Agents in Electronic Commerce: Component Technologies for Automated Negotiation and Coalition Formation

Tuomas Sandholm*
sandholm@cs.wustl.edu
Department of Computer Science
Washington University
St. Louis, MO 63130-4899
Phone: +1(314)935-4749
Fax: +1(314)935-7302

March 31, 1999, revised August 12, 1999.

Abstract

Automated negotiation and coalition formation among self-interested agents are playing an increasingly important role in electronic commerce. Such agents cannot be coordinated by externally imposing their strategies. Instead the interaction protocols have to be designed so that each agent is motivated to follow the strategies that the protocol designer wants it to follow. This paper reviews six component technologies that we have developed for making such interactions less manipulable and more efficient in terms of the computational processes and the outcomes:

1. OCSM-contracts in marginal cost based contracting,
2. leveled commitment contracts,
3. anytime coalition structure generation with worst case guarantees,
4. trading off computation cost against optimization quality within each coalition,
5. distributing search among insincere agents, and
6. unenforced contract execution.

Each of these technologies represents a different way of battling self-interest and combinatorial complexity simultaneously. This is a key battle when multiagent systems move into large-scale open settings.

*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 Introduction

Automated negotiation systems with self-interested agents are becoming increasingly important. One reason for this is the technology push of a growing standardized communication infrastructure—Internet, WWW, NII, EDI, KQML, FIPA, Concordia, Voyager, Odyssey, Telescript, Java, *etc*—over which separately designed agents belonging to different organizations can interact in an open environment in real-time and carry out transactions safely. The second reason is strong application pull for computer support for negotiation at the operative decision-making level [22, 10, 39]. For example, we are witnessing the advent of small transaction electronic commerce on the Internet for purchasing goods, information, and communication bandwidth. There is also an industrial trend toward virtual enterprises: dynamic alliances of small, agile enterprises that together can take advantage of economies of scale when available (e.g., respond to more diverse orders than individual agents can), but do not suffer from diseconomies of scale.

Multiagent technology facilitates such negotiation at the operative decision-making level. This automation can save labor time of human negotiators, but in addition, other savings are possible because computational agents can be more effective at finding beneficial short-term contracts than humans are in combinatorially and strategically complex settings. This is because computational agents can find, enumerate, and evaluate potential deals faster than humans, and because computational agents can be designed to act optimally on the users behalf based on game theoretic prescriptions that are often not easily comprehended by humans.

This paper discusses multiagent negotiation in situations where agents may have different goals, and each agent is trying to maximize its own good without concern for the global good. Such self-interest naturally prevails in negotiations among independent businesses or individuals. In building computer support for negotiation in such settings, the issue of self-interest has to be dealt with. In *cooperative distributed problem solving* [7, 5], the system designer imposes an interaction *protocol* [1] and a *strategy* (a mapping from history to action; a way to use the protocol) for each agent. The main question is what social outcomes follow given the protocol and *assuming that the agents use the imposed strategies.* On the other hand, in *multiagent systems* [30, 23, 18, 13], the agents are provided with an interaction protocol (aka. *mechanism*), but each agent will choose its own strategy. A self-interested agent will choose the best strategy for itself, which cannot be explicitly imposed from outside. Therefore, the protocols need to be designed using a *noncooperative, strategic* perspective: the main question is what social outcomes follow given a protocol that *guarantees that each agent's desired local strategy is best for that agent—and thus the agent will use it.* This approach is required in designing robust non-manipulable

[1] Here a protocol does not mean a low level communication protocol, but a negotiation protocol that determines the possible (valid, legal) actions that agents can take at different points of the interaction. The *sealed-bid first-price auction* is an example protocol, where each bidder is free to submit one bid for the item, which is awarded to the highest bidder at the price of his bid.

multiagent systems where the agents may be constructed by separate designers and/or may represent different real-world parties.

The rest of this paper overviews six component technologies that we have developed for such negotiations:

1. OCSM-contracts in marginal cost based contracting,
2. leveled commitment contracts,
3. anytime coalition structure generation with worst case guarantees,
4. trading off computation cost against optimization quality within each coalition,
5. distributing search among insincere agents, and
6. unenforced contract execution.

Each of these technologies is discussed at a high level, and pointers to the detailed technical papers on these topics are provided.

2 Technology 1: OCSM-contracts in marginal cost based contracting

A central part of automated negotiation systems is the ability to reallocate items (tasks, securities, bandwidth slices, megawatt hours of electricity, collectibles, *etc.*) among the agents. In many domains, significant savings can be achieved by reallocation. However, reallocation can be difficult if agents have preferences over combinations of items. Some tasks are inherently synergic, and should therefore be handled by the same agent. Some tasks have negative interactions, so it is better to allocate them to different agents. In other words, an agent's cost (and even feasibility) of handling a given task depends on what other tasks the agent will have. Furthermore, different agents may have different resources, and this leads to different capabilities and costs for handling tasks. This section discusses task allocation among self-interested agents in the following model that captures the above considerations. While we use the term "task", the items to be allocated can be anything else as well—financial securities, collectibles, resources, *etc.*—as long as the following model captures the setting.[2]

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*

[2]In settings such as securities reallocation where the items have positive value to each agent—unlike in task reallocation—the cost functions take on negative values.

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* $i \neq j$.[3,4]

The original contract net and many of its later variants lacked a formal model for making bidding and awarding decisions. More recently, we introduced such a formal model that gives rise to a negotiation protocol that provably leads to desirable task allocations among agents [22, 24, 25]. In that model, contracting decisions are based on marginal cost calculations, i.e., that model invokes 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.[5] This implies individual rationality of sequences of contracts.

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

$$MC^{add}(T^{contract}|T_q) = c_q(T^{contract} \cup T_q) - c_q(T_q)$$

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_q that the contractee has.[6]

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

$$MC^{remove}(T^{contract}|T_r) = c_r(T_r) - c_r(T_r - T^{contract}).$$

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.

[3] This definition generalizes what are called "Task Oriented Domains" [18]. 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 set of tasks. This is represented by a cost of infinity.

[4] Although a static version of the problem is discussed, the contracting scheme works even if tasks and resources (resources affect the cost functions) are added and removed dynamically.

[5] This differs from payoff maximizing agents of game theory [16]. 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 in the tree of possible future contracts. 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.

[6] Sometimes computing the value of the cost function for even a single task set is hard. For example, if the tasks are cities for a traveling salesman to visit, the computation is $\mathcal{NP}$-complete. Therefore, the marginal costs cannot actually be computed by subtracting two cost function values from each other in practice. Instead they have to be approximated [22, 24, 29].

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.

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 implementation based on marginal costs can be found in [22, 24, 29]. To our knowledge, this TRACONET (TRAnsportation COoperation NETwork) system was the first implementation of the contract net that used actual real-world marginal cost calculations as the basis of automated contracting [22, 20, 21]. Its scaling up was verified on large-scale real-world data from five independent dispatch centers.

2.1 Convergence to the globally optimal task allocation

In most contract net implementations, each contract concerns only one task, i.e. one task is moved from one agent to another against a payment [38, 34, 9]. Such an *original (O) contract* can be understood as a particular search operator in the global hill-climbing contracting algorithm that is used for task reallocation. When the contracting protocol is equipped with O-contracts only, it may get stuck in a local optimum where no contract is individually rational but the task allocation is not globally optimal.

To solve this problem, we recently introduced several new contract types: *cluster (C) contracts* [22, 20] where a set of tasks is atomically contracted from one agent to another against a payment, *swap (S) contracts* where a pair of agents swaps a pair of tasks (and potentially a sidepayment), and *multiagent (M) contracts* where more than two agents are involved in an atomic exchange of tasks (and potentially sidepayments) [25, 29, 24]. Each of the four contract types avoids some of the local optima that the other three do not:

Proposition 1 *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 [25].*

Unfortunately, even if the contracting protocol is equipped with all four of the contract types, the globally optimal task allocation may not be reached via IR contracts—even if there were an oracle for choosing the sequence of contracts:

Proposition 2 *There are instances of the task allocation problem where no IR sequence from the initial task allocation to the optimal one exists using O-, C-, S- and M-contracts [25].*

Clearly, no subset of the contract types suffices either. Another problem is that without an oracle, contracting may get stuck in a local optimum even if some IR sequence exists because the agents may choose some other IR sequence.

To address this shortcoming, we defined a new contract type, *OCSM-contract*, which 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 2 ([25, 24]) *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*.*

So OCSM contracts allow moving from a task allocation to any other task allocation with a single contract. It could be shown that an IR sequence always exists from any task allocation to the optimal one, if the contracting protocol incorporates OCSM-contracts. However, a stronger claim is now made. The following proposition states that OCSM-contracts are sufficient for reaching the globally optimal task allocation 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 sequence. 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 3 *Let* $|A|$ *and* $|T|$ *be finite. If the contracting protocol allows OCSM-contracts, any hill-climbing algorithm (i.e., any sequence of IR contracts) finds the globally optimal task allocation in a finite number of steps (without backtracking) [25, 24].*[7]

Proof. With OCSM-contracts there are no local optima (that are not global optima) since a global optimum can be reached from any task allocation in a single contract. This last contract will be IR, because moving to the optimum from some suboptimal allocation improves welfare, and this gain can be arbitrarily divided among the contract parties. Thus, the algorithm will not run out of IR contracts before the optimum has been reached. With finite $|A|$ and $|T|$, there are only a finite number of task allocations. Since the algorithm hill-climbs, no task allocation will be repeated. Therefore, the optimum is reached in a finite number of contracts. □

Proposition 3 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

[7] If the cost functions, $c_i(\cdot)$, have certain types of special structure, it can be guaranteed that the global optimum is reached even with less powerful contract types [25].

task allocation is reached may be impractically large—albeit finite. For example on a large-scale real-world distributed vehicle routing problem instance, the TRACONET contracting system never reached even a local optimum even with just O-contracts—with each of the five agents executing on its own Unix machine [22]. 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 $\frac{|A|^{2|T|} - |A|^{|T|}}{2}$ 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 in practice. Instead, the contracting should occur as long as there is time, and then have a solution ready: the anytime character of this contracting scheme becomes more important. See [3] for experimental results on the anytime characteristics of the different contract types.

3 Technology 2: Leveled commitment contracts

In traditional multiagent negotiation protocols among self-interested agents, once a contract is made, it is binding, i.e., neither party can back out [18, 22, 25, 3, 8, 13, 33, 6, 41]. Once an agent agrees to a contract, it has to follow through with it no matter how future events unravel. Although a contract may be profitable to an agent when viewed *ex ante*, it need not be profitable when viewed after some future events have occurred, i.e., *ex post*. For example, in business-to-business electronic commerce when a company has contracted to manufacture a component for another company, the former company may get a more profitable offer from another party, and may want to undo the earlier contract so as to be able to handle the latter. Similarly, a contract may have too low of an expected payoff *ex ante*, but in some realizations of the future events, the same contract may be desirable when viewed *ex post*. Normal full commitment contracts are unable to efficiently take advantage of the possibilities that such—probabilistically known—future events provide.

On the other hand, many multiagent systems consisting of cooperative agents incorporate some form of decommitment possibility in order to allow the agents to accommodate new events. For example, in the original contract net protocol, the agent that had contracted out a task could send a termination message to cancel the contract even when the contractee had already partially fulfilled the contract [38]. This was possible because the agents were not self-interested: the contractee did not mind losing part of its effort without a monetary compensation. Similarly, the role of decommitment possibilities among cooperative agents has been studied in meeting scheduling using a contracting approach [35]. Again, the agents did not require a monetary compensation for their efforts: an agent agreed to cancel a contract merely based on the fact that some other agent wanted to decommit. In such multiagent systems consisting of cooperative agents, each agent can be trusted to use such an externally imposed strategy

even though using that strategy might not be in the agent's self-interest.

Some research in game theory has focused on utilizing the potential provided by probabilistically known future events by *contingency contracts* among self-interested agents. The obligations of the contract are made contingent on future events. There are games in which this method provides an expected payoff increase to both parties of the contract compared to any full commitment contract [17]. Also, some deals are enabled by contingency contracts in the sense that there is no full commitment contract that both agents prefer over their fallback positions, but there is a contingency contract that each agent prefers over its fallback.

There are at least three problems regarding the use of contingency contracts in automated negotiation among self-interested agents. First, the agents might not know the entire space of possible future events. Even if the real-world party that the agent represents knows the possible events, programming that information into the agent can be prohibitively complex and error-prone. Second, contingency contracts get cumbersome as the number of relevant events to monitor from the future increases. In the limit, all domain events (changes in the domain problem, e.g., new tasks arriving or resources breaking down) and all negotiation events—contracts from other negotiations—can affect the value of the obligations of the original contract, and should therefore be conditioned on. Furthermore, these future events might not affect the value of the original contract independently: the value of the original contract may depend on combinations of future events [29, 22, 18]. Thus, there is a potential combinatorial explosion of events to be conditioned on. Third, verification of the unraveling of the events may not be viable. Sometimes an event is only observable by some of the agents. The observing agents might lie to the nonobserving agents about the event in case the event is associated with a disadvantageous contingency to the observing agents. Thus, to be viable, contingency contracts would require an event verification mechanism that is not manipulable and not prohibitively complicated or costly.

We devised *leveled commitment contracts* as another instrument for taking advantage of the possibilities provided by probabilistically known future events [30, 24]. Instead of conditioning the contract on future events, a mechanism is built into the contract that allows unilateral decommitting. This is achieved by specifying in the contract decommitment penalties, one for each agent. If an agent wants to decommit—i.e., to be freed from the obligations of the contract—it can do so simply by paying the decommitment penalty to the other party. Such contracts are called leveled commitment contracts, because the decommitment penalties can be used to choose a level of commitment. The method requires no explicit conditioning on future events: each agent can do its own conditioning dynamically. Therefore, no event verification mechanism is required either.

While the leveled commitment contracting protocol has intuitive appeal and several practical advantages [24], it is not obvious that it is beneficial. First, the breacher's gain may be smaller than the breach victim's loss. Second, agents might decommit insincerely. A truthful agent will decommit whenever its best

outside offer plus the decommitting penalty is better than the current contract. However, a rational self-interested agent will be more reluctant in decommitting. It will take into account the chance that the other party might decommit, in which case the former agent gets freed from the contract obligations, does not have to pay a decommitting penalty, and will collect a decommitting penalty from the other party. Based on the same reasoning, the other contract party will be reluctant to decommit as well. Due to such reluctant decommitting, contracts may end up being kept even though breaking them would be best from the social welfare perspective.

We analyzed this issue formally [30, 24]. A Nash equilibrium analysis was carried out where both contract parties' decommitting strategies (characterized by how good an agent's outside offer has to be to induce the agent to decommit) were best responses to each other. Both agents were decommitting insincerely, but neither was motivated to change the extent of his lie given that the other did not change. It was shown that even under such insincere decommitting, the leveled commitment protocol outperforms the full commitment protocol. First, it enables contracts by making them IR in settings where no full commitment contract is IR (the reverse cannot happen because leveled commitment contracts can emulate full commitment by setting the penalties high enough). Second, leveled commitment contracts increase both contract parties' expected payoffs over any full commitment contracts.

Recently we developed an algorithm for determining the optimal contracts [32]. The algorithm takes as input a piecewise linear probability distribution of the contractor's best future outside offer and a piecewise linear probability distribution of the contractee's best future outside offer. It outputs the range of optimal individually rational contract prices and the optimal penalties as a function of the contract price, i.e., the penalties that maximize the sum of the agents' expected payoffs. The optimization takes into account that rational agents decommit strategically in Nash equilibrium. The contract optimizer also solves for the Nash equilibria for any given contract, i.e., it determines how good each agent's outside offer has to be to trigger that agent to decommit. From this, the optimizer determines the decommitting probabilities. Using the algorithms, we provide a free client-server based contract optimizing service on the web (`http://ecommerce.cs.wustl.edu/contracts.html`) as part of *eMediator*, our next generation electronic commerce server. We invite the reader to try it.

Making multiple contracts sequentially introduces additional complications because a decommitment may motivate the victims to decommit from some of their other contracts. We have studied methods of increasing the decommitment penalties over time so as to reduce such cascade effects to an efficient level [1]. One of the key results is that infinite decommit-recommit loops cannot be avoided via any schedule of increasing the penalties if the timing is done locally from the time the contract was made. Instead, an element of global time (e.g., from the beginning of the entire negotiation) has to be used to avoid such loops. Finally, we have experimented with leveled commitment among agents that do lookahead into the future contracts vs. myopic agents that do not [2].

4 Technology 3: Anytime coalition structure generation with worst case guarantees

Coalition formation is another key issue in multiagent systems. By forming coalitions, i.e., coordinating their activities within each coalition, the agents can often reach considerable cost savings. In electronic commerce, coalition formation occurs, for example, in buyers pooling together to coordinate larger orders to obtain quantity discounts (such as at `www.accompany.com`), in sellers potentially maintaining cartel pricing, and in producers forming dynamic supply chains. As is often done [11, 43, 37, 12], this section discusses coalition formation in *characteristic function games*. In such games, each coalition S is associated with its value v_S. Coalition formation includes three activities:

1. *Coalition structure generation*: formation of coalitions by the agents such that agents within each coalition coordinate their activities, but agents do not coordinate between coalitions. Precisely this means partitioning the set of agents into exhaustive and disjoint coalitions. This partition is called a *coalition structure (CS)*. For example, in a game with three agents, there are seven possible coalitions: {1}, {2}, {3}, {1,2}, {2,3}, {3,1}, {1,2,3} and five possible coalition structures: {{1}, {2}, {3}}, {{1}, {2,3}}, {{2}, {1,3}}, {{3}, {1,2}}, {{1,2,3}}.

2. *Solving the optimization problem* of each coalition. This means pooling the tasks and resources of the agents in the coalition, and solving this joint problem. The coalition's objective is to maximize monetary value: money received from outside the system for accomplishing tasks minus the cost of using resources. (In some problems, not all tasks have to be handled. This can be incorporated by associating a cost with each omitted task.)

3. *Dividing the value* of the generated solution among agents. This value may be negative because agents incur costs for using their resources.

These activities may be interleaved, and they are not independent. For example, the coalition that an agent wants to join depends on the portion of the value that the agent would be allocated in each potential coalition.

4.1 Coalition structure generation

Classically, coalition formation research has mostly focused on the payoff division activity. Coalition structure generation and optimization within a coalition have not previously received as much attention. Research has focused [11, 43] on superadditive games, i.e., games where $v_{S \cup T} \geq v_S + v_T$ for all disjoint coalitions $S, T \subseteq A$. In such games, coalition structure generation is trivial because the agents are best off by forming the grand coalition where all agents operate together.

Superadditivity means that any pair of coalitions is best off by merging into one. Classically it is argued that almost all games are superadditive because, at

worst, the agents in a composite coalition can use solutions that they had when they were in separate coalitions.

However, many games are not superadditive because there is some cost to the coalition formation process itself. For example, there might be coordination overhead like communication costs, or possible anti-trust penalties. Similarly, solving the optimization problem of a composite coalition may be more complex than solving the optimization problems of component coalitions. Therefore, under costly computation, component coalitions may be better off by not forming the composite coalition [31]. Also, if time is limited, the agents may not have time to carry out the communications and computations required to coordinate effectively within a composite coalition, so component coalitions may be more advantageous.

In games that are not superadditive, some coalitions are best off merging while others are not. In such settings, the social welfare maximizing coalition structure varies, and coalition structure generation becomes highly nontrivial. The goal is to maximize the social welfare of the agents A by finding a coalition structure

$$CS^* = \arg \max_{CS \in \text{ partitions of A}} V(CS),$$

where

$$V(CS) = \sum_{S \in CS} v_S$$

The problem is that the number of coalition structures is large ($\omega(|A|^{|A|/2})$, see [27]), so not all coalition structures can be enumerated unless the number of agents is extremely small—in practice about 15 or fewer. Instead, one would like to search through a subset ($N \subset$ partitions of A) of coalition structures, and pick the best coalition structure seen so far:

$$CS^*_N = \arg \max_{CS \in N} V(CS)$$

Taking an outsider's view, the coalition structure generation process—e.g., a negotiation—can be viewed as search in a *coalition structure graph*, Figure 1. Now, how should such a graph be searched if there are too many nodes to search it completely?

One desideratum is to be able to guarantee that this coalition structure is within a worst case bound from optimal, i.e., that

$$k = \min\{\kappa\} \text{ where } \kappa \geq \frac{V(CS^*)}{V(CS^*_N)}$$

is finite, and as small as possible. Let us define n_{min} to be the smallest size of N that allows us to establish such a bound k.

We assume that each coalition's value is nonnegative ($v_S \geq 0$). However, if some coalitions' values are negative, but each coalition's value is bounded from below (i.e., not infinitely negative), one can normalize the coalition values by subtracting at least $\min_{S \subseteq A} v_S$ from all coalition values v_S. This rescales the coalition values so that $v_S \geq 0$ for all coalitions S. This rescaled game is strategically equivalent to the original game [11].

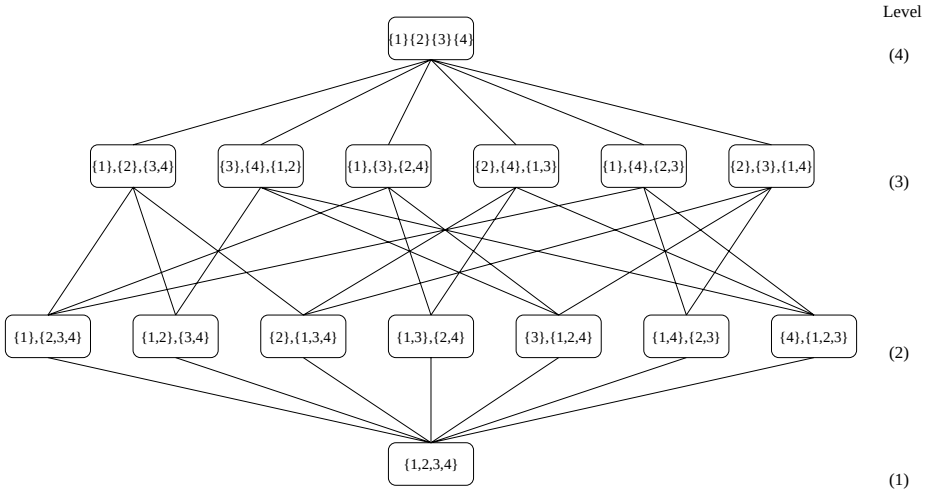

Figure 1: *Coalition structure graph for a 4-agent game. The nodes represent coalition structures. The arcs represent mergers of two coalition when followed downward, and splits of a coalition into two coalitions when followed upward.*

4.2 Minimal search to establish a bound

The following proposition establishes the minimal amount of search that is required to guarantee a solution that is within a bound from optimum:

Proposition 4 *To bound k, it suffices to search the lowest two levels of the coalition structure graph (Figure 1). With this search, the bound $k = |A|$, this bound is tight, and the number of nodes searched is $n = 2^{|A|-1}$. No other search algorithm (than the one that searches the bottom two levels) can establish a bound k while searching only $n = 2^{|A|-1}$ nodes or fewer [27].*

Interpreted positively, this means that—somewhat unintuitively—a worst case bound from optimum can be guaranteed without seeing all CSs. Moreover, as the number of agents grows, the fraction of coalition structures needed to be searched approaches zero, i.e., $\frac{n_{min}}{|\text{partitions of A}|} \to 0$ as $|A| \to \infty$. This is because the algorithm needs to see only $2^{|A|-1}$ coalition structures while the total number of coalition structures is $\omega(|A|^{|A|/2})$.

Interpreted negatively, the proposition shows that exponentially many coalition structures (in the number of agents)[8] have to be searched before a bound can be established. This may be prohibitively complex if the number of agents is large—albeit significantly better than attempting to enumerate all coalition structures. Viewed as a general impossibility result, the proposition states that no algorithm for coalition structure generation can establish a bound in general characteristic function games without trying at least $2^{|A|-1}$ coalition structures.[9]

[8] However, this search is linear in the number of possible coalitions.

[9] In restricted domains where the v_S values have special structure, it may be possible to establish a bound k with less search. Shehory and Kraus have analyzed coalition structure

This sheds light on earlier algorithms. Specifically, all prior coalition structure generation algorithms for general characteristic function games [37, 12]—which we know of—fail to establish such a bound. In other words, the coalition structure that they find may be arbitrarily far from optimal.

4.3 Lowering the bound via further search

The following algorithm will establish a bound in the minimal amount of search, and then rapidly reduce the bound further if there is time for more search. If the domain happens to be superadditive, the algorithm finds the optimal coalition structure immediately.

Algorithm. COALITION-STRUCTURE-SEARCH-1 [27]

1. Search the **bottom** two levels of the coalition structure graph.
2. Continue with a breadth-first search from the **top** of the graph as long as there is time left, or until the entire graph has been searched.
3. Return the coalition structure that has the highest welfare among those seen so far.

As was discussed earlier, before $2^{|A|-1}$ nodes have been searched, no bound can be established, and at $n = 2^{|A|-1}$ the bound $k = |A|$. By seeing just one additional node, i.e., the top node, the bound drops in half ($k = \frac{|A|}{2}$). Then, to drop k to about $\frac{|A|}{3}$, two more levels need to be searched. Roughly speaking, the divisor in the bound increases by one every time two more levels are searched (the exact drop of the bound is presented in [27]). So, the anytime phase (step 2) of COALITION-STRUCTURE-SEARCH-1 has the desirable feature that the bound drops rapidly early on, and there are overall diminishing returns to further search, Figure 2.

4.4 Comparison to other algorithms

All previous coalition structure generation algorithms for general characteristic function games [37, 12]—that we know of—fail to establish any worst case bound because they search fewer than 2^{a-1} coalition structures. Therefore, we compared COALITION-STRUCTURE-SEARCH-1 to two other obvious candidates:

- **Merging algorithm**, i.e., breadth first search from the top of the coalition structure graph. This algorithm cannot establish any bound before it has searched the entire graph [27].

generation in one such setting [36]. However, the bound that they compute is not a bound from optimum, but from a benchmark (best that is achievable given a preset limit on the size of coalitions), which itself may be arbitrarily far from optimum.

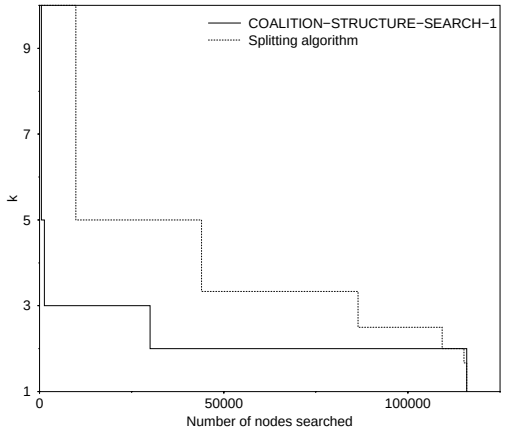

Figure 2: *Ratio bound k as a function of search size in a 10-agent game.*

- **Splitting algorithm**, i.e., breadth first search from the bottom of the graph. This is identical to COALITION-STRUCTURE-SEARCH-1 up to the point where 2^{a-1} nodes have been searched, and a bound $k = a$ has been established. After that, the splitting algorithm reduces the worst case bound much slower than COALITION-STRUCTURE-SEARCH-1 [27].

While that comparison was based on worst case performance, a recent paper compares the average case performance of these three algorithms experimentally using four different ways of choosing the coalition structure values [15]. All of the algorithms performed orders of magnitude better than their worst case. While each of the algorithms dominated the others in different settings, COALITION-STRUCTURE-SEARCH-1 performed the most consistently across settings, and its performance was close to that of the best out of the three algorithms in each of the four settings.

4.5 Variants of the coalition structure generation problem

One would like to construct an anytime algorithm that establishes a lower k for any amount of search n, compared to any other anytime algorithm. However, such an algorithm might not exist. It is conceivable that the search which establishes the minimal k while searching n' nodes ($n' > n$) does not include all nodes of the search that establishes the minimal k while searching n nodes. This hypothesis is supported by the fact that the curves in Figure 2 cross in the end. However, this is not conclusive because COALITION-STRUCTURE-SEARCH-1 might not be the optimal anytime algorithm, and because the bad cases for the splitting algorithm might not be the worst cases.

If it turns out that no anytime algorithm is best for all n, one could use information (e.g., exact, probabilistic, or bounds) about the termination time

to construct a *design-to-time algorithm* that establishes the lowest possible k for the specified amount of search.

So far we have discussed algorithms that have an *off-line search control* policy, i.e., the nodes to be searched have to be selected without using information accrued from the search so far. With *on-line search control*, one could perhaps establish a lower k with less search, because the search can be redirected based on the values observed in the nodes so far. With on-line search control, it makes a difference whether the search observes only values of coalition structures, $V(CS)$, or values of individual coalitions, v_S, in those structures. The latter gives more information, and in such settings, algorithms that capitalize on that information can be used [27, 26]. For example, if a value, v_S, is known for every coalition $S \subseteq A$, then the optimal coalition structure can be computed in $O(3^{|A|})$ time using dynamic programming [27].

None of these variants (anytime vs. design-to-time, and off-line vs. on-line search control) would affect the result that searching the bottom two levels of the coalition structure graph is the unique minimal way to establish a worst case bound, and that the bound is tight. However, the results on searching further might vary in these different settings.

5 Technology 4: Trading off computation cost against optimization quality within each coalition

Under unlimited and costless computation, each coalition would solve its optimization problem exactly, which would define the value, v_S, of that coalition. However, in many practical domains it is too complex from a combinatorial viewpoint to solve the problem exactly. Instead, only an approximate solution can be found. In such settings, self-interested agents would want to strike the optimal tradeoff between solution quality and the cost of the associated computation.

We address this issue [31] by adopting a specific model of bounded rationality where each agent has to pay for the computational resources that it uses for deliberation. A fixed computation cost $c_{comp} \geq 0$ per computation time unit is assumed. The domain cost associated with coalition S is denoted by $c_S(r_S) \geq 0$, i.e., it depends on (decreases with) the allocated computation resources r_S, Figure 3 left. For example, in a vehicle routing problem, the domain cost is the sum of the lengths of the routes of the coalition's vehicles.[10] The functions $c_S(r_S)$ can be viewed as *performance profiles* [4, 42] of the problem solving algorithm. They are used to decide how much time to allocate to each computation. With this model of bounded rationality, the value of a coalition with bounded-rational agents can be defined. Each coalition minimizes the sum of solution

[10] In games where the agents receive revenue from outside—e.g., for handling tasks—this revenue can be incorporated into $c_S(r_S)$ by subtracting the coalition members' revenues from the coalition's domain cost.

cost (i.e., domain cost, which decreases as more computation is allocated) and computation cost (which increases as more computation is allocated):

$$v_S(c_{comp}) = -\min_{r_S}[c_S(r_S) + c_{comp} \cdot r_S]. \quad (1)$$

This coalition value decreases as the computation time unit cost c_{comp} increases, Figure 3 right. Intuitively, as the unit cost of computation increases, agents

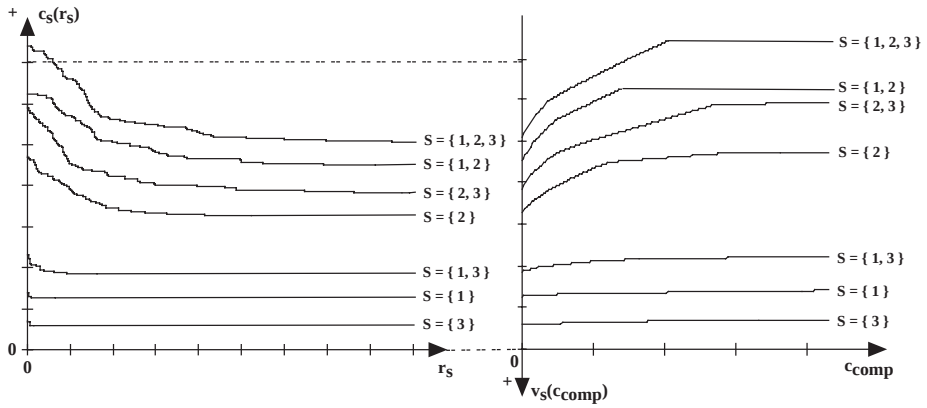

Figure 3: *Example experiment from a vehicle routing domain with agents 1, 2, and 3. Left: performance profiles, i.e., solution cost as a function of allocated computation resources. The curves become flat when the algorithm has reached a local optimum. Right: bounded-rational coalition value as a function of computation unit cost. The value of each coalition is negative because the cost is positive. The curves become flat at a computation unit cost c_{comp} that is so high that it is not worthwhile to take any iterative refinement steps: the initial solutions are used (their computation requirements are assumed negligible).*

need to pay more for the computation or they have to use less computation and acquire worse solutions accordingly. Our model also incorporates a second form of bounded rationality: the base algorithm may be incomplete, i.e., it might never find the optimal solution. If the base algorithm is complete, the bounded-rational value of a coalition when $c_{comp} = 0$ equals the rational value ($v_S(0) = v_S^R$). In all, the bounded-rational value of a coalition is determined by three factors:

- The *domain problem*: tasks and resources of the agents (e.g., trucks and delivery orders in a vehicle routing problem). Among rational agents this is the only determining factor.
- The *execution architecture* on which the problem solving algorithm is run. Specifically, the architecture determines the unit cost of computation, c_{comp}.
- The *problem solving algorithm*. Once the coalition formation game begins, the algorithm's performance profiles are considered fixed. This model in-

corporates the possibility that agents design different algorithms for different possible allocations of computation resources. We make no assumptions as to how effectively the algorithm uses the execution architecture. This is realistic because in practice it is often hard to construct algorithms that use the architecture optimally. For example, Russell and Subramanian have devised algorithms that are optimal for the architecture in simple settings, but in more complex settings they had to resort to an asymptotic criterion of optimality [19].

From our model of bounded rationality, the social welfare maximizing coalition structure can be determined. Similarly, the stability of the coalition structure can be determined: can the payoff be divided so that no group of agents gets higher payoff by moving out of the coalition structure by forming their won coalition? To avoid studying coalition games on a case by case basis, we have theoretically shown classes of performance profiles for which the welfare maximizing coalition structure and its stability can be determined directly without using Equation 1 and enumerating all possible coalition structures [31].

We have also experimented with our model of bounded rationality in a real-world vehicle routing problem. The main findings were the following. First, computational cost often does away with superadditivity, so it is no longer the case that every pair of coalitions is best off merging—which would imply optimality of the grand coalition. This is because the optimization problem of the composite coalition is significantly harder than the optimization problems of the component coalitions. Second, stability of the coalition structure is very sensitive to the problem instance, and varies in practice. Third, the coalition structure that our normative theory of bounded rational agents prescribes is closer to what human agents would choose based on domain specific considerations (such as adjacency of the dispatch centers and combinability of their loads) than is the classical normative prescription for agents whose rationality is unlimited [31].

This work on coalition formation under costly optimization within each coalition can be tied together with the nonexhaustive search for a welfare maximizing coalition structure (Section 4). The coalition structure generation algorithm can be used to search for a coalition structure, and only afterwards would the coalitions in the chosen coalition structure actually attack their optimization problems. If the performance profiles include uncertainty, this separation of coalition structure generation and optimization does not work e.g., because an agent may want to redecide its membership if its original coalition receives a worse optimization solution than expected. Recently, we have also studied coalition formation in conjunction with belief revision among bounded rational agents [40].

6 Technology 5: Distributing search among insincere agents

This section discusses a method of distributing any given search algorithm among self-interested agents. Distribution of search may be desirable because the search can be done more efficiently in parallel, and the agents will share the burden of computation. The method assumes that each agent has the information required to search the part of the space allocated to it.

As an example, this method can be used to distribute the algorithm COALITION-STRUCTURE-SEARCH-1. Self-interested agents prefer greater personal payoffs, so they will search for coalition structures that maximize personal payoffs, ignoring the ratio bound, k. The following algorithm can be used to motivate self-interested agents to exactly follow the socially desirable search. The randomizations in that algorithm can be done without a trusted third party by using a distributed nonmanipulable protocol for randomly permuting the agents [43].

Algorithm. DISTRIBUTED SEARCH FOR SELF-INTERESTED AGENTS

1. **Deciding what part of the coalition structure graph to search**. This can be decided in advance, or be dictated by a central authority or a randomly chosen agent, or be decided using some form of negotiation.

2. **Partitioning the search space among agents**. Each agent is assigned some part of the coalition structure graph to search. The enforcement mechanism in step 4 will motivate the agents to search exactly what they are assigned, no matter how unfairly the assignment is done. One way of achieving *ex ante* fairness is to randomly allocate the set search space portions to the agents. In this way, each agent searches equally on an expected value basis, although *ex post*, some may search more than others. Another option is to distribute the space equally among agents, or have some agents pay others to compensate for unequal amounts of search.

3. **Actual search**. Each agent searches its part of the search space, and tells the others which CS maximized $V(CS)$ in its search space.

4. **Enforcement**. Two agents, i and j, will be selected at random. Agent i will re-search the search space of j to verify that j has performed its search. Agent j gets caught of mis-searching (or misrepresenting) if i finds a better CS in j's space than j reported (or i sees that the CS that j reported does not belong to j's space at all). If j gets caught, it has to pay a penalty P. To motivate i to conduct this additional search, we make i the claimant of P. There is no pure strategy Nash equilibrium in this protocol. (If i searches and the penalty is high enough, then j is motivated to search sincerely. But then i is not motivated to search

since it cannot receive P.) Instead, there may be a mixed strategy Bayes-Nash equilibrium where i and j search truthfully with some probabilities. By increasing P, the probability that j searches can be made close to one. The probability that i searches goes close to zero, which minimizes enforcement overhead.

5. **Additional search**. The previous steps can be repeated if more time to search remains. For example, the agents could first do step 1 of COALITION-STRUCTURE-SEARCH-1. Then, they could repeatedly search more and more as time allows.

6. **Payoff division**. Many alternative methods for payoff division among agents could be used here. The only concern is that the division of $V(CS)$ may affect what CS an agent wants to report as a result of its search, since different CSs may give the agent different payoffs—depending on the payoff division scheme. However, by making P high enough compared to the $V(CS)$ values, this consideration can be made negligible compared to the risk of getting caught.

The method above is applicable in settings without a trusted third party. If a trusted third party exists, that party can be the one that conducts the re-search. This has the advantage that the re-searching party is not attempting to avoid search, but will conduct the re-search sincerely independent of the others' strategies. In such a mechanism, every agent's best response is to search sincerely if P is high enough.

7 Technology 6: Unenforced contract execution

After negotiation, the deals need to be executed. In conventional commerce, deals are usually enforced by law. For example, if a car dealership does not deliver the automobile after the customer has paid for it, the customer can resort to litigation. However, such enforced protocols are problematic in electronic commerce, e.g., over the Internet. First, adequate laws for electronic commerce may be lacking, or the transacting agents (human or computational) may be governed by different laws, e.g., they may be sited in different countries. Also, the laws might not be strictly enforced, or enforcing them—e.g., by litigation—might be impractically expensive. We would like the agents' electronic commerce transactions to work properly independent of such fluctuations in enforcement. Secondly, an electronic commerce party may vanish at any point in time, e.g., by logging out. Thus, the laws cannot be enforced unless the vanished agent represented some real-world party and the connection between the agent and the accountable real-world party can be traced.

Current electronic commerce technology is based on such enforced transactions. The problems of traceability and trust are being tackled, for example, by establishing trusted third parties like banks, credit card companies, and escrow intermediaries for electronic commerce, as well as by attempting to build

cybercommunities of trust. The developing infrastructure for electronic commerce among computational agents is also following the approach of enforced traceable transactions. For example, Telescript technology strives to strictly and accountably tie each computational agent to the real-world party that it represents.

Instead, we present a method that allows transactions to be carried out without enforcement. This enables transactions in settings where the parties cannot identify each other, or where litigation is not viable. From the perspective of computational agents, it allows the agents to be more autonomous because they do not have to be strictly tied to the real-world parties that they represent. In cases where this type of unenforced exchange is possible, it is preferable to the strictly enforced mode of exchange due to savings in enforcement costs (e.g., litigation costs, or operation costs of trusted third party intermediaries) and insensitivity to enforcement uncertainty.

The fulfillment of a mutual contract can be viewed as one agent delivering and the other agent paying, in money or some commodity. We propose a method for carrying out such an exchange without enforcement. The exchange is managed so that for both agents—supplier and demander—at any point in the exchange, the future gains from carrying out the rest of the exchange are larger than the gains from terminating the exchange prematurely by vanishing. For example, vanishing may be beneficial to a demander agent if the supplier agent has delivered much more than what the demander has yet paid for.

By intelligently splitting the exchange into smaller chunks, the agents can avoid situations where at least one of them is motivated to vanish. In other words, each agent only delivers a portion of its deliverables at a time. At the next step, the agents deliver some more, *etc.* The method is most suitable for settings where dividing the goods into chunks is relatively inexpensive, such as is often the case for example with information goods and computational services. We will call a sequence of deliveries and payments *safe* if neither agent is motivated to vanish at any point in the exchange. Specifically, the exchange is safe if it can be carried to completion according to a game theoretic solution concept called subgame perfect Nash equilibrium.

Some chunkings allow safe exchange while others do not. We devised algorithms that find a safe chunking if one exists for any given exchange [24, 28]. The sequence of delivering the chunks matters as well: some sequences are safe while others are not. The obvious candidate algorithms for sequencing fail to guarantee safety of the sequence. We devised a nontrivial sequencing algorithm that provably finds a safe sequence if one exists, and always terminates in quadratic time in the number of chunks. The algorithm works for settings where agents value each chunk independently. If the chunks are interdependent in value, the sequencing cannot be done in polynomial time in general, but dynamic programming can be used to carry out the sequencing significantly faster than by trying all sequences.

8 Conclusions

Multiagent systems consisting of self-interested agents are becoming ubiquitous; automated negotiation and coalition formation are playing an increasingly important role in electronic commerce. Such agents cannot be coordinated by externally imposing the agent's strategies. Instead the interaction protocols have to be designed so that each agent is motivated to follow the strategies that the protocol designer wants it to follow.

This paper reviewed six component technologies that we have developed for making such interactions less manipulable and more efficient in terms of the computational processes and the outcomes:

1. Marginal cost based contracting and OCSM-contracts. Marginal cost based contracting is an anytime reallocation scheme where every agent's utility improves monotonically over time, and agents and goods/tasks can arrive dynamically. The combinatorial contract types avoid local optima in the search for desirable allocations.

2. Leveled commitment contracts. Backtracking is a well-known method for avoiding local optima and accommodating new events in single agent settings. In multiagent systems consisting of self-interested agents, backtracking is difficult to implement. Leveled commitment contracts are a backtracking scheme for such negotiation settings, with provably desirable properties despite strategic breaching.

3. Anytime coalition structure generation with worst case guarantees. The scheme finds coalition structures that are provably within a bound from optimum in the minimal search time, and then improves the bound further via additional search. The intuitive approach of starting coalition negotiations from all agents operating individually, and then negotiating mergers, is highly inefficient from a worst case perspective. Instead, the negotiation should start from all agents in a grand coalition, and then trying all splits of the grand coalition into exactly two coalitions. After that, it is desirable to move to the stage where agents operate separately, and begin to negotiate mergers.

4. Trading off computation cost against optimization quality within each coalition. This technique uses a quantitative model of bounded rationality to normatively prescribe which coalitions should form, and how the value should be divided among the agents. In general, as computerized agents become more common in electronic commerce, theories of how to optimally use each agent's limited computational resources will become crucial.

5. Distributing search among insincere agents. This is a general method for implementing parallelization among self-interested parties that otherwise might avoid some of the search effort that they are assigned. This technique could be used for solving key combinatorial problems in electronic commerce, such as coalition structure generation, and winner determination in combinatorial auctions [26].

6. Unenforced contract execution. By splitting an exchange into chunks and by appropriately sequencing the chunks, divisible goods can be exchanged safely without enforcement under certain conditions. This disintermediates electronic commerce because the exchange will not rely on a third party escrow company.

In microeconomics and game theory, substantial knowledge exists of impossibility results and of constructive possibility demonstrations of interaction protocols and strategies for self-interested agents [16, 14]. However, the computational limitations of the agents deserve more attention. It is clear that such limitations have fundamental impact on what strategies agents want to use, and therefore also on what protocols are desirable, and what is (im)possible. This is one area where microeconomics and computer science fruitfully blend.

In the future, systems will increasingly be designed, built, and operated in a distributed manner. A larger number of systems will be used by multiple real-world parties. The problem of coordinating these parties and avoiding manipulation cannot be tackled by technological or economic methods alone. Instead, the successful solutions are likely to emerge from a deep understanding and careful hybridization of both.

Acknowledgments

This paper is an overview of several more detailed articles. Some of them are joint work with coauthors: Kate Larson, Martin Andersson, Sandeep Sikka, Samphel Norden, Victor Lesser, Onn Shehory, and Fernando Tohmé.

References

[1] Martin R Andersson and Tuomas 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] Martin R Andersson and Tuomas W Sandholm. Leveled commitment contracts with myopic and strategic agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 38–45, Madison, WI, July 1998.

[3] Martin R Andersson and Tuomas 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.

[4] Mark Boddy and Thomas Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285, 1994.

[5] Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[6] John Q. Cheng and Michael P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

[7] E Durfee, V Lesser, and D Corkill. Cooperative distributed problem solving. In A Barr, P Cohen, and E Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume IV, pages 83–147. Addison Wesley, 1989.

[8] Eithan Ephrati and Jeffrey S Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 173–178, Anaheim, CA, 1991.

[9] Cheng Gu and Toru Ishida. Analyzing the social behavior of contract net protocol. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away; MAAMAW'96, Lecture Notes in Artificial Intelligence 1038, Springer-Verlag*, pages 116–127, 1996.

[10] N Jennings, P Faratin, T Norman, P O'Brien, B Odgers, and J Alty. Implementing a business process management system using ADEPT: A real-world case study. *International Journal of Applied Artificial Intelligence*, 1999. To appear.

[11] James P Kahan and Amnon Rapoport. *Theories of Coalition Formation*. Lawrence Erlbaum Associates Publishers, 1984.

[12] Steven Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 414–419, Seattle, WA, July 1994.

[13] Sarit Kraus, Jonathan Wilkenfeld, and Gilad Zlotkin. Multiagent negotiation under time constraints. *Artificial Intelligence*, 75:297–345, 1995.

[14] David M Kreps. *A Course in Microeconomic Theory*. Princeton University Press, 1990.

[15] Kate S. Larson and Tuomas W. Sandholm. Anytime coalition structure generation: An average case study. In *Proceedings of the Third International Conference on Autonomous Agents (AGENTS)*, pages 40–47, Seattle, WA, May 1999.

[16] Andreu Mas-Colell, Michael Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

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

[18] Jeffrey S Rosenschein and Gilad Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, 1994.

[19] Stuart Russell and Devika Subramanian. Provably bounded-optimal agents. *Journal of Artificial Intelligence Research*, 1:1–36, 1995.

[20] Tuomas W Sandholm. A strategy for decreasing the total transportation costs among area-distributed transportation centers. In *Nordic Operations Analysis in Cooperation (NOAS): OR in Business*, Turku School of Economics, Finland, 1991.

[21] Tuomas W Sandholm. Automatic cooperation of area-distributed dispatch centers in vehicle routing. In *International Conference on Artificial Intelligence Applications in Transportation Engineering*, pages 449–467, San Buenaventura, CA, 1992.

[22] Tuomas 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.

[23] Tuomas 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, December 1996.

[24] Tuomas 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.

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

[26] Tuomas W Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 542–547, Stockholm, Sweden, 1999. Extended version: Washington University, Department of Computer Science technical report WUCS-99-01, January.

[27] Tuomas W Sandholm, Kate S Larson, Martin R Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 1999. To appear. Early version appeared at the National Conference on Artificial Intelligence (AAAI), pages 46–53, 1998.

[28] Tuomas W Sandholm and Victor R Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 694–701, Montreal, Canada, August 1995.

[29] Tuomas W Sandholm and Victor R Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS)*, pages 328–335, San Francisco, CA, June 1995. Reprinted in *Readings in Agents*, Huhns and Singh, eds., pp. 66–73, 1997.

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

[31] Tuomas W Sandholm and Victor R Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence*, 94(1):99–137, 1997. Special issue on Economic Principles of Multiagent Systems. Early version appeared at the International Joint Conference on Artificial Intelligence, pages 662–669, 1995.

[32] Tuomas W Sandholm, Sandeep Sikka, and Samphel Norden. Algorithms for optimizing leveled commitment contracts. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 535–540, Stockholm, Sweden, 1999. Extended version: Washington University, Department of Computer Science technical report WUCS-99-02.

[33] Tuomas W Sandholm and Fredrik Ygge. On the gains and losses of speculation in equilibrium markets. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 632–638, Nagoya, Japan, August 1997.

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

[35] Sandip Sen and Edmund Durfee. The role of commitment in cooperative negotiation. *International Journal on Intelligent Cooperative Information Systems*, 3(1):67–81, 1994.

[36] Onn Shehory and Sarit Kraus. Task allocation via coalition formation among autonomous agents. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 655–661, Montreal, Canada, August 1995.

[37] Onn Shehory and Sarit Kraus. A kernel-oriented model for coalition-formation in general environments: Implemetation and results. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 134–140, Portland, OR, August 1996.

[38] Reid 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, December 1980.

[39] Katia Sycara. Multi-agent compromise via negotiation. In M. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence, Volume II*. Pittman Publishing Ltd and Morgan Kaufmann, 1989.

[40] Fernando Tohmé and Tuomas W Sandholm. Coalition formation processes with belief revision among bounded rational self-interested agents. *Journal of Logic and Computation*, 9(97–63):1–23, 1999. An early version appeared in the IJCAI-97 Workshop on Social Interaction and Communityware, pp. 43–51, Nagoya, Japan.

[41] Michael Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, 1:1–23, 1993.

[42] Shlomo Zilberstein and Stuart Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1–2):181–213, 1996.

[43] Gilad Zlotkin and Jeffrey S Rosenschein. Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 432–437, Seattle, WA, July 1994.