Contract Types for Satisficing Task Allocation: I Theoretical Results

Tuomas W. Sandholm
sandholm@cs.wustl.edu
Department of Computer Science
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

We analyze task reallocation where individually rational (IR) agents (re)contract tasks among themselves based on marginal costs. A task allocation graph is introduced as a tool for analyzing contract types. Traditional single task contracts always have a short path (sequence of contracts) to the optimal task allocation but an IR path may not exist, or it may not be short. We analyze an algorithm for finding the shortest IR path.

Next we introduce cluster contracts, swaps, and multiagent contracts. Each of the four contract types avoids some local optima that the others do not. Even if the protocol is equipped with all four types, local optima exist. To attack this problem, we introduce OCSM-contracts which combine the ideas behind the four earlier types into an atomic contract type. If the protocol is equipped with OCSM-contracts, any sequence of IR contracts leads to the optimal task allocation in a finite number of steps: an oracle—or speculation—is not needed for choosing the path (no subset of OCSM-contracts suffices even with an oracle). This means that the multiagent search does not need to backtrack.

This is a powerful result for small problem instances. For large ones, the anytime feature of our multi-contract-type algorithm—with provably monotonic improvement of each agent's solution—is more important. [1]

Introduction

Multiagent systems are becoming increasingly important due to strong technology push and application pull. The capability of (re)allocating tasks among agents is a key feature in such systems. In many domains, significant savings can be achieved by reallocating tasks among agents. Some tasks are inherently synergic, and should therefore be handled by the same agent. On the other hand, some tasks have negative interactions, in which case it is better to allocate them to different agents. Furthermore, different agents may have different resources which leads to different capabilities and costs for handling tasks. This paper studies task allocation among self-interested agents in the following model which captures the above considerations.

Definition 1 *Our* task allocation problem *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] [3]

We use a variant of the contract net approach—which is a distributed method for mutual selection of contractors and contractees (Smith 1980)—for satisficing task (re)allocation.

The next section presents the use of marginal costs as a basis for making individually rational (IR) contracts. Then, classic contracts of one task at a time are studied. After that, three new contract types—cluster contracts, swaps, and multiagent contracts—are introduced and analyzed. Finally, combinations of these contracts types are discussed.

Marginal cost based contracting

The original contract net (Smith 1980) lacked a formal model for making bidding and awarding decisions.

[1]Supported by NSF CAREER award IRI-9703122 and NSF grant IRI-9610122. A one-page poster on this work was published in (Sandholm 1997). An experimental analysis of these contract types is presented in (Andersson & Sandholm 1998; 1997).

[2]This definition generalizes what are called "Task Oriented Domains" (Rosenschein & Zlotkin 1994). Specifically, we allow asymmetric cost functions among agents (e.g. due to different resources). We also allow for the possibility that some agent may be unable to handle some sets of tasks. This is represented by a cost of infinity.

[3]Although we analyze the static version of the problem, the contracting scheme that we will present works even if tasks and resources (resources affect the cost functions) are added and removed dynamically.

Such a model would allow one to develop methods that provably lead to desirable task allocations among agents. We follow the approach of (Sandholm 1993), where contracting decisions are based on marginal cost calculations. In so doing we invoke the concept of *individual rationality* on a per contract basis. A contract is individually rational (IR) to an agent if that agent is better off with the contract than without it. [4] This implies individual rationality of sequences of contracts.

Specifically, a contractee g accepts a contract if it gets paid more than its marginal cost

$$MC^{add}(T^{contract}|T_g) = c_g(T^{contract} \cup T_g) - c_g(T_g)$$

of handling the tasks $T^{contract}$ of the contract. The marginal cost is dynamic in the sense that it depends on the other tasks T_g that the contractee already has.

Similarly, a contractor h is willing to allocate the tasks $T^{contract}$ from its current task set T_h to the contractee if it has to pay the contractee less than it saves by not handling the tasks $T^{contract}$ itself:

$$MC^{remove}(T^{contract}|T_h) = c_h(T_h) - c_h(T_h - T^{contract}).$$

In the protocol, agents then suggest contracts to each other, and make their accepting/rejecting decisions based on these marginal cost calculations. An agent can take on both contractor and contractee roles. It can also recontract out tasks that it received earlier via another contract. The scheme does not assume that agents know the tasks or cost functions of others.

With this domain independent contracting scheme, the task allocation can only improve at each step. This corresponds to hill-climbing in the space of task allocations where the height-metric of the hill is social welfare $(-\sum_{i \in A} c_i(T_i))$. The fact that the contractor pays the contractee some amount between their marginal costs (e.g. half way between) causes the benefit from the improved task allocation to be divided so that no agent is worse off with a contract than without it.

[4] This differs from payoff maximizing agents of game theory (Mas-Colell, Whinston, & Green 1995; Fudenberg & Tirole 1991). Such an agent may reject an IR contract e.g. if it believes that it could be better off by waiting for a more beneficial contract that cannot be accepted if the former contract is accepted (e.g. due to limited resources). Similarly, such an agent may accept a non-IR contract in anticipation of a synergic later contract that will make the combination beneficial. Our approach is more practical because each contract can be made by evaluating just a single contract (each contract party evaluating one new task set) instead of doing exponential lookahead into the future. Our deviation from game theory comes at the cost of not being able to normatively guarantee that a self-interested agent is best off by following the strategy (of accepting any IR contracts) that we propose.

The scheme is an *anytime algorithm*: contracting can be terminated at any time, and the worth (payments received from others minus cost of handling tasks) of each agent's solution increases monotonically. It follows that social welfare increases monotonically. Details on an asynchronous distributed contract net implementation based on marginal costs can be found in (Sandholm 1993).

Classic contracts of one task at a time (O-contracts)

In most contract net implementations, each contract regards only one task (Smith 1980; Sen 1993; Gu & Ishida 1995). We now formalize this contract type:

Definition 2 *An* O-contract *is defined by 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.*

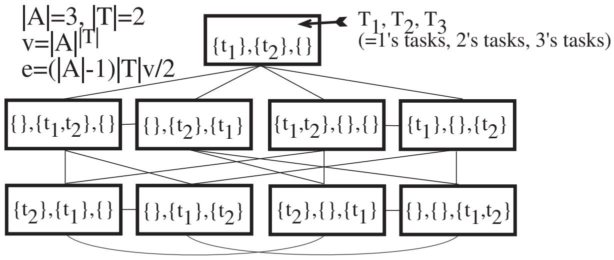

Figure 1: *Task allocation graph. The vertices represent task allocations, and the edges represent O-contracts.*

For any problem instance, the optimal task allocation can be reached via O-contracts:

Proposition 1 (Path) *A path of O-contracts always exists from any task allocation to the optimal one. The length of the shortest such path is at most* $|T|$.

Proof. The path can be constructed by moving one task at a time from the agent that initially has it to the agent that has it in the globally optimal task allocation. □

This means that if agents could carry out full lookahead—which is impossible in all but the smallest problem instances—O-contracts would suffice to reach the optimal task allocation because agents would be willing to temporarily move to worse task allocations in order to finally reach the optimal one. However, when agents use individual rationality as a criterion for contracting, they will not accept the temporary decrease in social welfare. It turns out that this leads to local optima—even if there were an oracle for choosing the sequence of IR contracts:

Proposition 2 (IR path) *In some instances of our task allocation problem, no path of IR O-contracts exists from the initial allocation to the optimal one. The length of the shortest IR path (if one exists) may be greater than* $|T|$. *However, the shortest IR path is never longer than* $|A|^{|T|} - (|A|-1)|T|$. [5]

Since the number of task allocations is $|A|^{|T|}$, this means that not all task allocations can be visited on the shortest IR path to the optimum.

Computing the shortest IR path

We mainly use the task allocation graph to analyze contract types. However, if the problem instance is not very large, the graph can actually be constructed and used to choose a sequence of contracts. To avoid unnecessary negotiations, it is desirable to minimize the number of contracts while still reaching the optimal task allocation. The approach is to first determine the optimal task allocation via a linear pass through the vertices. Next, breadth-first-search is run on the graph starting from the initial task allocation. The only difference is that when inserting children into the open list, the algorithm omits children that have lower social welfare than the parent (these contracts would not be IR). When the search reaches the optimal task allocation for the first time, the shortest IR path has been found. If the search terminates without reaching the optimal vertex, no IR path exists.

It is known that breadth-first-search runs in $O(v+e)$ time, where v is the number of vertices and e is the number of edges in the graph (Cormen, Leiserson, & Rivest 1990). [6] There is one vertex for each task allocation, so $v = |A|^{|T|}$. The number of edges is the same at each vertex: $|T|(|A|-1)$. This is because in an O-contract, any task can be transferred to any agent except its current holder. So, $e = \frac{v}{2}|T|(|A|-1)$. Therefore, the total running time is $O(|A|^{|T|} + \frac{|A|^{|T|}}{2}|T|(|A|-1)) \subset O(|T||A|^{|T+1|}) \subset O(v\sqrt{v})$.

Sparseness of O-contracts

To get an intuition about the search space in which contracting occurs, we show that the task allocation graph becomes arbitrarily sparse as the problem size increases:

Proposition 3 *Let* $|A| \geq 2$ *and* $|T| \geq 2$. *Now,* $\frac{e}{\#\textit{edges in fully connected graph}} \rightarrow 0$ *as* $|T| \rightarrow \infty$ *as well as when* $|A| \rightarrow \infty$.

Proof. $\frac{e}{\#\text{edges in fully connected graph}} = \frac{\frac{|A|^{|T|}}{2}|T|(|A|-1)}{\frac{v^2-v}{2}} = \frac{|T|(|A|-1)}{|A|^{|T|}-1}$. This approaches 0 when $|A| \geq 2$ and $|T| \rightarrow \infty$. It also approaches 0 when $|T| \geq 2$ and $|A| \rightarrow \infty$. □

Cluster contracts (C-contracts)

Using one task per contract is insufficient: the agents may get stuck in a local optimum (if the cost or feasibility of carrying out a task depends on the carrying out of other tasks, or agents have asymmetric cost functions). We now formalize a contract type that addresses this problem:

Definition 3 *A* cluster contract (C-contract) *is defined by 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.*

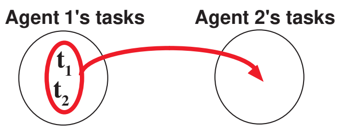

Figure 2: *Example of a cluster contract.*

C-contracts induce a set of edges in the task allocation graph that is disjoint from the set of edges induced by O-contracts. This leads to C-contracts avoiding some of the local optima of O-contracts (O-contract also avoid some of the local optima of C-contracts). This issue is formalized later in Thrm 8.

The need for larger transfers is well known in centralized iterative refinement optimization (Lin & Kernighan 1971; Waters 1987), but has been historically ignored in automated negotiation. Recently, the TRACONET system extended the contract net to handle task interactions by having the announcer cluster tasks into sets to be negotiated atomically (Sandholm 1993). Alternatively, the bidder could have done the clustering by counterproposing. Later, a protocol was presented that generalizes this by allowing either party to do the clustering at any stage of the protocol (Sandholm & Lesser 1995).

In some non-automated allocation settings, the need for clusters has recently been realized. For example, when the Federal Communications Commission (FCC) auctions airwave bandwidth for restricted geographical areas, the bidders' valuations for the auctioned items depend on what other items they are awarded (McAfee & McMillan 1996). For example, some bidders want to receive a cluster of awards that allows them to establish nationwide coverage. In the FCC auctions, explicit

[5] Some proofs are omitted for brevity.

[6] This assumes that the graph uses an adjacency list representation. If not, building this representation from a distance matrix requires $\Theta(v^2)$ time.

clustering was not used, but the agents could construct the clusters from individually auctioned items. A simultaneous ascending auction was used, where each agent sees the other agents' bids. Therefore, the agents could see which clusters they were likely to get, and bid accordingly. The auction was carried out in stages, where each stage had a quantitative activity rule. This disables any bidder from staying out at first, and then bidding on the last moment when it knows the others' bids. If the latter were possible, few agents would want to bid up front. The auction terminated when no agent wanted to raise its bids. The auction designers supported profitable clustering also by allowing agents to withdraw from their bid—e.g. if an agent did not get the cluster that it strived for. If an agent withdrew, the item was opened for reauctioning. If the new highest bid was lower than the old one, the withdrawing agent had to pay the difference. Alternatively, the decommitting penalties could have been fixed when the item was first auctioned. Deceitful decommitting strategies in such protocols have been game theoretically analyzed in (Sandholm & Lesser 1996).

In general, a cluster can encompass any number of tasks. [7] A task allocation is called k-optimal if no beneficial cluster contract of any k tasks can be made between any two agents. Now, m-optimality does not imply n-optimality when $m < n$. More surprisingly, n-optimality does not imply m-optimality.

Global optimality implies k-optimality for all k. However, the reverse is not true. It also turns out that C-contracts may not lead to the optimal task allocation—even if there were an oracle for choosing the path of C-contracts of any cluster size:

Proposition 4 (No path) *There are instances of the task allocation problem where no path (IR or not) of C-contracts leads from the initial task allocation to the optimal one.*

Proof. Say that there are two agents, and one task. Since there is only one task, no C-contract is possible. However, the allocation is not optimal if the agent with the higher cost of handling the task has the task, and the costs of handling no tasks are zero. □

Another problem is that without an oracle, contracting may get stuck in a local optimum even if some IR path from the initial allocation to the optimal one exists because that path may not be chosen.

[7] TRACONET used clusters of 2 or 3 tasks per announcement. Heuristic methods for choosing the tasks to cluster and sequencing the different cluster sizes were also studied.

Swap contracts (S-contracts)

Sometimes there is no task set size such that transferring such a set from one agent to another (C-contract or O-contract) enhances the task allocation. Yet there may be a beneficial *swap* of tasks where the first agent subcontracts a task to the second and the second subcontracts another task to the first (Sandholm & Lesser 1995). We now formalize this concept:

Definition 4 *A* swap contract (S-contract) *is defined by 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*.*

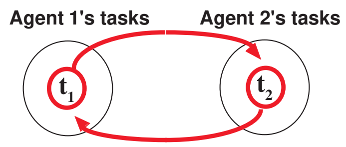

Figure 3: *Example of a swap contract.*

S-contracts induce a set of edges in the task allocation graph that is disjoint from the sets of edges induced by O- and C-contracts. This leads to S-contracts avoiding some of the local optima that O- and C-contracts cannot resolve (O- and C-contracts in turn avoid some of the local optima of S-contracts). This issue is formalized later in Thrm 8.

The protocols needed for cooperative agents and those needed for self-interested agents differ. Cooperative agents can be assumed to take care of each others tasks without compensation whenever that is beneficial for the society of agents. Self-interested agents need some compensation to take care of some other agent's task. This compensation can be organized as barter trade: one agent takes care of some of another agent's tasks if the latter agent takes care of some of the former agent's tasks. Barter trades that benefit both agents (IR deals) do not always exist even if it were profitable to move a task from one agent to another. Secondly, identifying beneficial barter exchanges is more complex than identifying one way transfers of tasks—especially in a distributed setting. A finer resolution of cooperation among self-motivated agents can be achieved by a monetary compensation mechanism: an agent pays another agent to take care of some of its tasks. The need for swaps shows that payment based exchanges cannot replace all barter exchanges. What is needed is the monetary exchange method (that allows infinitely divisible side-payments) but also a linking mechanism that allows swapping tasks atomically among agents (S-contracts).

Although S-contracts are necessary, they are not sufficient for reaching the global optimum:

Proposition 5 (No path) *There are instances of the task allocation problem where no path (IR or not) of S-contracts leads from the initial task allocation to the optimal one.*

Proof. If some agent has a different number of tasks in the initial task allocation than what that agent would have in the optimal allocation, then the optimal allocation cannot be reached. This is because S-contracts preserve the number of tasks that each agent has. □

Also, even if an IR path exists, the contracting agents may use some other path of IR contracts and get stuck in a local optimum. Furthermore, because S-contracts preserve the number of tasks that each agent has, they can only reach a very small subset of task allocations:

Proposition 6 *Let $|A| \geq 2$ and $|T| \geq 2$. As $|T| \rightarrow \infty$, or $|A| \rightarrow \infty$, the fraction of task allocations that are reachable via any (IR or not) S-contracts from any given initial vertex approaches zero.*

This implies that the chance of reaching the optimal task allocation via any (even one with an oracle for picking the path) hill-climbing S-contracting algorithm vanishes.

Multiagent contracts (M-contracts)

Even if negotiations have reached a local optimum with respect to mutual (O-, C-, and S-) contracts of any size, solution enhancements may be possible if tasks were transferred among more than two agents. [8]

[8] Sathi and Fox studied the role of grouping buy and sell bids into cascades which may involve multiple agents (Sathi & Fox 1989). Their setting is simpler than ours in that the value of a contract to an agent does not depend on which other ones of the agent's bids get accepted. In our setting, an agent's valuation of a contract depends significantly on which other ones of the agent's bids get accepted (marginal cost depends on which other tasks the agent has). Their algorithms also differ from ours in that they do not allow recontracting once a contract has been made. They present three heuristic algorithms for choosing the order in which to execute possible contracts. The order is important because one contract can preclude another if the two involve the allocation of the same resource. Because recontracting is not used, the former contract cannot be undone to allow the latter—even if the latter is more beneficial. The heuristics focus on handling more constrained requests first. They do not guarantee that the optimal solution is reached. Some of the algorithms are distributed while others involve centralized matching of bids.

Definition 5 *A* multiagent contract (M-contract) *is defined by 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.*

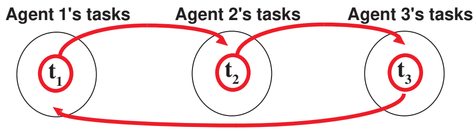

Figure 4: *Example of a multiagent contract.*

Decentralized multiagent contracts can be implemented for example by circulating the contract message among the parties and agreeing that the contract becomes valid if every agent signs. M-contracts induce a set of edges in the task allocation graph that is disjoint from the sets of edges induced by O-, C-, and S-contracts. This leads to M-contracts avoiding some of the local optima that O-, C-, and S-contracts cannot resolve (O-, C-, and S-contracts in turn avoid some of the local optima of M-contracts). This issue is formalized later in Thrm 8.

Although M-contracts are necessary, they are not sufficient:

Proposition 7 (No path) *There are instances of the task allocation problem where no path (IR or not) of M-contracts leads from the initial task allocation to the optimal one.*

Proof. Say that there are two agents, and one task. Since there are only two agents, no M-contract is possible. However, the allocation is not optimal if the agent with the higher cost of handling the task has the task, and the costs of handling no tasks are zero. □

Another problem is that contracting may get stuck in a local optimum even if some IR path from the initial allocation to the optimal one exists because the agents may choose some other path of IR contracts.

Merging the types: OCSM-contracts

No one of the presented four contract types is sufficient for reaching the global optimum via IR contracts—even if there were an oracle for choosing the path of contracts—because no IR path need exist to the optimal task allocation. However, each of the four contract types avoids some of the local optima that the other three do not:

Proposition 8 *For each of the four contract types (O, C, S, and M), there exist task allocations where no IR* *contract with the other three contract types is possible, but an IR contract with the fourth type is.*

Proof. The following example shows this for O-contracts. Let there be one task t_1 and two agents: 1, 2. Let the current task allocation be $T_1 = \{t_1\}$, $T_2 = \emptyset$. Let the task handling costs be $c_1(\emptyset) = 0$, $c_1(\{t_1\}) = 2$, $c_2(\emptyset) = 0$, $c_2(\{t_1\}) = 1$. The O-contract of moving t_1 to 2 would decrease the global cost by 1. At the same time, no C-, S-, or M-contract is possible.

The following example shows this for C-contracts. Let there be two tasks: t_1, t_2, and two agents: 1, 2 (Fig. 2). Let the current task allocation be $T_1 = \{t_1, t_2\}$, $T_2 = \emptyset$. Let the task handling costs be $c_1(\emptyset) = 0$, $c_1(\{t_1\}) = c_1(\{t_2\}) = 4$, $c_1(\{t_1, t_2\}) = 5$, $c_2(\emptyset) = 0$, $c_2(\{t_1\}) = c_2(\{t_2\}) = 2$, $c_2(\{t_1, t_2\}) = 3$. So, the current global cost is 5. Moving t_1 to 2 would increase the global cost to 6. So would moving t_2 to 2. So, no O-contract is IR. No S- or M-contract is possible at all. However, the C-contract of moving both t_1 and t_2 to 2 would decrease the global cost to 3.

The following example shows this for S-contracts. Let there be two tasks: t_1, t_2, and two agents: 1, 2 (Fig. 3). Let the current task allocation be $T_1 = \{t_1\}$, $T_2 = \{t_2\}$. Let the task handling costs be $c_1(\emptyset) = 0$, $c_1(\{t_1\}) = 2$, $c_1(\{t_2\}) = 1$, $c_1(\{t_1, t_2\}) = 5$, $c_2(\emptyset) = 0$, $c_2(\{t_1\}) = 1$, $c_2(\{t_2\}) = 2$, $c_2(\{t_1, t_2\}) = 5$. So, the current global cost is 4. Because each agent has only one task, no C-contract is possible. Because there are only two agents, no M-contract is possible. Moving t_1 to 2 would increase the global cost to 5. So would moving t_2 to 1. Thus no O-contract is IR. However, the S-contract of moving t_1 from 1 to 2 and simultaneously t_2 from 2 to 1 would decrease the global cost to 2.

The following example shows this for M-contracts. Let there be three tasks: t_1, t_2, t_3, and three agents: 1, 2, 3 (Fig. 4). Let the current task allocation be $T_1 = \{t_1\}$, $T_2 = \{t_2\}$, $T_3 = \{t_3\}$. Say that the task handling costs are $c_1(\emptyset) = 0$, $c_1(\{t_1\}) = 2$, $c_1(\{t_2\}) = 5$, $c_1(\{t_3\}) = 1$, $c_1(\{t_1, t_2\}) = c_1(\{t_1, t_3\}) = c_1(\{t_2, t_3\}) = 10$, $c_1(\{t_1, t_2, t_3\}) = 15$, $c_2(\emptyset) = 0$, $c_2(\{t_1\}) = 1$, $c_2(\{t_2\}) = 2$, $c_2(\{t_3\}) = 5$, $c_2(\{t_1, t_2\}) = c_2(\{t_1, t_3\}) = c_2(\{t_2, t_3\}) = 10$, $c_2(\{t_1, t_2, t_3\}) = 15$, $c_3(\emptyset) = 0$, $c_3(\{t_1\}) = 5$, $c_3(\{t_3\}) = 2$, $c_3(\{t_2\}) = 1$, $c_3(\{t_1, t_2\}) = c_3(\{t_1, t_3\}) = c_3(\{t_2, t_3\}) = 10$, $c_3(\{t_1, t_2, t_3\}) = 15$. So, the current global cost is $2 + 2 + 2 = 6$. C-contracts are impossible here because no agent has more than one task. Any one of the six possible O-contracts would increase global cost to $0 + 2 + 10 = 12$. Any one of the three possible swaps would increase global cost to $1 + 2 + 5 = 8$. Therefore, no O-, or S-contract is IR. However, the M-contract of moving t_1 from 1 to 2, t_2 from 2 to 3, and t_3 from 3 to 1 would decrease the global cost to 3. □

Unfortunately, even if the contracting protocol is equipped with all four of the contract types, the optimal task allocation may not be reached via IR contracts—even with an oracle: [9]

Proposition 9 (IR path) *There are instances of the task allocation problem where no IR path from the initial task allocation to the optimal one exists using O-contracts, C-contracts, S-contracts and M-contracts.*

Proof. Think of a deal where one agent gives a task to another agent, and the other agent gives two tasks to the first agent. This can be made to be the only welfare increasing deal by picking the cost functions appropriately. However, this deal is not possible via an O-, C-, S-, or M-contract. □

Clearly, no subset of the contract types suffices either:

Corollary 1 (IR path) *There are instances of the task allocation problem where no IR path from the initial task allocation to the optimal one exists using any pair or triple of the following contract types: O-contracts, C-contracts, S-contracts and M-contracts.*

Proof. Proposition 9 shows that an IR path may not exist even with all four contract types. Removing contract types can only remove edges from the task allocation graph. Thus, removing contract types cannot introduce new paths. □

Another problem is that without an oracle, contracting may get stuck in a local optimum even if some IR path exists because the agents may choose some other IR path.

To address these shortcomings, let us define a new contract type, *OCSM-contract*, that combines the characteristics of O-, C-, S-, and M-contracts into one contract type—where the ideas of the four earlier contract types can be applied simultaneously (atomically).

Definition 6 *An* OCSM-contract *is defined by 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 induce a fully connected task allocation graph. On the other hand, O- (Fig. 1), C-, S-, and M-contracts each induce disjoint sets of edges which are strict subsets of the edges of the fully connected graph. The union of the edge sets induced by O-, C-, S-, and M-contracts is a strict subset of the edges of the fully connected graph.

We could proceed to show that a path and an IR path always exist from any task allocation to

[9] The optimal task allocation can be reached with these four contract types if one allows a non-IR path. This is because even O-contracts alone suffice for that (Prop. 1).

the optimal one if the contracting protocol incorporates OCSM-contracts. However, we show something stronger. The following proposition states that OCSM-contracts are sufficient for reaching the global task allocation optimum in a finite number of contracts. The result holds for any sequence of IR OCSM-contracts, i.e. for any hill-climbing algorithm that uses OCSM-contracts: an oracle is not needed for choosing the path. This means that from the perspectives of social welfare maximization and of individual rationality, agents can accept IR contracts as they are offered. They need not wait for more profitable ones, and they need not worry that a current contract may make a more profitable future contract unprofitable. Neither do they need to accept contracts that are not IR in anticipation of future contracts that make the combination beneficial. Furthermore, these hill-climbing algorithms do not need to backtrack.

Proposition 10 *Let $|A|$ and $|T|$ be finite. If the contracting protocol allows OCSM-contracts, any hill-climbing algorithm (i.e. any path of IR contracts) finds the globally optimal task allocation in a finite number of steps (without backtracking).*

Proof. An OCSM-contract can move from the current task allocation to any other in a single step. Thus the optimal task allocation can be reached from any task allocation: there are no local optima. Clearly the move to an optimal task allocation is IR because an optimal task allocation has higher welfare than any other. Thus the hill-climbing algorithm does not need to backtrack in order to reach the optimum.

Because agents only make IR contracts, every contract (strictly) improves social welfare. Therefore, no task allocation is visited more than once. Because there are a finite number of tasks T and agents A, the number of task allocations ($|A|^{|T|}$) is finite. It follows that the global optimum is reached in a finite number of steps. □

OCSM-contracts are also necessary: no weaker set of contract types suffices—even if there were an oracle to choose the order in which to apply them:

Proposition 11 *If there is some OCSM-contract that the protocol does not allow, there are instances of the task allocation problem where no IR path exists from the initial allocation to the optimal one.*

Proof. If some OCSM-contract is missing, the task allocation graph contains (at least) two vertices that do not share an edge. Let the initial and optimal allocations be these two. The social welfares can be chosen so that in all vertices adjacent to the initial one, welfare is lower than in the initial one. Thus no IR path can lead away from the initial task allocation. □

Proposition 10 gives a powerful tool for problem instances where the number of possible task allocations is relatively small. On the other hand, for large problem instances, the number of contracts made before the optimal task allocation is reached may be impractically large—albeit finite. For example on a large-scale real-world distributed vehicle routing problem instance, TRACONET (Sandholm 1993) never reached even a local optimum even with just O-contracts—with multiple hours of negotiation time on five Unix machines. Another problem is that although any OCSM-contract can be represented in $O(|A|^2+|T|)$ space, the identification of welfare increasing contracts may be complex—especially in a distributed setting—because there are $\frac{v^2-v}{2} = \frac{|A|^{2|T|}-|A|^{|T|}}{2}$ possible OCSM-contracts, and the evaluation of just one contract requires each contract party to compute the cost of handling its current tasks and the tasks allocated to it via the contract.

With such large problem instances, one cannot expect to reach the global optimum. Instead, the contracting should occur as long as there is time, and then have a solution ready: the anytime character of our contracting scheme is more important. The presented marginal cost based (re)contracting method guarantees that the algorithm has a feasible solution even if terminated at any time (assuming that the initial solution was feasible). Furthermore, it guarantees that the worth of each agent's solution (payoffs received from others minus cost of handling tasks) increases monotonically. It follows that no agent is worse off after the negotiation than it was under the initial solution where it handled its own tasks. It also follows that the social welfare increases monotonically during the negotiation.

Conclusions

The task allocation graph is a useful tool for analyzing contracts among IR agents. We showed that traditional O-contracts always have a short path to the optimal task allocation but an IR path may not exist, or it may not be short. We also analyzed an algorithm for finding the shortest IR path, and showed that the graph becomes arbitrarily sparse as problem size increases.

Next we introduced cluster contracts, swaps, and multiagent contracts. Unlike O-contracts, these do not even guarantee that a non-IR path to the optimum exists. However, each of the four contract types avoids some local optima that the others do not.

Even if the contracting protocol is equipped with all four types, local optima exist. To attack this problem, we introduced OCSM-contracts which combine the ideas from the four earlier types into an atomic contract type. If the protocol is equipped with OCSM-contracts, any sequence of IR contracts leads to the globally optimal task allocation in a finite number of steps: an oracle—or speculation—is not needed for choosing the path (no subset of OCSM-contracts suffices even with an oracle).

This is a powerful result for small problem instances. For large ones, the number of contracts before the optimum is reached may be impractically large. Identifying profitable OCSM-contracts may also be complex. In such settings, the anytime feature of our multi-contract-type algorithm—with provably monotonic improvement of each agent's solution—is more important for finding satisficing solutions in reasonable time. One practical way of sequencing the contract types is to first apply simpler contracts, and use the more complex ones if a local optimum is reached and more negotiation time is available.

Future research includes studying restricted domains where the cost functions have some special structure. In such settings, the necessity results on the contract types do not apply since a weaker set of contract types could suffice.

The equivalent of a complex contract (C-, S-, M-, or any OCSM-contract) can be accomplished by a sequence of O-contracts if the agents are willing to take risks. Even if no O-contract is IR, the agents can sequentially make all the O-contracts that sum up to a more complex beneficial contract. Early in this sequence, the global solution can degrade until the later contracts enhance it. When making the early commitments, at least one of the agents has to risk taking a permanent loss in case the other agent(s) do not agree to the later contracts that are needed to make the sequence of contracts profitable. This risk can be reduced—as much as desired—by a leveled commitment contracting protocol where agents can back out of contracts by paying penalties (Sandholm & Lesser 1996). If the later contracts in the sequence do not happen, the agent can decommit from the earlier ones. Future work should more closely analyze how best to combine OCSM-contracts and leveled commitment contracts.

References

Andersson, M., and Sandholm, T. W. 1997. Contract types for optimal task allocation: II experimental results. Technical Report WUCS-97-36, Washington University, Department of Computer Science. To appear.

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

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

Fudenberg, D., and Tirole, J. 1991. *Game Theory.* MIT Press.

Gu, C., and Ishida, T. 1995. A quantitative analysis of the contract net protocol. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 449. In the poster collection.

Lin, S., and Kernighan, B. W. 1971. An effective heuristic procedure for the traveling salesman problem. *Operations Research* 21:498–516.

Mas-Colell, A.; Whinston, M.; and Green, J. R. 1995. *Microeconomic Theory.* Oxford University Press.

McAfee, R. P., and McMillan, J. 1996. Analyzing the airwaves auction. *Journal of Economic Perspectives* 10(1):159–175.

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

Sandholm, T. W., and Lesser, V. R. 1995. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, 328–335. Reprinted in Readings in Agents, Huhns and Singh, eds., pp. 66–73.

Sandholm, T. W., and Lesser, V. R. 1996. Advantages of a leveled commitment contracting protocol. In *Proceedings of the National Conference on Artificial Intelligence*, 126–133.

Sandholm, T. W. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the National Conference on Artificial Intelligence*, 256–262.

Sandholm, T. W. 1997. Necessary and sufficient contract types for optimal task allocation. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 87. Poster session abstracts.

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 of *Research Notes in Artificial Intelligence.* 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.

Waters, C. D. 1987. A solution procedure for the vehicle-scheduling problem based on iterative route improvement. *Journal of the Operational Research Society* 38(9):833–839.