A Market-based Adaptation for Resolving Competing Needs for Scarce Resources

Rui Wang, Tracy Mullen, Viswanath Avasarala, John Yen
College of Information Sciences and Technology,
The Pennsylvania State University,
University Park, PA 16802 USA
{rwang, tmullen,vavasarala jyen}@ist.psu.edu

Abstract

The dynamic nature of many real-world domains (e.g., military, emergency first response and hurricane relief, etc) requires adaptive resource allocation to respond to changes in the environment that trigger additional resource requirements. Since the total resources are limited, there are often conflicts among various tasks regarding their resource needs. Thus, resources must be reallocated in order to maximize global utility for the current situation. This problem is further complicated when scarce resources are owned by distributed teams, each of which needs to allocate resources among tasks assigned to them, because each team has limited information about the other teams' resources and states. In this paper, we propose a market-based approach that uses an agent-based auction mechanism to enable teams to communicate and coordinate their utility information about possibly competing resource needs. As a result, the teams can collaboratively assess trade-offs among competing needs to allocate resources efficiently.

1. Introduction

Uncertain, dynamic environments afford challenging domains for multi-agent systems (MAS) applications, especially when they face the situation with scarce resources. Although centralized mechanisms to solve the optimization problem can provide quality-guaranteed solutions, they are often not scalable and can run into a "bottleneck" problem. Distributed solutions can be more flexible and more reliable; although those benefits are obtained via extra efforts in coordinating multiple agents. The coordination problem becomes more challenging when scarce resources are distributed among different teams initially, since each team has limited knowledge about the other teams' resources and states. This paper presents recent research on designing and implementing a market-based mechanism to optimize the reallocation of limited resources dynamically among distributed teams by making trade-offs among competing resource needs based on exchanging utility-based price information.

Market mechanisms (e.g., auctions) have been widely applied to software agents in the e-business. Under the right conditions, each individual agent tries to maximize its own profit in the market, which under the right conditions leads to a globally efficient outcome [4]. Market architectures connect sellers with buyers using price to provide a means for low-cost communication of value. There are several market-based implementations for MAS successfully demonstrated in simulations [4, 15] and physical robot-based applications [9]. For those market-based systems mentioned above, they have a common feature that a centralized auctioneer is used to allocate resources among agents or teams. In contrast, our system doesn't have a fixed auctioneer agent but each participating agent can dynamically take the role of auctioneer as well as buyer and seller. While the auctioneer also often serves as a trusted third party, since all the agents are cooperative in our system, this aspect is not a concern.

There are two different types of agents in a team. One type is task agents, who are responsible for executing a task plan when its preconditions and resource requirement are met. The other is a coordinator agent, who tries to not only satisfy the resource requirement of an assigned task for the team it belongs to, but also to maximize the global utility of scarce resources. One coordinator collaborates with coordinators in other teams to solve this resource optimization problem dynamically. Here we use a market-based mechanism to enable coordinator agents to assess competing resource needs and adaptively allocate the needed resources efficiently.

In this paper, we investigate how an agent coordinator leveraged on R-CAST architecture can correctly evaluate the utility of one resource item or a bundle of resources to its own plan, and how to reach an agreement on reallocating scarce resources by sharing the evaluation information with other planners. Background knowledge on R-CAST agent architecture and the market-based optimization are given in Section 2. The resource optimization mechanism is elaborated in Section 3. The problem domain, simulation environment and scenario design are described in Section 4. Some preliminary

experiment results are presented in Section 5, and Section 6 concludes the paper.

2. Background

2.1. The R-CAST Agent Architecture

The R-CAST agent architecture [5] is built on top of the concept of shared mental models [2], the theory of proactive information delivery [7], and the recognition-primed decision (RPD) model [11]. A collaborative RPD decision process was implemented in R-CAST with features such as relevant information sharing, decision process monitoring, and decision adaptation.

The major components of R-CAST are presented in Figure 1. The RPD module uses domain knowledge, past experiences and the current situation awareness to produce a new or adapt an existing decision.

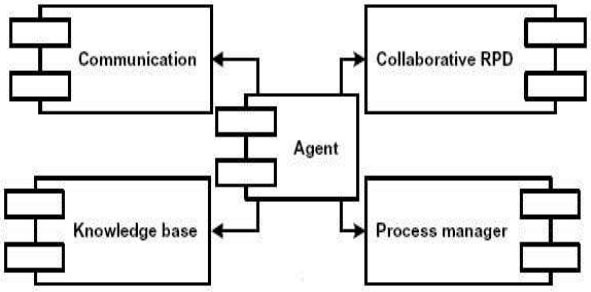

Figure 1 [4]: The R-CAST agent architecture

This study takes advantages of two important modules in R-CAST, which are the knowledge base (KB) and the process manager. The knowledge base is a forward-chaining rule-based system, which enables the agent to maintain what it believes regarding the external world and the other agents. The unique feature is that it is able to reason about missing information relative to the current tasks, and proactively find ways to satisfy the inferred information requirements. Also, it is proof-preserving in the sense that the proof-trace of a query is preserved, which is important when information link-analysis is needed, and can be very useful in planning for information gathering in subsequent activities. The process manager manages the templates of predefined plans, each of which contains preconditions, termination conditions, effects, and a process body. In addition, we extend the plan templates to include resource requirements for this study, which will be described later in Section 3. The process manger can instantiate plan instances from appropriate templates. The execution of plan instances is scheduled by the process manger based on the constraints associated with the instances and the KB's current state.

2.2. The Market-based Optimization

In Section 1, we mentioned some related work in market-based optimization. Another example is file allocation in a distributed computer system [14]. Market-based approaches enable a natural decomposition of the problem, which is well suited for distributed environments. Here we briefly review how market mechanisms can be applied to such distributed resource allocation problems.

Auction mechanisms determine who should get the goods and at what prices. In many auction environments, items being traded are complements. For example, a left shoe and right shoe are complements that have greater utility acquired together than either acquired individually. Combinatorial auctions [3, 10, 13] allow bidders to bid on combinations of items, and thus directly express these synergies. Each bidder can bid on item bundles, and the auctioneer chooses the set of bids that maximizes the total value.

However, selecting the winning bids, call the general winner-determination problem (WDP), is NP-complete for combinatorial auctions [3, 17]. There are two types of approaches to optimal winner determination in the general case. The first one is using powerful general-purpose mathematical programming software. A general optimal winner determination problem can be formulated as a mixed integer problem so it can be run directly on standard highly optimized software packages such as CPLEX [1]. The second approach is developing search algorithms specifically for winner determination, combining AI search techniques and domain-specific heuristics. Recently researchers had made a great deal of progress in developing algorithms solving WDP efficiently in this way. For example, the BOB algorithm [16, 17] can solve auctions involving hundreds of items and thousands of bids within 10 seconds. Since our research doesn't focus on designing algorithms to solve WDP more efficiently, here we adopt the first approach and use optimization software based on integer programming to optimize the resource allocation. Thus, different agents can place bids on bundles of resource items they need. And the auctioneer decides winners of this combinatorial auction to maximize the utility of the total resource items.

A great deal of research has been devoted to solving the resource allocation problem via auction mechanism explained above. However, there are some hidden assumptions simplifying this problem. First, most of them start directly with resource allocation without considering alternative options of resources to complete a task. Second, the value of each task's expected utility function

only has two values: zero and another number. In other words, the expected outcome of a task is either success or failure and there is no reward for partially completed tasks. Third, resource allocation doesn't adapt to changes in the situation. Our work addresses those issues and uses a market-based approach to allocate resources adaptively.

3. Resource Management

Resource allocation plays an important role in successful plan execution. Some resources are physical, such as materials, machines or money. Some resources are more abstract facts like possibilities and availabilities. Different from information, which can be shared by multiple agents at the same time, resources can be consumed only by one agent at a time. In this paper, we focus on scarce resources, which are in limited supply so that competing needs for them may arise. Many planning systems have integrated the scheduling of resources by including some sort of Constraint Satisfaction Problem or Linear Programming problem solver [6, 12, 13]. In all these cases, the "best" or optimal solution may mean maximizing profits, minimizing costs, or achieving the best possible quality, given scarce resources.

3.1. Resource Representation

An agent coordinator maintains its belief about the status of all known resources in the knowledge base. In order to efficiently organize such knowledge, there are two structures designed for representing resources. The first is "resource class", which is used to describe multiple instances of the same type resources (i.e., instances with the same functionality) or uncountable resources which may be measured (e.g., 5 gallons of gasoline). Also, a resource class has a *type* field indicating whether this kind of resource is consumable or reusable. The second is "resource instance", which represents an individual resource item. A resource instance is associated with a class name that it belongs to. It also has a unique id and domain-specific attributes. It provides more detailed information such as the *ownership*, its current *status* and what plans may use it within what time periods. Formats of those two structures are listed below:

```
(Resource_Class (class ?n)
   (number/amount ?x)
   (type ?t)
   (attributes + values)
)

(Resource_Instance (class ?n) (id ?i)
(attributes + values)
(ownership ?owner)
(status available/occupied/reserved)
((plan ?id)(begin_action ?ida)(end-
   action ?idb))
 )
```

As mentioned in the previous section, a predefined plan template specifies its requirement for resources. In order to ensure that the instantiated plan can be executed successfully, this minimum resource requirement must be fulfilled. In other words, the resource requirement can be viewed as a set of constraints just like the preconditions to satisfy. Let's take a look of the example of a plan template below:

```
(plan deliver_to(?dest ?obj)
    (res-requirement (helicopter 2)(pilot
2)
     alternatives (1 (truck 3)(driver 3))
    )
    (termcondition (current_loc ?dest =))
    (utility 500)
    (process
           (seq
              (load ?obj)
              (move_to ?dest)
              (unload ?obj)
             )
    )
 )
```

This plan template defines a task "deliver the object y to the destination x " when the variable *?obj* is bound to an object y (e.g., a pile of sandbags) and the variable *?dest* is bound to a place's name x (e.g., a leaking levee at New Orleans). The minimum resources required by this plan are two helicopters and two pilots or three trucks and three drivers. The first set of resources is the default option, while the second one is an alternative to satisfy the resource requirement. It is easy to understand why the default option is preferred since it's much faster to deliver the object to the destination using helicopters than using trucks, especially when the delivery task is in an urgent situation.

Similar to how information needs are generated in a CAST agent [8], a coordinator agent built on top of R-CAST can infer missing resources by comparing the resource requirement defined in the target plan and the current information on resource states in its knowledge base. As soon as the coordinator figures out what resources are missing to perform a task for its team, it will send out requests for those missing resources to other

coordinators whose teams may be potential providers, and trigger the process of market-based resource optimization.

3.2. Market-based Resource Optimization

3.2.1 Resource Auctions

The market for agent coordinators to reallocate resources is based on auctions for resources that multiple tasks are competing for. When an agent coordinator detects missing resources for its team to carry out a scheduled task, it generates a request for a set of those resources denoted by A with a maximum price that it is willing to pay for getting them and sends the request to the auctioneer. The maximum price is calculated based on the instantiated task's estimated utility, which will be further explained later. Then, the auctioneer forwards the request to all other coordinators whose teams may possess the requested resources. Assume each coordinator receiving such a request has a set of its own resources is B .and let $\Re = \mathrm{A} \bigcap \mathrm{B}$. Each coordinator generates a bid for each element in its power set of $\Re$ except the empty set. The bid price is decided by estimating how the resource(s) affects the coordinator's own team completing its assigned task and the task's expected utility. All bids from all potential providers are sent to the auctioneer, who determines the winning bid(s). The role of an auctioneer can be dynamically assigned to any coordinator based on their current work load, since usually the optimization process involves much computational cost.

3.2.2 Bid Estimation

Given an instantiated task (a plan instance) T_i , there are two major factors affecting its expected utility $EU(T_i)$:

- The probability that T_i is fulfilled to certain degree (let S_t denote such a status of the task): $P(S_t)$
- The utility of T_i at the status S_t : $U[T_i(S_t)]$

The utility of a partially completed task can be easily calculated based the percentage of completing given goals. However, the probability that the task T_i reaches a status S_t depends on many possible factors like conditions, resource requirements, or even uncertainty. Let

$$U[T_i(S_t)] = func(S_t), t = 1,...,m$$

$P(T_i \ \ in \ \ S_t) = func(C,R)$, C denotes conditions and R denotes resource status when T_i is executed, and T_i ends up into S_t , then the expected utility of T_i is:

$$EU(T_i) = \sum_{S_t} U[T_i(S_t)] \times P(T_i \ \ in \ \ S_t) = \sum_{t=1}^{m} f(S_t) \times f(C,R)$$

A simplified version of the stated problem assumes each task will only have two possible cases after executed: succeeded or failed. Suppose in a task T , there are n conditions and m resource requirements. For each condition C_i , the probability that C_i is satisfied is $P(C_i = ture) = P_i$ (i = 1…n); for each resource requirement R_j , the probability that R_j is satisfied is $P(R_j = ture) = P_j$ (j = 1…m).

So the probability that T will succeed is:

P_{succ} = ($\prod_{i=1}^{n} P_i \ \prod_{j=1}^{m} P_j$) (1- P_e), where P_e is the probability of exception, and the probability that T will fail is: 1- P_{succ} .

Suppose the utility of task T is zero if it fails, and the utility of a successfully executed task T is U_t , thus the expected utility of task T is:

$$EU(T) = P_{succ} \times U_t + (1 - P_{succ}) \times 0$$

$$= (\prod_{i=1}^{n} P_i \ \prod_{j=1}^{m} P_j) (1- P_e)$$

The U_t can be determined by combining the base utility value defined in the plan template (e.g., `(utility 500)`)and the variable binding utility rules in the plan instance.

The maximum willing price for missing resources is primarily calculated based on the expected utility of the task. However, when a resource coordinator decides the bid price responding a request, it also has to consider the cost switching resources from its current team to the requesting team.

3.2.3 Combinatorial Auction

We use combinatorial auctions to handle bids for bundled resources. Let B_i denote the bundle of missing resources for a task and p_i is the bidding price for B_i .

Bidders (i.e., resource coordinators who are seeking for missing resources) submit n bids as bundle/price pairs (B_i, p_i). Given the fact that the auctioneer may accept any combination of non-conflicting bids and charge the sum of the associated prices (or called OR bidding), and a decision variable $x_i \in \{0,1\}$ for each bid (B_i, p_i), the WDP for combinatorial auctions becomes the following problem:

Maximize $\sum_{i=1}^{n} p_i \cdot x_i$ subject to $\sum_{i \in Bids(r)} x_i \leq 1$ for all resource items r, where $Bids(r) = \{i \in [1,n] \mid r \in B_i\}$

We implemented Sandholm's CABOB algorithm [17] in the auctioneer agent for solving the WDP. The auctioneer collects all bids and runs the algorithm to get the value of x_i for each bid. The optimal allocation to those bidders is determined based on the result.

In addition, the auction should be done in a timely manner, which means there is a deadline associated with a task's resource requirement. The task must have all required resources before the deadline or it will fail to be executed. Since it could be delayed for a resource seller to report its bids to the auctioneer, when the deadline is approaching, the auctioneer has to make the decision with incomplete information. Even though such an "optimal" solution is not really optimal, it avoids the failure of executing a task, which may result into significant utility loss. Since concurrent resource requests usually have different deadlines, the auction may carry out in an iterative way.

4. Setting Stage for Experiments

In this section we describe the problem domain and scenario design used in the experiments.

4.1. Problem Domain

In a dynamic environment, intelligent agents usually have to face unexpected changes, which will force them to adapt current plans in order to keep the execution from failure. Since those agents are cooperative, they may exchange information and resources in order to maximize the global utility of those plans under the updated situation. The problem studied here is how an agent evaluates the tradeoff so that it can make a correct decision whether to switch its own resources to its teammates or not. The following conditions are typical in the domain being studied: first, the total resources are limited; second, there may be multiple options, each of which has different resource requirements, to accomplish a task; third, there is an initial resource allocation so that different teams have private resources; fourth, there is cost associated with sharing information about own resources to other teams; last, information of utilities (both resource utility and plan utility) is distributed initially.

4.2. A Hurricane Relief Scenario

In this scenario, a Category 5 hurricane hits a Major Metropolitan Area (MMA). Sustained winds are at 160 mph with a storm surge greater than 20 feet above normal. As the storm moves closer to land, massive evacuations are required. Certain low-lying escape routes are inundated by water anywhere from 5 hours before the eye of the hurricane reaches land. In addition to the massive destruction caused by the hurricane itself, there are also areas within the MMA and scattered inland areas that have sustained severe damage from tornadoes that were generated by the storm. Storm surges and heavy rains cause catastrophic flooding to low lying areas. Rainfall from the hurricane, in combination with earlier storms, causes significant flooding in multiple states along the coast. We focus on the following three hurricane relief tasks:

1) Delivering foods to a large group people who have been isolated in a flooded area (priority level: 3, deadline: in 24 hours).

2) Transferring sands bags to a specific place in order to fix a leaking levee (priority level: 4, deadline: in 10 hours).

3) Rescuing a few of persons in a dangerous situation or they will be overflooded soon (priority level: 5, deadline: in 2 hours).

Each task is associated with priority information and the deadline that it must complete before. The priority level contributes to the expected utility of a task and the deadline determines when the task's preconditions and resource requirements must be satisfied. The most critical resource shared among those tasks is a troop of helicopters operated by pilots. One helicopter may switch from one task to another task dynamically. Such a switch is not arbitrary but really depends on many facts in the current situation (e.g., the priority of an emerging task, the task's deadline, and the current location of helicopters, etc).

The scenario is designed as Figure 2 shows. One helicopter H1 is in the process of carrying out the task Deliver_Food (dest, food, deadline), which requires the helicopter to arrive at the destination with loaded food before the predefined deadline. Another helicopter H2 is engaged in a task named Fix_Levee (loc, bags, deadline), which H2 has to transfer sand bags to the specific location to fix the broken dam. Meanwhile, there is an emerging task Rescue_People (dest, safe_place, deadline) which

requires at least one helicopter as the resource to rescue a group people from a very dangerous place before they will be flooded at the estimated deadline. More helicopters are desirable in case that there are too many people to be rescued. Both coordinators A1 and A2, representing tasks Deliver_Food and Fix_Levee respectively, receive the request from the coordinator A3 of the task Rescue_People. Each of them needs to decide whether to switch its own helicopter from itself to the emerging task in order to maximize the global outcomes. In order to avoid redundant resource transferring, the resource requester will also make a decision to select the right provider in case that there are multiple willing providers. In this case, supposing A3 places a maximum willing price 500 on its request for one helicopter: (?h, 500, Rescue_People), and A1 generates a bid for its own helicopter H1 at a price 300: Bid (H1, 300, Deliver_Food) and A2 generates a bid for its own helicopter H2 at a price 400: Bid (H2, 400, Fix_Levee), it turns out the bid from A1 will be accepted finally and H1 will be switched from the task Deliver_Food to the task Rescue_People. The expected global utility will be 900, which is more than the previous expected global utility 700 if we don't consider the switching cost here.

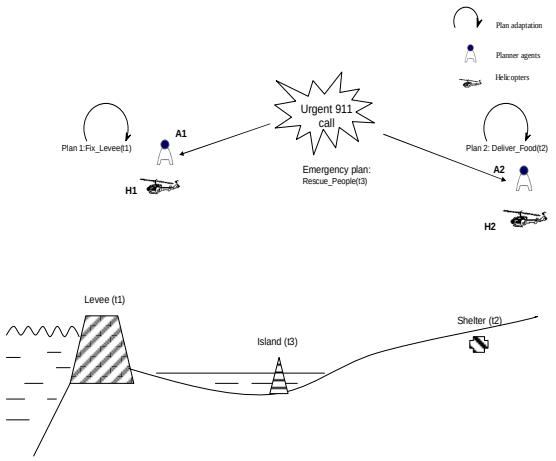

Figure 2: A scenario of hurricane relief

5. Experimental Results & Analysis

We tested the proposed approach in a simulation of hurricane relief scenario. There are three types of tasks which are described in the previous section. Multiple instances of each task are triggered randomly in the simulator. The initial resource allocation is configurable before starting the run. We are using the following configuration for initial resource allocation in the experiment:

Table 1: Initial Resource Allocation

Task:	Deliver_Food	Rescue_People	Fix_Levee
# Instances	1	1	1
Resource instances	r1, r2, r3	r6, r7	r3

The simulation starts with the initial configuration. There are ten total resource instances [r1, …, r10]. As the time goes on, emerging tasks are generated randomly and raise competing resource requests. The total utility of all succeeded tasks are calculated after the process terminates. We use the number of total triggered tasks (successful or unsuccessful) as the control variable to demonstrate how our approach performs as it increases comparing to the other approach that uses a simple "request and reply" way to handle resource requests.

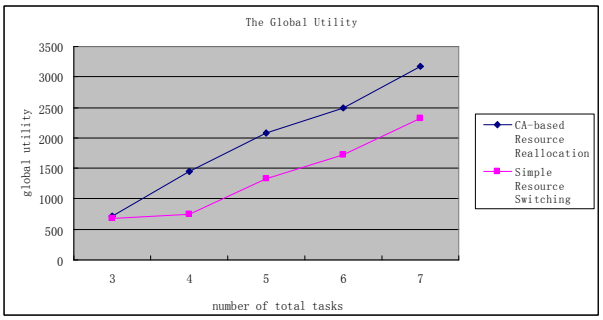

Figure 3: The global utility of different solutions

From the diagram above, we can see that the global utility is significantly enhanced using our solution, while the case that a resource provider always gives out the requested resources doesn't fully utilize the limited resources.

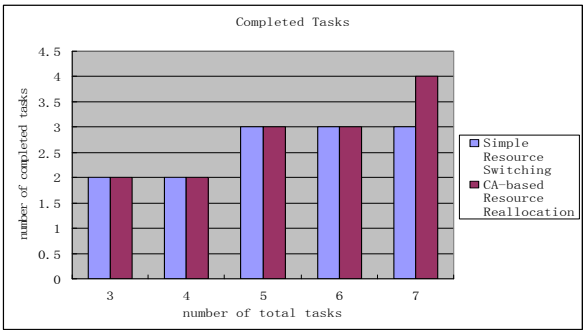

Figure 4: The number of completed tasks

However, from the figure 4, we can see that the number of completed tasks may not improve using the resource optimization comparing to the strategy of switching resources simply upon requests. It is because some task instances may have very high expected utilities. So it is worthwhile to switch resources from several other tasks of low expected utilities to satisfy such a task's resource requirement. Thus, even though the total complete number of tasks may decrease in some cases, the global utility is still maximized.

6. Conclusion

We have introduced a new market-based adaptation for coordinating distributed tasks to solve their competing resource needs and make the maximum utilization of scarce resources. Experimental results in a simulated hurricane relief scenario show that resources can be reallocated dynamically and efficiently through the auction mechanism and generate an optimal solution in a timely manner.

Currently, we are performing further experiments involving more types of resources in a more complex environment, which will show the scalability and robustness of our adaptation mechanism. An interesting issue to be explored is alternative resource reasoning. Alternative resources not only make the solution more flexible, but also help the coordinator choose the most suitable option to compete the task not only for its own sake but also for the global objective. In addition, in order to reduce unnecessary communication costs, we are trying to enable a coordinator to infer who are most likely to possess the needed resources so that it can only send requests to those agents instead of broadcasting to all others.

References

[1] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the 4th International Conference on Multiagent Systems*, pages 39-46, 2000.

[2] J. A. Cannon-Bowers, E. Salas, and S. Converse. Cognitive psychology and team training: Training shared mental models and complex systems. *Human Factors Society Bulletin*, 33:1-4, 1990.

[3] P. Cramton, Y. Shoham, and R. Steinberg. Introduction to Combinatorial Auctions. *Combinatorial Auctions*, MIT Press, 2005.

[4] M. B. Dias, D. Goldberg, and A. Stentz. Market-based multirobot coordination for complex space applications. In *the 7th International Symposium on Artificial Intelligence*, Robotics and Automation in Space(i-SAIRAS), 2003.

[5] X. Fan, S. Sun, M. McNeese, and J. Yen. Extending the recognition-primed decision model to support human-agent collaboration. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 945-952. ACM Press, 2005.

[6] S. A. Wolfman and D. S. Weld. The LPSAT engine and its applications to resource planning. In *Proceedings of the sixteen International Joint Conference on Artificial Intelligence (IJCAI-99)*, San Mateo, CA, Morgan,1999.

[7] X. Fan, J. Yen, and R. A. Volz. A theoretical framework on proactive information exchange in agent teamwork. *Artificial Intelligence*, 169:23-97, 2005.

[8] X. Fan, R. Wang, S. Sun, J. Yen, and R. A. Volz. Context-Centric Needs Anticipation Using Information Needs Graphs. *Journal of Applied Intelligence*, Vol. 24, No. 1, 2006.

[9] B. P. Gerkey and M. J. Mataric. Sold!: Auction methods for multirobot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*, 18(5):758-768, October 2002.

[10] L. Jimsberger and B. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of ICMAS-2000*.

[11] G. A. Klein. Recognition-primed decisions. In W. B. Rouse, editor, *Advances in man-machine systems research*, volume 5, pages 47-92. JAI Press, 1989.

[12] J. Koehler. Planning under resource constraints. In Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98), John Wiley & Sons, 1998.

[13] W. Walsh, M. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. ACM Conference on Electronic Commerce, 2000.

[14] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705-717, 1989.

[15] T. Mullen, V. Avasarala, D. L. Hall. Customer-Driven Sensor Management. *IEEE Intelligent Systems*, 21(2): 41-49, March/April 2006.

[16] T. Sandholm and S. Suri. BOB: Improved winner determination in combinatorial auctions and generalizations. In *Artificial Intelligence*, 2003.

[17] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *International Joint Conference of Artificial Intelligence*, pages 520–526, 1999.