ROBUST MULTI-ROBOT COOPERATION THROUGH DYNAMIC TASK ALLOCATION AND PRECAUTION ROUTINES

Sanem Sariel
Istanbul Technical University, Computer Engineering Department, Maslak, Istanbul, Turkey

Tucker Balch
Georgia Institute of Technology, College of Computing, Atlanta, GA, USA

Nadia Erdogan
Istanbul Technical University, Computer Engineering Department, Maslak, Istanbul, Turkey

Keywords: Distributed AI, robotics, multi-agent systems.

Abstract: In this paper, we present the design and implementation of a multi-robot cooperation framework to collectively execute inter-dependent tasks of an overall complex mission requiring diverse capabilities. Given a heterogeneous team of robots and task dependencies, the proposed framework provides a distributed mechanism for assigning tasks to robots in an order that efficiently completes the mission. The approach is robust to unreliable communication and robot failures. It is a distributed auction-based approach, and therefore scalable. In order to obtain optimal allocations, effective bid evaluations are needed. Additionally to maintain optimality in noisy environments, dynamic re-allocations of tasks are needed as implemented in dynamic task selection and coalition maintenance scheme that we propose. Real-time contingencies are handled by recovery routines, called Plan B precautions in our framework. Here, in this paper, we present performance results of our framework for robustness in simulations that include variable message loss rates and robot failures. Experiments illustrate robustness of our approach against several contingencies.

1 INTRODUCTION

We propose **D**istributed and **E**fficient **M**ult**i** **R**obot - **C**ooperation **F**ramework (DEMiR-CF) for multi-robot teams that must cooperate/coordinate to achieve complex missions including tightly coupled tasks that require diverse capabilities and collective work. It combines *auctions*, *coalition maintenance scheme* and recovery routines called *Plan B precaution routines* to provide an overall system that finds (near-) optimal solutions in the face of noisy communication and robot failures. The efficiency of our framework was tested for the multi-robot multi-target exploration problem in an earlier work (Sariel and Balch, 2006). In this paper, we evaluate the performance of the framework for robustness on complex missions against several noisy or faulty situations. The *Plan B precaution routines* embedded in the framework enable the system to dynamically respond to the contingencies and still complete the mission. The proposed framework strives to provide (near-) optimal task allocations with effective bid evaluations while simultaneously and effectively responding to contingencies and maintaining the solution quality.

Multi-robot coordination problem has been an active area of research where both centralized and distributed approaches are considered. In most cases, the selected domain missions include independent subtasks which can be executed by a single robot. Due to the real world requirements and issues of robustness, the centralized approach has usually not been successful in these implementations. The distributed approach and, specifically when complemented with the auction based methods based on Contract Net Protocol (Smith, 1980), shows great promise for multi-robot task allocation because of its scalability. The task allocation problem in the face of environmental changes and the optimality analysis are investigated separately on earlier distributed systems. According to (Dias et al., 2005), existing market mechanisms/auction based approaches are not fully capable of re-planning task distributions, changing the decomposition of tasks continually, rescheduling commitments, and re-planning coordination during execution. Our research aims at addressing these shortcomings.

Even though task allocation issues have been studied extensively, there is not yet a complete frame-

Sariel S., Balch T. and Erdogan N. (2006).
ROBUST MULTI-ROBOT COOPERATION THROUGH DYNAMIC TASK ALLOCATION AND PRECAUTION ROUTINES.
In *Proceedings of the Third International Conference on Informatics in Control, Automation and Robotics*, pages 196-201
DOI: 10.5220/0001212901960201

work which is fully capable of integrating task allocation into real time execution for complex missions. A classification for multi-robot task allocation problem and a review of the existing approaches are given in (Gerkey and Mataric, 2004). According to this classification, our framework lies in the single-task robots, instantaneous assignment with time-extended view of the problem and multi-robot tasks horizon. Contingency handling is not directly integrated into task allocation and execution in earlier frameworks. Our work differs at this point, as it integrates contingency handling into task allocation and achieves a dynamic allocation scheme through the task execution. The framework maintains allocation (near-) optimality during task execution, even in the presence of noisy and/or faulty conditions.

2 FRAMEWORK OVERVIEW

DEMiR-CF is designed to address complex missions of tightly coupled tasks requiring diverse capabilities and simultaneous and synchronous task execution. Optimal solutions are sought using effective bid evaluation strategies. The framework also effectively responds to limited or noisy communications and robot failures during execution. These failures are recognized by *Plan B precaution routines* and corresponding actions are taken. *Dynamic task selection* and *Coalition maintenance scheme* provide task (re-)allocations to maintain the solution quality. Each robot equipped with DEMiR-CF software executes a part of the domain mission, achieving allocation, execution and system consistency maintenance in a fully distributed fashion without a central coordinator. Our system relies on a set of task dependencies that are compiled *a priori* by a mission commander, or generated by a planner. The task dependencies, a specification of the mission, and a specification of the robot teams' capabilities are distributed (reliably) to the robots before the mission execution begins. At that point, each robot autonomously decides which task to execute and negotiations are implemented in a fully distributed fashion.

2.1 Task Representation

One of the important mechanisms of the framework is the guaranteed validity of the generated plans for executing tasks by robots. Although there is not a central planner, the robots execute tasks by considering dependencies among them. We use a simple representation for tasks with ordering and dependency constraints, similar to but simpler than some representations in TAEMS (Decker, 1996). We consider hard ($h-dep$) and soft dependencies ($s-dep$). If task T_2 is hard dependent on the task T_1, it cannot be executed before the task T_1 is completed. In this case, there is a strict ordering between tasks. If the task T_2 is soft dependent on the task T_1, it can be executed while the task T_1 is being executed or before its execution begins. However the task T_2 cannot be completed until the task T_1 is completed. Therefore some portions of the two tasks can be executed simultaneously. The required capabilities and the number of robots required to execute the task are also encoded into the representation. Robots, which lack the necessary capabilities for a task, are not eligible for executing and do not consider such tasks for execution. A task can only be executed when the required number of robots are assigned to execute.

2.2 Roles

Some tasks may require simultaneous performance by a group of robots. We have chosen the coalition organizational paradigm to organize the task execution. (Horling and Lesser, 2005). These coalitions can contain one or more robots according to the number of robots required to execute tasks. Members of coalitions are selected by auctions. The auctioneers are also active robots in the system. The overall objective is completing a mission (M) consisting of tasks T_i ($0 < i \leq ||M||$) with a multi-robot team ($r_j \in R$, $0 < j \leq ||R||$) in a cost optimal manner. Each coalition (C_i) is formed to execute a task T_i of the overall mission M. Sizes of coalitions vary according to the required number of robots ($reqno_i$) to execute the task. The capabilities (cap_j) of each robot r_j in a coalition should be a superset of the required capability set for the task T_i ($reqcap_i$) to be eligible. A robot (r_j) may take different roles for the task T_i, such as auctioneer, bidder (B_{ij}), coalition leader (CL_i) or coalition member (CM_i) during runtime.

- An Auctioneer robot is responsible for managing auction negotiation steps and selecting $reqno_i$ suitable members of a coalition.
- A Bidder robot is a candidate to become a member of a coalition to execute a task.
- A Coalition Leader is the robot responsible for maintaining the coalition and providing synchronization while performing the coalition task.
- A Coalition Member is one of the members of the coalition, and it executes a portion of the task.

A robot r_j may be in more than one B_{ij} roles for different tasks, but may not be in more than one role of auctioneer, CM_i or CL_i at the same time. The auctioneer is responsible to select the required number of robots (the coalition leader and members) for task execution. The auctioneer may or may not take place in

the coalition for which it offers an auction. The coalition leader maintains the coalition and keeps track of the members' conditions and their updated information. After the execution of the task is completed, the coalition ends. Each robot is allowed to take part in only one coalition until it leaves the coalition. Coordination between coalition members is implemented through synchronization messages. We assume that robots are not able to infer the state of others by observation; although such capabilities would only provide more reliability (e.g. (Balch and Arkin, 1994)).

2.3 Precautions

To deal with the uncertainties that may arise due to message losses, each robot keeps track of the models of the known tasks and the other robots in its world knowledge. The current statuses of the known tasks are represented in finite state machines (FSM). The state transitions are initiated either by own motivations or incoming information from the other robots. There are eight task states in each FSM. Initially all tasks are in *free* state, meaning that none of the robots is executing/auctioning and they are not achieved yet. Then, according to the status of the task, the state can be specified as: *auctioned*, *selected*, *ready for execution*, *awarded*, *being executed by own/others*, or *achieved*. When messages are received from the other robots, the task and robot models are updated accordingly. In some cases a new message may imply a conflict with the robot's current model of the task state. In this case, the appropriate precaution routines are activated to either correct the model, or initiate a recovery. Recovery operations may include warning the other robots or changing the task states and dynamically switching among tasks if profitable. Such inconsistencies usually arise when robots are not accurately informed about the tasks that are completed, under execution, or under auction. It is assumed that robots are trusted and benevolent.

To keep consistency, each robot executing a task is responsible for broadcasting an execution message, informing others that the task is under execution and the robot is alive. Thus, robots are informed about the task and they assume that the task is under execution for a period of time. If a robot does not receive any messages related to a task for a certain period of time, the task is assumed to be *free*. Robot failures are detected by checking the latest communication information with the robot. When parallel auctions for the same task are detected, the robot with the minimum cost value continues the auction negotiation processes. After receiving an auction message, a robot checks the validity of the auction by considering the task information before sending its bid value. If the robots get an auction offer for a task that has hard dependencies, then it assumes that these dependent tasks are also achieved and the own world knowledge is updated respectively. Similar update actions are carried out when execution, cancellation messages are received.

For synchronized coordination, the coalition leader sends synchronization messages to all coalition members. These messages are also used for recognizing failures. The leader also receives the updated cost information in execution messages from the members. If the messages cannot be received during a predefined time period, the existence of a problem is assumed and the coalition execution is ended.

2.4 Dynamic Task Allocation

Each robot updates its world knowledge according to the incoming messages. Before selection of an action, each robot should perform some routine tasks for different roles. Auctioneers perform the negotiation processes. Leaders and members of the coalitions carry out synchronization actions that ensure the simultaneous execution of tasks synchronously. Next, considering the updated world knowledge and the situations of the processes for different roles, each robot determines a rough schedule where tasks for which it meets the capability requirements are considered and several constraints are taken into consideration. Then, it generates a prioritized list of tasks, from which it selects the most suitable and precedence feasible one and takes an action accordingly. This approach is known as List Scheduling (Graham, 1966) and shown to be $2-(1/m)\ OPT$ for makespan minimization objective on tasks with precedence constraints (m is the number of machines/robots). Different priority rules may be used. Here in this work, we use Least Flexible Job First (LFJ) rule in a precendence feasible manner.

The selected action may be participation to a coalition or becoming an auctioneer auctioning for a *free* task. The dynamic action selection algorithm (Algorithm 1) is performed if the robot is free or released. The releasing mechanism provides a way to switch tasks effectively if another advantageous situation arises. Decision on releasing a robot from a coalition can only be made by the coalition leader. This restriction prevents the system from frequent switching and contributes to the achievement of tasks which require simultaneous execution.

2.5 Cost/Bid Evaluation

The main objective to achieve a global goal (e.g. minimization of total energy consumed or goal achievement time) determines how robots should perform cost evaluation. Unless effective bid evaluation strategies are designed, it is not possible to observe globally optimal solutions for NP-Hard problems, and additional adjustments are required to change allocations

Algorithm 1 Action Selection Algorithm for r_j.

for each known task T_i **do**
  if the $reqcap_i \subseteq cap_j \wedge h - dep_i$ are completed **then**
    if the task is in one following states: under execution and cost profitable to join || awarded || free **then**
      add the task in a priority queue ordered by LFJ
    end if
  end if
end for
T_s = queue.top()
if r_j is in a coalition of T_k(should be released) **then**
  cancel executing the task, send a "leaving" message to the CL_k
end if
if state(T_s) == free **then**
  begin auction negotiation process
else
  the current task = T_s
  if T_s is an awarded task **then**
    send "accept" message to the auctioneer
  end if
end if

with an additional cost of communication as in combinatorial auctions. Difficulty of the problem arises when communication is limited and robots should autonomously perform task allocation at the same time with execution. Simultaneous execution requirements make the problem more challenging because each robot should be in its most suitable coalition in a future formation and estimate it correctly before making a decision in a distributed fashion and with as minimum communication as possible. In our earlier work (Sariel and Balch, 2006), we have shown that by effective bid evaluation approaches, globally near optimal solutions can be observed in an auction based system. The incremental assignment approach and dynamic task selection scheme eliminate redundant considerations for environments in which the best solution is highly probable to change, and effective bidding strategies ensure the high solution quality with a time-extended view of the problem.

2.6 Coalition Maintenance Scheme

To ensure maintaining optimality against environmental changes, we propose a dynamic reconfiguration approach. Coalition members can leave a coalition when there is sufficient number of members to execute the task. The coalition leader is responsible to broadcast the maximum cost of execution for the task by one of the coalition members in each execution step. In the decision stage, each robot receiving these messages evaluates the maximum cost value of each coalition. If a robot detects that its cost is lower than the maximum cost of the coalition and it is released from its current coalition, it sends a join request message to the coalition leader. The leader, getting a join request message, directly adds the robot to the coalition. If the coalition leader detects that the size of the coalition is larger than required, it can release coalition members with the maximum cost value for the current task. Getting a released message, a robot can proceed to select another suitable task. When the coalition leader considers the size of the current coalition, it also checks the failures. Since each robot in the coalition broadcasts "under execution" and updated "cost" messages, their failure can be detected by the coalition leader. The failed robots are also released from the coalition. If the number of members to execute the task is below the required for a period of time, the coalition leader cancels the coalition. An illustrative example of coalition reconfiguration is given in Fig. 1. Such a situation may occur if a robot is not reachable when the auction announcement is made {a}. When the situation changes {b}, the robot may take over the role of the member with the maximum cost value in this coalition {c}. The released member can select another task to execute.

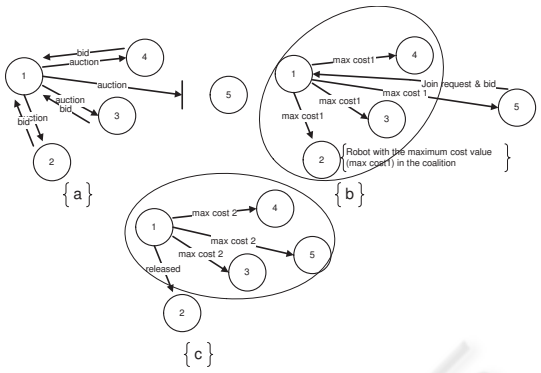

Figure 1: Dynamic Coalition Reconfiguration.

3 EXPERIMENTAL RESULTS

We have conducted experiments to test the performance of the framework in terms of the total time to complete the overall mission for collective construction domain. In the first stage of the experiments, the performance is evaluated against message losses for a complex mission that consists of tasks with hard and soft dependencies among each other. In the second stage of the experiments, the effectiveness of the framework and the recovery solutions is analyzed when the robot failures are injected into the system. The experiments are conducted on an abstract simulator simulating robots' updated locations, the message exchange and execution of the missions. The results are from 100 independent runs with random seeds. The maximum simulation step number is

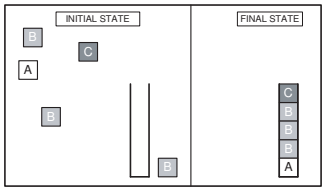

Figure 2: Initial and Final States of the Test Domain.

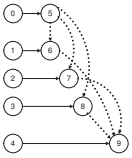

Figure 3: Task dependency graph of the mission.

selected as 200. In the designed experiment, there are three types of objects: A, B and C. The mission contains tasks for locating these objects and building up the construction while obeying the specified ordering restrictions. In the construction, the objects should be moved into the destination in the predetermined order: A, B and C. The initial and final stages can be seen in Fig. 2. The requirements for the tasks and their precedence constraints are given in Table 1. The third and fourth columns represent the hard and soft dependencies respectively. The required capabilities for the task and the required number of robots to execute are specified in the last two columns. The capabilities of having blob finder, sonar sensor, prods for objects A and C and prods for object B are listed as 0, 1, 2 and 3 respectively. Based on the specifications of the overall mission, the task dependencies can be represented graphically as in Fig. 3, in which the solid and dashed lines represent the hard and soft dependencies, respectively.

The capabilities of the robots are given in Table 2. There is one redundant robot in each set to analyze robustness of the framework against failures. It should be noted that, if the requirements are not met, there is no way to execute the corresponding task.

Instead of selecting the failure step randomly, we've selected the step number in which the failure occurs as the most important time step to complete the mission, to fairly analyze the robustness. Based on the results of no failure case, we've observed that, for most of the runs, the 25th time step is a critical time step when the robots have already finished executing tasks that do not require coordination, and try to coordinate for other tasks. The failed robot is selected randomly. The mission completion ratios over all runs are illustrated in Table 3. In this table, the main columns present results of experiments with no

Table 1: Task Specifications.

ID	Description	$h-dep$	$s-dep$	$reqcap$	$reqno$
0	Locate-Obj-A	-	-	0, 1	1
1	Locate-Obj-B	-	-	0, 1	1
2	Locate-Obj-B	-	-	0, 1	1
3	Locate-Obj-B	-	-	0, 1	1
4	Locate-Obj-C	-	-	0, 1	1
5	Move-Obj-A-to-Dest	0	-	0, 1, 2	1
6	Move-Obj-B-to-Dest	1	5	0, 1, 3	2
7	Move-Obj-B-to-Dest	2	5	0, 1, 3	2
8	Move-Obj-B-to-Dest	3	5	0, 1, 3	2
9	Move-Obj-C-to-Dest	4	6, 7, 8	0, 1, 2	3

Table 2: Robot Set.

SetID	RobotID	cap
1	0, 8	0, 1
2	1, 2, 3, 4	0, 1, 2
3	5, 6, 7	0, 1, 3

failure, 1 and 2 robots failures, respectively. An additional column is added for each experiment with failures to report the number of runs in which one or two of randomly failed robot is a coalition leader. One can infer from these results that the completion of the mission is not guaranteed when the message loss rate is over 50%. Since synchronization is highly required for multi-robot coordination of tasks 6-9, message losses cause synchronization to be lost and also the auction negotiation steps are not completed reliably. However it should be noted that the framework can easily handle message loss rates smaller than 50% (a very high message loss rate for real life experiments). In those cases, in spite of robot failures, even if the failed robots are coalition leaders, the mission is completed successfully as if there were no robot failures. In the 2 failures case, the mission is not completed in some runs, because the randomly selected robots belong to the same set and when they fail, mission completion is not possible. For the message loss rate 100%, the number of runs in which the failed robot is a coalition leader is 100. This is because, in this case, each robot is the coalition leader of its own task with $reqno = 1$, and since the robots cannot communicate, the execution of the tasks with $reqno = 1$ (0-4) are completed in later steps and all robots try to execute all these tasks by themselves. The mission completion time results can be seen in Fig. 4. The mission completion time increases for increasing message loss rates logarithmically. However, even when message loss rate is 70%, the mission can be

Table 3: Mission Completion Ratios.

Message Loss Rate (%)	#of runs in which the mission is completed				
	no failure	1 robot failure		2 robots failure	
	#runs	#runs	$\#CL_i$	#runs	$\#CL_i$
0	100	100	22	83	39
10	100	100	22	79	45
20	100	100	27	81	33
30	100	100	27	78	42
40	100	100	20	77	31
50	98	95	19	72	27
60	59	48	15	30	27
70	4	3	13	2	18
80	0	0	5	0	19
90	0	0	4	0	25
100	0	0	100	0	100

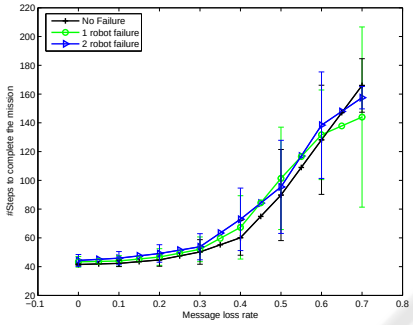

Figure 4: Mission Completion Time for different message loss rates, averaged over 100 runs.

completed in some runs. As expected, no failure case has a better success rate. However, even if there are failures, results are still very close to the no failure case. Therefore, we conclude that the framework can easily handle robot failures.

The total number of messages also increases logarithmically for the changing message loss rates as illustrated in Fig. 5. The robots continuously query the *free* tasks and try to coordinate the simultaneous task execution. The number of messages for no failure case is also greater when the message loss rate increases, due to the number of robots and their efforts to coordinate. When the results are evaluated, it can be concluded that the framework handles message losses as well as possible.

4 CONCLUSION

We evaluate our multi-robot cooperation framework on complex missions with inter-related tasks requiring heterogeneity and coordination. The recovery solutions provided by precaution routines for different

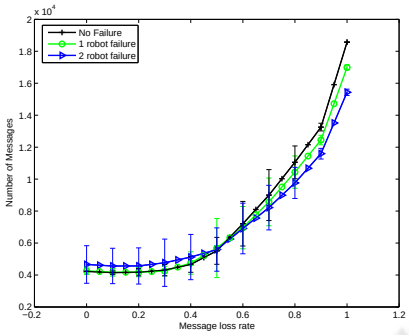

Figure 5: Total Number of System Messages for different message loss rates, averaged over 100 runs.

kinds of failures ensure the completeness of the approach. Experiments to validate the robustness of the approach are conducted in the simulated object construction domain with message losses and robot failures. Even for very high message loss rates, the robot team can still complete the mission using the proposed framework. Our next step in this research is the real field experiments on real robots and extensive cost/bid designs and analyses for different domains.

REFERENCES

Balch, T. and Arkin, R. C. (1994). Communication in reactive multiagent systems. *Autonomous Robots*, 1(1):1–25.

Decker, K. (1996). *Foundations of Distributed Artificial Intelligence*, chapter TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms, pages 429–448. John Wiley and Sons.

Dias, M. B., Zlot, R. M., Kalra, N., and Stentz, A. (2005). Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Carnegie Mellon University, Robotics Institute.

Gerkey, B. and Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation. *Intl. J. of Robotics Research*, 23(9):939–954.

Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581.

Horling, B. and Lesser, V. (2005). A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review*, 19(4):281–316.

Sariel, S. and Balch, T. (2006). Efficient bids on task allocation for multi-robot exploration. In *The 19th International FLAIRS Conference*.

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