An Application of Automated Negotiation to Distributed Task Allocation*

Michael Krainin, Bo An, Victor Lesser
Department of Computer Science
University of Massachusetts Amherst
{krainin, ban, lesser}@cs.umass.edu

Abstract

Through automated negotiation we aim to improve task allocation in a distributed sensor network. In particular, we look at a type of adaptive weather-sensing radar that permits the radar to focus its scanning on certain regions of the atmosphere. Current control systems can only computationally handle the decision making for a small number of radars because of the complexity of the process. One solution is to partition the radars into smaller, independent sets. Redundant scanning of tasks and loss of cooperative scanning capabilities can occur as a result. With negotiation we can reduce these occurrences, helping to ensure that the correct radars scan tasks based on the overall social welfare. We develop a distributed negotiation model where on each cycle the overall system utility improves or remains constant. Experimental results show that as compared to the centralized task allocation mechanism, the proposed distributed task allocation mechanism achieves almost the same level of social welfare but with a significantly reduced computational load.

1. Introduction

This paper explores the use of negotiation for a network of short-range, adaptive radars called NetRad [11], which work together as described in the next section in order to improve weather detection at low elevations. NetRad adaptive radars are controlled by the Meteorological Command and Control (MCC) system, which gives them instructions as to where to scan based on emerging weather conditions. In the context of MCCs, a task is a weather phenomenon in need of being scanned. The process of allocating tasks to the radars, however, takes an amount of time exponential in the number of radars and weather-tracking tasks which the MCC must consider. As outlined in [4], one approach to handling these combinatorics as the radar network is scaled up is to partition the network control so that there are multiple MCCs. Each MCC is responsible for controlling a set of radars, and no radar is in the radar set of more than one MCC. Unfortunately, if MCCs do not know each others' scanning strategies, redundant scanning can occur. Additionally, in certain situations it is very advantageous for a task to be scanned by multiple radars, but some of these opportunities can be lost when these radars are controlled by different MCCs. To prevent these undesirable behaviors, we will use automated negotiation as a means of task allocation among MCCs.

Many approaches have been used for task allocation in the past. Often there is a focus on conserving limited resources [10][17], but the sensors in this application are resource-rich, and the primary concerns are speed of computation and quality of scans. The process needed for allocating tasks is similar to that of coalition formation [1][2]; however, the agents involved are not individual radars but rather groups of radars. Each decision to either join or quit a coalition involves a potentially very costly computation. Although [17] deals with allocating tasks in a clustered sensor network, its goal of energy minimization is not appropriate for this application.

Our approach to negotiation is one which uses marginal utility calculations to determine which MCC benefits most from altering its configuration. In doing so we monotonically increase the social welfare. This approach still allows for concurrency in the network and also resolves issues of how multiple MCCs can interact to handle tasks on partition boundaries. The concurrency that results when each MCC runs its own optimization is particularly important because a centralized approach may only handle a few radars and a few tasks given its approximately 30 second time limitation; whereas, spanning the United States could require thousands of radars. Through automated negotiation, we can allow for concurrency and properly handle issues involving multiple MCCs. Although we can potentially in-

*This work was supported primarily by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

crease processing power and communication bandwidth at the centralized site, the distributed resource allocation approach has some additional advantages such as higher reliability and scalability. The spacial locality of tasks also makes partitioning a particularly appealing option.

The rest of the paper is structured as follows. Section 2 introduces the NetRad radars and Meteorological Command and Control more thoroughly. Section 3 discusses the simulator used to obtain the results later in the paper. The negotiation scheme proposed for this application as well as its implementation are in Section 4. Section 5 gives the results of our experiments. In Section 6, we investigate scenarios in which the negotiation performs particularly well or particularly poorly. Section 7 describes the use of this protocol in the physical weather-sensing network. Finally, our conclusions and possible future work are contained in Section 8. Related work is discussed throughout.

2. Architecture Overview

The MCC system is a closed-loop control system in that it responds to the emerging weather events based on detected features in the radar data and end-user concerns that may vary over time. End-users such as forecasters, emergency managers, and researchers can provide information as to what sort of data they are looking for and how frequently. This end-user data goes into determining the priority of any given task. Consequently, the MCC ranks the importance of tasks so as to give preference to the data users want.

The NetRad radar is a radar specially designed for the purpose of quick detection of low-lying meteorological phenomena such as tornadoes. They are short-range radars used in dense networks, thereby alleviating blind-spots caused by the curvature of the Earth. NetRad radars additionally do not use the traditional sit-and-spin strategy; rather, they can be focused to scan only in a particular volume of space. By using the combined data gathered by the NetRad radars, scanning strategies may be created which target radar scanning more effectively than a simple sit-and-spin technique.

The MCC gathers moment data from the radars and runs detection algorithms on the weather data. The results of this analysis lead to a set of weather-scanning tasks of interest for the next radar scanning cycle. The MCC then determines the best set of scans for the available radars that will maximize the sum of the utility associated with the chosen tasks according to a utility function based on the end-user priorities. These scans are used by the NetRad radars on the next cycle of this process each cycle being approximately thirty seconds.

The utility for a task from a single radar is the priority of the task multiplied by a factor meant to represent the quality of the data that would result from the scan. This factor is itself a combination of factors: (coverage factor) $\times$ (distance factor + elevation factor). These represent the amount of the scanning task that is covered by the scan of a radar, the distance of the scanning task region of interest from the radar, and the number of elevations scanned respectively. The elevation factor is weighted nine times as heavily as the distance factor; the exact values are determined by a step function. Using this formula, the MCC considers both the importance of the data to the user and how well it would be able to collect that data. For a more in-depth description of the MCC system, see [18], or see [7] for more details on the utility function.

3. Simulator

To test out new ideas, we have created an abstract simulation of the NetRad radar system. The primary focus of the simulator is to determine how best to decentralize control and as such abstracts away details of the actual system. The simulator, built on top of the Farm simulator framework [5], consists of a number of components. Radars are clustered based on location, and each cluster of radars has a single MCC. Each MCC has a feature repository where it stores information regarding tasks in its spacial region, and each task represents a weather event. It also has an optimization function, which takes the tasks from the feature repository and returns scans for each of the radars belonging to the MCC as well as the value of the scans as determined by the utility function. The simulator additionally contains a function that abstractly simulates the mapping from physical events and scans of the radars to what the MCC eventually sees as the result of those scans. Depending on the elevations scanned, the number of radars scanning, the type of task, and the speed of scanning, it assigns error values to the attributes of the task within certain bounds. In this way, the MCCs do not see exactly what is there but rather something slightly off. Through this process, MCCs discover and track the movement of the weather events.

Tasks are abstracted and are represented as circular areas. Each task has a position, a velocity, a radius, a priority, a preferred scanning mode, and a type. The preferred scanning mode is meant to encapsulate the elevations that need to be scanned to get the most accurate results in scanning the event. Tasks may be one of a few different types: storms, rotations, reflectivity, or velocity. Each of these types has its own distributions for the characteristics described above. The tasks are added into the system by an event generator that places a set number of weather events randomly in the region the radars occupy. The event generator sets all other attributes of the tasks randomly within certain bounds which depend on the type of task.

Tasks may also be either pinpointing or non-pinpointing meaning either there is, or is not, a significant gain by scan-

ning the associated volume of space with multiple radars at once. The utility gained from scanning a pinpointing task increases with the number of radars scanning the task; whereas, the utility for a non-pinpointing task is the maximum of the utilities from the individual radars. This final utility for a task is calculated the same way in the simulator as it is in the real system. Similarly, the optimization function used by MCCs in the simulator for maximizing these utilities is the same as the one used in the real system.

Several abstractions are used to simplify the simulator. Most notably, time is discretized into units of system cycles. Rather than having a beam sweep across moving targets, the scans are instead represented by wedges of scanning area, and all motion of tasks occurs between cycles. This is a reasonable approximation as each cycle represents only 30 seconds of time, and the radars have a range of approximately 30 kilometers. The meteorological phenomena do not change significantly at these small time scales.

4. Distributed Negotiation Model

Although switching from a centralized to a distributed system increases speed of calculation of scanning strategies, it also decreases quality of scanning. Each MCC ends up with only a partial view of the physical space; tasks crossing partition boundaries can thus be lost where they would not in a centralized system. This problem can be alleviated by allowing MCCs to share abstract level data such as task location with each other for tasks lying on boundaries. Each MCC also does not know what other MCCs will do, which results in two distinct ways scanning quality can decline. The first source of quality degradation is the loss of the ability to cooperatively scan pinpointing tasks on boundaries, which can be solved by coordinating scans between MCCs and sharing resulting raw data. The second source of lessened quality is redundant scanning. Most tasks do not require scanning by multiple radars, but without any extra mechanism, each MCC will believe it should scan all tasks on partition boundaries. This can also be helped by coordinating scans. Through negotiation, we can help prevent all of these causes of poorer scanning.

Negotiation has been used in distributed sensor networks in the past; however, previous techniques are not entirely appropriate for these weather-sensing radars. For example, [15] uses an argumentation-based approach to coalition formation: an initiator attempts to recruit other sensors to scan a specific task. In our domain, a per task negotiation would not be feasible due to time limitations.

Contract-net based negotiation schemes [8][13] face similar limitations in the NetRad radar network. Although the contract net protocol's generality allows for both cooperative and self-interested agents by allowing each agent to make bids based on a local utility function, it is not a suitable protocol. The agents are strictly cooperative in this domain, with the relevant measure being the social welfare. The need to consider specific subsets of tasks to add or remove leads to an excessive number of marginal utility calculations. Every time an MCC's neighbor changes its scanning strategy, that MCC must perform potentially as many optimizations for marginal utility calculations as the size of the powerset of the boundary tasks belonging to it. As we will see, by properly structuring the negotiation and taking the focus of the negotiation away from individual tasks or sets of tasks, we can reduce this number to one.

Our negotiation scheme is a hillclimbing algorithm with the goal being to monotonically increase the social welfare each round. We chose utilitarian social welfare as the relevant measure because we want to maximize the average value we receive per task. As [3] discusses, there are other measures which could be used instead such as the egalitarian social welfare and the elitist social welfare. For any given timestep, we are not looking to make sure that each task is at least partially scanned, so an egalitarian model would not make sense for our purposes. Also, while it can make sense to scan certain tasks very thoroughly, we do not always want to focus on maximizing the utility for a single task while starving others. Therefore, we chose utilitarian social welfare as the measure to maximize.

To do this, we use marginal utility calculations to determine what configuration change will produce the greatest overall benefit. Negotiation proceeds in rounds, and MCCs need only utilize their optimization functions once per round. Each marginal utility calculation made reflects any configuration changes that were made in previous rounds of the negotiation. In a way, the optimization run each round is analogous to a search for a specific type of contract. This contract is similar to the OCSM-contracts discussed in [14], but with the additional constraint that it is only one agent making any changes.

4.1. Definitions

For negotiation, there are multiple MCCs, which we will denote $M = \{M_1, M_2, \ldots, M_{|M|}\}$. These MCCs control radars $R = \{R_1, R_2, \ldots, R_{|M|}\}$, where $R_i = \{r_{i,1}, r_{i,2}, \ldots, r_{i,|R_i|}\}$ is the set of radars controlled by M_i. Over the area the radars scan there are a number of tasks, which are weather phenomena to be scanned: $T = \{t_1, t_2, \ldots, t_{|T|}\}$. $T_i \subseteq T$ is the set of tasks observable by M_i's radars. Out of the tasks in T_i, some are boundary tasks, meaning they lie in the region of more than one MCC, and we will call this set T_i^o.

A configuration, or scanning strategy, of M_i is a set of sweeps for R_i and is defined as $C_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,|R_i|}\}$, where $s_{i,j}$ is the sweep for radar $r_{i,j}$. The global scanning strategy is just $C =$

$\{C_1, C_2, \ldots, C_{|M|}\}$. In Sections 2 and 3, we defined the notion of utility of a single task based on the scanning strategy of some collection of radars. For the scanning strategy $C^{partial} \subseteq C$ of some number of MCCs, we may represent this utility for a task t_i with the notation $U\left(t_i, C^{partial}\right)$.

The utility for a set of tasks is defined to be the sum of the utilities for the individual tasks. With this definition we may define the social welfare $\omega = U(T, C) = \sum_{t_i \in T} U(t_i, C)$.

In our experiments, we define a neighbor of M_i to be any MCC whose radars overlap in space with those controlled by M_i. Communication among neighboring MCCs can be decreased by refining the definition to be just those MCCs that share a task with M_i. During negotiation, an MCC M_i will know not only its configuration C_i, but also the configurations $C_i^{neigh} \subset C$ of the neighboring MCCs. With this information, M_i may calculate the utility of T_i for its own configuration and its neighbors configurations, $U\left(T_i, \{C_i\} \cup C_i^{neigh}\right)$. Thus it may also calculate the marginal utility for a change in configuration from C_i to C_i' as $U^{marg}(C_i') = U\left(T_i, \{C_i'\} \cup C_i^{neigh}\right) - U\left(T_i, \{C_i\} \cup C_i^{neigh}\right)$. Note that only the scans in C_i^{neigh} that cover tasks in T_i are considered in calculating U.

4.2. The Negotiation Model

Before MCCs can begin the main stages of negotiation, each MCC M_i communicates with its neighbors to make sure its set of boundary tasks T_i^o is in agreement with the boundary tasks of other MCCs. If one MCC's radars scanned a region more recently than those of a neighboring MCC, the one that scanned more recently likely has a better idea of the tasks in the region. The most recent task information is used by all MCCs. M_i's initial value for C_i is calculated with $C_i^{neigh} = \emptyset$. Each MCC shares its initial configuration with other MCCs, and the configurations collected by M_i become the initial C_i^{neigh}. Now that each MCC has these initial values, the main stages of negotiation can begin. Rounds are run until a time limit of approximately 30 seconds is reached. This time limit is set because each system pulse is restricted to a very brief time period in order to keep the radars sufficiently adaptive. Each round k consists of the following sub-stages:

1. Sub-stage k^1 (proposing) : Each MCC M_i computes its best configuration given C_i^{neigh} as $C_i' = argmax_{C_i'} U\left(T_i, C_i' \cup C_i^{neigh}\right) - U\left(T_i, C_i \cup C_i^{neigh}\right)$. This is equivalent to simply $argmax_{C_i'} U\left(T_i, C_i' \cup C_i^{neigh}\right)$. M_i additionally calculates the marginal utility of this potential move, $U^{marg}(C_i')$ as defined above. Note that if neither C_i nor C_i^{neigh} changed from the previous round, then C_i' and $U^{marg}(C_i')$ from the previous round can be used again without any additional computation. M_i then sends its marginal utility and C_i' to each of its neighbors.

2. Sub-stage k^2 (agreement selection): Each MCC M_i received the marginal utilities from each of its neighbors during sub-stage k^1. M_i changes its configuration to C_i' if it has a higher marginal utility than any of its neighbors, and the marginal utility is greater than zero. In the case of a tie, the MCC with higher index number is the one to change its configuration. In this way when one MCC changes its configuration, none of its neighbors will change theirs.

3. Sub-stage k^3 (informing of changes): Each MCC sends out its current configuration to all of its neighbors. Although most MCCs will not have changed their configurations, this technique is used for simplicity. An MCC may not know if its neighbor with greatest marginal utility actually ended up changing its configuration. In practice the number of messages needed in this sub-stage could likely be decreased. For each M_i, the configurations it receives become the new C_i^{neigh}.

This approach is similar to the DBA algorithm [16] and the LID-JESP algorithm [12], which makes use of the DBA algorithm. These algorithms also take a hillclimbing approach to multi-agent optimization. In LID-JESP, agents begin with a random policy then each agent calculates its best policy given its neighbors' current policies; if an agent's gain is greater than its neighbors', it changes its policy and sends its new policy to its neighbors. This is the basic structure of our algorithm as well; however, we restrict ourselves to considering policies for a single timestep at a time due to the enormous size of the action and observation spaces and the time-restrictions faced by MCCs. To reduce the number of iterations needed, we start each MCC not with a random configuration but with one that is likely somewhat close to its final configuration.

Our algorithm also shares some of the problems faced by LID-JESP. Primarily, there is some loss of concurrency by requiring an MCC's marginal utility to be greater than all of its neighbors'. This is particularly problematic in the case where chains of MCCs not making moves are formed. For example if M_i's marginal utility is greater than its neighbor M_j's, which is in turn greater than its neighbor M_k's, neither M_j nor M_k will make a move. One possible solution is the type of alteration introduced in the SLID-JESP algorithm [6], which is to allow any move with positive marginal utility with a certain probability. An alternative to this is a mediation-based approach discussed further in Section 4.4.

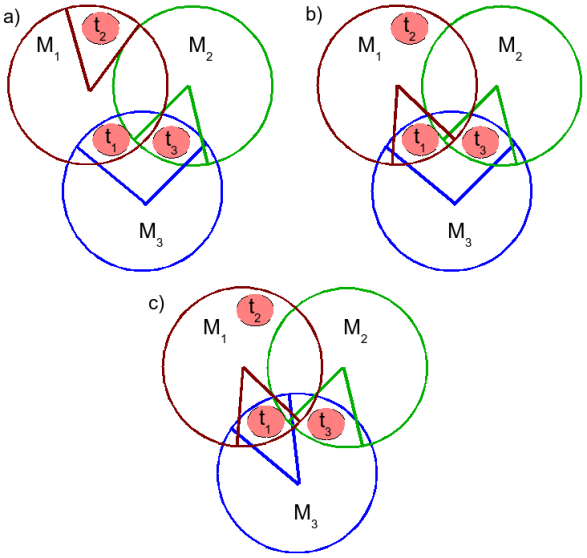

Figure 1. Three radars belonging to different MCCs

In Section 5 we will explore simple modifications to the negotiation protocol to allow neighboring MCCs to make moves simultaneously. The first idea we test is the most obvious technique to try, which is to allow any MCC with a positive marginal utility to make its move regardless of its neighbors' actions. This is akin to using SLID-JESP with the probability for making a move set to one.

The other modification we explore performs a more advanced check of neighbors' moves. Suppose an MCC M_i has a positive marginal utility. For each neighbor M_j of M_i, M_i determines whether its move conflicts with its neighbors. If M_i's marginal utility is greater or if the two MCCs have no common boundary tasks, then M_j presents no reason for M_i not to make its move. Otherwise, for each shared boundary task t_k, M_i compares $U\left(t_k, \{C_i'\} \cup \left\{C_j'\right\}\right)$ to $U\left(t_k, \{C_i\} \cup \left\{C_j'\right\}\right)$ and $U\left(t_k, \{C_i'\} \cup \{C_j\}\right)$. The purpose of the comparison is to determine how the marginal utilities change when the assumed configurations change. As long as M_i can assure that both marginal utilities will remain positive, M_i's move does not conflict with M_j's move. If an MCC has no conflicts with neighbors, then it makes its move. Clearly, this modification requires that marginal utility messages also include the proposed new configuration, but the hope is that by increasing concurrency, this modification will reduce the number of rounds needed.

Fig. **??** illustrates how negotiation works. t_1 is a pinpointing task, so it gets a low value if scanned by only one radar and a high value if scanned by more than one. The negotiation begins with each MCC sending messages to its neighbors containing descriptions of the common tasks between them. The MCCs then calculate their initial configurations. Let us suppose that M_1 would scan t_2, M_2 would scan t_3, and M_3 would scan t_1 and t_3 as shown in part a of Fig. 1. Each MCC then sends its initial configuration to its neighbors.

The first round commences with each MCC determining its best move based on the configurations it received from its neighbors. M_1 can now improve the social welfare considerably by switching to the pinpointing task t_1 since M_1 now knows M_3 is scanning it. M_2 does not have any other configuration options. M_3 can slightly improve the social welfare by focusing just on t_1 since M_2 is taking care of t_3. Each MCC sends its marginal utility to its neighbors. M_1 likely has the highest marginal utility, so it would change its configuration. The round ends with each MCC sending out its configuration (part b of Fig. 1).

The second round starts with MCCs again calculating the marginal utilities for their best moves. M_1 and M_2 do not have any beneficial moves. M_3 on the other hand can still improve the social welfare by scanning only t_1. The MCCs again send out their marginal utilities, after which M_3 changes its configuration. The MCCs once more send out their configurations. Part c of Fig. 1 shows the final configurations. More rounds may be run in the time remaining after the second round, but no new moves would be made because no MCC has a beneficial move remaining.

4.3. Negotiation Implementation

Algorithm 1 MCC algorithm

```
round = 0
while time spent negotiating is less than the limit do
  if round == 0 then
    send and collect messages describing boundary tasks
    calculate configuration C_i without any C_i^neigh
    send and collect messages describing configurations
    collected configurations become C_i^neigh
  end if
  if C_i or C_i^neigh has changed then
    recalculate best C_i' and marginal utility
  end if
  send and collect messages containing marginal utilities
  if this agent's marginal utility is highest then
    update C_i to C_i'
  end if
  send and collect messages with new configurations
  set C_i^neigh to the collected configurations
  round++
end while
```

Each MCC uses Alg. 1 for negotiating with other MCCs. Configurations and marginal utilities are calculated as defined in Sections 4.1 and 4.2. The algorithm requires no centralized entity. Synchronization is accomplished by requiring MCCs to wait for messages from each neighbor before proceeding, and the negotiation process is terminated by each MCC if it has been running for a pre-specified amount of time.

The algorithm makes the following assumptions. We assume that all agents are cooperative. We can safely assume this as it is the goal of each MCC to maximize the social welfare. A second assumption is that the MCCs will all know when to terminate negotiations and do so at the same time. This can be accomplished by synchronizing the clocks of the MCCs. The algorithm additionally assumes that MCCs know the geometric layout of their neighbors' radars; this is information that need only be sent once, so the assumption is reasonable. One final assumption is that neighbors will always respond, and their responses will get through. For reliable radar nodes and a reliable data transport protocol such as TCP, this is fairly reasonable, but to make the algorithm more robust, one could add a timeout. If no response is received within a certain amount of time from an MCC, that MCC's neighbors simply remove the non-responsive MCC from their lists of neighbors. Thus this type of negotiation allows for fault tolerance should there be a failure of communication. One assumption we do not make is that MCCs have the same representation of tasks. Since utility calculations are based on scanning strategy, an MCC does not need to know how its tasks relate to its neighbor's tasks.

4.4. System Properties

Proposition 1. *If an MCC makes a move, all of its neighbors keep their configurations.*

Proof. An MCC M_i only makes a move if its marginal utility is greater than its neighbors', or the marginal utilities are the same and M_i's index is highest. Each of M_i's neighbors receives M_i's marginal utility, so each sees that either its own marginal utility is less than M_i's or its index is lower than M_i's. Therefore, if M_i makes a move, none of its neighbors do. □

The above property is necessary because M_i calculates its marginal utility as $U\left(T_i, \{C_i'\} \cup C_i^{neigh}\right) - U\left(T_i, \{C_i\} \cup C_i^{neigh}\right)$. If C_i^{neigh} also changes and becomes $C_i^{neigh\,\prime}$, then the actual change in social welfare is $U\left(T_i, \{C_i'\} \cup C_i^{neigh\,\prime}\right) - U\left(T_i, \{C_i\} \cup C_i^{neigh}\right)$. It may be that this new difference is negative even if the original difference is positive.

As a sidenote, the possibility that the new difference could in fact be greater than the old difference leads to interesting possibilities. For instance, if multiple MCCs have moves that by themselves are beneficial, it may be worthwhile to begin a mediation process such as that used in [9] to determine the possible benefits of performing multiple moves simultaneously. The usefulness of such a technique would depend on how often conflicts arise between potential moves, how often it is beneficial for multiple moves to be made together, and how often they are all eventually made. We aim to achieve a similar goal as mediation in our experiments in which we change when an MCC is allowed to make a move.

Proposition 2. *We may guarantee that for the same set of tasks, we will never do worse with negotiation than without negotiation.*

Proof. Suppose an MCC M_i changes its scanning strategy from C_i to C_i' at some point during negotiation, thus changing the global configuration from C to C'. The change in social welfare is $U(T, C') - U(T, C)$, which is $\sum_{t_j \in T} U(t_j, C') - \sum_{t_j \in T} U(t_j, C)$. All of these terms cancel except for those t_j for which $U(t_j, C') \neq U(t_j, C)$. Since the only change to C was from M_i's configuration change, the only tasks for which this inequality could be true are those which M_i is capable of scanning, which is T_i.

Thus the change in social welfare is $U(T_i, C') - U(T_i, C)$. But since the only tasks involved here are those of M_i, only a subset of the global configuration has any affect on this value. In particular, any MCC capable of scanning a task $t_j \in T_i$ must be a neighbor of M_i because t_j lies in a region also scannable by M_i. Therefore the subset of the global configuration affecting tasks in T_i is C_i^{neigh} together with M_i's configuration.

This now makes the change in social welfare $U\left(T_i, \{C_i'\} \cup C_i^{neigh}\right) - U\left(T_i, \{C_i\} \cup C_i^{neigh}\right)$. This is, however, the definition of $U^{marg}(C_i')$. MCCs only make moves when the marginal utility is positive, so the change in social welfare from M_i's move must be positive. Thus any changes that arise from negotiation can only result in an increase in social welfare. □

Proposition 3. *Alg. 1 may complete without finding the optimal solution.*

Proof. Fig. 9 depicts a scenario where negotiation results in a sub-optimal allocation. The MCCs would both be better off scanning the pinpointing task t_2, but neither MCC has a positive marginal utility until the other commits to scanning the pinpointing task. Therefore, M_1 continues scanning t_1 and M_2 continues scanning t_3. □

Proposition 4. *If at least one MCC has a positive marginal utility for a round, a move will be made in the round.*

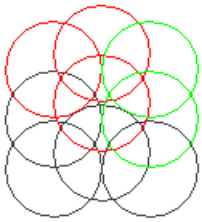

Figure 2. Radar layout used for experiments

Proof. Let M_i be the MCC with highest marginal utility. If there are multiple MCCs tied for the highest marginal utility, let M_i be the one with highest index. Since there is an MCC with positive marginal utility, M_i must also have positive marginal utility. M_i cannot be required to keep its configuration based on its neighbors marginal utilities and indices based on the way we selected M_i. Since M_i has positive marginal utility and no restrictions from making a move, it makes the move. □

5. Results

5.1. Experimental Settings

Here we examine the effectiveness as well as the cost of negotiation embodied in Alg. 1. The first set of results were obtained using the radar arrangement depicted in Fig. 2, in which there are nine radars divided among three MCCs. Each trial performed for these results was a simulation of five thirty-second system cycles. Because of the possibility of weather-events going unseen, the variations in results between trials could be large, but averaged over 5,000 trials, the resulting graphs are quite smooth. For these trials, negotiation was allowed to proceed until a round concluded without any move as opposed to running rounds until a time limit. The purpose of this was to see for how many rounds the negotiation would ideally be allowed to run. All results given are as a function of the number of tasks in the boundaries between MCCs. Specifically, this measurement is of the number of tasks which are scannable by two or more MCCs out of 12 total tasks.

For comparison, the results in Fig. 3 show the performance of a system which allows cooperative scanning of pinpointing tasks if it happens to occur but does not allow negotiation for coordinating actions. So for example, if two MCCs happen to scan the same pinpointing task, they may share their data to get a good result for that task. Also shown are the results for a system for which MCCs are separate entities, which do not interact with each other. This technique has no cooperative scanning, so the values for a boundary tasks are counted differently; it is the highest value received

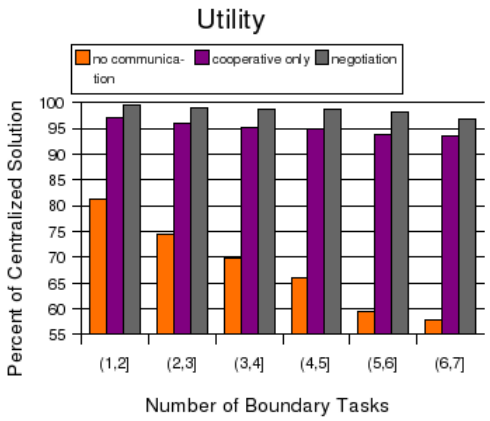

Figure 3. The performance as compared to a centralized system for the nine radar system

from a single MCC. In this way there is no double counting of boundary tasks. The percent drop in utility shown for each technique is with respect to a centralized system, meaning all nine radars belong to a single MCC.

We also include results in Fig. 5 and Fig. 6 for a system for which there is a total of 50 tasks over 49 radars, divided into 12 MCCs. We do these experiments to see how well the negotiation protocol scales up as we add more tasks, radars, and MCCs. We also use these trials with larger numbers of MCCs in order to evaluate the effects of the concurrency we lose as a result of not allowing neighboring MCCs to make moves in the same round. We compare the results for the regular negotiation protocol to those of the two modifications to the protocol described in Section 4.2. Again, the first modification allows any MCC with positive marginal utility to make a move. The second modification has MCCs do a more advanced check of neighbors' moves before making a move.

5.2. Observation I

The negotiation provides a result generally within 5% of that of a centralized system as shown in Fig. 3. Because of Prop. 3, the performance is still lower than the centralized system, though. While it is clear from Fig. 3 that cooperative scans alone can help out considerably as compared to the system with no cooperative scanning, negotiation provides an extra advantage. By reducing redundant scanning and setting up additional cooperative scans, negotiation brings the performance of a decentralized system near that of a centralized one.

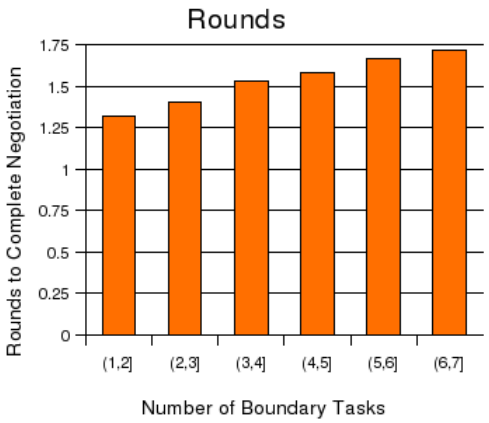

Figure 4. The number of rounds required to complete negotiations with nine radars, three MCCs, and 12 tasks

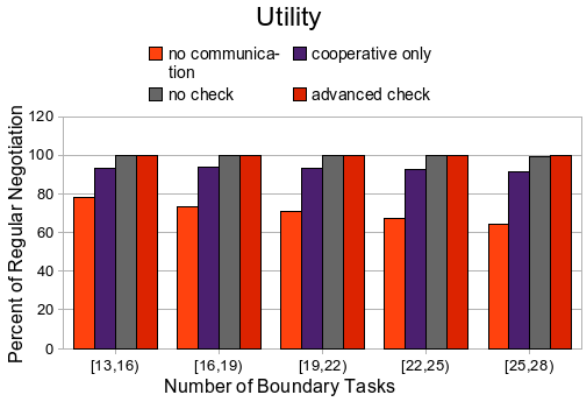

Figure 5. The performance as compared to the standard negotiation protocol for the 49 radar system

5.3. Observation II

Based on the number of rounds in Fig. 4, we can see that the number of optimizations which must be performed by each MCC is quite low. Looking back at the algorithm, one can see that the maximum number of calculations each MCC will have to do is equal to the number of rounds in the negotiation plus one. For this layout of MCCs, the number of calculations equals the number of rounds plus one. The reason for the equality is that all MCCs neighbor all others, so if none of them make a move, the negotiation simply terminates. In the general case, an MCC and its neighbors could all keep their configurations, resulting in some MCCs not having to recalculate their configurations in the following round. Additionally, the number of messages each MCC needs to send is linear in the number of rounds that are run. Messages are broadcast by MCCs a fixed number of times each round.

5.4. Observation III

As noted in the previous observation, the number of optimizations required of each MCC is approximately the number of rounds in the negotiation. Fig. 4 shows that the number of rounds needed is generally fairly low, at least for this setup of MCCs and tasks. Because each MCC is a neighbor of the other two MCCs in the example used, the number of moves made in the negotiation is equal to exactly the number of rounds minus one. What this suggests is that for this setup, not many moves are needed. For larger numbers of radars and MCCs, the number of moves goes up, but the neighbor relation does not form a $|M|$-clique, so multiple moves can be made in the same round. Thus the number of rounds needed would remain fairly low. Even in trials with 49 radars, the number of rounds needed is generally under four, as shown in Fig. 6. The ramifications of this are that few optimizations are required of each MCC, making negotiation a reasonable option for a real-time system.

5.5. Observation IV

The utility for the standard negotiation protocol is almost identical to the utilities for the two modifications to the protocol that were described in Section 4.2, as shown in Fig. 5, but the number of rounds varies. It is not surprising that the ability for neighboring MCCs to make simultaneous moves does not increase the utility. The modifications allow for simultaneous moves given that each MCCs wants to make its move to begin with, but the modifications do not allow reasoning during marginal utility calculations about what combinations of moves would be beneficial. Thus situations such as those illustrated in Fig. 9 are not improved by the modifications. That would require additional mechanisms.

Although the modified protocols provide similar utilities, we can see in Fig. 6 that the number of rounds needed can be quite different. A negotiation that uses a more advanced check requires fewer rounds as it increases concurrency in the network by letting neighbors make moves during the same round. The same often also happens with the modification that does no check of neighbors' marginal utilities, but there is another effect which brings the average number of rounds up significantly. Because in general we cannot guarantee that the social welfare increases when neighboring MCCs make simultaneous moves, neighboring MCCs can get into situations in which they loop between con-

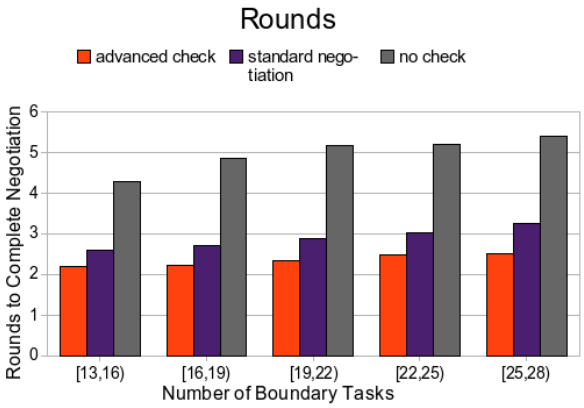

Figure 6. The number of rounds required to complete negotiations with 49 radars, 12 MCCs, and 50 tasks

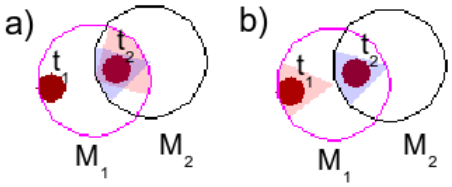

Figure 7. Negotiation can eliminate redundant scanning.

figurations. We cap the number of rounds at 20 to keep these looping agreements from continuing indefinitely. Another way to prevent these loops from continuing is to use a probability lower than one for MCCs to make simultaneous moves given that they both have positive marginal utilities. These results demonstrate that there are ways to reduce the number of rounds needed for negotiation, and even for a system as large as 49 radars, we can keep the number of rounds quite low.

6. Scenario Analysis

There are particular situations for which negotiation provides a particularly large advantage. One such situation is demonstrated in Fig. 7. In this example, t_2 has a higher priority than t_1 and neither task is pinpointing. M_1 and M_2 both scan t_2 when no negotiation is used. Without negotiation M_1 assumes it is better to scan the higher priority task. Since t_2 is not a pinpointing task, M_1 would be better off scanning t_1. Negotiation allows M_1 to take M_2's configuration into account when comparing its options, and thus it

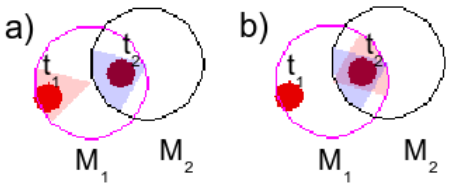

Figure 8. Negotiation can set up cooperative scans.

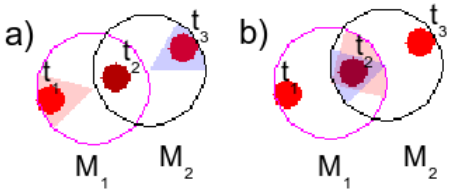

Figure 9. No change in configuration from a to b can be made in a single negotiation round.

leads to both tasks being scanned. This sort of improvement is based on the reduction of redundant scanning.

Another situation in which negotiation provides an advantage is in setting up cooperative scans. In Fig. 8, t_1 is a non-pinpointing task, and t_2 is a pinpointing task. t_2 gets a lower value than t_1 when its volume is scanned with only one radar but a much higher value than t_1 when it is scanned by multiple radars. M_1 scans t_1 when there is no negotiation because it does not know M_2 is scanning t_2. Knowing M_2's configuration, though, it becomes much more beneficial for M_1 to switch to scanning t_2 so that the two MCCs may cooperatively scan t_2. Negotiation allows the MCCs to set up cooperative scans that would not have occurred otherwise.

Despite the advantages of negotiation, there are still drawbacks. One example is shown in Fig. 9. In this situation, we would want both MCCs scanning the pinpointing task t_2 as shown in part b of the figure, as opposed to scanning t_1 and t_3. Neither MCC can make a move until the other does. A move for either MCC to t_2 would result in a drop in social welfare. In general, the type of situation that this negotiation cannot deal with well are those for which temporary drop in social welfare must be endured in order to achieve a greater gain later.

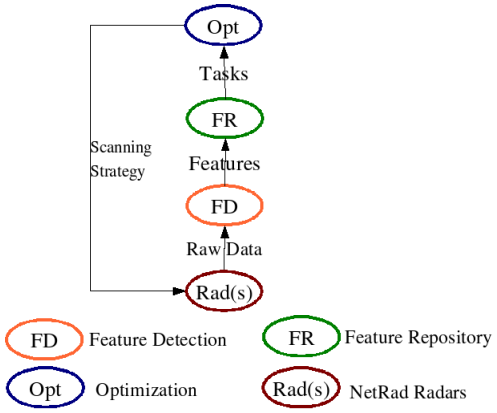

Figure 10. The centralized model

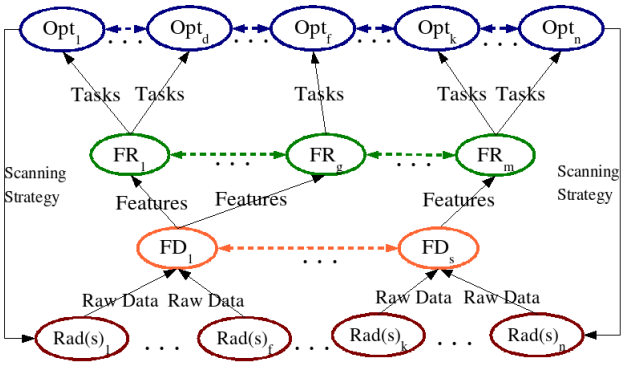

Figure 11. A generalized hierarchy

7. Real World Usage

Due to the promising results for this negotiation protocol in simulation, it has been implemented for the physical radar network as well. The removal of abstractions has some interesting effects on the protocol and invites some new lines of investigation. Most notably, in the physical system the control flow is more tiered than in the simulator. The optimization and the feature repository are separate entities as is the feature detection process. In normal, centralized operation, only one of each of these components runs, and together they form the MCC as depicted in Fig. 10. This is also the model used by the simulator, except that multiple MCCs may run concurrently. We now have the option, though, of using more general hierarchical structures of feature detection algorithms, feature repositories, and optimization modules shown in Fig. 11.

Inherent in this freedom of structure is the possibility to create self-organizing structures and self-repairing structures. By adapting the structure based on emerging weather conditions, we can reflect the set of tasks in the structure of the MCC, thereby reducing the number of boundary tasks. In this way, the system may be able to organize in ways that make cooperation easier or reduce the need for communication. Dynamic structures also allow for increased fault tolerance and the ability to compensate for varying computational loads throughout the system.

Also made clear from the components of the real system is that data can be shared at various levels of abstraction. In the simulator, all weather-phenomena represented at the task level, but in reality there is a spectrum ranging from the raw radar data to the tasks. MCCs can share raw data or some compressed form of it, features resulting from the detection algorithms, or tasks resulting from clustering the features.

For pinpointing tasks, data from multiple radars must be combined because individual radars can only measure velocity in the radial direction. The most straightforward way to combine the velocities is at the level of raw data by adding components of the velocity vector field. Doing so requires MCCs to share raw data, which is represented by the dashed arrows between detection algorithms in Fig. 11.

There are a number of ways we can cut down on the amount of information that needs to be sent when transmitting raw data. The general techniques are restricting the volumes for which any data is shared and compressing the raw data by averaging over larger volumes. Obviously, data corresponding to volumes that can only be scanned by one radar need not be shared; only the boundary regions must be shared. We can further restrict what data need to be sent by using some additional domain knowledge. Using indicators such as the single radar detections, MCCs can get a good estimate of where possible regions of interest are, and MCCs can push and pull data based on these regions of interest. Similarly, based on the level of interest of a region, the data can be compressed at varying levels of granularity. Using techniques such as these, multi-radar detections can be done efficiently in a decentralized MCC system.

An alternative to sharing the raw data for pinpointing tasks is to use the radial components alone to do velocity detections on a per MCC basis. MCCs can then share these detections and merge them. This technique provides a savings in communication cost, but it has its tradeoffs[1]. For one, the radial component for even a very strong wind could be very small, so to ensure there are features for those winds an MCC must have a low threshold for adding velocity features. Thus, there could be a lot of features which must be communicated and processed. Another problem is in actually combining these radial velocity detections into a set of detections resembling what a centralized detection algorithm would produce.

By adding the components of features that roughly overlap, MCCs can get a general picture of the combined veloc-

[1]Special thanks to Eric Lyons for his analysis.

ity detections. For the analysis currently done in the MCC system, there is no major loss. The problems come in when a more detailed vector field is needed. For instance, the best indicators for predicting how a storm will change are the updrafts, which require measurements of vorticity. Thus, for more advanced detecting and predicting, raw data in some form will have to be shared. For simple detection of velocity features, data can be shared and combined at the feature level.

For non-pinpointing tasks, what gets detected is reflectivity, which is a scalar quantity. Therefore, single radar data is sufficient to perform the necessary detections. MCCs can share these detections with each other to get a better idea of the weather in the corresponding volumes, but no altering of these features is needed in the way it was when combining radial component velocity detections. MCCs still benefit from sharing these detections, though, as one MCC may not have scanned a volume as recently as another, and certain radars may experience more attenuation then others when looking at particular volumes.

Once MCCs have all of their features, they can begin negotiation. Before negotiation, MCCs share data at the feature level (depicted with dashed arrows between feature repositories) as opposed to sharing all of the data from the radars because it is much less data to be sent, but it is not as coarse as the task level. It also carries the additional benefit of allowing each MCC to easily decide its own regions of interest. Finally, we depict the communication that occurs during the negotiation with the arrows between optimization nodes in Fig. 11. The negotiation consists of messages containing marginal utilities and scanning strategies. These messages are of negligible size in comparison to the raw radar data.

We can save a lot in communication and processing costs by not transporting all of the raw data, but in doing so, we run the risk of tasks having different representations in different MCCs. This is generally not of great significance as the negotiation operates at the level of scanning strategies rather than individual tasks; however, the differences in representation could conceivably change the decisions MCCs make. Clustering of features into tasks in a decentralized network can result in two smaller tasks where the central clustering would return one. This raises two issues. The first is that if the centroid of a task is outside of a radar's range, the radar gets no value for that task; splitting a task results in one of the tasks having a centroid in range. Thus, decentralized clustering promotes scanning the edges of tasks if it can, which may actually be a better design than the original centroid constraint. The main concern, though, is ensuring that MCCs do not doubly count what was previously one task because it becomes two. Intuitively, we should not have problems because each MCC considers only one of these subtasks when computing the marginal utilities; however, more testing is required to determine the effects of task decomposition.

To evaluate the effectiveness of negotiation outside of the simulator, we need some way to compare the centralized and decentralized systems when run in the same weather scenarios. We can do so by using archived or emulated radar detections. To the extent that is possible with the amount of data in those detections, we can create a set of oracle tasks from the data. These oracle tasks represent the tasks a radar network would perform if it had perfect information and unlimited scanning resources. We can run the system in a variety of modes and compare the effectiveness of the resulting scans with respect to the maximum possible value for the oracle tasks. Additionally, we can do these tests in a number of ways. To see how the techniques will perform given identical tasks, we can use the oracle tasks as the task input. To see the effects of clustering, we can use the same features as input but allow the clustering to be done independently of the oracle clustering. Finally, if we want to see downstream effects, we can selectively input features based on the scans in the previous heartbeat. Comparing against an oracle gives us a way to measure effectiveness with real weather data.

8. Conclusions and Future Work

We introduced a hillclimbing approach to negotiation for a time-restricted domain based upon the DBA algorithm. We applied this algorithm to an abstract simulation of NetRad radars to show its usefulness in distributed radar networks. Based on the results obtained, negotiation seems to do an excellent job of approaching the performance of a centralized system. Additionally, the low number of optimizations required of each MCC makes it a great candidate for a decentralized technique because of the time-restricted nature of weather radars.

We have implemented this negotiation protocol in the actual NetRad system and have begun tests on this system. Some interesting questions arise in terms of sharing task information. We would like to investigate the ramifications of different data sharing techniques and choose a technique which best balances the amount of data being sent and the cost we incur in having differing task representations.

There are also some possible changes that could be made to the negotiation protocol. For example, an anytime version of the optimization could be used in order to allow negotiation to proceed at varying degrees of precision. Also, so far we have only focused on accepting agreements which increase the social welfare. By doing so we guarantee we do not lose value; however, we may often get stuck in a poor local optimum. If this is the case, it may sometimes be beneficial to accept a poor agreement to help in the long run. Applying simulated annealing to this application would be a

possibility to pursue. Another possibility to look into would be multilateral negotiation in order to allow neighboring MCCs to make moves that they otherwise would not consider. As opposed the the methods explored in Section 5 for allowing neighbors to make simultaneous moves, we would be looking to alleviate situations like the one shown in Fig. 9 rather than simply increasing concurrency. A multilateral approach could conceivably result in configurations which would not otherwise be reached. By modifying the negotiation in various ways, we hope to be able to get closer to the optimal configuration.

References

[1] J. Chen, C. Zang, W. Liang, and H. Yu. Auction-based Dynamic Coalition for Single Target Tracking in Wireless Sensor Networks. *The 6th World Congress on Intelligent Control and Automation*, 1:94–98, 2006.

[2] V. D. Dang, R. K. Dash, A. Rogers, and N. R. Jennings. Overlapping Coalition Formation for Efficient Data Fusion in Multi-Sensor Networks. In *Proc. of the 21st Nat. Conf. on Artificial Intelligence*, pages 635–640, 2006.

[3] U. Endriss, N. Maudet, F. Sadri, and F. Toni. Negotiating socially optimal allocations of resources. *Journal of Artificial Intelligence Research*, 25:315–348, 2006.

[4] B. Horling and V. Lesser. Distribution Strategies for Collaborative and Adaptive Sensor Networks. In *Proc. of the Int. Conf. on Integration of Knowledge Intensive Multi-Agent Systems*, pages 497–504, 2005.

[5] B. Horling, R. Mailler, and V. Lesser. Farm: A Scalable Environment for Multi-Agent Development and Evaluation. In *Advances in Software Engineering for Multi-Agent Systems*, pages 220–237. 2004.

[6] Y. Kim, R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Exploiting Locality of Interaction in Networked Distributed POMDPs. In *2006 Spring Symp. Distributed Plan and Schedule Management*, pages 41–48, 2006.

[7] J. F. Kurose, E. Lyons, D. McLaughlin, D. Pepyne, B. Philips, D. Westbrook, and M. Zink. An End-User-Responsive Sensor Network Architecture for Hazardous Weather Detection, Prediction and Response. In *Proceedings of the Second Asian Internet Engineering Conference, AINTEC*, pages 1–15, 2006.

[8] P. Lou. Negotiation-Based Task Allocation in an Open Supply Chain Environment. *Proc. of the I MECH E Part B Journal of Engineering Manufacture*, 220:975–985, 2006.

[9] R. Mailler, V. Lesser, and B. Horling. Cooperative Negotiation for Soft Real-Time Distributed Resource Allocation. In *Proc. of 2nd Int. Joint Conf. on Autonomous Agents and MultiAgent Systems*, pages 576–583, 2003.

[10] G. Mainland, D. C. Parkes, and M. Welsh. Decentralized adaptive resource allocation for sensor networks. In *Proc. of the 2nd USENIX Symp. on Networked Systems Design and Implementation*, 2005.

[11] D. McLaughlin, V. Chandrasekar, K. Droegemeier, S. Frasier, J. Kurose, F. Junyent, B. Philips, S. Cruz-Pol, and J. Colom. Distributed Collaborative Adaptive Sensing (DCAS) for Improved Detection, Understanding, and Prediction of Atmospheric Hazards. In *9th AMS Symp. on Integrated Observing and Assimilation Systems for the Atmosphere, Oceans, and Land Surface*, January 2005.

[12] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed POMDPs: A Synthesis of Distributed Constraint Optimization and POMDPs. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence*, pages 133–139, 2005.

[13] T. Sandholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. *11th Nat. Conf. on Artificial Intelligence*, pages 256–262, 1993.

[14] T. W. Sandholm. Contract types for satisficing task allocation: I theoretical results. In *Proc. of the AAAI Spring Symp.: Satisficing Models*, pages 68–75, 1998.

[15] L.-K. Soh and C. Tsatsoulis. Reflective Negotiating Agents for Real-Time Multisensor Target Tracking. In *Proc. of the Int. Joint Conf. on Artificial Intelligence*, pages 1121–1127, 2001.

[16] M. Yokoo and K. Hirayama. Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction Problems. In *Proc. of the 2nd Int. Conf. on Multiagent Systems*, pages 401–408, 1996.

[17] M. Younis, K. Akkaya, and A. Kunjithapatham. Optimization of Task Allocation in a Cluster-Based Sensor Network. In *Proc. of the 8th IEEE Int. Symp. on Computers and Communications*, pages 329–334, 2003.

[18] M. Zink, D. Westbrook, S. Abdallah, B. Horling, E. Lyons, V. Lakamraju, V. Manfredi, J. Kurose, and K. Hondl. Meteorological Command and Control: An End-to-end Architecture for a Hazardous Weather Detection Sensor Network. In *Proc. of the ACM Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services*, pages 37–42, 2005.