the large. Therefore, in large-scale problem instances it is important to be able trade off solution quality against negotiation time. For example, the agents may want to find the best solution that is obtainable in a given amount

[3]Sathi and Fox (1989) (Sathi & Fox 1989) studied a simpler version of multiagent contracts where bids were grouped into cascades.

of time. This paper studies how the different contract types affect that tradeoff.

Contracting system

In principle our contracting system implementation can be used to solve reallocation problems with any number of agents and items. The simulations of this paper focus on the multiagent TSP domain with up to 8 agents and 8 tasks per problem instance. For all combinations of numbers of agents between 2 and 8, and numbers of tasks between 2 and 8, 1000 TSP instances were randomly generated.

Each problem instance was solved five times, four of which used myopically individually rational (*i.e.* hill-climbing) contracting. In the first run, O-contracts were used until a local optimum was found. In the second, C-contracts were used until a local optimum was found. In the third, S-contracts were used until a local optimum was found. In the fourth, M-contracts were used until a local optimum was found. In addition, an exhaustive enumeration of task allocations was conducted in order to find the globally optimal allocation. This corresponds to the outcome that would be reached via myopically individually rational contracting using OCSM-contracts.

In the experiments, each problem instance was tackled in two phases. First, all possible TSPs were solved (for each salesman, there is one TSP corresponding to each subset of cities).[4] Second, the four contracting runs and the exhaustive enumeration of task allocations were conducted.

Contract sequencing

During the contracting run, contracts of the particular type (O, C, S, or M) were applied repeatedly. The algorithm knows that a local optimum has been reached when 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. The agents are numbered from 1 to $|A|$, and each agent's tasks from 1 to $|T_i|$.

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, each O-contract has been tried. However, the process repeats because some O-contracts may have made other O-contracts profitable that were not profitable before. The process stops when no O-contracts have been made during one of these loops were all of them are tried.

Sequencing of cluster contracts In a C-contract one agent moves at least two tasks to another 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, each C-contract has been tried. However, the process repeats because some C-contracts may have made other C-contracts profitable that were not profitable before. The process stops when no C-contracts have been made during one of these loops were all of them are tried.

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, each S-contract has been

[4]The IDA* search algorithm (Korf 1985) 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 underestimating the cost function of the remaining nodes by the minimum spanning tree (Cormen, Leiserson, & Rivest 1990) of those nodes (that is, of 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).

tried. However, the process repeats because some S-contracts may have made other S-contracts profitable that were not profitable before. The process stops when no S-contracts have been made during one of these loops were all of them are tried.

Sequencing of multiagent contracts In an M-contract tasks are being moved between at least three agents. Each agent can transfer at most one task to each other agent. If an M-contract increases social welfare, side payments can be used so that each contract party is better off than before the contract.

First, all combinations where only agent $|A|$ transfers tasks to 3 other agents are tried. The combinations of agents receiving tasks are in order: (1,2,3), (1,2,4), …, (1,2,$|A|$-1), (1,3,2), (1,3,4), …, ($|A|$-1,$|A|$-2,$|A|$-3). For each of these combinations all possible tasks transfers are tried. For agent combination (1,2,3) that is (from agent $|A|$ to agent 1, from agent $|A|$ to agent 2, from agent $|A|$ to agent 3): (1,2,3), (1,2,4), …, ($|T|_{|A|}$,$|T|_{|A|}-1$,$|T|_{|A|}-2$). Then, all combinations where only agent $|A|$-1 transfers 3 tasks to other agents are tried in the same manner as above.

After that, contracts where agent $|A|$ transfers tasks to 4 other agents are tried. After that agent $|A|$-1 tries to transfer tasks to 4 other agents, *etc.* After that, the loop is repeated with giving tasks to 5 other agents, then 6, *etc.*

After individual agents have tried to move their tasks, all combinations of two agents try to move their tasks. First agents $|A|$ and $|A|$-1 try to transfer their tasks (all combinations of their tasks are tried) to all combinations of agents. The order of all agents that will try to transfer their tasks is: ($|A|$), …, (1), ($|A|$,$|A|$-1), …, (1,2), ($|A|$,$|A|$-1,$|A|$-2), … If one of the agents does not have the task needed, that combination is skipped.

As soon as a contract is performed, the scheme starts over from the beginning. The process stops when no M-contracts have been made during one loop were all of them are tried.

Results

To compare the solution quality obtained by the different contract types, the *ratio bound* was used. Let x_j^l denote the social welfare of the task allocation achieved by protocol $l \in$ {O,C,S,M} 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

$$\overline{r}^l = \frac{1}{1000} \sum_{j=1}^{1000} r_j^l$$

This average ratio bound was calculated for all possible combinations of numbers of agents and numbers of tasks.

The differences of the ratio bounds between the contract types were also calculated for statistical significance testing. The difference in ratio bounds between two different contract types, k and l, applied to the same problem instance j, is

$$r_j^{kl} = r_j^k - r_j^l$$

The mean difference between the contract types is

$$\overline{r}^{kl} = \frac{1}{n} \sum_{j=1}^{n} r_j^{kl}$$

Comparison of social welfare

Compared to the other contract types, the mean ratio bound for O-contracts, $\overline{r}^O$, does not vary as much in the number of agents or tasks. The ratio bound increases slightly with both the numbers of agents (Figure 2) and the number of tasks (Figure 3). The ratio bound for O-contracts varies between 1.1 and 1.2, which means that the social welfare using O-contracts is 10% - 20% from optimal.

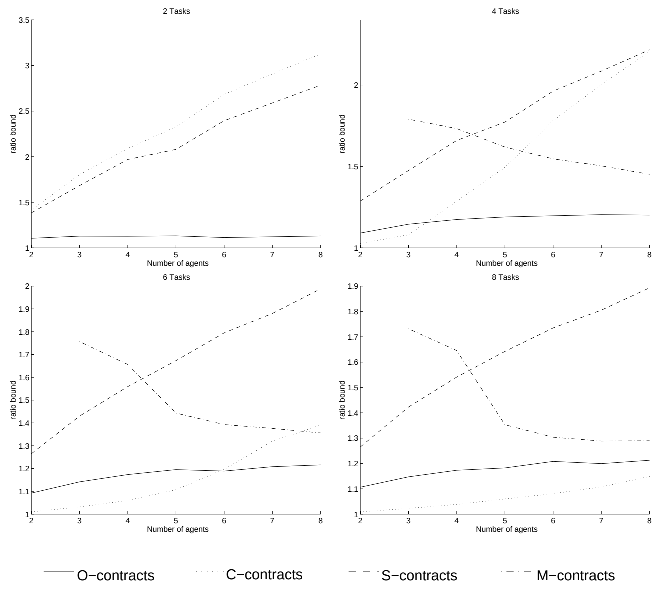

Figure 2: *Ratio bound as a function of the number of agents. The graphs for M-contracts do not include any values for two agents or two tasks since at least three agents and three tasks are needed in an M-contract.*

As the number of tasks increases, the mean ratio bound, $\overline{r}^C$, for C-contracts decreases (Figure 3), *i.e.*, using C-contracts leads to local optima that are closer to the global optimum when the number of tasks is large. While the decrease is monotonic, it is greatest for small numbers of tasks. The ratio bound increases as the number of agents increases (Figure 2). This is especially noticeable in the cases with few tasks (2-5). For greater numbers of tasks, the increase in the ratio bound is smaller (Figure 2, bottom left).

The mean ratio bound, $\overline{r}^S$, for S-contracts also decreases with the number of tasks, and increases with

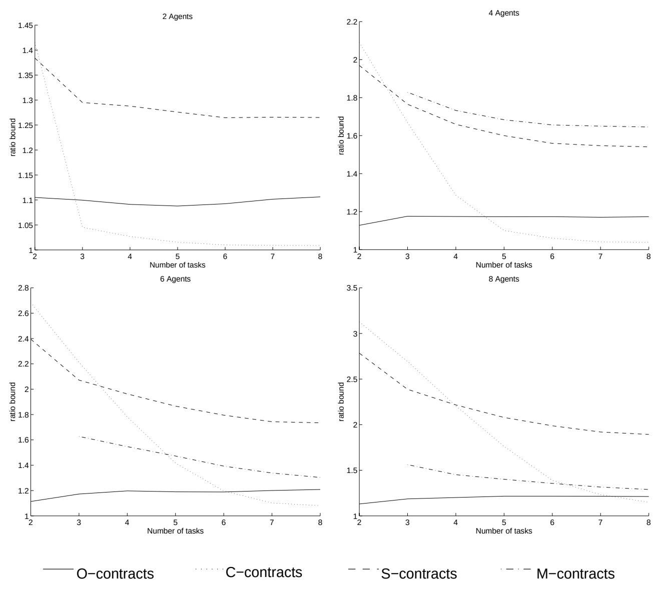

Figure 3: *Ratio bound as a function of the number of tasks.*

the number of agents. However, for S-contracts the increase in the ratio bound is considerable even for large numbers of tasks.

As expected, M-contracts perform better both when the number of agents increases (Figure 2) and when the number of tasks increases (Figure 3). In other words, the mean ratio bound, $\overline{r}^M$, decreases with the number of tasks and agents. This is obvious in the bottom right graph in Figure 3. Extrapolating from these results suggests that M-contracts could reach a lower ratio bound than any of the other contract types for much greater numbers of agents and tasks than eight.

Figure 4 shows that O-contracts always perform better than S- and M-contracts. C-contracts provide a lower ratio bound than O-contracts when the number of tasks is greater than the number of agents. For those numbers of agents and tasks, C-contracts are the best contract type also when compared to S- and M- contracts. So, the top left graph in Figure 4 summarizes which contract types are best for which numbers of agents and tasks.[5]

Computational aspects

The number of contracts that has to be tried before reaching a local optimum varies considerably across the contract types, as does the number of contracts that is needed to verify that a local optimum has been reached, Figure 5. As expected, the number of contracts made

[5]The black and white areas represent results that are significant at the 0.05 confidence level of the mean difference ratio bounds in a paired t-test. The gray areas represent results that are not significant at the 0.05 level, yet one of the contracts is better. In the dark gray areas the latter contract is better while in the lighter gray areas the former is better. While the paired t-test formally assumes normal distributions, we use it because it has been shown to be very robust against distributional variations (Cohen 1995).

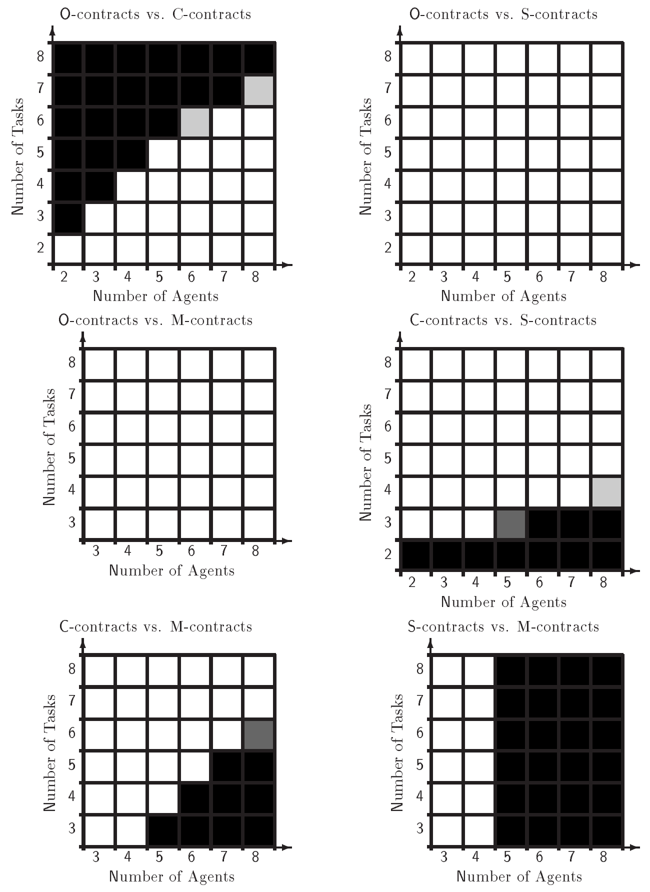

Figure 4: *Pairwise comparison of the differences in ratio bounds $\overline{r}^{kl}$ for O-, C-, S-, and M-contracts. Above each graph the two contract types under comparison are stated. The darker the color in a square is, the better the latter contract type.*

and tried before reaching a local optimum, and the number of contracts needed to verify the local optimum increase with the number of tasks. These numbers are polynomial in tasks for all contract types (the curves are sublinear on a logarithmic scale). Similarly, the numbers are polynomial in agents—these curves are omitted for brevity.

O-contracts perform on average the largest number of contracts before reaching a local optimum, followed by C-contracts, S-contracts, and M-contracts. The fact that O-contracts only move one task in each contract is likely to contribute to this result. This result is interesting since O-contracts are desirable because they require the smallest number of contracts to *verify* that a local optimum has been reached. On the other hand, O- and C-contracts need to try a larger number of contracts before reaching a local optimum than S- and M-contracts do. In the case of six agents and six tasks, O-contracts and C-contracts still need less than 100 contracts to reach a local optimum—except in a small number of cases. With the exclusion of some exceptional cases where several thousand contracts are needed, M-contracts find local optima after a small number of con-

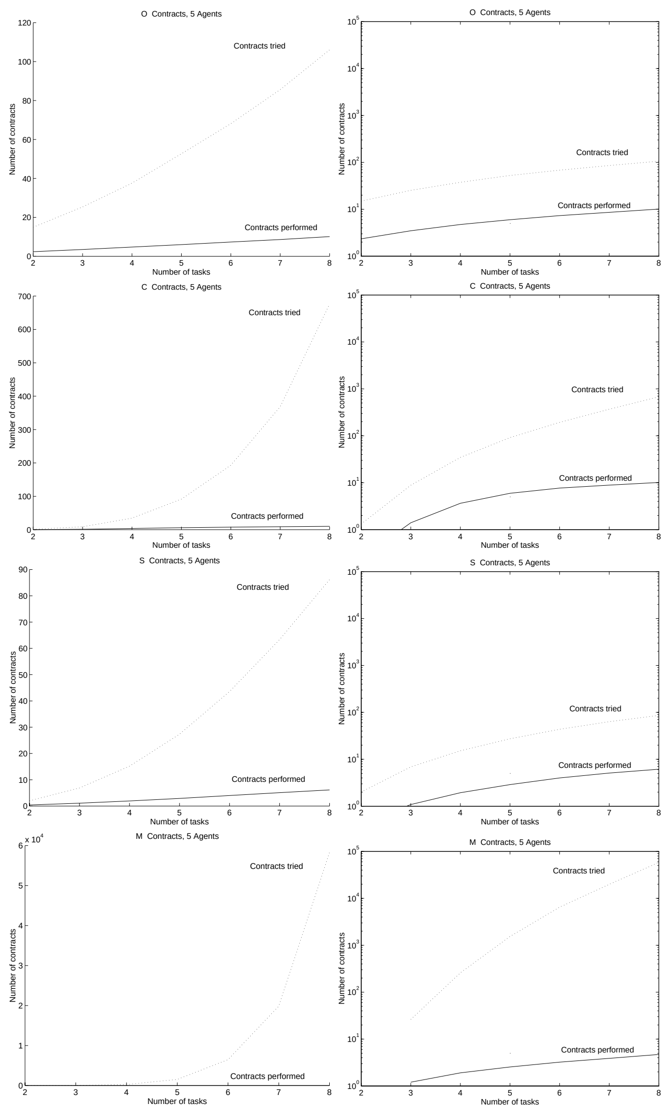

Figure 5: *Number of contracts tried (including those to verify the local optimum) before the system reached a local optimum (dotted line), and number of contracts performed before reaching the local optimum (solid line). Left graphs have linear scales. The scales on the value axes differ between the graphs. The value axes of the graphs on the right have logarithmic scales.*

tracts have been tried. This may be affected by the specific order of trying M-contracts. Note that the discussion until now concerns the number of contracts needed to reach a local optimum, not the number needed to verify that one has been reached. M-contracts need by far the greatest number of contracts to verify that the solution is a local optimum. Then, in order, C-contracts, S-contracts, and O-contracts follow.

The CPU-time used for negotiation is proportional to the number of contracts tried, but the constant of proportionality varies greatly between the contract types: it is much greater for M-contracts than O-, C-, and S-contracts. This is because M-contracts are more complicated than the other contract types, and because many contracts need to be checked to verify that a local optimum has been reached.

Dynamics of contracting

The typical final task allocations are very different between the contract types. C-contracts tend to concentrate the tasks to one agent or a few of them. O-contracts tend to spread the contracts to all the agents. Due to the sequencing of M-contracts, the tasks tend to be allocated too often to agent 1. One could avoid such anomalies by randomly picking M-contracts to try. However, a systematic scheme is necessary to verify that a local optimum has been reached. The number of tasks per agent cannot change at all when S-contracts are used, which contributes to their poor performance. As is desired from an anytime contracting perspective, contracts performed earlier often improved the social welfare more than later contracts.

Conclusions

The capability to profitably reallocate items is a key feature in automated negotiation. Currently, the most widely used contract type allows for only one task at a time to be moved from one agent to another (O-contracts). New contract types (cluster (C), swap (S), multiagent (M), and OCSM-contracts) were recently introduced to avoid some of the local optima in which O-contracts can get stuck when used by myopic self-interested agents. They are all based on moving several tasks in a single contract. This reduces the number of local optima in the search space of task allocations for hill-climbing-based contracting algorithms.

OCSM-contracts guarantee that a global optimum is reached in a finite number of contracts independent of the order of the contracts. Although this is a powerful result for small problem instances, in large-scale problems the number of steps needed to reach the global optimum may be impractically large. In such problems it is better to accept the best achievable solution in a limited amount of time than to strive for the global optimum. Marginal cost based contracting allows this because it is an anytime algorithm: each agent's payoff increases monotonically in time. However, the rate of increase depends on which contract types are used. To determine how best to increase social welfare quickly in time, we compared the five contract types on an example problem called the multiagent TSP.

The results regarding the social welfare of the local optima of the different contract types provide guidelines to system builders regarding what contract types to use in different environments when computation is limited. We also presented timing results which can be used in the choice of contract type if there is not enough time to even reach a local optimum. In addition, our results help in the choice of contract type when certain properties of the final outcome are desired, *e.g.*, that tasks are distributed among multiple agents or concentrated to just a small number of agents. For six agents and six tasks, a local optimum was reached within the first 100 contracts tried, with the exclusion of some exceptional cases. This is important since several hundred - sometimes several thousand - contracts were often tried before it could be verified that a local optimum had been reached. M-contracts reached a local optimum faster (when measured in the number of contracts tried or in the number of contracts performed before the optimum was reached) than the other contract types. However, M-contracts require more CPU-time per contract than O-, C-, or S-contracts. They also require a significantly larger number of contracts to be tried in order to verify that a local optimum has been reached.

For these relatively small problem instances, O- and C-contracts led to better local optima than S- and M-contracts. C-contracts performed best when the number of tasks was greater than the number of agents; otherwise O-contracts were best. Extrapolating to problems containing more agents and tasks, M-contracts may reach the best local optima. Despite the fact that O- and C-contracts lead to similar social welfare values, the typical task allocations are very different: O-contracts tend to spread the tasks among all agents while C-contracts tend to concentrate the tasks to only one agent or a few agents.

Sequencing of contracts within a particular contract type influences the results. Analyzing this effect further is part of our future research. Also, to improve the social welfare, more than one contract type can be used during contracting (Andersson & Sandholm 1998c). Further research is required to determine the optimal way to sequence the different contract types in order to obtain satisfactory social welfare with bounded negotiation time. There are several possible approaches: change the contract type for every single contract, apply many possible contracts (maybe all) of one contract type before changing the type, or find a local optimum using one contract type before changing to another contract type. There is also the question of which of the contract types should be interleaved with each other. 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 myopic individually rational agents will reach the globally optimal allocation, but they would lead to a local optimum faster than OCSM-contracts, and to higher average social welfares than O-, C-, S-, or M-contracts individually.

References

Andersson, M. R., and Sandholm, T. W. 1998a. Contract types for satisficing task allocation: II experimental results. In *AAAI Spring Symposium Series: Satisficing Models*, 1–7.

Andersson, M. R., and Sandholm, T. W. 1998b. Leveled commitment contracting among myopic individually rational agents. In *ICMAS*, 26–33.

Andersson, M. R., and Sandholm, T. W. 1998c. Sequencing of contract types for anytime task reallocation. In *Agents-98 Workshop on Agent-Mediated Electronic Trading (AMET)*. Reprinted in Springer Verlag LNAI 1571, pp. 54–69, 1999.

Cohen, P. R. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press.

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

Kalakota, R., and Whinston, A. B. 1996. *Frontiers of Electronic Commerce*. Addison-Wesley.

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

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

Rassenti, S. J.; Smith, V. L.; and Bulfin, R. L. 1982. A combinatorial auction mechanism for airport time slot allocation. *Bell J. of Economics* 13:402–417.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter*. MIT Press.

Rothkopf, M. H.; Pekeč, A.; and Harstad, R. M. 1998. Computationally manageable combinatorial auctions. *Management Science* 44(8):1131–1147.

Sandholm, T. W., and Lesser, V. R. 1995. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *ICMAS*, 328–335. Reprinted in *Readings in Agents*, Huhns and Singh, eds., pp. 66–73, 1997.

Sandholm, T. W., and Ygge, F. 1997. On the gains and losses of speculation in equilibrium markets. In *IJCAI*, 632–638.

Sandholm, T. W. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *AAAI*, 256–262.

Sandholm, T. W. 1996. *Negotiation among Self-Interested Computationally Limited Agents*. Ph.D. Dissertation, University of Massachusetts, Amherst. At www.cs.wustl.edu/ ~sandholm/ dissertation.ps.

Sandholm, T. W. 1997. Contract types for optimal task allocation: I theoretical results. WUCS-97-35, Washington University, Dept. of Computer Science.

Sandholm, T. W. 1998. Contract types for satisficing task allocation: I theoretical results. In *AAAI Spring Symposium Series: Satisficing Models*, 68–75.

Sathi, A., and Fox, M. 1989. Constraint-directed negotiation of resource reallocations. In Huhns, M. N., and Gasser, L., eds., *Distributed Artificial Intelligence*, volume 2, Pitman, chapter 8, 163–193.

Sen, S. 1993. *Tradeoffs in Contract-Based Distributed Scheduling*. Ph.D. Dissertation, Univ. of Michigan.

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