On Automated Contracting in Multi-enterprise Manufacturing

Tuomas W. Sandholm and Victor R. Lesser *
{sandholm, lesser}@cs.umass.edu
University of Massachusetts at Amherst
Computer Science Department
Amherst, MA 01003

Abstract

So far, most distributed scheduling systems have been designed for cooperative agents, and are inappropriate for self-interested agents, which exist for example in inter-firm interactions such as virtual enterprises. This paper discusses issues that arise in extending the Contract Net Protocol to operate among such self-interested agents whose rationality is bounded by combinatorial complexity. The new protocol allows varying the stage of commitment for flexibility in the contracting process. It also allows varying levels of commitment; associated advantages are reviewed. Next, the implications of bounded rationality are discussed. Tradeoffs between allocated computation and negotiation benefits and risk are enumerated, and the necessity of explicit local deliberation control is substantiated. The Vickrey auction mechanism for truth-promotion and counter-speculation reduction is shown insufficient for bounded rational agents. Finally, a negotiation termination mechanism is discussed that guarantees finding a local optimum in distributed iterative task allocation among certain agent types.

1 Introduction

There are three main types of distributed manufacturing environments based on the types of agents that constitute the system. We will call these agent types *cooperative*, *self-interested (SI)*, and *hostile*. Cooperative agents attempt to maximize social welfare, which is the sum of the agents' utilities. They are willing to take individual losses in service of the good of the society of agents. As an example, different production cells within an enterprise should act as cooperative agents. They should attempt to minimize production costs and maximize the revenues of the company as a whole—sometimes accepting local losses in order to facilitate production at other cells.[1] In *multi-enterprise manufacturing*, individual companies join together to form a so called *virtual enterprise*, which will take care of production tasks—usually more economically than the companies could when operating individually. In a virtual enterprise, each individual company is usually a self-interested agent: it wants to maximize its own profit while not caring about the other companies' profits within the virtual enterprise. In such cases, an agent is willing to accommodate other agents' tasks only for a (monetary) compensation. The third type of agent relationship that occurs in distributed manufacturing is hostile. As an example, one can consider companies that compete against each other in the same market. In such a setting the company agents can be viewed as maximizing their utility which increases with their own gains but also with other companies' losses. Even if an agent is self-interested on a strategic level, it may be in its interest to act in a hostile manner in operative level distributed production scheduling, e.g. to attempt to drive competitors out of business. Working together towards coordinated distributed scheduling seems least

*This research was supported by ARPA under ONR contract N00014-92-J-1698 and ONR contract N00014-92-J-1450. The content does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

[1]However, when cells involve humans, individual cells often need incentives to motivate them to perform efficiently locally. Such incentives can usually only be implemented through local evaluation, which often introduce a disparity between the cell's local goals and the company's goals.

fruitful and least feasible in the hostile setting. In many situations, the distributed manufacturing environment actually comprises of agents from more than one of the classes. For example, while the setting between companies is self-interested, the cell-wise distribution within each company is simultaneously cooperative.

Up to date, automated distributed production planning and scheduling systems have been almost exclusively developed to operate among cooperative agents [18, 7, 1, 16]. Some reasons for distributing such systems have been presented. First there is the bandwidth argument: information is mostly sustained at the local level and this saves communication costs. Detailed information regarding tasks, resources and their dynamic state is usually not transmitted between agents; only abstractions (meta-information) such as load profiles of each resource class of each agent are sent [18, 7]. This bandwidth argument originates mostly from the Cooperative Distributed Problem Solving work of Lesser et al. (see e.g. [6]), which was done in an interpretation domain. In such a domain, large amounts of low level information arrive through sensors and it may really be infeasible to transmit that information to a central location to be processed. On the other hand, in planning and scheduling domains, the problem can often (not always because some schedulers can use large databases, for example of prior operation statistics) be relatively succinctly described and transmitted, and the real complexity mainly stems from inherent combinatorics. Another problem with this line of argument is that search efficiency is usually reduced as information of the global problem becomes less precise. Moreover, the negotiation communication that is required for distributed search may actually exceed the amount of communication that would occur if all the information were just gathered to a central problem solver: often distributed problem solving messages have to be sent multiple times, e.g. due to backtracking. Distribution has also been argued for by claiming that parallelization increases problem solving efficiency. This holds for (nearly) decomposable hierarchical systems, but tightly coupled domains—including many scheduling domains—are not easily decomposable to independent subproblems that would facilitate efficient parallelization. Also, the real world natural distribution of a scheduling problem seldom equals the distribution that would be computationally most efficient. To gain from parallelization, search has to be implemented asynchronously between the agents, and this usually involves giving up desirable algorithm properties such as completeness [18]. Another issue supporting distribution in cooperative settings is reactivity. Supposedly agents can locally react to local changes faster than a centralized system could. To a certain extent this is true, but the communication time to and from a central site is often negligible compared to the rate of change in the physical domain. Distribution in cooperative domains has also been advocated using a decision autonomy argument: the agents prefer to maintain the possibility to make local decisions and not to submit to centrally made ones. The problem with this argument is that cooperative agents have the same goals and therefore they should be indifferent regarding the question of which one of them makes the decisions. Finally, distributed systems have been argued for by pointing out that they are more robust against failure at a single point than centralized systems. To date, the arguments in favor of distributing cooperative scheduling systems have mostly been weak: often the problem could be solved by keeping a centralized problem solver up to date about the distributed events and having it do the planning and scheduling.

Among self-interested agents the very self-interest introduces an inherent distribution into the domain. Agents do want to maintain local decision autonomy because they have private goals. Moreover, they do not necessarily pass information truthfully—such an agent lies whenever it benefits from doing so. An agent may reveal only some of its tasks or resources or it might untruthfully reveal tasks or resources that do not exist. Rosenschein and Zlotkin [9] have formally analyzed the issue of task revelation. Unfortunately their analysis does not apply to most scheduling domains because it assumes that each agent has sufficient resources to potentially handle all of the tasks of all agents, and that the agents have symmetric costs for handling tasks. As to other forms of untruthful behavior, an agent can lie about the cost that it is willing to pay to another agent for taking care of some of its tasks, or about the cost that it requires in order to accept some of the other agents' tasks. Ephrati [2] presented an initial approach to solving the problem of exaggerating agents using the Clarke tax voting mechanism in a meeting scheduling domain. Several unresolved issues remain concerning this problem, for example the applicability of the proposed mechanism to a sequence of decisions and to dynamic domains with different expectations of changing tasks and resources. Even further forms of lying exist, e.g. an agent can declare false load profiles on resources or otherwise lie about its dynamic status. Because synergy savings from joint problem solving are often available among scheduling agents and virtual enterprises are becoming a reality, and because self-interested agents are inherently distributed, support mechanisms for distributed planning and scheduling among self-interested agents are clearly called for. The developing inter-enterprise communication infrastructure (EDI, NII, KQML [3], Telescript [4] etc.) provides part of the appropriate technology push for distributed planning and scheduling applications. Such automated systems have to take into account the issues that arise from self-interest: an agent will take action only for a payment, and an agent may lie. Therefore most of the techniques developed for cooperative distributed scheduling are inappropriate for the self-interested setting.

This paper reviews our results and ongoing research on the implications of performing automated negotiations where agents are *self-interested* (SI) and must make negotiation decisions in real-time with *bounded or costly computation resources*. Such *bounded rationality* of the agents further complicates distributed scheduling: local decisions are suboptimal due to the inability to precisely compute the value associated with accepting a task. We cast such negotiations in the following framework. Each agent has a (possibly empty) set of tasks and a (possibly empty) set of resources it can use to handle tasks. These sets change due to domain events, e.g. new tasks arriving or resources breaking down. The agents can subcontract tasks to other agents by paying a compensation. This subcontracting process can involve breaking a task into a number of subtasks handled by different agents, or clustering a number of tasks into a supertask. A task transfer is profitable from the global perspective if the contractee can handle the task less expensively than the contractor, or if the contractor cannot handle it at all, but the contractee can. So, the problem has two levels: a *global task allocation problem*, and each agent's *local combinatorial optimization problem* defined by the agent's current tasks and resources. The goal of each agent is to maximize its payoff which is defined as its income minus its costs. Income is received for handling tasks, and costs are incurred by using resources to handle the tasks. We restrict ourselves to domains where the feasibility and cost of handling a task do not depend on what other agents do with their resources (the control of no resource is shared) or how they divide tasks among themselves, but do depend on the other tasks that the agent has—as is usually the case in scheduling. The global solution can be evaluated from a social welfare viewpoint according to the sum of the agents' payoffs.

The original contract net protocol (CNP) [17] did not explicitly deal with issues arising from self-interest and bounded rationality. A first step towards extending the CNP to deal with these issues was the work on TRACONET [10]. It provided a formal model for bounded rational (BR) self-interested agents to make asynchronously announcing, bidding and awarding decisions. It used a simple static approximation scheme for *marginal cost* [2] calculation to make these decisions. Each agent's contracting decisions are based solely on these marginal cost estimates. The monetary payment mechanism allows quantitative tradeoffs between alternatives in an agent's negotiation strategy. Within DAI, bounded rationality (approximate processing) has been studied with cooperative agents, but among SI agents, perfect rationality has been widely assumed, e.g. [9, 2]. We argue that in most real multiagent applications, resource-bounded computation will be an issue, and that bounded rationality has profound implications on both negotiation protocols and strategies, see e.g. [13]. Although the work on TRACONET was a first step towards this end, it is necessary to extend in significant ways the CNP in order for BRSI agents to deal intelligently and asynchronously with uncertainty present in the negotiation process. Our new protocol represents a family of different protocols in which agents can choose different options depending on both the static and dynamic context of the negotiation. To our knowledge, the new protocol subsumes the CNP and most—if not all—of its extensions. The next sections present an overview of the protocol and the tradeoffs involved. For more details see [15, 12, 11].

2 Commitment issues

In mutual negotiations, *commitment* means that one agent binds itself to a potential contract while waiting for the other agent to either accept or reject its offer. If the other party accepts, both parties are bound to the contract. When accepting, the second party is sure that the contract will be made, but the first party has to commit before it is sure. Commitment has to take place at some stage for contracts to take place, e.g. TRACONET was designed so that commitment took place in the bidding phase. However, this stage can be varied. Shorter protocols (commitment at the announcement phase) can be constructed as well as arbitrarily long ones (commitment at the awarding phase or some later stage). The choice of commitment stage can be a static protocol design decision or the agents can decide on it dynamically. For example, the focused addressing scheme of the CNP was implemented so that in low utilization situations, contractors announced tasks, but in high utilization mode, potential contractees signaled availability—i.e. bid without receiving announcements first [17, 19]. So, the choice of a protocol was based on characteristics of the environment. Alternatively, the choice can be made for each negotiation separately before that negotiation begins. We advocate a more refined protocol [15], where agents dynamically choose the stage of commitment of a certain negotiation during that negotiation. This allows any of the above alternatives, but makes the stage of commitment a negotiation strategy decision, not a protocol design decision. Unlike the CNP, but like the work of Sen [16], our protocol allows counteroffers: an agent can for example recombine tasks to be negotiated over, alter the task descriptions, alter the suggested decommitment penalties, or alter the suggested price.

In traditional multiagent negotiation protocols among SI agents, once a contract is made, it is binding, i.e. neither party can back out. In cooperative distributed problem solving (CDPS), commitments are often allowed to be broken unilaterally based on some local reasoning that attempts to incorporate the perspective of common good. A more general alternative is to use protocols with continuous levels of commitment based on a monetary penalty method, where commitments vary from unbreakable to breakable as a continuum by assigning a commitment breaking cost to each commitment separately. Among other things, the use of multiple levels of commitment allows:

[2] The *marginal cost* of adding a set of tasks to an agent's solution is the cost of the agent's solution with the new task set minus the cost of the agent's solution without it.

- a low commitment search focus to be moved around in the global task allocation space (because decommitting is not unreasonably expensive), so that more of that space can be explored among SI agents, which would otherwise avoid risky commitments,
- flexibility to the agent's local deliberation control, because marginal cost calculation of a contract can go on even after that contract has already been agreed upon,
- an agent to make the same low-commitment offer (or offers that overlap in task sets) to multiple agents. In case more than one accepts, the agent has to pay the penalty to all but one of them, but the speedup of being able to address multiple agents in committal mode may outweigh this risk,
- the agents with a lesser risk aversion to carry a greater portion of the risk. The more risk averse agent can trade off paying a higher price to its contractee (or get paid a lower price as a contractee) for being allowed to have a lower decommitting penalty, and
- contingency contracts by conditioning the payments and commitment functions on future negotiation events or domain events. These enlarge the set of mutually beneficial contracts, when agents have different expectations of future events or different risk attitudes [8].

We presented the details of such a protocol in [15] and formally analyzed its advantages in [11]. Because the decommitment penalties can be set arbitrarily high for both agents, the leveled commitment protocol can always emulate the full commitment protocol. Furthermore, there are cases where there is no full commitment contract among two agents that fulfills the participation constraints (agent prefers to agree to the contract as opposed to passing) for both agents, but where a leveled commitment contract does fulfill these constraints. This occurs even among risk neutral agents, for example when uncertainty prevails regarding both agents' future offers received, and both agents are assigned a (not too high or low, and not necessarily identical) decommitment penalty in the contract. Among risk neutral agents, this does not occur if only one of the agents is allowed the possibility to decommit (other agent's decommitment penalty is too high), or only one agent's future is uncertain. If the agents have biased information regarding the future, they may perceive that such a contract with a one-sided decommitment possibility is viable although a full commitment contract is not. In such cases, the agent whose information is biased is likely to take the associated loss while the agent with unbiased information is not.

3 Implications of bounded rationality

Interactions of self-interested agents have been widely studied in game theory and other fields of microeconomics (e.g. [5, 20, 8]), and lately also within DAI (e.g. [9, 2]). Most of the results assume perfect rationality of the agents: flawless deduction, optimal reasoning about future contingencies and recursive modeling of other agents. In terms of negotiation, perfect rationality implies that agents can compute their marginal costs for tasks exactly, immediately, and without computation costs. For example, the research in [9] assumes that an agent can instantly solve exponentially many NP-hard problems, which is untrue in most practical situations. In single agent settings, an agent's rationality is bounded because computation resources are costly, or they are bounded and the environment keeps changing—e.g. new tasks arrive and there is a bounded amount of time before each part of the solution is used. Contracting agents have the following additional real-time pressures:

- A counteroffer or an acceptance message has to be sent by a deadline—otherwise the negotiation terminates (using our protocol). If the negotiation terminates, the agent can begin a new negotiation on the same issues, but it will not have the other agent's commitment at first.
- Sending an outgoing offer too late may cause the receiving agent to make a contract on some of the same tasks with some other agent who negotiated earlier—thus disabling this contract even if the offer makes the deadline. In this case, the partner has to pay the decommitment penalty that it had declared.
- In our protocol, an agent may get paid less for handling tasks (or pay more for having tasks handled) or be required to commit more strongly or receive a weaker commitment from the negotiation partner if its response is postponed.

- The agent's cost of breaking commitments (after a contract is made) may increase with time.

This problem setup leads to a host of local deliberation scheduling issues. An agent has to decide *how much computation* it should allocate to refine its marginal cost estimate of a certain task set. If too much time is allocated, another agent may win the contract before the reply is sent, or not enough time remains for refining marginal costs of other task sets. If too little time is allocated, the agent may make an unbeneficial contract concerning that task set. If multiple negotiations are allowed simultaneously, the agent has to decide on *which sets of tasks* (offered to it or potentially offered by it) its bounded computation should be focused, and *in what sequence*. It may want to ignore some of its contracting possibilities in order to focus more deliberation time to compute marginal costs for task sets of some selected potential contracts. So, there is a tradeoff of getting more exact marginal cost estimates and being able to engage in a larger number of negotiations.

3.1 Pending offers

The CNP did not consider an agent's risk attitude toward being committed to activities it may not be able to honor, or the honoring of which may turn out unbeneficial. In our protocol, an agent can take a risk by making offers while the acceptance of earlier offers is pending. Contracting during pending commitments speeds up the negotiations because an agent does not have to wait for results on earlier commitments before carrying on with other negotiations. The work on TRACONET formalized the questions of risk attitude in a 3-stage (announce-bid-award) full-commitment protocol, and chose a risk taking strategy where each agent ignored the chances of pending commitments being accepted in order to avoid computations regarding these alternative future worlds. This choice was static, but more advanced agents should use a risk taking strategy, where negotiation risk is explicitly traded off against added computation regarding the marginal cost of the task set in the alternative worlds, where different combinations of sent pending offers are accepted.

3.2 Enlarging the offering/accepting/rejecting context

Usually, an agent does not know what offers it will receive in the future. There is a strategic tradeoff between accepting (or counterproposing) early on and waiting:

- A better offer may be received later.
- Waiting for more simultaneously valid offers enables an agent to identify and accept synergic ones: having more options available at the decision point enables an agent to make more informed decisions.
- Accepting early on simplifies costly marginal cost computations, because there are fewer options to consider. An option corresponds to an item in the power set of offers that an agent can accept or make.
- By waiting an agent may miss opportunities due to others making related contracts first.

Intuitively speaking, accepting the first profitable offer corresponds to the first-swap iterative refinement algorithm in the global task allocation space. Waiting for all the offers corresponds to the best-swap algorithm (assuming that no offers are missed by waiting). Undoing the first swap for a better swap is possible in our variable commitment protocol.

3.3 Anticipating the future

An intelligent agent should anticipate future negotiation events (contracts and offers) and future domain events (new tasks arriving, resources breaking down) in its negotiation strategy [15]. In order to bias its current negotiation decision towards accepting tasks that are synergic with anticipated future tasks, it is sufficient for the agent to take these future events into account in marginal cost estimation: this will cause the agent to anticipate with its negotiation decisions. The real marginal cost of a task is the difference in the *streams of payments and domain costs* when an agent has the tasks and when the agent does not have it. This marginal cost does not necessarily equal the cost that is acquired statically at the time of contracting (before the realization of unknown future events) by taking the difference of the cost of the agent's optimal solution with the task and the optimal solution without it. Furthermore, for bounded rational agents, the marginal cost may change as more computation is allocated to the solution including the task or the solution without the task.

In general, the marginal cost of a task set depends on which other tasks the agent has. Therefore, theoretically, the marginal cost of a task set has to be computed in all of the alternative future worlds, where different combinations of pending, to-be-sent, and to-be-received offers have been accepted, different combinations of old and to-occur contracts have been broken by decommitting (by the agent or its partners), and different combinations of domain events have occurred. Managing such contingencies formally using probability theory is intractable: costs of such computations should be explicitly traded off against the domain advantage they provide. An agent can safely ignore the chances of other agents decommitting only if the decommitment penalties are high enough to surely compensate for the agent's potential loss. Similarly, an agent has to ignore its decommitting possibilities if its penalties are too high. The exponential number of alternative worlds induced by decommitting options sometimes increases computational complexity more than the benefit from the gradual commitment scheme warrants. Moreover, the decommitting events are not independent: chains of decommitting complicate the management of decommitment probabilities. Thus, decommitment penalty functions that increase rapidly in time are often appropriate for BR agents.

3.4 Long term contracts: Social laws

Long term contracts can be made to avoid computing solutions to all situations from scratch or pruning parts of the global search space. Such social laws can be justified in the cost-based approach by accounting for each agent's real-time deliberation process: computation itself incurs cost. An example of a social law is the *preset distribution of some task types*, i.e. independent of which agent receives the task from the environment, the type of the task determines which agent will be the contractor. Another social law is the *audience restriction* [19]: a task is announced to only a subset of the agents. Social laws compromise solution quality, but save computation and communication costs and allow solutions to be found in a more timely manner.

4 Insufficiency of the Vickrey auction

The *Vickrey auction* was originally developed to promote truthful bidding and to avoid counterspeculation among self-interested agents. In this section we show that even in cases where that auction mechanism meets these desiderata when used among rational agents, it can fail to meet them with bounded rational agents. As an example, we use the announce-bid-award protocol with total commitment at the bidding phase and none elsewhere. In the Vickrey auction (second price sealed-bid auction), the contractor accepts the bid that offers the lowest required payment, but pays the required payment of the *second lowest* bid. This motivates each potential rational contractee to bid its true marginal cost, i.e. not to strategically overstate based on socially wasteful counterspeculation of opponents' bids [21]. Truthful bidding is the *dominant strategy*, i.e. it is the strategy that provides a bidder the highest expected payoff *no matter what the other potential contractees bid*, i.e. irrespective of their marginal costs and irrespective of their truthfulness. Therefore, an agent is motivated to bid truthfully even if it exactly knew the other contractees' bids. There are some limitations and deficiencies in the Vickrey auction even among rational agents:

- The mechanism is vulnerable to collusion. The bidders could coordinate their bid prices so that the second lowest bid stays artificially high (higher than the second lowest marginal cost among the bidders). In this manner the bidders could get a higher payment for carrying out the task than they would without colluding. For collusion to work, the bidders need to identify each other before the submission of bids—otherwise a non-member of the coalition could win the contract. Furthermore, the bidders would have to maintain an income redistribution method, because only one bidder gets the task and payment.
- The contractor may understate the second lowest offer to the lowest bidder unless that bidder can verify it. An understated second offer would give the contractee a lower payment than it would receive if the contractor were truthful. In a sense, a truthful auctioneer is assumed. Alternatively, cryptographic electronic signatures could be used by the bidders so that the contractor could actually present the second best bid to the winning contractee (and would not be able to alter it).
- The Vickrey auction promotes truthful bidding in *private-value auctions* where an agent's marginal cost for a task is totally determined locally and is independent of other agents' marginal costs. This is in contrast to *public-value auctions*, where an agent's marginal cost is entirely determined by other agents' valuations (such as in bidding for stocks). In contracting protocols where a contractee can recontract out a task that it contracted in, the contractee's marginal cost for a task is, at least to some extent, defined by the potential recontracting prices. Therefore, such settings are not pure private-value auctions, and truthful bidding is not necessarily the dominant strategy.
- When applying the Vickrey auction (or other auction) to a *series of contracts* (of possibly nonintersecting sets of tasks), an agent should take into account in the marginal cost determination of a certain task set the effect that that task set would have on the marginal costs of other task sets that will potentially be negotiated over in the future. Such future contingencies are subject to strategic behavior on part of the other agents, and furthermore, estimating the contingencies requires counterspeculation.

Let us look at a one-shot auction where no future contracts will take place, recontracting is impossible, agents cannot collude, and the contractor has to verify the second best bid to the winning bidder. Surprisingly, even under these restrictive assumptions that guarantee the operation of the Vickrey auction among rational agents, that auction mechanism can fail to solve the counterspeculation problem and to promote truthfulness among agents whose rationality is bounded by limited computation resources. The following simple scenario acts as a counterexample by showing the usefulness of counterspeculation. Say, agent C announces a task, and it is common knowledge that there are two bidders: A and B. We will analyze A's best bidding strategy. Let us assume that B bids truthfully. This is sufficient for our counterexample, because we are trying to show that truthful, counterspeculation-free bidding is not A's dominant strategy, i.e. that there exists a situation where A should act differently. Say that A does not know the task's exact marginal cost for agent B or for itself, but it can decrease the variance of the probability density functions of these costs by allocating more computation to each of these problems separately. If the distribution of B's marginal cost is entirely above the distribution of A's marginal cost, A can stop computing and submit any bid below B's distribution—thus getting the contract at the price that B announces. If this is not the case, A can maximize expected payoff by trading off further computing its own or B's marginal cost, and submitting some strategic bid. Therefore, A needs to outguess B: it has to estimate B's probability distribution in order to terminate its own marginal cost estimation.

The design of truth-promoting, counterspeculation-free protocols for BR agents is a virgin area of research. Most probably, protocols themselves will actively use timing to minimize counterspeculation and bid biasing.

5 Terminating the negotiations

Knowing when to terminate distributed search is difficult—especially for iterative refinement algorithms [10] that are not based on a systematic and often slow backtracking scheme. TRACONET used the following termination heuristic for distributed iterative refinement: an agent stops negotiating once it has made no contracts during a certain fixed number of negotiation iterations. Since then we have developed an exact termination protocol for iterative refinement negotiations. With cooperative, exactly computing agents it guarantees that negotiations end exactly when a local minimum with respect to all task exchanging operators and each agent's local refinement operators has been found. The idea is that the agents inform each other of situations when they have tried all possible local search operators and announced all possible (with respect to the restriction on the number of tasks per announcement) task sets without success. If an agent that has sent this stalemate status information to other agents later gets an award (tasks are allocated to it by contracting) or a domain event (new task from the environment or resource status change, e.g. breakdown), it retracts its stalemate status by sending a message to the other agents. When an agent is in stalemate status, it is still worth while for it to continue announcing and bidding, because other agents may not be in a stalemate. Once all agents are in a stalemate simultaneously, a local optimum has been reached, and the negotiations should be terminated until new domain events occur or new agents log onto the negotiation net. The termination protocol is appropriate in domains where there is relatively little domain volatility, i.e. new tasks and changes in resource status. If there is high domain volatility, a local optimum will never be reached, because the negotiations take a long time in comparison to the domain changes, and in general, each domain change redefines the set of local optima.

When used among BR cooperative agents, this method may terminate the negotiations before a local optimum is reached. For example, an agent may omit reacting to an announcement in order to allocate more time to process other messages. This makes the announcing agent think that the receiver of the announcement could not bid, i.e. that there was no possibility of a beneficial swap. This may lead to termination of the negotiations, while the agents erroneously believe that a local optimum has been reached. The termination protocol also fails to guarantee a local optimum (or even to terminate at all) when used among rational but SI agents. First, a SI agent may not bid (due to strategic manipulation) even if it could based on marginal cost calculations. Therefore an agent cannot infer the required amount of information from another agent's decision not to bid. Secondly, there is some cost to send each stalemate status messages, and it is not always in an agent's self-interest to do so. Future work includes developing a termination protocol that works among BRSI agents also as opposed to just cooperative, exactly computing ones.

6 Conclusions

Most distributed scheduling systems are designed to operate among cooperative agents, and are inappropriate for self-interested agents. This paper presented some of the issues involved in extending the CNP to be used among BRSI agents in a production scheduling domain. The new protocol allows the *stage* of commitment to vary, and can adjust the *level* of commitment, which has favorable implications on the distributed search. Implications of bounded rationality were discussed. It was shown that there are situations where the Vickrey auction works among rational agents, but fails among bounded rational ones. Finally, a negotiation termination protocol was discussed that guarantees a local optimum when used among cooperative, exactly computing agents, but may fail to do so among BRSI agents.

Many other issues arise in contracting among BRSI agents, but where omitted due to space limitations. For example, the protocol should facilitate mechanisms to avoid local optima: task clustering [15, 10], barter task exchanges [15], and contracts involving more than two agents [15]. Another issue is that sometimes the exchanges can be carried out without external enforcement [14]. There are also questions regarding message passing. First, message congestion is more likely to occur among SI agents than among cooperative agents, whose behavior can be centrally prescribed [15]. Secondly, a SI agent cannot be expected to reply to all messages. This makes practical protocol design more complicated, because reply motivation mechanisms or time-outs are required [15]. Finally, there are interesting issues regarding coalition formation among BRSI agents [13].

References

[1] P. Burke and P. Prosser. The distributed asynchronous scheduler. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, pages 309–339. Morgan Kaufmann, 1994.

[2] E. Ephrati. A non-manipulable meeting scheduling system. In *Proc. 13th International Distributed Artificial Intelligence Workshop*, Lake Quinalt, Washington, July 1994. AAAI Press Technical Report WS-94-02.

[3] T. Finin, R. Fritzson, and D. McKay. A language and protocol to support intelligent agent interoperability. In *Proc. of the CE & CALS Washington '92 Conference*, June 1992.

[4] General Magic, Inc. Telescript technology: The foundation for the electronic marketplace, 1994. White paper.

[5] D. M. Kreps. *A course in microeconomic theory.* Princeton University Press, 1990.

[6] V. R. Lesser and L. D. Erman. Distributed interpretation: A model and experiment. *IEEE Transactions on Computers*, C-29(12):1144–1163, 1980.

[7] D. Neiman, D. Hildum, V. R. Lesser, and T. W. Sandholm. Exploiting meta-level information in a distributed scheduling system. In *Proc. Twelfth National Conference on Artificial Intelligence (AAAI 94)*, Aug. 1994.

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

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

[10] T. W. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proc. 11th National Conference on Artificial Intelligence (AAAI 93)*, July 1993.

[11] T. W. Sandholm and V. R. Lesser. Advantages of a leveled commitment contracting protocol. Technical report, University of Massachusetts at Amherst Computer Science Department, 1995. In preparation.

[12] T. W. Sandholm and V. R. Lesser. Automated contracting among self-interested bounded rational agents. Technical report, University of Massachusetts at Amherst Computer Science Department, 1995. In preparation.

[13] T. W. Sandholm and V. R. Lesser. Coalition formation among bounded rational agents. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, Aug. 1995.

[14] T. W. Sandholm and V. R. Lesser. Equilibrium analysis of the possibilities of unenforced exchange in multiagent systems. In *Proc. 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, Aug. 1995.

[15] T. W. Sandholm and V. R. Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *Proc. First International Conference on Multiagent Systems (ICMAS-95)*, San Fransisco, June 1995.

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

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

[18] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, Special issue on distributed artificial intelligence SMC-21(6):1446–1461, Nov/Dec 1991.

[19] H. Van Dyke Parunak. Manufacturing experience with the contract net. In M. N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 10, pages 285–310. Pitman, 1987.

[20] H. R. Varian. *Microeconomic analysis.* New York: W. W. Norton, 1992.

[21] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.