Reverse Combinatorial Auctions for Resource Allocation in the Rescue Scenario

Silvia Suárez and Beatriz López
University of Girona, Spain
{sasuarez, blopez}@eia.udg.es

Abstract

In a disaster environment no single agent knows the disaster situation completely. So, rescue agents share their information in order to identify rescue tasks that are then allocated. In this paper we model the allocation problem as a reverse combinatorial auction and we apply some existing algorithms to solve it. Our results have been compared with the RoboCup classification of 2005.

Keywords
Resource Allocation, Combinatorial Auctions.

Introduction

Scheduling concerns the allocation of limited resources to tasks over time. It is a difficult problem that often requires ad hoc solutions. This is the case of the rescue scenario of the RoboCup rescue project [1]. In a simulated environment of an earthquake in a city and its first 300 cycles of simulation, rescue agents and victim agents interact. As in a real scenario, no agent has complete knowledge about the situation of the disaster. Coordination is required to share information about the situation and actuate consistently, so rescue agents collaborate to diminish the consequences of the disaster.

Rescue agents gather information on the area where they are placed, and report it to a central agent (station or office). This information includes victim localization, blocked roads, and fires. Next, the central agent facilitates information about some rescue tasks to an idle agent.

When deciding which resource agent should be assigned to a rescue task, a central agent has partial knowledge of the current situations of the agents. All the information concerning the agents' situations must be gathered in communication messages that require some time (simulation cycles in the RoboCup simulator). Rescue agents, conversely, know which actions are required to tackle a given task by considering several factors such as: blocked roads, other distended victims close to them, and their own survival rooms (since they can also be damaged), among others. It therefore seems appropriate that rescue agents are able to express their task preferences before central agents allocate them.

Auctions provide such a framework: agents make bids that represent their task performance preferences. A central agent, using the knowledge of the bids, can then make a final decision. In the rescue domain, in particular, when different tasks must be tackled at the same time, the appropriate framework is a combinatorial auction. In this paper we present the application of combinatorial auctions to coordinate the rescue agents in the RoboCup environment following the methods described in [5] and [6].

Rescue Domain Description

Disaster rescue is a complex domain in which it is hard to conduct real life experimentation. For this reason, we have used the simulator provided for the RoboCup [1] to simulate a disaster environment caused by an earthquake. In this simulated scenario, there are collapsed buildings, fires, blocked highways, people in a state of panic looking for a safe place and rescue agents such as fire brigades, ambulance teams, police forces and rescue headquarters. These last ones have to coordinate all kinds of rescue agents.

There are several communication and capability constraints in this scenario [3]. Regarding communication, for example, ambulance agents cannot detect victims and spend too much time looking for them. In this sense, effective communication among heterogeneous rescue agents about the position of victims is crucial to rescuing them.

With regard to planning and scheduling, agents have to make decisions about task execution. These decisions are related to which victim (in the case of an ambulance team) agents have to rescue first in order to maximize benefits from the system, such as reaching and rescuing the highest number of victims while minimizing damage in the surrounding area. In the case of police forces, for example, they need to determine which roads they should unblock first, taking into account that other kinds of rescue agents can be stopped by these blocked roads [2]. Besides, in the case of fire brigade agents, they first need to extinguish fires in buildings containing the highest number of potential victims.

Two main problems can be distinguished in this scenario: planning which tasks should be performed first (for example, a plan to rescue a victim should not be made before the road is unblocked) and scheduling which resources are used to perform the tasks. In previous works [2] we have studied temporal dependencies related to the planning problem. In this paper, we do not consider task dependencies and only deal with the allocation of a set of agents to rescue tasks.

Reverse Combinatorial Auction Formulation

A traditional reverse combinatorial auction is a type of combinatorial auction in which the buyer wants to obtain a set of items, M={1,2,…,m}, and the sellers submit a set of asks, $B=\{B_1,B_2,\ldots,B_n\}$. An ask is a tuple $B_j=(S_j,a_j)$, where $S_j \subseteq M$ is a set of items and a_j, $a_j \geq 0$ is an asking cost. How to determine the combination of winner bids (i.e. the ones that minimize the cost) is known as the winner determination problem. The integer programming formulation for the winner determination problem consists of labeling the asks as winning or losing so as to minimize the buyer’s cost under the constraint that the buyer obtains each item:

$$\text{Min} \sum_{j=1}^{n} a_j x_j \quad \text{subject to} \sum_{j|i \in S_j} x_j \geq 1 \quad i = 1,2,\ldots,m \qquad (1)$$

The X_j variable can take 0 or 1 value and there is one for each bid B_j. It means that if X_j=1, B_j is accepted. The constraint ensures that at least one bid is allocated for each item.

Rescue Formulation

In the rescue domain, the problem of assigning rescue agents to rescue tasks can be formulated as a reverse combinatorial auction problem. To do so, we consider that central agents are buyers, and rescue agents are sellers who are submitting asks for rescue tasks. We then define the following 4-step communication protocol:

1. The rescue agents send the local tasks they can detect in their surroundings.
2. The central agents receive the tasks from the rescue agents and return the complete list of tasks to each agent.
3. With information about all the tasks, the rescue agents send bids corresponding to combinations of tasks to the central agents.
4. The central agents determine the winners, using the winner determination algorithm for combinatorial auctions, and send the results to the rescue agents.

Since each message can be sent/received in a simulation cycle, the protocol requires four cycles. Consistently, a combinatorial auction is performed once every 4 simulation cycles.

Bid Generation

The rescue agents generate bids corresponding to combinations of tasks to be performed in sequential order. Bids consist of a list of tasks and the costs assumed by the agents to carry out these tasks. The cost is determined by the sum of distances from the agents to the tasks.

Solving the Winner Determination Problem

Winner determination algorithms are aimed at solving the optimization problem defined by a combinatorial auction as explained in the previous section. For our particular problem, we have used a formulation based on [5]. It is an A* search algorithm where the bids are organized in a structure called a bidtree. The bidtree is used to define a heuristic to order the bids in the A* search according to [6].

The Bidtree Formulation

A bidtree is a binary tree whose principal purpose is to support content-based lookup of bids [6]. The tree depth is m+1, where m is the number of tasks in the problem. The bids appear in the leaf nodes. Each level of the bidtree represents one task and the in and out branches show if the task belongs or not to the bid. For example, suppose we have some victims at some simulation time with the following identifications, Id_1, Id_2, Id_3, Id_4, and assume that our ambulance central has received the following bids, $B_1(s_1, 30)$, $B_2(s_2, 27)$, $B_3(s_3, 17)$, $B_4(s_4, 32)$, $B_5(s_5, 39)$, $B_6(s_6, 14)$, $B_7(s_7, 12)$, $B_8(s_8, 65)$, $B_9(s_9, 24)$, $B_{10}(s_{10}, 33)$, with the corresponding set of tasks as follows:

s_i= (s_1=[Id_2, Id_4], s_2= [Id1, Id_2], s_3=[Id_1], s_4=[Id_2, Id_4], s_5=[Id_1, Id_2, Id_3], s_6=[Id_4], s_7=[Id_1], s_8=[Id_1, Id_2, Id_3, Id_4], s_9=[id_1, id_4], s_{10}=[id_3, id_4]).

Taking into account that bid B3 has the same task set as bid B7 and that the cost of B3 is higher than the cost of B7, B3 can be erased from the bid set as it will never generate an optimal solution. The same is true for bid B4 with regard to bid B1.

Bid ordering

Taking into account how many times the tasks appear in the bids (bid count), the tasks can be ordered with either an increasing or a decreasing bid count. Using the increasing bid count, the order of tasks for the bidtree is:

id_1, id_4, id_2, id_3. The bidtree with this sorting is shown in Figure 1.

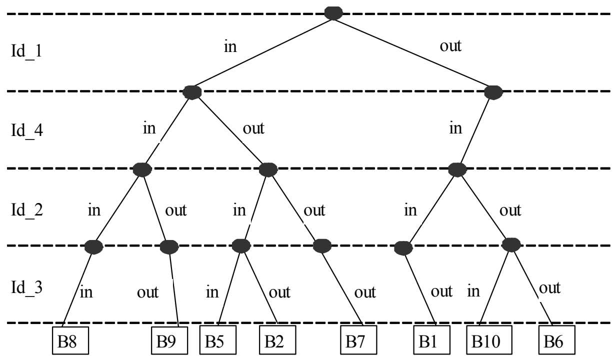

Figure 1. Bidtree with increasing bid count

On the other hand, taking into account a decreasing bid count, the order that we get is: id_3, id_2, id_1, id_4. The bidtree for this case is presented in Figure 2.

.

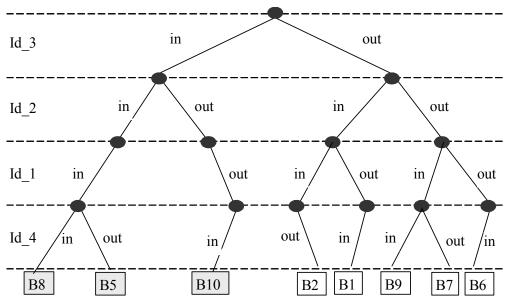

Figure 2. Bidtree with decreasing bid count

Generating the Search Tree

The bidtree is used to generate the search tree. The algorithm based on A* for generating the search tree is presented below.

Let B = {B_1, B_2,…B_n} be the set of bids submitted to be considered in the winner determination process. Let Bc={Bc_1, Bc_2,…Bc_m} be the subset of bids of B that do not include any tasks that have already been allocated and which are still available to be appended to the search path. Let Bl={Bl_1, Bl_2,…Bl_n} be the subset of bids in the leave nodes of the left subtree of the bidtree (in branch). These are the bids in shaded squares in Figure 2. Let Br= {Br_1, Br_2,…Br_n } be the subset of bids in the leave nodes of the right subtree of the bidtree (out branch). These are the bids in unshaded squares in Figure 2. Let $alloc_{\text{best}}$ be a subset of B which is the best allocation found so far (i.e. with the minimum cost), and C($alloc_{\text{best}}$) be the total cost of $alloc_{\text{best}}$. Let Bo={Bo_1, Bo_2,…Bo_n } be the set resulting from Bc $\cap$ Br. Analogously, let Bq(B_i)= {Bq_1,Bq_2,…Bq_n} be a set which stores the bids that conflict with bid Bi. In addition, for one partial allocation *alloc*={B_1…B_n}, it is possible to define:

$$\text{Bq}(alloc)= \bigcup_i B_i \qquad (2).$$

The steps of the algorithm are shown in Figure 3. This algorithm is exponential in the worst case, when b→ ∞ (b=number of bids), and finds the optimal solution, always the least costly path in the search tree, if any exists.

```
Set C($alloc_{best}$) = {}
For each b from Bl
  1. b=one bid from set Bl.
  2. alloc={}
  3. Bc(alloc)= B – Bq(alloc) – (alloc)  (3)
  4. Bo=Bc(alloc)$\cap$Br
  5. if Bo != {} then
         $b_j$= one bid from Bo which has the lowest cost.
         If $b_j$ then
             alloc=alloc $\cup$ bj
  6. if (|tasks(alloc)| = total)  then
         if C(alloc) < C($alloc_{best}$) then
             C($alloc_{best}$) = C(alloc)
             Return to point 1, next b.
     else
         Backtracking
  7. Return to point 3.
```

Figure 3. Algorithm to generate the search tree for the winner determination in combinatorial auctions based on A*

Regarding the decreasing order of our example (Figure 2), the Bl set is {B8, B5, B10}, and the Br set is {B2, B1, B9, B7, B6}. First, *alloc* B8 is explored. Since B8 contains all tasks, no further exploration is carried out. So, B8 is the first feasible solution found for the algorithm. Next B5 is explored. Regarding the Bc set, for instance, the B and Bq sets for *alloc* = B5 are as follows: B={B1, B2, B5, B6, B7, B8, B9, B10}, Bq(B5)= {B1, B2, B7, B8, B9, B10}. And according to step 3: Bc($alloc_{B5}$)= B-Bq(B5)-B5, Bc($alloc_{B5}$)= B6, and Bo=Bc $\cap$ Br, so Bo=B6. As Bo is composed just for one bid, no bid ordering which takes cost into account is required. Since {B5, B6} is a combination that contains all the tasks, it is another feasible solution. Finally, the total search space generated is presented in Figure 4. The best allocation found *alloc*$_{\text{best}}$ is {B5,B6}. And the cost C(*alloc*$_{\text{best}}$)= 55.

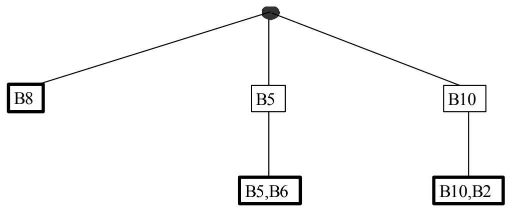

Figure 4. Search tree. Solutions provided in bold squares.

With the increasing bid ordering (Figure 1), the solution of the problem is the same. However, the number of nodes explored is higher. This is something that has been already studied in [6]. The ideal task ordering is one in which the left subtree (the in branch) is always maximally larger than the right subtree (the out branch).

Regarding our example, the ambulance agents who have submitted bids B5 and B6 are the winners and can carry out the set of tasks in their correspondent bids.

Results

We performed 10 simulations and we recorded the results according to the V score defined in the competition [1], which is the following:

$$V=(P + S/Sint) * sqrt(B/Bint) \qquad (4).$$

The higher V value for a map, the better the rescue operation (maximum V value is 97).

Our V average is 62.73. As a frame of reference, the scores of the first four teams in the final of the latest RoboCup Rescue competition (2005) for the Kobe scenario map were 83, 79, 69 and 58. So, our agents have obtained a good score in this frame. Note that in our experiments we have not used any planning method to take into account either task precedence (only re-scheduling as explained in [2]) or path planning computation for rescue agents. Therefore, we believe that our results are quite encouraging. If we include planning competences in our agents in the future, we believe that our results will be even more satisfactory.

Conclusions and Future Work

This paper presents the application of reverse combinatorial auctions for task scheduling in rescue operations following [5] and [6]. The RoboCup rescue simulator has been used as a test bed. The allocation process is repeated with every 4 cycles of simulation.

As a future work, we are thinking about introducing the expected utility theory [7] to improve the planning process in the ambulance team agents. In this sense, the utility attained by agents from scheduling victim rescues is maximized. We are also thinking of applying our approach to one decentralized scheduling model.

Acknowledgements

This work was supported by the Spanish MEC project TIN 2004-06354-CO2-02 and the DURSI-AGAUR 00296 SGR (2005).

References

[1] RoboCupRescue Simulator, http://www.robocup2005.org/roborescue/

[2] Suárez S., Collins J., López B. Improving Rescue Operation in Disasters. Approaches about Task Allocation and Re-scheduling. In Proceedings of PLANSIG 2005. London, UK, 2005.

[3] Takeshi M. How to develop a RoboCup rescue agent. http://ne.cs.uec.ac.jp/~morimoto/rescue/manual/

[4] Sandholm, T., Suri, S., Gilpin, A., and Levine, D. 2005. CABOB: A Fast Optimal Algorithm for Winner Determination in Combinatorial Auctions. Management Science 51(3), 374-390, special issue on Electronic Markets.

[5] Collins, J. "Solving Combinatorial Auctions with Temporal Constraints in Economic Agents." PhD thesis, University of Minnesota, June 2002.

[6] Collins J., Demir G., and Gini M. "Bidtree ordering in IDA* combinatorial auction winner-determination with side constraints." In Lecture Notes of Artificial Intelligence I2531, pages 17-33. Springer-Verlag, 2002.

[7] A. Babanov, J. Collins, and M. Gini. "Harnessing the search for rational bid schedules with stochastic search and domain-specific heuristics." In Proc. of the Third Int'l Conf. on Autonomous Agents and Multi-Agent Systems, submitted, New York, July 2004. AAAI Press.