Real Time Auction Based Allocation of Tasks for Multi-Robot Exploration Problem in Dynamic Environments

Sanem Sariel[1], Tucker Balch[2]

[1]Istanbul Technical University, Department of Computer Engineering, Maslak, Istanbul TURKEY TR34469
[2]Georgia Institute of Technology, College of Computing, Atlanta, Georgia, USA 30332
[1]sariel@cs.itu.edu.tr, [2]tucker.balch@cc.gatech.edu

Abstract

Single auction-based methods are known to be efficient for multi-robot problem solving. In this work, we investigate performance of our general multi robot coordination framework for multi robot multi target exploration problem under uncertainties in dynamic environments. Our framework offers a real time single item allocation method featuring different mechanisms for failure recovery. In multi robot exploration problem, a different version of well known NP-hard MTSP (Multiple Traveling Salesman Problem), each target is visited by at least one robot in its open tour. Overall objective function for cost optimization while visiting targets varies by different exploration domains. In this work, we present performance results for total cost minimization objective. There are many efficient centralized heuristic methods for generating close to optimal solutions. These heuristics may be used to allocate targets to robots. However, when the environment is dynamic and/or unknown, initially assigned targets may need to be reallocated during run time. In our framework, redundant calculations are eliminated by means of incremental assignments based on up-to-date situations of the environment. Offered precautions in the framework maintain the quality of solutions as close to optimal as possible. Experiments are conducted on simulations. The comparison of the proposed method is made with Prim Allocation method.

Introduction

Search and rescue operations, space exploration, reconnaissance/surveillance applications require effective multi robot exploration. Overall objective cost function of the robot team may be different in these domains. While time is a hard constraint in search and rescue operations, in space operations, the objective may be minimizing the overall cost value which is proportional to the total energy consumed by robots.

We propose a framework for heterogeneous teams of robots coordinating to complete complex missions that require diverse capabilities and collective work without a central planner/decision maker. Robots are always in valid plans satisfying constraints on interdependencies among tasks. Different objective functions can be optimized by this framework when effective cost functions and constraints are supplied. The framework also consists of precaution routines for failure recovery. Details of the proposed framework and mechanisms can be found in (Sariel 2005).

In this work, we investigate performance of some heuristic functions combined with our framework for multi robot exploration problem as a case study. This application domain is selected to analyze the deviations of results generated by the framework from optima. Optimal results can be easily obtained by effective integer programming formulation. For this particular case problem, robot team is homogeneous; tasks do not require different robot capabilities and can be executed independently.

Single robot multi target exploration problem (as a different version of TSP) is an NP-hard problem. For multi robot case, allocations of targets to robots are also very affective on the quality of the solution. After allocations are made, the problem turns into a TSP problem for each robot. When the knowledge of the robot is not certain or the environment is dynamic, initially allocated targets to robots are subject to chance. Therefore calculations for allocating all targets may become redundant. Our framework eliminates these redundant calculations by means of incremental assignments based on up-to-date situations of the environment. Moreover, robot systems are open to failures either in communication or robots. Especially robot failures may cause some of the initially allocated targets become unallocated, and these situations should be considered. Our framework can handle such situations by *precaution routines*. Communication failures may prevent initial allocations from being optimal. Our framework can also detect these situations and maintains high solution quality by *dynamic target selection* and *task exchange schemes*.

Related Work

Both combinatorial and single auction methods are studied for multi robot exploration problem in literature. In GRAMMPS (Brummit 1998), one of the earliest works on MTSP (Multi Traveling Salesman Problem) problem, mission planner works centrally either on one of the robots or as an operator. Mission planner selects the corresponding robot for each target. The system can regenerate plans when the environment is changed. Authors claim that for the initial state of the system, the

allocations may be sub-optimal. But in later steps when the number of open missions decreases, the system can find close to optimal solutions. By using the simulated annealing algorithm, a randomized search over all possible allocations is made. In their latest work, Dias and Stentz propose a market based scheme introducing leader approach for combinatorial task exchanges (Dias 2002). These leaders are responsible for multi-party multi-task optimizations for obtaining optimal results. In combinatorial auctions, different combinations of tasks are offered and bidders bid by considering all tasks in these combinations. However this method may become intractable for large instances or dynamic situations in which calculations should be made frequently. Especially when the environment is dynamic, allocations may become sub-optimal. Then a combinatorial exchange mechanism is necessary to maintain optimality. Computational requirements for combinatorial auctions increase drastically for dynamic environments.

Prim Allocation method is a single auction based allocation method. All the targets are assigned to the robots initially. Whenever world knowledge is changed, targets are reassigned accordingly with the same algorithm (Lagoudakis 2004). MSF (Minimum Spanning Forest) of all the targets and robots are constructed by this algorithm. Then routes of robots are constructed for each robot by depth first traversal of the MST's (Minimum Spanning Tree) and adding shortcuts. Since this algorithm is based on Prim Algorithm, its worst case performance is bounded by 2*OPT. The distances are considered from any of the nodes in robot routes while considering a new target to assign. Additional cost of visiting a target is not considered. Therefore some allocations may be sub-optimal.

TSP Heuristics

Although TSP problem is NP-hard there are many efficient heuristic methods in literature generating k-OPT solutions. Some of the heuristic methods use triangle inequality principle given in Eq. 1. The triangle inequality principle for cities i, j and k assumes that the shortest distance (c) between two cities is a straight line (direct route).

$$c_{ik} \leq c_{ij} + c_{jk} \qquad \text{Eq.1}$$

In the following heuristic function definitions, T is a partial tour and k is a city not on T, and $\{i, j\}$ is one of the edges of T.

Minimum Spanning Tree Heuristic (MST)

In this method, either Prim algorithm or Kruskal algorithm may be used to generate MST for the given set of cities. Then a depth first search of T is constructed. After introducing shortcuts into the depth first search, it is ensured that cities are visited only once.

Nearest Merger Heuristic (NMH)

This method corresponds to the Kruskal's minimum spanning tree algorithm. The algorithm starts with n partial tours, each consisting of a single city. Then successively merges the tours until a single tour containing all the cities is obtained. Each time trees to be merged are chosen so that min $\{cij: i \in T \text{ and } j \in T'\}$ is as small as possible.

Nearest Neighborhood Heuristic (NNH)

The algorithm starts with an empty tour T. Cities k and j, for which $c(j,k)$ minimized are found. $\{i, j\}$ is replaced by $\{i, k\}$ and $\{k, j\}$ to obtain a new tour including k. The algorithm continues until all the cities are added to T.

Nearest Insertion Heuristic (NIH)

The algorithm starts with an empty tour T. Cities k and j, for which $c(j,k)$ minimized are found. $\{i, j\}$ is the edge of T which minimizes $c(i, k) + c(k, j) - c(i, j)$, and it is replaced by $\{i, k\}$ and $\{k, j\}$ to obtain a new tour including k. The algorithm continues until all the cities are added to T.

Cheapest Insertion Heuristic (CIH)

The algorithm starts with an empty tour T. If T does not include all cities, for each k, the edge $\{i, j\}$ of T which minimizes $c(T, k) = c(i, k) + c(k, j) - c(i, j)$ is found. Then city k minimizing $c(T, k)$ is found. If $\{i, j\}$ is the edge of T for which $c(T, k)$ is minimized, it is replaced by $\{i, k\}$ and $\{k, j\}$ to obtain a new tour including k. The algorithm continues until all the cities are added to the T.

Farthest Insertion Heuristic (FIH)

The algorithm starts with an empty tour T. City k, which is the farthest of all the cities out of T, is inserted to T. The algorithm continues until all the cities are added to the T.

Christofides Algorithm (CA)

MST of the given set of cities is constructed. Minimum matching M^* for the set of all odd-degree vertices in T is constructed. An Eulerian tour for the Eulerian Graph that is the union of the T and M^* is found, and it is converted into a tour using shortcuts.

Except from FIH and NNH, all the algorithms presented above has a worst case solution bounded by 2*OPT. The worst case solution of the NNH is bounded by (log n)*OPT. CA solution is bounded by 1.5*OPT.

Multi Robot Exploration Problem

Multi robot exploration problem is a different version of MTSP (Multiple Traveling Salesman Problem). In this problem, each target (i.e. nodes or cities of TSP) should be visited by at least one robot in its open tour. Even in single robot TSP problem, there is a trade-off between the

solution quality and the computational requirements for generating optimal solutions. If the environment is dynamic and/or knowledge of the robots is not certain, allocations made initially may result in sub-optimal solutions in later steps because of the decisions based on uncertain information and estimates. A very simple illustrative example is given in Fig. 1. In (a) the estimated distances among targets and robots are seen. With this knowledge, r_1: t_3 and r_2: t_1, t_2 assignment gives a total cost of 13. However when the obstacle is detected, all the allocations should be changed as r_1: t_1, t_2 and r_2: t_3.

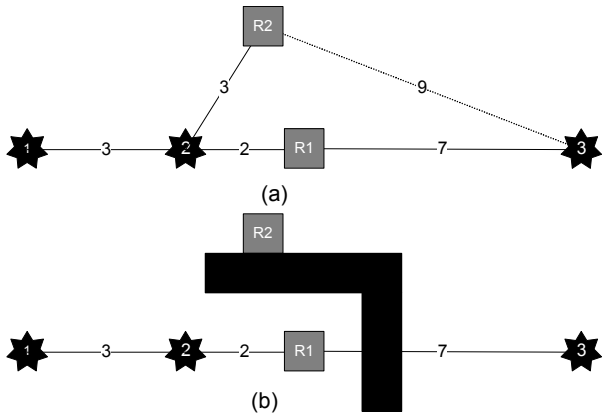


Figure 1. A simple illustrative example for reallocations (a) Robots do not have complete information on the locations of obstacles. t_3 is assigned to r_1; t_1 and t_2 are assigned to r_2 (b) Obstacle is detected; allocations should be changed.

Failures are also common in robot systems. Therefore in addition to the difficulties of dynamic environments, robots may also become inactive during runtime. That requires reallocation of all targets assigned to the failed robots when the world knowledge or the situation is changed. In this case, for maintaining optimality, information on the locations of the all active robots and unvisited targets should be reconsidered. As the frequency of updates on world information increases the calculations for reallocations may become intractable. Therefore applying an algorithm searching all over the target set and assigning all targets to robots may result in redundant calculations without benefit of solution quality.

Heuristic functions/algorithms used for finding near-optimal tours are convenient for environments with uncertainties (Lawler 1985). In this case, computation is implemented for a local instance of the overall problem. It is more logical to make local search for situations with uncertainties. However it should be noted that there is not a polynomial time algorithm guaranteeing optimal solution.

All single TSP heuristic algorithms presented in the previous section are designed to generate a complete single TSP tour. For multi robot exploration problem, more than one TSP tour should be generated for each robot. Moreover assignments of the targets to robots affect the solution quality to a great extent.

Clustering methods generate classes of the targets for the given number of clusters. For this particular problem, initial cluster number can be selected as the number of robots or it may also allowed to be smaller than the number of robots related to the selected objective function. One problem with these algorithms is that they only take into consideration of the distances between target pairs. However costs of adding targets to routes of robots should be considered while generating clusters. This consideration may complicate the clustering approach. This approach may be used for initial selection of targets to be considered.

Vehicle Routing Problem (VRP) (Ghiani 2003) from transportation and logistics research is similar to multi robot exploration problem. Especially dynamic and stochastic multi-depot VRP problem solutions can be used. In multi-depot VRP, vehicles may be in different depots, as in our case robots may be in different locations initially.

Prim Allocation Method

Prim Allocation method (Lagoudakis 2004) inspired from Prim Algorithm generates a MSF (Minimum Spanning Forest) of the targets and robots. MSF consists of separate robot trees. These trees are constructed by adding each unallocated target to the closest robot path containing the node with the minimum distance to the target, until all targets are allocated. In other words, addition of a new target is implemented by considering the distances between the target and nodes of the robot routes instead of considering the last position of the robot. The target is auctioned and the robot offering the smallest bid value is selected to allocate that target. Before robots run and visit the targets, all the allocations are made. Whenever the world knowledge is changed, allocations for the remaining unvisited targets are made with the same algorithm.

Real Time Single Auction Based Dynamic Allocation Scheme

Our general framework (Sariel 2005) is for a heterogeneous robot team working on a complex mission consisting of tasks requiring different capabilities and cooperative work and having dependencies on each other (acyclic precedence relation). In this framework, optimal solutions are sought while handling with common failures in robot systems. This framework is based on distributed market based approach. However to maintain optimality additional routines such as *dynamic target selection* and *task exchange schemes* (called as *coalition maintenance* in the general definition) are added to the framework.

In this work, we investigate the performance of our framework for multi robot exploration problem with total cost minimization objective. In our approach, the targets are allocated in an incremental way. Robots visit targets and then offer auctions for the next best unvisited target for them. Locations of robots at the time of the auction are considered while calculating distances to targets for all

experimented heuristic cost functions with our framework. Therefore routes are constructed online based on the world knowledge at hand. In the problem definition, the robots are not supposed to come back to their initial locations, or the minimization of the distance between last target of the route and the initial location is not required; i.e. routes are not closed. Since the targets are incrementally assigned to robots in our framework, considering robots' up-to-date location information is a better approach than considering all targets from routes of robot in the cost calculation for our framework.

Target Selection

Targets and related tasks are considered in each step when the world knowledge is changed if the robot is already assigned to a target or in each step when the robot is idle. Situations in which world knowledge is updated are arrival of new online targets, changes in knowledge about environment and distance estimations accordingly, new cost value information on tasks under execution or detection of robot failures.

This procedure is called as *selectTarget* in our framework for the multi robot exploration problem. *selectTarget* method is the modified version of the *selectAction* method (Sariel 2005) of the general framework and is given in Fig. 2.

```
selectTarget for rj
For each known and unvisited target ti
    if the target ti is not in one of the following types, skip
        The targets for which the robot is accepted for exchanging (with higher cost value than
        the robot has)
        Awarded targets by the auctioneers
        Free targets which are not under consideration
    else add the target ti in a priority queue ordered by cost with the priorities in given order
Select the minimum cost target in the priority queue

if rj has already been allocated to a different target
    cancel visiting the target at hand

if the selected target is a free target
    begin AuctionNegotitationProcess
else
    set the new target as the current target and begin executing
        if it is an awarded target
            send "accept become a member" message to the auctioneer
```

Figure 2. Target Selection method

The targets/tasks are prioritized based on the situations. If there are two targets at the same cost, the one with the highest priority, otherwise the minimum cost target is selected. The targets suitable to exchange have the highest, awarded targets by auctioneers have the second and the targets not being considered (free) have the lowest priority. The targets are pushed into a priority queue in which they are ordered based on their estimated costs and priorities. The minimum cost target (top element of the queue) is selected. If the robot is already assigned to a different target than the selected one, current task is canceled. If it is idle and the selected target is a free target, an auction negotiation process is initiated. Otherwise the target is set as the new assigned target and the robot begins executing it. If the target is an awarded target, a confirmation message is sent to the auctioneer.

Robots may consider targets at the same time in the general framework. However for obtaining optimal results for the given objective function, we just allowed only one robot to visit a target or offer an auction. If there is more than one robot offering an auction, the one having the smallest cost value continues auction negotiation process; other robots cancel their auction negotiation processes.

Auction Negotiation Process

In the auction negotiation process, the corresponding auctioned target is assigned to one of the robots (bidders) having the minimum cost value. Initially the auctioneer offers the auction. If the auction is invalid, a warning message is sent to the auctioneer. Invalidity may occur if the auctioneer has incomplete knowledge about the task status. Possible situations may be that the task related to the target is completed or it has already been executing. If the auction passes the validity check, the candidate robot becomes a bidder of the task, calculates the cost and sends the cost value as a bid. The other candidate robots behave as so. The auctioneer robot is also a bidder and generates a bid for the task at hand. It waits until the end of the deadline. If the auctioneer does not get bids from other robots until the deadline, it selects itself as the awarded bidder. Otherwise it ranks all the bids and selects the robot with the minimum cost/bid value.

A bidder robot may get confirmation from different auctioneers. However in the action selection step, it selects the minimum cost target for it. Therefore it sends a message to only one of these auctioneers.

It is allowed that more than one robot offer an auction for the same task. In this case, when the robots detect this conflict, the robot having the minimum cost value is the winner of this race. If the cost values are the same, the robot having the smaller identification number is the winner. This decision only provides to recover from the conflict. There may be other solutions but we avoid adding extra complexity to the negotiation process. Besides that, since these robots are in reliable communication range (they can detect that both them auctioned for the same task), the bid of each of them is considered in either case not affecting the optimality.

Task Exchange Mechanism

To keep consistency, each robot executing a task is responsible for broadcasting an *execution message* to inform others that the task is under execution, the robot is alive and about the cost value for that moment. Since in the decision stage, each robot taking these kinds of messages considers the cost value from each robot, if a robot detects that its cost is lower than the broadcasted cost of a task, it sends a *task exchange* message to the robot executing the task. The corresponding robot cancels its execution after

receiving *task exchange* message and the sender of the message begins executing the task.

Precautions

For dealing uncertainties because of the message losses, each robot keeps track of the models of known tasks (targets) and other robots in their world knowledge. Up-to-date knowledge of the known tasks is maintained by representing each known situation in a FSM.

When robots get information from others they update their world knowledge accordingly. Whenever communication becomes unreliable, world knowledge of each robot may be inconsistent. The proposed framework ensures an update mechanism when conflicts are detected to reduce inconsistency. When robots receive inconsistent messages, they either warn others or correct themselves. These inconsistencies occur when robots are not informed about the tasks that are completed, under execution or under auction. It is assumed that robots are trusted and benevolent.

Execution conflicts:
If a parallel execution of the same task is detected, all robots executing the same task (for the same target) cancel their execution. This is for ensuring optimality. A new auction is generated to consider all the robots' costs. The robots are allowed to generate different auctions for the same task. However one of them continues the auction negotiation process.

The robot getting execution messages is informed about the task and assumes that the corresponding task is under execution for a period of time. If a robot cannot get any message related to a task for a period of time, the task is assumed to be free. The *execution message* is a clue that the executer robot is still alive and the task is being executed. The same type of update is performed when robots offer auctions for tasks. In this case the world knowledge is updated as the robot is not executing a task and the task is under consideration.

Auction conflicts:
Whenever a robot gets an auction message, it first checks the validity of the auction by considering the task information. If it is a valid auction, it always sends bid information to the auctioneer. Therefore a more suitable task for the robot may be selected in future, whether it is currently executing a task or not. Updates are implemented when execution, cancellation messages are received.

Robot Failures:
Since world knowledge of a robot is updated based on incoming information from others, if a robot executing a task fails, after a period of time other robots not receiving messages from the failed robot mark the related task as left/free to consider it in the target selection process.

Cost Function and Heuristics:

In the proposed framework, the general cost function c_{ji} for robot r_j and target t_i is calculated as summation of two separate components namely base distance cost (c_{ji1}) and additional cost (c_{ji2}) as in Eq. 2.

$$c_{ji} = c_{ji1} + c_{ji2} \qquad \text{Eq.2}$$

These functions change based on the heuristic methods (HM). The base distance cost function for robot r_j and target t_i is calculated as given in Table 1 for different situations. It can be seen that when a robot is in target selection step, it always calculates the distance to target from the current location of itself. When another robot offers an auction for that target, it calculates the distance either from the current location (when idle) or from the assigned target t_k (when not idle) and sends the cost information to the auctioneer.

Table 1. Base distance cost

c_{ji1}	Robot is idle	Robot is already assigned to target t_k
Robot considering the target	$dist(r_j, t_i)$	$dist(r_j, t_i)$
Target is auctioned by another robot	$dist(r_j, t_i)$	$dist(t_k, t_i)$

Additional cost function (c_{ji2}) for robot r_j and target t_i depends on the selection of the heuristic method. We performed experiments on three different heuristic methods given in Table 2. These methods are Closest Cost (CC) Heuristic, Nearest Addition Cost (NAC) Heuristic and Farthest Addition Cost (FAC) Heuristic. We inspired from TSP heuristics while designing these cost functions.

Table 2. Cost function components for different heuristic methods

HM	Base cost	c_{ji2}
CC	c_{ji1}	0
NAC	c_{ji1}	$\begin{cases} dist(t_i,t_k) - dist(r_j,t_k) & if\ dist(t_i,t_k) > dist(r_j,t_k) \\ 0 & otherwise \end{cases}$
FAC	$\alpha * c_{ji1}$	$(1-\alpha) * (dist(t_{f1},t_{f2}) - \max(dist(r_j,t_{f1}), dist(r_j,t_{f2})))$

T_U is the unallocated target set, $T_{Uj} \in T_U$ is the set of targets closest to robot r_j. $t_k \in T_U$ is the target with the minimum distance $dist(t_i, t_k)$ to t_i. In CC heuristic method, c_{ji2} is 0. Therefore robots consider only distances to targets. NAC heuristic method provides one further look ahead for an additional cost of visiting target t_i. If the distance between the next closest target t_k to t_i is greater than the distance from the robot location to t_i that means selection of visiting target t_k may cause additional cost given in the Table 2. FAC heuristic method consider t_{f1} and t_{f2} in T_{Uj} which are the targets having the maximum $dist(t_{f1}, t_{f2})$ value. The basic idea behind this heuristic function is that if a circle is drawn containing all the

targets in T_{Uj}, t_{f1} and t_{f2} are the targets determining the diameter of the circle and they should also be visited. This method forwards robots to these farthest targets to some degree. By introducing a constant (α) this additional cost consideration can be adjusted. We selected α as 2/3 in the experiments.

Experimental Results

The experiments are conducted on a 100*100 grid environment with 20 targets and 1-4 robots. Target and robot locations are assigned randomly for each run. The results are presented as average of 50 runs for each set. In the experiments, the performance of the heuristic methods are measured on a static environment and compared with Prim Allocation method. Additionally result of a sample run in which robots do not have complete information on a dynamic environment is given as an illustrative example. The distances are taken as symmetric in our problem meaning that the distance between any two points are the same in both ways. The optimal results are obtained by CPLEX program. The integer programming formulation for obtaining optimal results can be found in (Lagoudakis 2004).

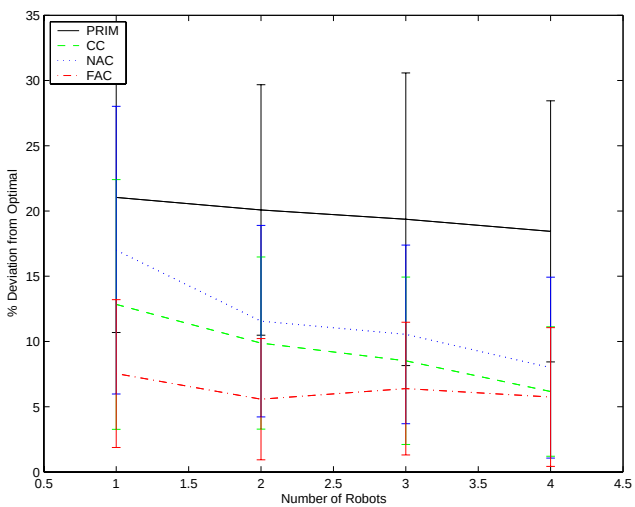


Figure 3. Performance results of the heuristic approaches combined with the framework and the Prim Allocation, averaged over 50 runs given with standard deviation

Results are presented as average percentage deviation from optimal values with standard deviation in Figure 3. As can be seen from the graphs, all heuristic methods combined with our framework outperform Prim Allocation method. The reason for even closest cost heuristic (CC) generates better results is taking into account of the up-to-date location of the robot. Although initially some targets are close to a robot, they may become farther when the robot moves or vice versa. FAC heuristic is the most successful of all, it generates close to optimal results even for experiment with one robot (it should be noted that in this case, the problem is a single TSP problem). NAC heuristic performance approaches to the other two successful heuristics (CC and FAC) for the experiment with 4 robots. Deviation from optimum decreases when the number of robots increases because of the increase in density and the decrease in distances for all approaches.

Reconfigurations in a dynamic environment implemented by the proposed framework can be seen in Figure 4. In this scenario, there are 6 targets and 3 robots in the environment. Robots are shown as small squares. Targets are marked as crosses. World knowledge of robots is not certain. In (a) the environment and in (b) world knowledge of robots is given. Robots begin executing the tasks. In (c) robot r_1 detects the obstacle and marks the target as unreachable. In (d) robot r_3 detects the obstacle in which case the target is reachable. However robot r_2 has the smallest cost for this target. Then robot r_3 cancels the execution. Robot r_2 considers all the targets and decides on the closest target (e). Robot r_2 fails in (f). Robots r_1 and r_2 detect this failure. Robot r_1 completes the mission.

Conclusion

In this work we investigate performance of our general framework for a multi-robot team in an exploration problem. Overall mission is exploring a set of targets with minimization of total cost objective. The system can handle real time communication and robot failures.

Experiments on different heuristic cost functions combined with the framework validate that the results are close to optimal. Decisions on each new situation and allocations of targets in an incremental way make the results closer to optimal. Recovery solutions provided by precaution routines for different kinds of failures ensure the approach is complete. The approach is also suitable for dynamic version of the problem in which targets are generated on-line.

Some stochastic approaches may be used for forwarding to robots or for their decision making. Robots may select high cost calculations guaranteeing optimality by using Linear Programming methods if it is certain that the environment will not change. However the situations are subject to change for dynamic environments, and it is more reasonable to perform a local search by using a heuristic function as in presented in this work. Currently combined tests with communication and robot failures for dynamic environments are being experimented. In this work, we analyzed performance on an objective function for minimizing the total cost. As a future work, we are going to perform experiments for different objective functions on multi robot exploration problem in different domains.

Acknowledgments

This work is supported by Siemens TURKEY, Tincel Kultur Vakfi TURKEY and NSF. Authors would like to thank to Michail G. Lagoudakis for providing optimal CPLEX results.

References

Brummit, B. and Stentz, A., 1998. GRAMMPS: A Generalized Mission Planner for Multiple Robots in Unstructured Environments. In Proceedings of the IEEE International Conference on Robotics and Automation.

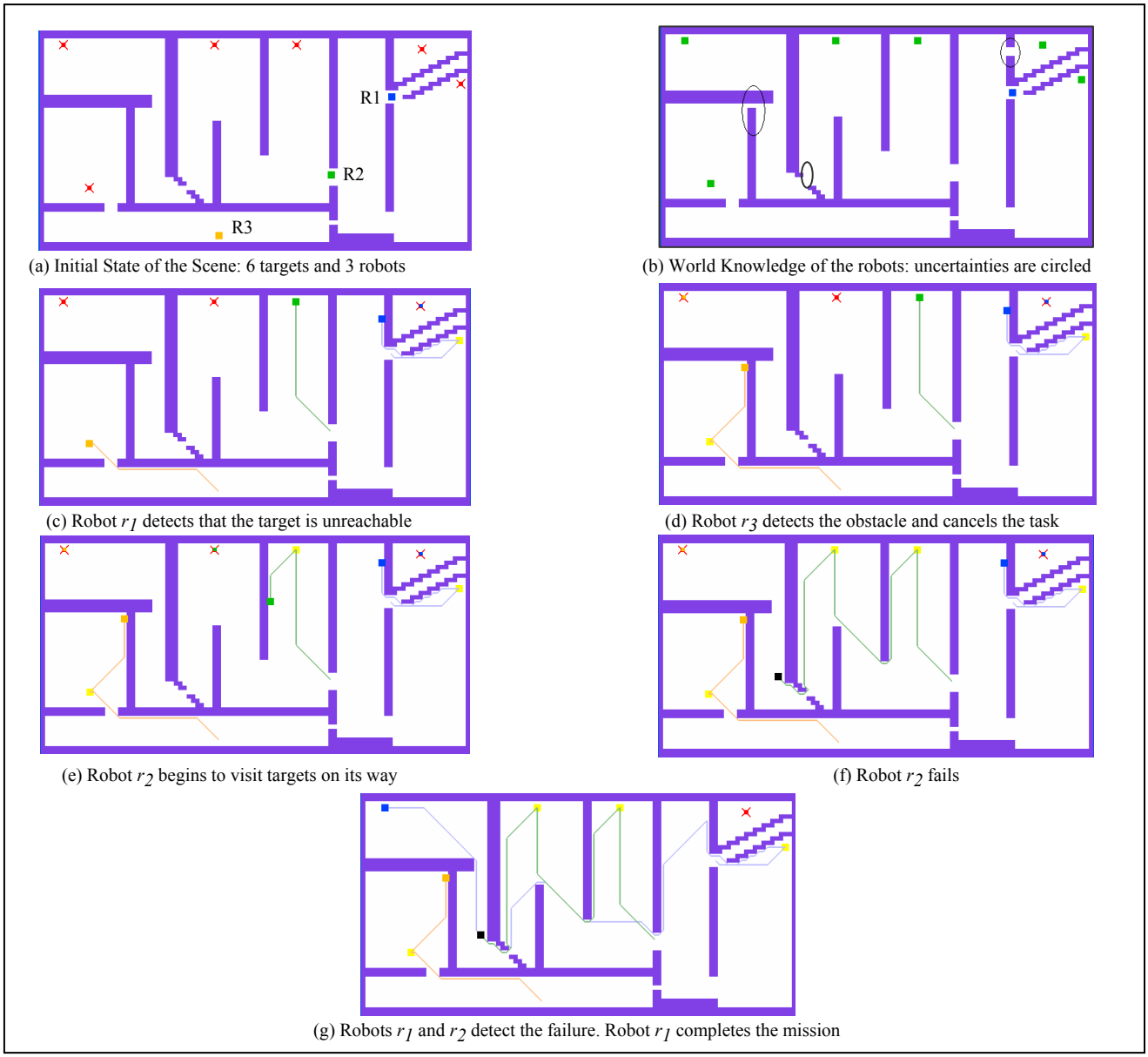
Dias, M.B. and Stentz, A., 2002. Opportunistic Optimization for Market-Based Multirobot Control. In Proceedings of the IEEE/JRS International Conference on Intelligent Robots and Systems (IROS).

Ghiani G. et al. 2003. Real-Time Vehicle Routing: Solution Concepts, Algorithms and Parallel Computing Strategies. *European Journal of Operational Research* 151: 1-11.

Lagoudakis, M. G. et al., 2004. Simple Auctions with Performance Guarantees for Multi-Robot Task Allocation. In Proceedings of the IEEE/JRS International Conference on Intelligent Robots and Systems (IROS).

Lawler, E.L., and Lenstra, J.K., eds. 1985. *The Traveling Salesman Problem.* John Wiley and Sons Ltd.

Sariel S. and Balch T., 2005. *A Robust Multi-Robot Coordination in Noisy and Dangerous Environments*. GVU Technical Report: GIT-GVU-05-17, Georgia Institute of Technology.



(a) Initial State of the Scene: 6 targets and 3 robots

(b) World Knowledge of the robots: uncertainties are circled

(c) Robot r_1 detects that the target is unreachable

(d) Robot r_3 detects the obstacle and cancels the task

(e) Robot r_2 begins to visit targets on its way

(f) Robot r_2 fails

(g) Robots r_1 and r_2 detect the failure. Robot r_1 completes the mission

Figure 4. A sample scenario in a dynamic environment