J Intell Robot Syst (2011) 63:323–358
DOI 10.1007/s10846-011-9558-4

A Generic Framework for Distributed Multirobot Cooperation

Sanem Sariel-Talay · Tucker R. Balch · Nadia Erdogan

Received: 7 December 2009 / Accepted: 14 March 2011 / Published online: 25 May 2011

Abstract DEMiR-CF is a generic framework designed for a multirobot team to efficiently allocate tasks among themselves and achieve an overall mission. In the design of DEMiR-CF, the following issues were particularly investigated as the design criteria: efficient and realistic representation of missions, efficient allocation of tasks to cooperatively achieve a global goal, maintenance of the system coherence and consistency by the team members, detection of the contingencies and recover from various failures that may arise during runtime, efficient reallocation of tasks (if necessary) and reorganization of team members (if necessary). DEMiR-CF is designed to address different types of missions from the simplest to more complex ones, including missions with interrelated tasks and multi-resource (robot) requirements. Efficiency of the framework is validated through experiments in three different types of domains.

Keywords Multirobot systems · Distributed multirobot cooperation · Task allocation · Incremental task selection · Robustness

1 Introduction

The real dynamics of physical task performance enforce the unplanned actions to be taken. Since the world is beyond the control of robots and changes continuously in real-world applications, the difficulty of the multirobot task execution problem

S. Sariel-Talay (✉) · N. Erdogan
Department of Computer Engineering, Istanbul Technical University,
Maslak, Istanbul, Turkey
e-mail: sariel@itu.edu.tr

T. R. Balch
College of Computing, Georgia Institute of Technology,
Atlanta, GA, USA

goes beyond the task allocation problem. In particular, multirobot systems deal with difficulties arising from noisy sensor information, unexpected outcomes of actions, environmental limitations (especially in communication) and the presence of failures of hardware. Furthermore, the problem instance may be evolving through real time which is another important research challenge. All these factors may affect the overall solution. Against this background, this research addresses issues of real-time execution when managing an overall team by a central authority is not possible due to limitations of the real-world environments. Therefore, each individual robot should find a way to solve the global problem from a local perspective in a decentralized way.

The main contribution of this research is the design of a general framework, **D**istributed and **E**fficient **M**ulti-**R**obot-**C**ooperation **F**ramework (DEMiR-CF), that can be used to solve problems on a wide variety of application domains. DEMiR-CF is suitable for multirobot teams cooperating to achieve a global mission. Team members cooperate to fulfill a mission by dividing the labor of task execution through individual decisions that coordinate their actions, contributing to the achievement of the goal in a distributed manner. The behaviors of the robots that run on DEMiR-CF are shaped by architectural components that are in parallel with the BDI software model. In essence, each robot maintains its own beliefs about the current states of mission tasks and resources. Its actions are driven by its desires that overlap with the goals of the overall mission and its intentions to select and execute particular tasks of the mission. The robots determine their subgoals by constructing their schedules in a computationally tractable way. They announce their intentions on task execution by single-item auctions which are cleared in a distributed manner. DEMiR-CF demonstrates a robust execution. Each robot is continuously involved in failure detection and recovery as well. Robots keep and monitor the states of the system models locally and execute precaution actions when necessary in order to realize the common goal.

The overall multirobot execution problem is formulated as the Cooperative Mission Achievement Problem ($CMAP$) and each individual robot contributes to solve the $CMAP$ by solving the formulated Coordinated Task Selection Problem ($CTSP$) for itself. These two problem formulations are introduced to represent the generalized problem and stated formally in [29]. The proposed framework is capable of solving $CMAP$ and $CTSP$ for each robot.

The framework ensures an efficient way of integrating task planning, allocation and execution for multi-entity (agent/robot) teams independent of the underlying low-level behavior architecture, yet without ignoring it. The integrated components of the framework ensure solving the $CMAP$ in a robust and efficient way. As a result, global planning, scheduling and execution is carried on by the cooperative work of robots performing under DEMiR-CF. Even though an incremental task selection approach is adopted, a global plan consideration is also preserved to make the system both sound and complete.

The performance evaluations of the framework were implemented both in simulators and on real robots for different application domains. Each application domain is a separate problem domain which desires in depth research.

The Multiple Traveling Robot Problem (MTRP) to explore several targets is the first application domain on which initial tests were performed. This domain forms a basis for different application domains. Although tasks of this domain are

independent of each other, there is a combinatorial structure of the problem if efficiency of the solution is concerned. Therefore, optimization of the generated solutions is investigated. Some heuristic cost functions to solve the $CTSP$ is proposed to be used with DEMiR-CF in [32]. The performance of the framework together with the proposed heuristics is compared with that of one of the well known allocation approaches.

The evaluations on NAVY domains were performed where cooperation of underwater vehicles is achieved for homeland security missions, such as mine countermeasure mission. In this domain, the robot team is modeled as a heterogeneous team and the mission is constructed from different types of tasks, where each one requires to be performed by a different vehicle. The domain is modeled as containing both the coverage problem and the MTRP in itself. Robustness of the framework against both communication and robot failures and the efficiency of its response to the dynamically varying conditions is tested in simulations.

In general, DEMiR-CF targets complex missions involving tasks with resource constraints and interrelations. Therefore, these types of domains are perfect candidates to apply the full functionality of the framework. In the last experimental setup, specifically, pick-up/delivery and object construction domains are investigated as complex domains. Different from previous experiments, the robots involve in more complex tasks where they interact with the objects in the environment. The objective is not only optimizing cost functions but also obeying rules and resolving constraints on task execution during runtime. The base mechanisms of DEMiR-CF are used for designing the solution. The efficiency of the proposed solution for complex domains is evaluated through simulation and real-robot experiments.

The rest of the paper is organized as follows. Section 2 presents the formulated cooperative multirobot mission achievement problem and the real-time issues that should be addressed in multirobot systems. Related work is reviewed in Section 3. DEMiR-CF framework and its components are presented in Section 4. Three case studies of DEMiR-CF are presented in Section 5. The first case study investigates the MTRP domain and presents how DEMiR-CF is used for this domain. In the second case study, NAVY missions are addressed and the task representation and allocation strategies of DEMiR-CF are presented for this domain. The third study evaluates the performance of DEMiR-CF on complex missions with resource constrained and interrelated tasks. The computational analysis of the approach is presented in Section 6. Finally, Section 7 concludes the paper.

2 Problem Statement and Motivation

General multirobot task allocation problem for complex missions may be formulated based on the well known Resource Constrained Project Scheduling Problem (RCPSP) [7]. RCPSP is known to be an NP-Hard problem [38] in Operations Research (OR). Beyond this base problem, unpredictability of the exact processing times of tasks, unstable cost values during runtime and inconsistencies due to uncertain information form the main difficulties of the task allocation problem for robot systems. Particularly, robots deal with the constraints of real-world missions which may change their forms by introducing new online tasks during execution, making the problem more challenging besides the real-world dynamism.

Evolving circumstances that may change the overall solution to a real-world problem can be listed as:

- Self failure detection: Robots detecting their own failure.
- Robots detecting the failure of another robot.
- Change in the estimated task execution cost/time: Environmental dynamics, uncertain knowledge, or hardware problems may cause delays in task execution or early achievements of tasks. Uncertain sensor and/or localization information may also result in incorrect estimations.
- Change in the task definitions: Task dependencies, priorities, or the overall objective (goal) may change. Some tasks may become invalid during runtime.
- New online tasks introduced by human operators or discovered by the robots themselves.
- New robots being released, or some failed robots being repaired or recovering from trap-like threats.
- Intervention and manual changes on the assignments by external agents.

Some of these situations may arise after either internal or external events. Given these contingencies, even the result of an approach capable of finding the optimal solutions may become suboptimal under the uncertainties of the real-world applications. Verification of the solution optimality is also a difficult issue for real-world applications.

2.1 Formulation of the Cooperative Mission Achievement Problem

The adapted version of RCPSP to the formulation of the multirobot task allocation problem in investigation on project tasks is given as follows. A complex mission consists of a set of tasks $T = \{t_1, ..., t_n\}$ which have to be performed by a team of robots $R = \{r_1, ..., r_m\}$ each having a certain set of capabilities (cap_j). The tasks are interrelated by two types of constraints. First, precedence constraints are defined between activities. These are given by the relations $t_i \prec t_k$, where $t_i \prec t_k$ means that task t_k cannot start before task t_i is completed. Second, a task t_i requires a certain set of capabilities $reqcap_i$ and certain number of robots (resources) $reqno_i$ to be performed.

Using the given notation, Scheduling Problem (ScP) is defined as determining starting times of all tasks in such a way that:

- at each execution time, the total $reqno_i$ for a task t_i is less than or equal to the number of available robots ($R_{Sj} = \cup r_j$) with $reqcap_i \subseteq cap_j$ (Condition-1).
- the given precedence conditions (Condition-2) are fulfilled, and
- the makespan $C_{\max} = \max(C_i), 1 \leq i \leq n$ (Objective, O) is minimized, where $C_i = S_i + p_i$ is assumed to be the completion of task t_i, where S_i is the actual starting time and p_i is the actual processing time respectively.

This problem can also be stated as a multiprocessor task scheduling problem and it is proved to be an NP-Hard problem by [6].

Beyond this base problem, the real-time issues presented earlier add further dimensions into this problem. It's not always possible to estimate the exact processing times (p) of tasks in real-world missions, especially those in which robots are

involved. However, to form a complete schedule, it is necessary to make an approximation in terms of the best knowledge available.

The cost values are unstable during runtime and usually inaccurately estimated. Since the sensor values of the robots are noisy and world knowledge is uncertain, inconsistencies are unavoidable. Particularly, robots also deal with real-world missions that may change their forms by introducing new online tasks during execution, making the problem more challenging besides the real-world dynamism.

The difficulty of the task allocation/reallocation problem arises when communication is limited and robots should autonomously perform task allocation at the same time as task execution. Simultaneous execution requirements make the problem more challenging because each robot should be in its most suitable execution in a future formation and estimate it correctly before making a decision with the minimum communication possible.

The Coordinated Task Selection Problem ($CTSP$) is a part of the Scheduling Problem (ScP) and is stated for each robot. When each $CTSP$ is solved in a time-extended manner, an overall schedule of the tasks and their executors can be found. The new formulated task selection problem that robots try to solve is given as follows: $CTSP$ is an online selection problem for each robot (after either being idle or completing a task) to determine the next task t_i to be selected in such a way that:

- task t_i is not achieved yet,
- total $reqno_i$ for task t_i is less than or equal to the number of available robots ($R_{Sj} = \cup r_j$) with $reqcap_i \subseteq cap_j$ (Condition-1),
- the given precedence conditions (Condition-2) are fulfilled,
- and the selected objective (O) is minimized.

Including the $CTSP$, the Cooperative Mission Achievement problem ($CMAP$) for each robot is formulated as follows:

1. Select the task in such a way that the $CTSP$ is satisfied,
2. Determine the most appropriate robot (coalition) according to communication or beliefs to execute the task; resolve conflicts, if any,
3. Execute the selected task efficiently, if it is appropriate to execute, and
4. Simultaneously respond to contingencies and return to Step 1, when necessary, until the mission is achieved.

The second step in the formulation of $CMAP$ is required due to the uncertainties in the knowledge of robots. If robots had complete knowledge over the world state, then this step would become redundant. Real-world limitations make the fourth step an inevitable part of this problem.

3 Background and Related Work

Different frameworks proposed in literature touch separate aspects of the research questions presented here. Current research in multirobot coordination addresses the issue of dynamic task allocation in the presence of robot failures and environmental changes. Earlier experiments are usually on either simulated or real robot teams implementing simple and independent tasks. However, complex task domains

including heterogeneous robot teams working on interrelated tasks facing with time constraints, sensing uncertainties, and non-deterministic actions have not yet been fully investigated with a complete framework.

According to the taxonomy presented in [20] Multirobot Task Allocation (MRTA) problems can be classified within three different dimensions: Single-task robots (ST) vs. Multi-task robots (MT); Single-robot tasks (SR) vs. Multirobot tasks (MR); Instantaneous assignment (IA) vs. Time-extended assignment (TA). Different multirobot systems can be classified based on these dimensions.

Although tasks can be classified according to single/multiple robot requirements on task execution, the general multirobot team may be either a homogeneous or a heterogeneous group. The heterogeneity may be in the possessed capabilities or in the task execution performance. For example, the robots may have the same equipments capable of achieving all the tasks of the mission but may differ in abilities such as speed. Dahl et al. [10] address this issue in their task allocation method through vacancy chains. This method ensures differentiation between robots based on their individual performances different than the physical and sensor capabilities. High-value tasks are assigned to the high-performance robots.

Task dependency has been analyzed in some earlier multirobot cooperation schemes. Alami et al.'s [2] initial work on multirobot coordination presents a generic scheme based on a distributed plan-merging process. Although optimality is not guaranteed, Plan Merging Operation (PMO) provides a coordinated plan. A deadlock resolution method is implemented in a distributed manner. M+ scheme [5] combines local planning and negotiation for task allocation, and cooperative reaction to contingencies. In their framework, a mission is a set of partially ordered tasks. Each robot has its own local world knowledge. Tasks are allocated through negotiation processes. Alami and Botelho [1] introduces the mechanism concept in their framework M+CTA as an improvement to the M+ scheme. Each robot has an individual plan and tasks are initially decomposed and then allocated. After this planning step, robots negotiate with each other in order to incrementally adapt their plans in the multirobot context.

Instantaneous task assignment becomes profitable as it provides a dynamic solution allocating tasks to robots whenever resources are available [19, 25]. Parker [25] has presented, one of the earlier works for instantaneous multirobot task assignment, with a behavior based framework, ALLIANCE, and further extended it by integrating learning into the system in L-ALLIANCE. When tasks are assigned instantaneously, the global solution quality may be degraded if the decisions are made by just using the up-to-date knowledge available, ignoring the global solution quality.

Multirobot task allocation is better viewed as a scheduling problem when there are interrelations and dependencies among tasks. When the resources are not adequate (e.g., problem solving time is limited), Branch and Bound or MILP methods may not be convenient. In this case, heuristic methods are preferred to find a good solution in reasonable time. Furthermore, finding a solution with these approaches in a decentralized setting may need considerably high efforts in both computation and communication. Paquet [24] models the multiagent task assignment problem as a scheduling problem for the RoboCupRescue simulation domain. The main objective is the maximization of the number of rescued civilians in a simulated disaster environment.

As desJardins et al. [12] state, an agent should plan continually for dynamic environments or under time constraints. It is better to delay plan refinement as long as possible, so that detailed decisions are made with as much information as possible. They argue that simply combining distributed and continual planning methods independently may not be sufficient and more intelligent integration approaches are needed.

According to the classification of multiagent organizations given in [21], coalitions (agent groups) are formed to perform tasks in cooperation. Coalitions are suitable for meeting the simultaneous resource requirements of executing tasks with a subteam of robots. Shehory and Kraus [33] present one of the earlier algorithms for coalition formation for cooperative multiagent systems. During coalitional value calculations, agents' capabilities are taken into consideration. In multirobot systems, the cost values are a function of not only capabilities but also the physical conditions which change during execution, such as robot's location, object/subject's location, etc. When the robots decide to perform a task, both subjects in the environment and robots' physical entities (e.g., fuel) change. Vig and Adams [36] state the differences of multirobot and multiagent coalition formation issues from sensor possessive point of view. Locational sensor capabilities are considered which may be applicable in the beginning of mission execution. However, another important factor in multirobot systems for evaluating coalitions is the changing cost values during runtime. They assume robot capabilities do not change, however, this is not the case for the costs.

ASyMTRe [26], uses reconfigurable schema abstraction for collaborative task execution providing sensor sharing among robots. In ASyMTRe, connections among the schemas are dynamically formed at runtime. Their approach is used to form low level coalitions to share robot capabilities.

The dynamic task allocation problem has been investigated in the face of robot failures (including partial/complete failures) or environmental changes. In most cases which include failure detection, the test domain missions include independent subtasks that can be executed by a single robot. ALLIANCE [25], provides a mechanism to handle robot failures through using the motivational behaviors. Dias et al. [14] investigate the performance of their framework, Traderbots, against different kinds of failures such as communication failures, partial malfunction or death. In their work, execution conflicts between robots are resolved by continuous auctioning by one of the robots. Gerkey and Mataric [19] evaluates the MURDOCH against the robot failures for tightly coordinated task execution in which there is a leader giving appropriate directives against the changing situation of the system after failures. In M+ [5], contingencies in task execution (task failures) are handled by re-planning for execution of the goals at hand by a robot. The watch-out task introduced in Lemaire et al.'s [22] work has an interesting property providing cooperative work against the communication failures.

Recent studies have revealed that the distributed approach and, especially when complemented with the auction based methods, has shown great promise for multirobot task allocation because of its scalability. M+ [5], one of the most successful architectures for distributed task allocation and achievement, addresses many real-time issues, including plan merging paradigms. MURDOCH [19] is a framework that achieves publisher/subscriber type allocation for instantaneous assignment. Dias [13] proposes a combinatorial auction-based task allocation scheme: TraderBots. Zlot and Stentz's [39] work on task tree auctions is presented as an extension to the

Traderbots approach to address more complex tasks which can be decomposed into task trees. In the combinatorial auction methods, even when effective methods are used to define bundles of items, communication requirements and efforts for negotiations and bidding grow exponentially with the task size. Lemaire et al. [22] propose a task allocation scheme for multi-UAV (Unmanned Aerial Vehicles) cooperation with balanced workloads of robots. Gancet et al. [18] have proposed a framework for multi-UAV coordination. Their approach supports both centralized and distributed coordination and uses synchronization signals to coordinate synchronization among UAVs.

Market-based approach seems to ensure a scalable way of solving the multirobot task allocation problem. However, following the remarks made by Dias et al. [15], existing market mechanisms are not fully capable of re-planning task distributions, changing decomposition of tasks, rescheduling commitments, and re-planning coordination during execution. From dynamic events dimension, there is not a formalized study of response speed for any multirobot coordination approach. Scalability in the market-based approaches may be limited by the computation and communication needs that arise from increasing auction frequency, bid complexity and planning demands. On the other hand, OR methods may not be directly applicable to the multirobot coordination for complex tasks due to real-time execution constraints.

DEMiR-CF is different from earlier work by ensuring both instantaneous assignment procedures incrementally and forming rough schedules to consider the problem as a whole from global perspectives. The rough schedule generation process takes polynomial time even for complex tasks. Combinatorial task exchange mechanisms as in market-based approaches are not used in DEMiR-CF. Auctions are used by robots to announce intentions on task execution and selection of the appropriate task executers to deal with the incomplete world information. Only single items are allocated incrementally during task execution. Contingency handling mechanisms are directly integrated into the dynamic task selection mechanisms which in turn facilitate recovering from failures, reconfiguring robots during runtime dynamically and efficiently, and maintaining system consistency. These utilities are ensured by autonomous robots implementing DEMiR-CF in a completely distributed manner without central authorities and/or complete knowledge injected manually.

4 DEMiR-CF

In practical applications, computing the true optimal solutions is not always required due to several reasons [28] such as the incorrect modeling of the underlying problem or lack of sufficient time to find the optimal solution. These are common cases in robot applications along with the real-time issues presented in Section 2. As a solution method to CMAP that meets these limitations, a dynamic and distributed task allocation scheme, **D**istributed and **E**fficient **M**ulti-**R**obot-**C**ooperation **F**ramework (DEMiR-CF) [29], is proposed to coordinate robots that cooperate to fulfill different parts of a mission. DEMiR-CF combines *The Dynamic Priority-Based Task Selection Scheme*, *Distributed Task Allocation* and *Coalition Maintenance Schemes* as cooperation components and *The Precaution Routines*, some of which are implemented by *The Coalition Maintenance/Dynamic Task Selection Scheme*. These components are integrated into a single framework to provide an overall system that

finds efficient solutions for real-time task execution. The modules that embody the framework and information flow among them are given in Fig. 1. Each robot keeps a model, up-to-date state information, of the other robots and the mission tasks. *The Model Update Module*, *The (System) Consistency Checking Module*, and *The Dynamic Task Selector Module* perform *precaution routines* by either updating the model maintained by the robot or activating warning mechanisms. Model updates are initiated by either incoming information from the other robots or the information perceived by the robot itself. If a system inconsistency exists, *The Consistency Checking Module* is responsible for initiating warning mechanisms and informing the corresponding robots. *The Dynamic Task Selector Module* employs *Dynamic Priority-Based Task Selection Scheme (DPTSS)* to select the most suitable task by considering the model of the robot. *The Distributed Task Allocation Scheme* ensures distributed task allocation by executing the required negotiation procedures for the selected task. *The Execution/Coalition Scheme* implements synchronized task execution and coalition maintenance procedures. Task models are updated according to the selected task and the task currently in execution.

A sample flow of the operations for a robot in the framework (as depicted in Fig. 1) is summarized below:

1. Initially, the mission task definitions are delivered to the robots.
2. Each robot selects the most suitable candidate task to execute through global cost consideration (dynamic task selection or switching).

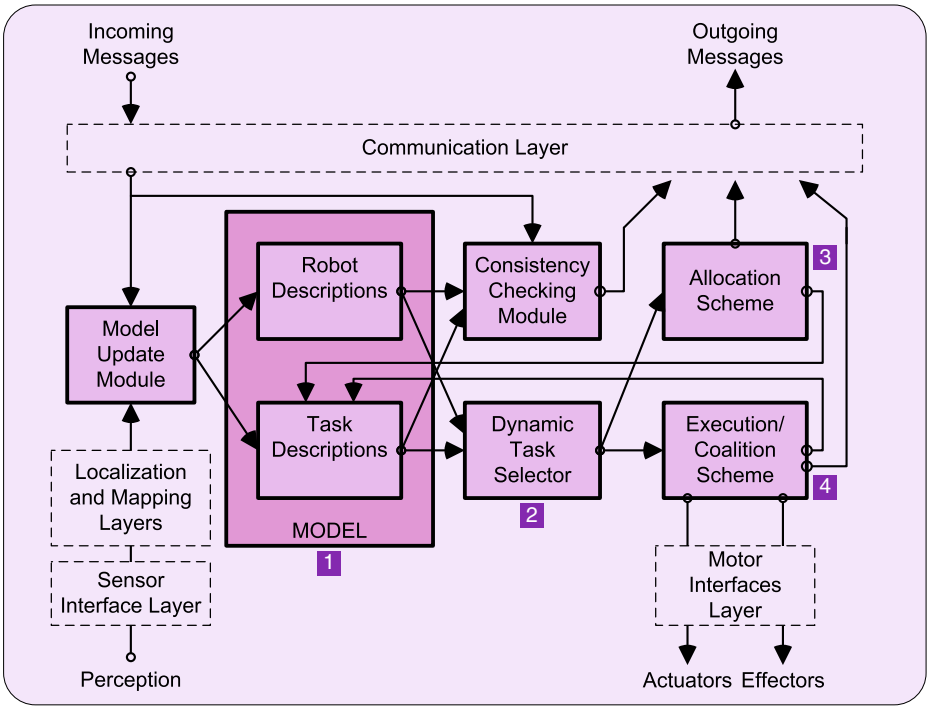


Fig. 1 DEMiR-CF: integrated modules and their interactions

3. Robots offer auctions for the tasks they have selected. During auction steps, inconsistencies are cleared and conflicts are resolved.
4. Task assignments are made for the announced tasks, making sure that each robot takes part in the most suitable execution when the global solution quality is considered.
5. Dynamic task selection or switching proceeds simultaneously with task execution. This allows the robot to switch between tasks when executing another task becomes more profitable than continuing with the current task, and to handle the real-time contingencies at the same time. Hence, corresponding auction and selection procedures (second through fourth items) are applied continually.

DEMiR-CF is designed with the capability to deal with real-time situations. The framework can efficiently respond to these events and maintain the solution quality simultaneously with real-time task execution.

The overall objective of the robot team ($r_j \in R$, $0 < j \leq ||R||$) in the framework is to achieve a mission (M) consisting of independent or interrelated tasks T_i ($0 < i \leq ||M||$), by incremental assignment of all $T_i \in M$ to $r_j \in R$ while optimizing the specified objective function. Tasks are preemptive, that is the activity of task execution can be split during runtime if another advantageous situation arises or environmental conditions impel.

Coalitions ($Coal_i$) [21] are formed to meet simultaneous execution requirements of tasks (T_i) synchronously by a group of robots. An example of such a task that needs to be executed by a coalition of robots is pushing a heavy object which cannot be moved by a single robot. The size of a coalition vary according to the minimum number of robots required ($reqno_i$) to execute the corresponding task. A coalition[1] may involve only a single robot for a task to be achieved.

4.1 Mission Representation

A mission in DEMiR-CF is represented by a directed acyclic graph (DAG) where each node represents a task and the directed arcs (conjunctive arcs) represent the precedence constraints among tasks. Interrelations among tasks can be represented by either adjacency-lists or matrices where each node represents a task. Tasks are represented as septuples containing information regarding task execution requirements and task status: $< id, type, reqcap, deplist, reqno, relinfo, precinfo >$.

1. *id*: A system-generated unique task identification number common to all robots before mission execution.
2. *type*: A description of task type and corresponding action definitions.
3. *reqcap*: Requirements defining special sensors and capabilities required to execute the task.
4. *deplist*: The two types of dependencies representing precedence relations. A hard dependency implies sequential execution of the related tasks while a soft dependency allows parallel execution.
5. *reqno*: The minimum number of robots required to execute the task, determined either before mission execution or during runtime.

[1]The term coalition is also used for a single robot executing a task for the sake of generality.

6. *relinfo*: Descriptive information regarding the task type, such as the latest location of an object, the target location, etc.
7. *precinfo*: Precaution information used for contingency handling: the task state, the estimated task achievement time and the current execution cost.

Representation of a task can be dynamically modified during execution. In particular, *relinfo*, *precinfo* and *reqno* are subject to change during execution.

4.2 Dynamic Priority-Based Task Selection Scheme

In DEMiR-CF, the robots make instantaneous decisions (from their local perspectives) which are both precedence and resource feasible in the context of the global time-extended view of the problem. While the completion of the mission is the highest priority objective, performance related objectives can additionally be targeted. Each robot initially forms a rough schedule of its activities for an overall time-extended resolution of the mission. Since these schedules are highly probable to change in dynamic environments and robots also have the real-time burdens of path planning, mapping etc., the formed rough schedules are tentative and constructed by computationally cheap methods (explained in Section 4.2.1). Therefore, robots come up with their rough schedules and refine their plans during actual fast execution when information available in the current context enables them to make specific, detailed decisions.

Task selection and allocation is performed by evaluating each task according to the selected cost function. Depending on the objective, different cost functions can be defined, from the simplest functions to more complex and composite evaluations. Cost evaluation is one of the key issues to make the framework suitable for different domains in an efficient and effective manner. Cost functions are analyzed in Section 4.3.

Since schedules are subject to change, the focus is on an approach in which tasks are not scheduled initially but instead allocated to robots incrementally, without ignoring the overall global solution quality. Therefore, the main objective becomes determining a particular task to be assigned whenever it is convenient in a precedence and resource feasible manner, instead of scheduling all the tasks from scratch. Although not a concern during assignments, preemption (i.e., yielding) is possible to maintain the solution quality and to handle failures during execution. Therefore, the allocation problem turns into a selection problem and is stated as $CTSP$, which is introduced earlier.

Note that $CTSP$ presented earlier is an optimization problem as in ScP, and it is desirable to find a solution by considering the problem from the global perspective. Therefore, the instantaneous task selection scheme needs to be strengthened by considering the problem as a whole, with the designed cost evaluation functions. Depending on the objective function, either priorities or penalties can be applied to find a near-optimal solution ensuring a time-extended view of the problem. This issue is analyzed in detail in Section 4.3.

The following definitions are needed to present the proposed formulation to solve $CMAP$.

Definition (*suitable task and suitable robot*) t_i is a suitable task for robot r_j, if $reqcap_i \subseteq cap_j$ and r_j is a suitable robot for t_i.

Definition (*executable task*) t_i is an executable task, if it is a *suitable task* and at least $reqno_i$ number of robots can be assigned for its execution.

Definition (*task in execution*) t_{iej} is a task in execution by robot r_j or coalition C_j. T_{ie} is a union of tasks in execution.

Definition (*eligible task*) t_{Ej} is an eligible task, if it is an *executable task* and is neither in execution (t_{ie}) nor achieved. T_{Ej} is a union of *eligible tasks* for robot r_j.

Definition (*ineligible task*) t_ϕ is an ineligible task if it is not an *executable task* or it is already achieved. T_ϕ is a union of *ineligible tasks*.

Definition (*predecessor task set*) $P(t_i)$ is defined as the set of all predecessor tasks of task t_i.

Definition (*active task*) t_{Aj} is an active task if it is *executable* and tasks in $P(t_{Aj})$ are completed. $T_{Aj}(\subseteq T_{Ej})$ is a union of the *active tasks* for robot r_j.

Definition (*inactive task set*) is the set of all inactive tasks (t_{Ij}), $T_{Ij} = T_{Ej} \setminus T_{Aj}$ contains the tasks that are *suitable* but not *executable* yet for robot r_j.

Definition (*critical task*) A critical task t_{Cj} is a task that has inflexibility from the point of view of resources and robot r_j is suitable for that task. T_{Cj} is a union of *critical tasks* for robot r_j.

Definition (*rough schedule*) A rough schedule S_{Rj} for robot r_j is a priority queue of mission tasks that r_j assumes it will execute.

4.2.1 Rough Schedule Generation Scheme

Each robot r_j generates its rough schedule as a dynamic priority queue similar to runqueues, by considering its critical task set (T_{Cj}), the eligible tasks (T_{Ej}), the conjunctive arcs (if any) and the requirements. This priority list is generated by considering cost values for the tasks (Section 4.3). If there are no new online tasks or invalidations, the order of the tasks which are connected by the conjunctive arcs remains the same in the priority queue, even though there may be additional intermediate entries into the queue at runtime.

Since each robot r_j has different capabilities, the eligible task sets (T_{Ej}) and the priority queue entries will be different. Sometimes uncertain information (e.g., related to a local online task) or unexpected (internal or external) events (e.g., detection of a fuel leakage) may result in this difference, even when robots possess the same type of capabilities. The critical tasks may be determined either by negotiations or by beliefs. Critical task information is used for determining the task requirements in terms of consumable resources such as power, fuel etc.

Intuitively, robots do not deal with the ineligible tasks (T_ϕ), the union of tasks that are already achieved or those that are not eligible from the capabilities perspective while forming the rough schedules. The eligible tasks ($T_{Ej} = T \setminus T_\phi$) for robot r_j consists of active and inactive tasks. The rough schedule of a robot is generated

by execution of Algorithm 1. $curcs_j$ represents the current capacity of robot r_j, $remcs_j$ remaining capacity after executing all of its critical tasks and $reqcs(t_i)$ the required capacity for task t_i in terms of a consumable resource (e.g., fuel). In the rough schedule generation algorithm, while the rough schedule is being formed, the remaining capacity of the robot is also monitored. If the capacity of the robot is not sufficient for executing all of its critical tasks and the mission is believed to be unachievable accordingly, then the robot may select an active task to execute even if it is not a critical task for the robot. However, if the mission is believed to be achievable, the robot may select to stay idle until its critical tasks become active. This selection is done after forming the rough schedule. The active task on the top of the rough schedule that can be executable is the most suitable task to be executed for the robot since costs are considered during the priority list generation. The priority values to form rough schedules are determined based on the mission and the objective function which will be explained in Section 4.3. Sometimes the rough schedule of the robot may be empty. In this case, the robot selects to stay idle as determined in the DPTSS algorithm.

Algorithm 1 GenerateRoughSchedule for robot r_j

input: Eligible task set (T_{Ej}), active task set (T_{Aj}), critical task list (L_{Cj}), current capacity ($curcs_j$) of robot r_j, the required capacities for tasks t_i ($reqcs(t_i)$)
output: Rough schedule (S_{Rj}) of tasks, the top most suitable active task (t_s)

$t_s = \phi$; $remcs_j = curcs_j$

/*Priority list formation may be different depending on the application domain and the cost evaluations*/
S_{Rj} = GeneratePriorityList(T_{Ej}, T_{Aj})

/*Determines if the mission is achievable from a local perspective*/
for each $t_i \in L_{Cj}$ **do**
  $remcs_j = remcs_j - reqcs(t_i)$
  if $remcs_j < 0$ **then**
    /*Mission is not achievable*/
    $remcs_j = curcs_j$
    break
  end if
end for

if $S_{Rj} \neq \phi$ and ($top(S_{Rj}) \in L_{Cj}$ or $remcs_j - reqcs(top(S_{Rj})) \geq 0$) **then**
  $t_s = top(S_{Rj})$
end if

4.2.2 DPTSS Algorithm

In the proposed incremental allocation approach, the fundamental decision that each robot must make is the selection of the most suitable task from the active task set (T_A) by considering eligible task set (T_E). Algorithm 2 presents the DPTSS in which

a rough schedule is generated before making a decision. The four different decisions made by robots after performing DPTSS are:

- continue to execute the current task (if any),
- join a coalition,
- form a new coalition to perform an available task, or
- stay idle.

The dynamic task switching scheme is used by robots to dynamically switch between tasks if updates in the world knowledge compel. Therefore, issues related to both online scheduling and scheduling under uncertainty are addressed.

Algorithm 2 DPTSS Algorithm for robot r_j

input: Mission (M) task descriptions, current task of robot r_j (t_{iej})
output: Action to be performed depending on the selected task (t_s)

Determine the T_{Ej}, $T_{Aj} \subseteq T_{Ej}$

for each ($t_{iek} \in T_{ie}$ and the execution cost is lower than announced) **do**
  $T_{Aj} = T_{Aj} \bigcup t_{iek}$
end for

L_{Cj} = GenerateListOfCriticalTasks(T_{Ej})
$t_s = t_{iej}$

if ($T_{Aj} \| T_{Ej} \| L_{Cj}$ is changed) **then**
  $[S_{Rj}, t_s]$ = GenerateRoughSchedule (T_{Ej}, T_{Aj}, L_{Cj}, $curcs_j$)
end if

if $t_s \neq \phi$ **then**
  if $t_s = t_{iej}$ **then**
    Continue with the current execution
  else
    if $t_s \in T_{ie}$ **then**
      Join the coalition
    else
      Offer an auction to form a new coalition or directly begin execution
    end if
  end if
else
  Stay idle
end if
$t_{iej} = t_s$

The DPTSS process is repeated whenever a robot completes its current task execution or detects a change in its world knowledge. Instead of regenerating the rough schedule at each call of the DPTSS, the rough schedule may be repaired whenever it is desirable.

4.3 Cost Evaluation for Rough Schedule Generation and Dynamic Task Selection

Cost evaluation has a high impact on the solution quality for systems that need some forms of optimization procedures, and research in this area requires more investigation. Unless efficient cost evaluation strategies are designed, it is not possible to observe globally high quality solutions for NP-Hard problems, and additional adjustments are required to change allocations with an additional cost of communication as in combinatorial auctions.

According to the taxonomy given in [20], multirobot task allocation problems are divided into two classes based on the mission description: instantaneous vs. time-extended. Most multirobot architectures offer solutions for instantaneous assignments. DEMiR-CF lies in between these two extreme approaches due to its incremental allocation strategy by rough schedule generation scheme. The global solution quality is improved by a time-extended view of the problem. However, the approach is also capable of offering solutions for instantaneous changes on the task assignments. Incremental assignments eliminate redundant considerations for environments in which a current best solution is highly probable to change, and efficient and intelligent bidding strategies ensure solutions to be close to optimal with a time-extended view of the problem in a computationally tractable way.

Depending on the selected application domain and a regular objective function, cost functions should be designed appropriately. Different types of rough schedule generation schemes can be performed by using the designed cost functions. At this point, it is necessary to distinguish the rough schedule generation schemes for independent tasks and interrelated tasks with workforce constraints.

4.3.1 Independent Tasks with Combinatorial Structures

There are several OR methods to allocate independent tasks to robots. Optimal results can be obtained by an efficient Integer Programming (IP) formulation in IP solvers (e.g., the commercial IP solver CPLEX [9]). Branch and Bound algorithms may also be used to perform the search. The performance of the approach is dependent on the selected branching and bounding schemes [35].

Heuristic approaches may also be used to find approximate solutions. The methods are classified into two subclasses: Classical Heuristics and Meta-heuristics. In the classical heuristic approach, standard construction and improvement methods are applied to the solution. These approaches perform limited exploration of the search space and typically produce good results within modest computing times. In the meta-heuristics approach, the algorithm can perform a search through a large solution space. Evolutionary algorithms, Tabu Search and Simulated Annealing methods are the common methods belonging to this class. Although the quality of the solution is much higher than that of the classical approaches, time complexity increases dramatically in these approaches. The applied procedures are usually context dependent and require finely tuned parameters [35].

All these cost evaluation procedures can be used in DEMiR-CF. Each robot may simply perform the explained operations to generate the rough schedules and select a task to perform. However, as experiments illustrate [32], classical heuristic cost evaluations, which are more applicable to robots, produce highly acceptable and efficient results.

4.3.2 Interrelated Tasks with Multirobot Requirements

The interrelations among tasks and resource requirements are represented as directed acyclic graphs in each robot's world knowledge. The generated rough schedules respect the precedence and resource constraints. For the makespan objective in general, branch and bound methods can be applied. However, since there are interrelations and resource dependencies, reaching a consensus by communication among robots may sometimes be intractable.

There are two efficient heuristic methods to generate feasible schedules: Forward-Backward Schedule Generation Schemes [27] for the RCPSP and another method which generates a topological sort of the directed acyclic graph fed by different priority rules [30]. There are various priority rules [8] that can be applied for evaluating cost values to select tasks. These heuristics may be used in DEMiR-CF.

4.4 Task Allocation

DEMiR-CF uses the standard auction procedures of CNP [34] to announce the *intentions* of robots on task execution and to select the *reqno* number of robots for a coalition in a cost-profitable, scalable and tractable way. Additionally, *Plan B Precaution Routines* are designed to check validity, consistency and coherence in these negotiation steps. Each robot intending to execute a task announces an auction offer after forming its rough schedule and selecting the most suitable task for itself by DPTSS.

4.4.1 Distributed Task Allocation Scheme

Basically, auction announcements are ways to illustrate intentions to execute tasks for which $reqno = 1$ or to select members of coalitions to execute tasks for which $reqno > 1$. Therefore, if more than one robot declares intentions to execute the same task, the more suitable one is selected in the auction by considering the cost values. Auction negotiations and selection of the suitable robots are performed in a completely distributed manner by the auctioneers. Single task items are auctioned and allocated in auctions. Auction negotiation implemented in the framework consists of the standard steps to clear an auction. Robots can get the necessary task details from the auction offers, and then check the validity of these auctions. If an auction is invalid, related precaution routines (explained in Section 4.6) are activated. Otherwise, the candidate robots send their cost values as bids. The other candidate robots may behave as auctioneers as well. If the auctioneer does not receive the required number (*reqno*) of bids (also counting in its own bid) from the other robots by the predefined deadline, it cancels the auction. Otherwise, it ranks all bids and assigns the best suitable robot with the lowest cost value to the executable task (if $reqno = 1$), or suitable coalition members (if $reqno > 1$). The framework allows multiple auctions to be carried out simultaneously. Validity controls are performed to ensure system consistency as a part of the *Plan B Precautions* on top of the standard CNP protocol procedures. If the auction is for an already achieved task, it becomes invalid and if this situation is detected by any of the candidates, a warning message is sent to the auctioneer.

Interactions among robots is assumed to be implemented by explicit communication. However, if robots are capable of observing each other, this utility can also be

used by DEMiR-CF to improve the performance and the solution quality. Different types of communication are available for robot systems. While ground robots use RF channel, underwater robots may use acoustic communication channel.

An extensive set of communication primitives and different message types are designed for robots to have a common ontology. These messages can be implemented in KQML [16] or FIPA [17] compliant formats. Most of the time, the broadcast type of message propagation is used in DEMiR-CF. P2P messaging for some special negotiations is also used.

4.4.2 Roles

Members of coalitions are selected by auctions. The auctioneers are also active robots in the system. A robot (r_j) may take different roles for task t_i, such as auctioneer, bidder (B_{ij}), coalition leader (CL_i) or coalition member (CM_i) during runtime.

- An Auctioneer is responsible for managing auction negotiation steps and selecting $reqno_i$ of suitable members of a coalition.
- A Bidder is a candidate to become a member of a coalition to execute a task.
- A Coalition Leader is the robot responsible for managing 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 multiple roles as an auctioneer, a CM_i or a 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, selected by the auctioneer as the one with the minimum cost for executing the task, manages 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. It is assumed that robots are not able to infer the state of others by observation, although such capabilities would only provide more reliability (e.g., [4]).

4.5 Coalition Maintenance/Dynamic Task Switching Scheme

The framework suggest an incremental task selection and switching scheme for behaving myopically while thinking globally using bid evaluation heuristics instead of using complicated re-allocation procedures. Provided with an efficient bid evaluation heuristic, the dynamic task selection scheme ensures task switching whenever it is profitable. Each robot, independent of its current status from executing a task or not, can offer a new auction or select to execute a task already being executed by another robot with a worse cost value than it will cost for itself. If task switching occurs with a coalition member, the corresponding coalition member is released from the coalition and becomes a suitable robot for other tasks.

To ensure maintaining the solution quality against environmental changes, a dynamic reconfiguration approach is proposed. Coalition members can leave a coalition when there is a sufficient number of members to execute the task. The

coalition leader is responsible for broadcasting the maximum cost of execution for 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. Getting a released message, a robot can proceed to select another suitable task to execute after then. When the coalition leader considers the size of the current coalition, it also checks the failures. The failed robots are also released from the coalition. If the number of members to execute the task is below the required number for a period of time, the coalition leader cancels the coalition. Such a situation may occur if a robot is not in the communication range at the time of auction announcement. When the situation changes, this robot may take over the role of a member in the coalition.

The decision regarding which robot/agent is a member of which coalition (task execution) is an important issue. Since coalitions are disjoint in the focused case, assigning a robot/agent to a coalition may prevent another advantageous situation in which one of the already assigned robots may take a role in a near future formation. Further negotiations, other than auctions are needed to reach consistent agreements.

One important issue that should be addressed in robot systems is ensuring ways to plan for the global problem from local views. This can be achieved through extensive bid evaluation designs and additional routines to improve solution quality. The statements given in [12] for continual planning are generalized here. Apart from the integration of planning and execution, task allocation/scheduling should also be integrated into a continual planning process. Currently most multiagent/multirobot coordination architectures implement decomposition, allocation and execution steps sequentially, and respond to the contingencies (mostly embedded in the model) in the execution phase. However, this integration is provided in DEMiR-CF by means of the *Plan B Precaution Routines* which are explained in the following section.

4.6 Plan B Precautions: A System-wide Contingency Handling Mechanism

In DEMiR-CF, information is not assumed to be complete. However, the framework can exploit communication when it is reliable. The consistency related with task states is ensured by the precautions taken in a completely distributed manner. *The precaution routines* are embedded into the framework to enable the system to react dynamically to various failure modes and to recover from them. The current implementation uses explicit communication to detect conflicts and contingencies. However, failures in communication can also be handled by the precaution routines.

4.6.1 Representation of The System Model In Each Robot's World Knowledge

Each robot keeps the models of the system tasks and other robots in its knowledge base. Models of different robots may become inconsistent due to uncertainties, incomplete knowledge, assumptions etc. It is not always possible to share a common world knowledge in decentralized systems as in the case of DEMiR-CF.

Models for single and multirobot tasks are maintained as Finite State Machines (FSM) where each task can be in a different state. Multirobot task models are

illustrated in Fig. 2. The only difference between single and multirobot task models is mainly the synchronization issue that robots need to achieve when participating in a coalition [29]. The state transitions are activated by *The Model Update Module* for both types of FSMs. Different task states that tasks can be in are as follows:

- *available*: This is the initial state of the tasks that are neither in execution nor in auction.
- *uncertain* (interpreted as state *available*): This state is activated whenever there is incomplete information regarding a task which was previously being *auctioned* or *executed*.
- *self_auctioned*: A task in this state is under the auction negotiation process managed by the corresponding robot as an auctioneer.
- *others_auctioned*: A task in this state is under the auction negotiation process managed by another robot as an auctioneer.
- *awarded*: A task is in this state if the robot either selects itself or is awarded the task at the end of an auction negotiation process.
- *self_inexec_wait_sync*: A task in this state is being executed by the corresponding robot in a coalition with more than one robot but not synchronized yet.
- *self_inexec*: A task in this state is being executed by the corresponding robot in a coalition and synchronized or the coalition involves only the robot itself.
- *others_inexec*: A task in this state is being executed by the other robots.
- *achieved*: This is the final state of the tasks that are achieved.
- *unachievable*: This state is used for the tasks that are not traversable or achievable.

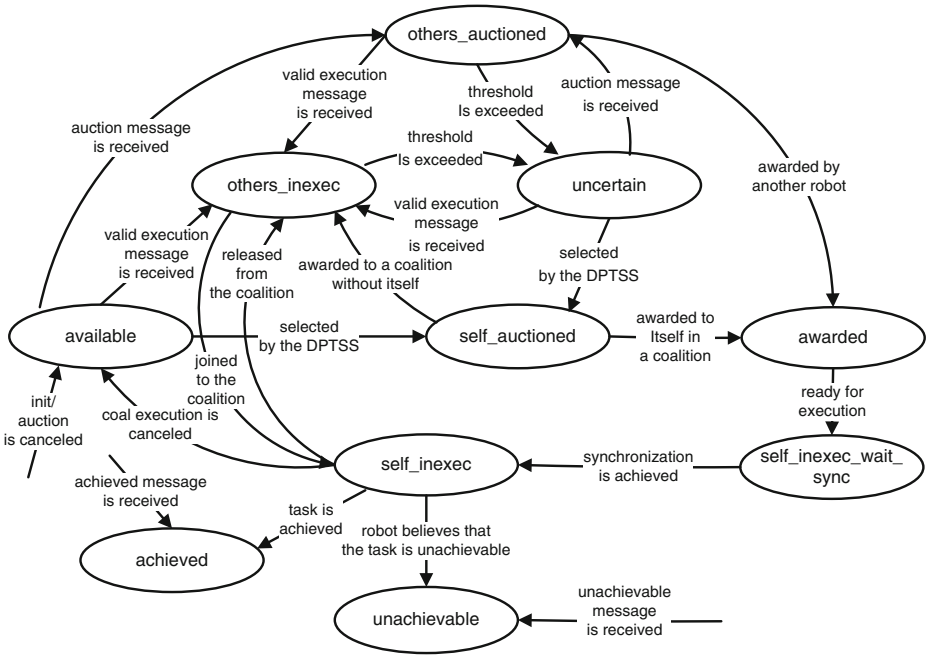


Fig. 2 States of the FSMs for multirobot task models

Robot models are also stored in FSMs, where each robot model has a state which is assumed by the corresponding robot. A robot may be in one of the following states:

- *idle*: A robot is assumed to be in this state when there is no evidence that it is executing any task.
- *executing*: A robot is assumed to be running properly and executing a task whenever there is supporting evidence.
- *failed*: A robot is assumed to have failed when there is no evidence that it has been running properly for a long time.
- *auctioneer*: A robot is assumed to have auctioned for a task when there is recent evidence.

4.6.2 Plan B Precaution Routines

Recovery operations may include warning other robots about the problem or changing the model accordingly. Inconsistencies usually arise in real-world operations when robots are not informed about tasks that are achieved, that are still under execution or under auction. To maintain system consistency, robots use explicit communication and broadcast the following information:

- tasks known to have been achieved in predefined time periods to prevent redundant executions. This feature provides a bucket-brigade type of information sharing which handles communication range limitations.
- newly discovered online tasks which are not yet achieved.
- task execution message containing an updated cost value and estimated task achievement deadline information acting as clues for the executer robot to be still alive and the task being under execution.
- task achievement message when a task is over.
- cancellation message if task execution is canceled.
- task invalidation message when an invalid situation is detected.

Designed precautions are given in Table 1. Most of the contingencies are detected by checking models, and corresponding model updates are implemented (Tables 2

Table 1 Precautions for contingencies and conflicts

Contingency or conflict by inconsistencies	Precaution
Any message from an unrecognized system robot is received.	Robot model is created with the corresponding state derived from the message.
Any message related to an unrecognized task is received.	Task is added to the task list with the corresponding state.
An already achieved task is announced as a new task being executed/canceled/auctioned.	Warning message is sent to the sender.
A task being executed/auctioned is announced as being executed/auctioned.	Only the robot with the minimum cost continues to the operation.
Cancellation message is received for a task already being executed by the receiver.	Robot state is set *idle*.
A cancellation message is received for a task being executed by the sender robot.	Task and robot states are set as *available* and *idle*, respectively.

Table 2 Model checking for tasks and system robots

Status	Action
The time duration from the latest communication with a robot is longer than the threshold.	Robot state is set as *failed*. Related task state is set as *uncertain*.
Task in execution is not achieved although the estimated deadline is reached.	Task state is set as *uncertain*.
Task state is *others_auctioned* for longer than predefined time period.	The task state is set as *uncertain*.

and 3). If robots can observe each other implicitly, model updates can be implemented in a similar manner.

One standard way of detection of robot failures is sending heart-beat signals. However in DEMiR-CF, incoming messages from other robots are taken as clues for their states. More complicated prediction models may be used for more accurate failure prediction. Some misleading beliefs such as setting the state of a robot as *failed* although it is running properly may cause parallel executions. This is a desired feature for the mission completion point of view. Designed precautions resolve these kinds of inconsistencies if communication resources permit in later steps. In the design of precautions, it is assumed that robots are trusted and benevolent.

5 Case Studies

DEMiR-CF has been evaluated for different domains in both simulation and real environments with robots. WEBOTS [37] simulator, and US NAVY's realistic ALWSE-MC (Autonomous Littoral Warfare Systems Evaluator-Monte Carlo) [3] simulator are used as simulation environments and Khepera II robots are used as physical real hardware to carry out experiments.

5.1 DEMiR-CF on the MTRP Domain

The Multiple Traveling Robot Problem (MTRP) [32], the real-world version of the well known NP-hard Multiple Traveling Salesman Problem (MTSP), asks for finding routes of a team of robots to visit a set of targets while optimizing an objective.

Table 3 Model updates related to the messages

Message type	Action
Any type	Current time is registered as the latest comm. time with the robot and for the task.
"achieved"—valid	The robot and task states are set as *idle* and *achieved*, respectively. If the task is in consideration (in schedule or in execution), it is canceled.
"execution"—valid	If there are other tasks with state *self_inexec*, these states are changed as *uncertain*.

Examples of such objectives are total path length minimization, time minimization, average energy minimization, makespan minimization, etc. The overall solution quality is dependent on both the quality of the solution constructed by the paths of robots and the efficient allocation of the targets to robots. MTRP forms an important basis for several domains such as Search and Rescue (SR), Space Exploration, Object Construction, Pick-up/Delivery, etc. All these domains contain MTRP ingredients in their problem representations even though their implementations and the complementary objectives may be different.

Although a single type of task is considered in MTRP, it is NP-Hard due to the combinatorial structure of the problem. The problem area is well studied in the field of OR and optimal solutions can be obtained by Integer Programming (IP) formulations [32]. However, these approaches may become impractical when the size of the mission is even moderate or the cost values change frequently due to uncertain knowledge, unpredictable changes in the environment (including failures) or in the structure of the mission (e.g., online generated tasks). Furthermore, robots have continuous path planning burdens for target sets in dynamic environments. Expensive computational efforts for initial allocations may become redundant as conditions change.

5.1.1 Rough Schedule Formation for MTRP

The integrated components that make up the DEMiR-CF framework are successfully implemented for the MTRP domain. During mission execution, each robot (r_j) initially generates its rough schedule, and then, selects the most suitable target for itself among the targets in its rough schedule (T_{Rj}). T_{Rj} is constructed as a set of targets that are closest to the robot r_j, among unvisited targets (T_U) using an efficient heuristic cost evaluation [32]. Targets in T_{Rj} are considered as the candidate targets for robot r_j. Next, each robot proceeds to announce its intention to execute the selected task (to visit a target). Therefore, before selecting the most suitable target, each robot constructs these rough route sets. This heuristic does not compel an actual commitment, and the targets in rough schedules are not necessarily assigned to the corresponding robots in future auctions. Instead, the approach ensures a global view to the problem from a local perspective.

5.1.2 Results for MTRP

An extensive analysis of DEMiR-CF on the MTRP domain is given in an earlier work [32]. In this earlier work, different heuristic cost functions integrated within the framework are evaluated and the results are compared with optima that are generated by using an Integer Programming formulation running on a commercial IP solver, CPLEX. Task allocation approach of DEMiR-CF is also analyzed and the results are given with comparisons to the other heuristic approaches. It has been observed that DEMiR-CF integrated with the designed heuristic cost functions produces results that are close to optima.

Overall results from this domain [32] present the efficiency of integration of target allocation and route construction by DEMiR-CF. This integration and incremental allocation is useful for eliminating redundant evaluations in highly dynamic

or unknown environments. The rough schedule generation scheme forms loose commitments, which if needed, can be canceled in the future. Thus, it offers a way to reconsider the problem globally when it is appropriate. The approach is efficient with its polynomial complexity. *Plan B precautions* ensure that the $CMAP$ is successfully solved. If real resources permit, failures are handled to maintain system consistency. Even though a certain amount of additional communication overhead is injected into the system by the *Plan B Precaution Routines*, communication efficiency is also achieved. Empirical evaluations of the system are performed on real robots as well. The experimental results reveal the success of the approach as a whole with its integrated components and its applicability on even very simple and small robots like Khepera II [32]. Videos of the real robot scenarios in which different allocation strategies and failure recovery procedures are applied are available online at [11].

5.2 DEMiR-CF on NAVY Missions

DEMiR-CF is also evaluated in NAVY domains where the cooperation of underwater vehicles is required for homeland security missions, such as the Mine Countermeasure Mission (MCM) [31]. Naval Mine CounterMeasures are actions taken to counter the effectiveness of underwater mines. In general, recognizing proud mines on the seafloor is not overly difficult; the difficulty arises with the abundance of non-mine objects on the sea floor that possess mine-like characteristics (e.g., geologic outcroppings, coral, man-made debris, etc.). This ample supply of false alarms has necessitated the following strategy typically employed by the Navy: detect and classify the mine-like objects (MLOs) with high-coverage rate sensors (e.g., sidelooking sonar), employ advanced signal processing techniques for maximal false alarm reduction, then revisit the remaining MLOs with identification-quality assets (e.g., electro-optic sensors) to confirm them as mines or dismiss them as false alarms. The reference mission in this research is to detect, classify, and identify underwater mines in a given operational area simulated in a PC-based software, ALWSE-MC (Autonomous Littoral Warfare Systems Evaluator-Monte Carlo), analysis package designed to simulate multiple autonomous vehicles performing missions in littoral regions including mine reconnaissance, mapping, surveillance, and clearance. This mission employs two types of vehicles: unmanned underwater vehicles (UUVs) which are free swimming AUVs and possess large-footprint sensors (e.g., side-scan sonar) for detection and classification (D/C) of mines and sea-floor crawlers equipped with short-range, identification-quality sensors (e.g., camera). The crawlers have the ability to stop at an object and take a picture with a camera.

This domain has a challenging structure since it is performed underwater and communication is achieved through acoustic modems. In this domain, the robot team is modeled as a heterogeneous team, and the reference mission consists of two different types of tasks each of which can be executed by a different type of vehicle. The domain is modeled to include both the coverage problem and the MTRP. Two types of tasks are defined for vehicles: “visit waypoint” (w) and “identify MLO” (t). The task representation includes the capabilities required for each type

of task: $reqcap_w$ contains side-scan sonar and $reqcap_t$ contains cameras besides the standard capabilities of AUVs common in both types of vehicles. The Naval Mine countermeasure domain has appropriate characteristics to deploy teams of robots and let them cooperate to achieve the overall mission. It should be noted that, the objectives and the limitations of this domain are similar to those of both Search and Rescue and Space Exploration domains.

5.2.1 Rough Schedule Formation for NAVY Missions

The coverage mission (M_C) contains predefined number of waypoints ($w_i \in M_C, 0 < i \leq ||M_C||$) to be visited by all UUVs ($R_{UUV} \subset R$). One way of task representation is to directly assign a task for each waypoint. However, this representation has a drawback of high communication requirements for efficient completion of the mission. Instead, tasks are represented as interest points of regions/search areas ($W_k = \cup w_i$, $\forall w_i$ is unvisited, and $W_k \subseteq M_C$). These regions, represented as rough schedules, are determined by the robots during runtime dynamically although the waypoint locations are fixed at known coordinates. Therefore, both the allocation of the waypoints to the robots and the paths constructed to traverse these waypoints are determined online by negotiations. Negotiating the interest points of the corresponding regions instead of the individual waypoints reduces the communication overhead. Regions determined by different UUVs may vary during runtime, and may sometimes overlap. However, the uncertainty related to the region determination is within an acceptable range, especially when the cost is compared to the requirements of complete knowledge sharing by representing each waypoint as a task. Before defining the regions, the relative distance values, are determined for each unvisited waypoint. The coverage area assignments are performed by having robots generate rough schedules for the regions to be covered and negotiate over these regions.

While covering the environment, robots simultaneously sense "mine like objects" to generate new online tasks. They transmit this information to different types of robots that can visit targets and perform correct classification. With this mode of operation, the problem turns into MTRP and is solved in a decentralized way for continually appearing online tasks. Therefore, depending on the types of robots, different types of rough schedules are generated for coverage and classification missions. Regions are represented as rough schedules for the coverage mission, whereas target sets are formed as rough schedules for the classification mission.

5.2.2 Results for NAVY Missions

The online task handling performance of DEMiR-CF on MCM domain is validated through experiments. The robustness of the framework against both communication and robot failures and the efficiency with which it responds to dynamic changes in the environment are tested in the ALWSE simulator. The related results are given in [31]. A sample illustration of region allocations are given in Fig. 3. In this figure, three robots determine their regions (Search Areas) as their rough schedules and visit waypoints in those regions (Fig. 3a). Different markers in the figure correspond to the waypoints of a different robot's region. The robots patrol their corresponding

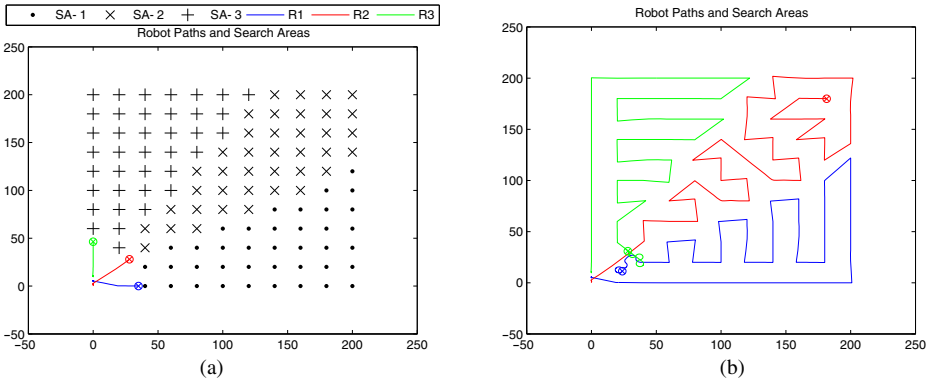
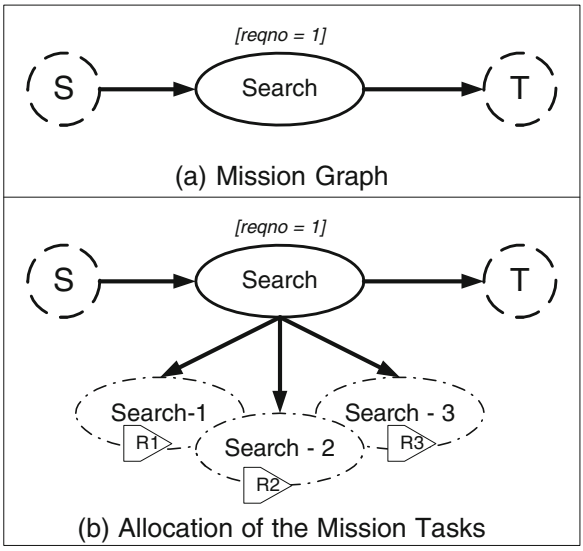


Fig. 3 Robots patrol the area by dividing the overall coverage area into three regions to be searched

regions which are determined after the negotiations (Fig. 3b). Therefore, although there is only a single task on the higher level, the robots create instances of the Search Task (Search 1–3) as if each instance is another separate task (Fig. 4). If there are no other tasks to perform, the robots only search the area.

This domain has been extended by introducing online tasks whose arrival times are not known in advance in a naval homeland security mission. The results of this experiment are reported here. In this case, the overall mission is searching a predefined area in order to protect the deployment ship from any hostile intents. The initial mission graph for the extended MCM mission is given in Fig. 4. The scenario in the ALWSE simulator is illustrated in Fig. 5. Initially, the mission consists of only the Search Task. Although $reqno = 1$ for this task, since there are no other tasks and the robots have enough fuel capacities, all robots involve in the execution of the search task as a coalition and divide the area to be searched. The robots begin searching the

Fig. 4 Initial mission graph consists of only the search task. The three robots are assigned to the search task to explore the whole environment



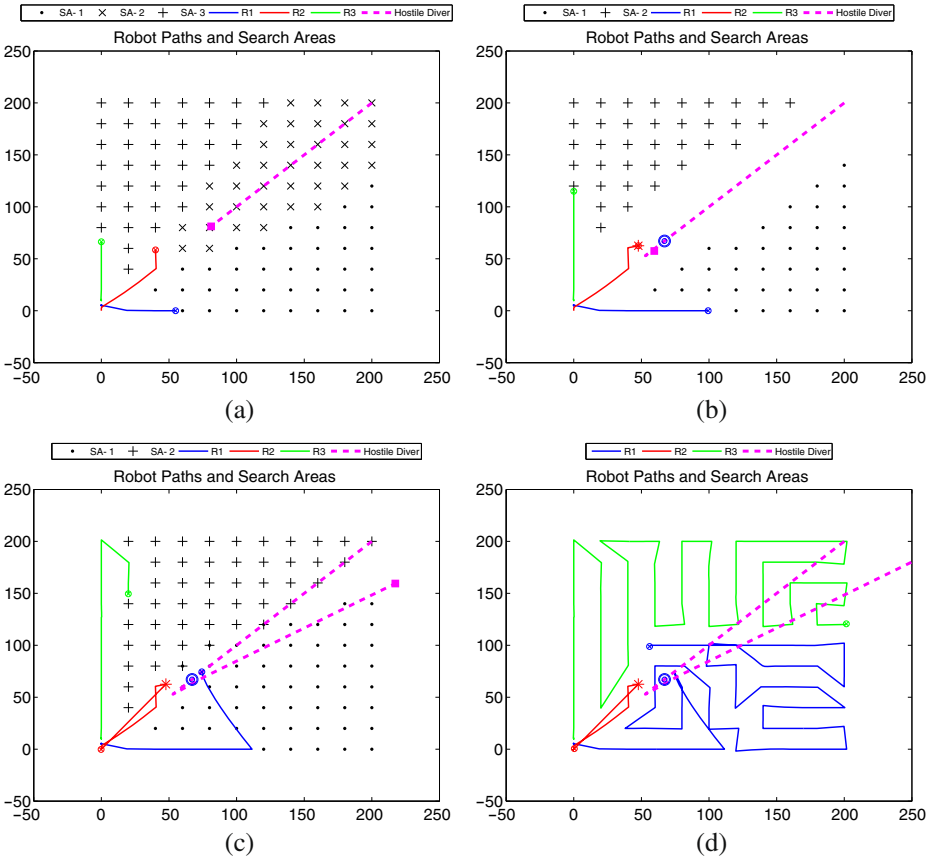


Fig. 5 A sample execution trace under highly dynamic task situations involving failures after the shots by the hostile diver. **a** R2 recognizes the hostile intent and **b** activates the intercept task for itself. **c** R2 is attacked by the hostile vehicle and returns to the deployment ship. R1 takes control of the intercept task. **d** After the hostile diver disappears, R1 can continue executing the search task. Note that, the regions (rough schedules) are dynamically updated

area (Fig. 5a). Whenever R2 recognizes the hostile intent (Fig. 5b), it adds the related interception task in its rough schedule and also sends the related task information to the other robots while activating and selecting this task. At this time, although interception task is in the rough schedules of R1 and R3, they continue their search task by diving the area into two subregions since the relevant reaction is provided by R2. Whenever, R2 is attacked by the hostile vehicle, it informs the others about the cancellation of the intercept task execution while returning to the deployment ship for recovery. R1 takes control of the intercept task. R3 continues to the execution of the search task. The hostile intention disappears (Fig. 5c). R1 and R3 continue their search task execution by diving the area into two regions (Fig. 5d). Note that, rough schedules and accordingly, the regions are changed during runtime [31]. The updated mission graph for this scenario is illustrated in Fig. 6.

Since the hostile diver may be destructive by using missiles, executions may need to be preempted and the task execution authority is exchanged during run time. The

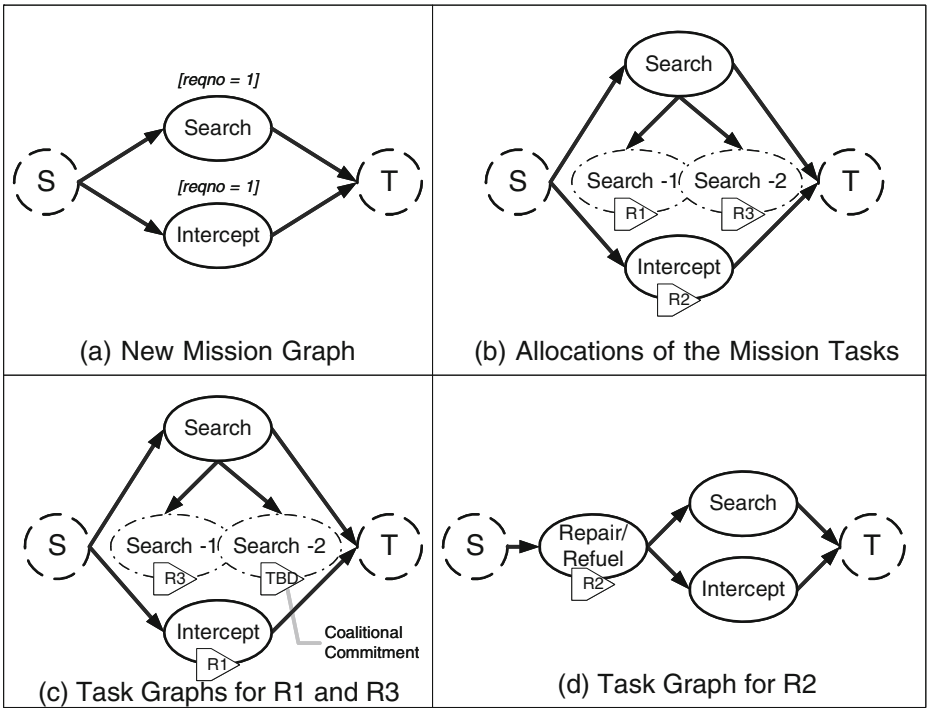


Fig. 6 The mission graph and the allocations evolve through time accordingly

robots may need to generate local tasks (e.g., Repair/Refuel Task, which is generated by R2 after being shot by the hostile diver unexpectedly) as in Fig. 6d. Therefore, the rough schedules are different for the robots even when they work cooperatively (Fig. 6c and d). In Fig. 6c, although executing the Intercept Task, R1 can make a coalition commitment assuming it will succeed in a predefined time period (described as TBD). At that time, R2 cannot make any coalition commitment for the search task because its future operations depend on its recovery.

Cost evaluations for the tasks are computed by considering the task facilitating composite (multi) objective missions. While the robots try to optimize the fuel levels for the search task, the intercept task requires immediate response and time minimization. Therefore, different cost evaluations are carried out for different tasks. The MCM mission cost evaluation is adopted for the search task. For the intercept task, the expected time to achieve (intercept the diver) the task is taken as the cost value.

Auction announcements are used both to maintain the models of the other robots in the system and to announce clues for the intentions. Emergency tasks (e.g., Intercept Task) require immediate action. Standard auction steps for these types of tasks are not suggested. Instead, either a one-step auction is performed or the task is directly executed, which is the approach adopted in the experiments. However, in this case, parallel executions may occur and should be resolved. This facility is provided in the framework by precaution routines. Although parallel executions are allowed to handle the emergencies, these inefficient allocations are resolved when

recognized. In a sample scenario for a limited communication range, the parallel executions arise for the emergency tasks such as the intercept task as in Fig. 7. However, these inconsistencies are resolved by the activation of the corresponding precaution routines whenever the robots enter into the communication range. In this scenario, R3 switches to the search task after detecting the parallel execution. R1 continues to execute the intercept task. The intercept task is assumed to be achieved whenever the hostile threat is believed to have disappeared completely.

5.3 DEMiR-CF on Complex Missions

As a final testbed, object construction domain is selected to use and validate the full functionality of DEMiR-CF. Complex missions investigated in this domain involve tasks with resource constraints and interrelations. Multirobot task allocation problem is better viewed as a complex scheduling problem if there are interrelations among tasks. Therefore, this domain forms the basis of the design objective of DEMiR-CF and explained in detail here.

Task interrelations may correspond to shared resources, producer/consumer, simultaneity and task-subtask dependencies [23]. The Pick-Up/Delivery domain tasks can be classified in this class because of the producer/consumer type of dependency relation for the pick-up and delivery tasks. More complicated interrelations may be placed in mission representations. Simultaneous execution requirements imply tightly coupled task execution where the success of the actions taken by each robot are highly dependent on the actions of others.

5.3.1 Rough Schedule Formation for Complex Missions

The dynamic and incremental task selection, distributed allocation and contingency handling mechanisms of DEMiR-CF are used in the design of the solution for complex mission domains. Although the base mechanisms are the same, the rough schedule generation scheme and the cost functions are designed accordingly to meet

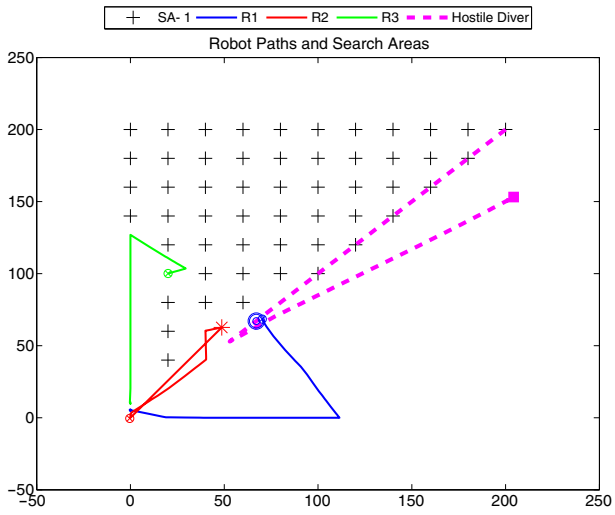


Fig. 7 Under limited communication ranges, parallel executions may occur and are resolved when detected

the ordering and resource constraints. As the core principle of DEMiR-CF, the robots make instantaneous decisions (from their local perspectives) which are both precedence and resource feasible in the context of the global-time extended view of the problem. While the completion of the mission is the highest priority objective, performance related objectives can additionally be targeted. Each robot initially forms a rough schedule of its activities for an overall time extended resolution of the mission. Since these schedules are highly probable to change in dynamic environments and the robots also have the real-time burdens of path planning, mapping etc., the rough schedules formed are tentative and constructed by computationally cheap methods (explained in Section 4.3). Therefore, the robots using the framework come up with their rough schedules and refine their plans during actual fast execution when information available in the current context enables them to make specific and detailed decisions.

Instead of scheduling all tasks in one step, as the DPTSS algorithm (Algorithm 2) implies, tasks are allocated to robots incrementally, considering the global solution quality. The main objective of the proposed scheme is the incremental allocation of tasks by taking into account the precedence and resource constraints whenever a new task is activated, instead of scheduling all tasks from scratch.

The critical tasks may be determined by either negotiations or beliefs. To eliminate intractable communication overhead, a rough belief update approach is used to form the critical tasks as in Algorithm 3. Each critical task is assigned a probability value to indicate its criticality. Critical task information is used for determining the task requirements such as power, fuel etc. The generated list of critical tasks is used to determine whether the given mission is achievable with the current resources from the perspective of a single robot. This issue is analyzed in the DPTSS algorithm.

Algorithm 3 GenerateListOfCriticalTasks for robot r_j

input: The eligible task set T_{Ej}
output: The list of critical tasks (L_{Cj})

$L_{Cj} = \phi$
for each $t_i \in T_{Ej}$ **do**
  Determine the number of suitable robots
  if $numofsuitablerobots = 0$ **then**
    $P_{ct}(t_i) = 1$
  else
    $P_{ct}(t_i) = \frac{reqno_i}{numofsuitablerobots}$
  end if
  if $P_{ct}(t_i) \geq 0.5$ **then**
    Insert t_i in L_{Cj} prioritized by the $P_{ct}(t_i)$
  end if
end for

The rough schedule of a robot constitutes a topological order of the directed acyclic graph of the eligible mission tasks. While generating the rough schedules,

both precedence constraints and cost values are considered. Basically each rough schedule is a priority list (T_o, topological order) determined by Algorithm 4. While forming the topologically ordered prioritized schedule list, a depth first search (DFS) is performed to topologically order the tasks by using the estimated task completion times. Next, the tasks are inserted into the list according to their completion times. If a task is an active task, its priority key is computed as a combination of the precedence and the cost value. Tasks with equal precedence are ordered according to their cost values. The rough schedule of a robot is generated by considering the constructed priority list as presented in Algorithm 1 (Section 4).

Algorithm 4 GeneratePriorityList for robot r_j

input: Eligible task set (T_{Ej}), active task set (T_{Aj})
output: Topologically ordered and prioritized schedule list: S_{Rj}

$S_{Rj} = \phi$, $S_{Temp} = \phi$
$S_{Temp} = DFS(T_{E_j})$ /*List generated by a depth-first search, the tasks are ordered by ascending order of their estimated task completion times*/
for all $t_i \in S_{Temp}$ **do**
  if $t_i \in T_{Aj}$ **then**
    insert t_i in S_{Rj} as ordered by the cost value and the precedence
  else
    insert t_i to the front of S_{Rj}
  end if
end for

The cost evaluation has a tremendous impact on the solution quality. Each task type requires a different cost evaluation to efficiently solve the problem. As an example, while estimating the cost value for picking up an item, the distance between the robot location and the estimated location of the item may be considered. However, to form a globally efficient allocation, the locations of the other items must be considered as well. This is validated in the MTRP domain.

To decide on the selection of the robots to execute specific tasks in a distributed setting is a challenging issue. Tasks are allocated by single-item auction negotiations as explained in Section 4.4. A common situation appears when different auctions are offered at the same time either for the same task or for different tasks. If there are conflicting auctions for the same task, only the robot offering an auction with the smallest cost value continues with the auction negotiation process. Furthermore, robots may need to cancel or postpone their offers for auctions if there is synergy between tasks when announcements of offers from others are received. In the case of the conflicting auctions for different tasks, a resource-based rule (related to the *reqno* of the tasks) borrowed from OR, Greatest Resource Requirements (GRR), is used [8]. Another important design criteria is determining the bid values to be sent to the other robots. Intuitively, robots only send bid values for the active and critical tasks for themselves. If a robot receives an auction message when executing a task, it sends the bid value by considering the final destination of the current task as the location of itself.

5.3.2 Results for the Complex Domain

The first set of experiments is targeted to analyze the scalability of the proposed approach on the pick-up/delivery mission in which the tasks are interrelated by picking up and delivery constraints [30]. All picked up items are collected in the center of the environment. In the fully flexible version, while there are precedence constraints between pick-up and delivery tasks, there are no interrelations between pick-up tasks for different items. The items are distributed in the environment at fixed locations for each run. The robot locations are randomly determined. Figure 8 illustrates the mission completion times and the total path length traversed by the robots for sets with different number of robots. As expected, the time to complete the overall mission is greatly reduced with the increasing number of robots, validating the scalability of the approach. Since the items are delivered to the center of the environment, an extreme variation for the expected utility in the total path length traversed by robots is not expected, as illustrated in the graph.

Another complex mission scenario for a complex mission which includes tasks for moving boxes and picking-up and delivering items to a desired location is given in Fig. 9 with five participating robots in WEBOTS. Initial locations of the objects are indicated with the red arrows in the figure. In the first scenario, two items are picked up and delivered to the destinations by the robots possessing grippers. The two robots simultaneously and independently push the two boxes. One of the robots stays idle during mission execution. In the second scenario, since a two-robot resource requirement exits to push one of the boxes, the two robots without grippers form a coalition and push the heavy box synchronously.

The real robot implementation of DEMiR-CF is illustrated by a sample mission which includes interrelated tasks for pushing a box, carrying a cylindrical object to a final destination and then inspecting the environment. The mission DAG can be given indicating the interrelations between push, carry and inspection tasks respectively as in Fig. 10. This mission can be achieved by a heterogeneous robot team with Khepera II robots having different equipments. While the objects can only be carried by the robots with grippers, the inspection task requires possessing a camera. The box can be pushed by any of the three robots. However, due to the

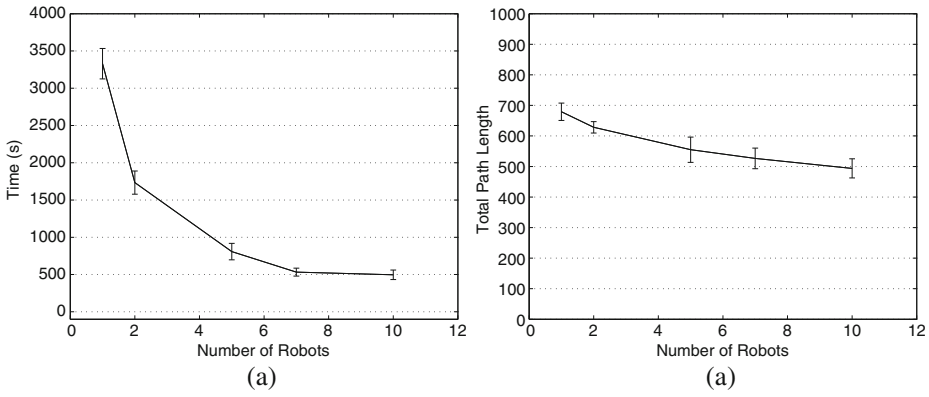


Fig. 8 **a** Mission completion time (s) and **b** total path length traversed by the robots (mm) for the pick-up/delivery mission, with task number fixed at 20

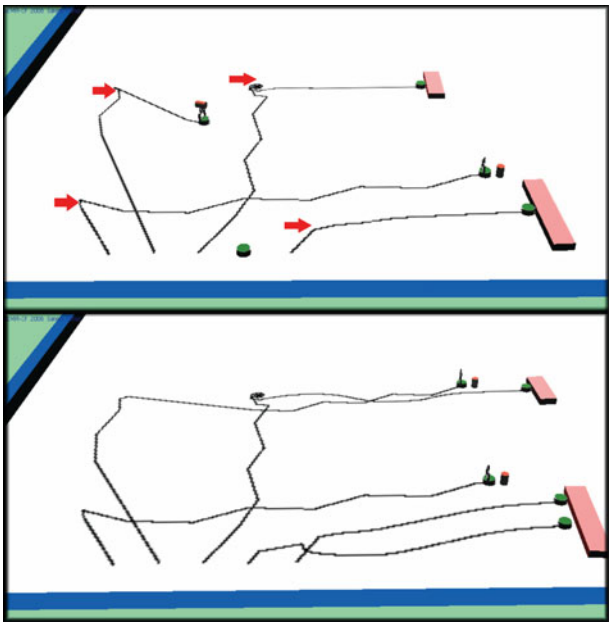


Fig. 9 Scenarios 1 and 2: Robots push and carry boxes to a final destination. *Top* Each task can be executed by a single robot. *Bottom* The larger box requires the two robots simultaneously push the box

cost evaluations and the critical task list consideration, allocations are implemented accordingly. Robots obey the interrelation constraints and each robot takes part in a suitable task execution for itself in which the decision is made in a completely distributed manner. The execution scenario is illustrated in Fig. 11. The video of this scenario is available online at [11].

6 Computational Analysis of the Approach

DEMiR-CF for MTRP offers a polynomial time solution. Calculating cost values for all n tasks takes $O(n\log(n))$. Cost and queue initialization is implemented each in $O(n)$. Top element selection and deletion is performed in $O(\log(n))$. Therefore, the total complexity is bounded by $O(n\log(n))$. In the worst case, where the

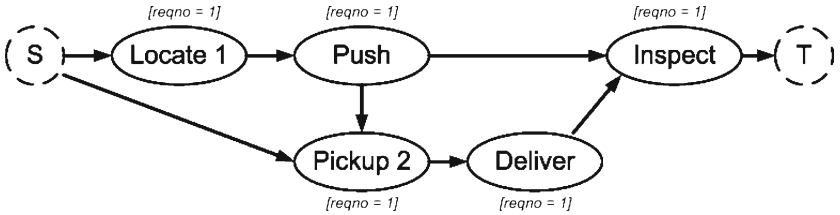


Fig. 10 The graph of the real mission scenario. The ordering constraints between tasks are indicated by the directed edges and *reqno* values are attached to the nodes

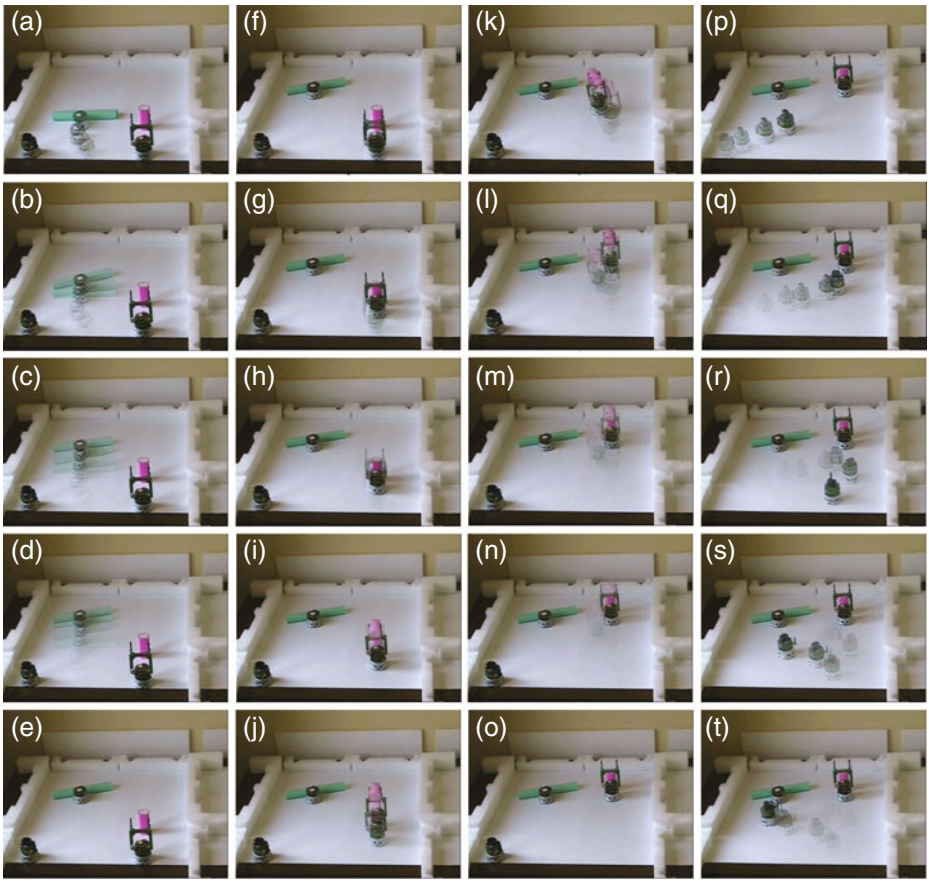


Fig. 11 Khepera II robots achieve the overall complex mission of pushing/carrying the objects to the final destinations and inspecting the area

environment is dynamic and cost values change frequently in the order of $O(l)$, the total complexity becomes $O(ln\log(n))$ for each robot.

For the complex mission domains, the critical task list generation takes $O(n\log(n))$ for all n number of tasks. Achievability of the mission is determined in $O(n)$. The complexity of the rough schedule generation is bounded by the topological list generation algorithm which is in the order of $O(n+e)$ (where e is the number of conjunctive arcs, i.e., hard dependencies). Therefore, the total complexity becomes $O(n(e+n\log(n)))$. If $(e << n)$, the complexity of the proposed approach reduces to $O(ln^2\log(n))$.

7 Discussion and Conclusion

DEMiR-CF is a multirobot cooperation framework to solve the Cooperative Mission Achievement Problem ($CMAP$). $CMAP$ asks for achieving a complex mission of either independent or interrelated tasks with multirobot requirements for their

execution. The problem should be solved in a cost efficient manner while simultaneously handling unplanned events and contingencies. The Coordinated Task Selection Problem ($CTSP$) is a task selection problem for a robot participating in a multirobot team running to solve $CMAP$. Each robot involved in the $CMAP$ proceeds by generating incremental solutions to $CTSP$ at runtime until the mission is completed.

DEMiR-CF is designed for complex missions including interrelated tasks that require diverse (heterogeneous) capabilities and simultaneous execution. The framework combines *Dynamic Priority-Based Task Selection Scheme*, *Distributed Task Allocation* and *Coalition Maintenance Schemes* as cooperation components and *Plan B Precaution Routines*. These components are integrated into a single framework to provide an overall system that finds efficient solutions for real-time task execution. DEMiR-CF eliminates redundant scheduling efforts by means of the incremental assignments based on the current state of the environment. It can also handle contingencies by the precaution routines embedded in its integrated structure. Communication failures may sometimes prevent allocations from being optimal. DEMiR-CF can also detect these situations and maintains high solution quality through a dynamic task selection/exchange scheme.

The Dynamic Priority-Based Task Selection Scheme forms the basis of incremental task selection for each robot. Each robot maintains a rough schedule of future tasks before deciding on an active task to execute. Rough schedules ensure ways to consider the problem as a whole although the decisions are made locally by each robot. A topological list generation approach for forming rough schedules is used to solve $CTSP$ and to resolve the constraints on tasks. *The Distributed Task Allocation Scheme* ensures the selection of appropriate robots for corresponding tasks in a decentralized way. Applicable cost functions are proposed for robots to achieve an overall complex mission of interrelated tasks. *Coalition Maintenance and The Dynamic Task Switching Scheme* ensures dynamic reconfiguration of allocations. *Plan B Precaution Routines* ensure system consistency, coherence and robustness in a decentralized way. Robots keep the models of the system tasks and the other robots to maintain an up-to-date representation of the current status of the execution environment. *Plan B Precaution Routines* include both recovery routines that failing robots can execute and warning mechanisms that aim to correct behaviors of other robots in the system.

DEMiR-CF differs from the earlier work in terms of its instantaneous assignment procedure by forming rough schedules as computationally efficient time-extended resolution of a complex mission. Single-item auctions are used by robots to announce intentions about task execution and to select the appropriate task executers. Contingency handling mechanisms are directly integrated into the dynamic task selection mechanisms, which in turn facilitate recovering from failures dynamically and efficiently, reconfiguring robots during runtime, and maintaining system consistency. These utilities are ensured by autonomous robots implementing DEMiR-CF in a completely distributed manner without central authorities and/or complete knowledge injected manually.

DEMiR-CF has been evaluated in different domains in both simulation and real environments with robots. The results of the experiments support the motivating claim that in real-world execution environments, an incremental task selection approach eliminates redundant efforts that are introduced by allocating all tasks from scratch on an unexpected change. Rough schedule generation scheme that

forms loose commitments which if needed, can be canceled in the future, offers a way to reconsider the problem globally when it is appropriate. *Plan B precautions* ensure that the $CMAP$ is successfully solved at the end of mission execution. If real resources permit, failures are handled to maintain system consistency. Even though a certain amount of additional communication overhead is injected into the system by the *Plan B precaution routines*, communication efficiency is also achieved. The scalability of DEMiR-CF is validated through experiments. Efficiency is analyzed and validated both in theory and practice. As results illustrate, $CTSP$ is solved by each robot implementing the incremental task selection, distributed task allocation and contingency handling mechanisms of DEMiR-CF to achieve the overall $CMAP$ for complex missions.

In conclusion, DEMiR-CF has been designed and implemented as a generalized framework for cooperative multirobot mission achievement. Several performance tests are carried out for different domain implementations of the framework. It has been demonstrated that DEMiR-CF is an efficient, scalable, robust and complete framework for a multirobot system. Furthermore, it is also shown to be applicable on even very small and simple robots with limited computational capabilities, such as Khepera II.

References

1. Alami, R., Botelho, S.C.: Plan-based multi-robot cooperation. In: Advances in Plan-Based Control of Robotic Agents (2001)
2. Alami, R., Ingrand, F., Qutub, S.: A scheme for coordinating multi-robot planning activities and plans execution. In: Thirteenth European Conference On Artificial Intellingence (1998)
3. ALWSE-MC: http://nswcpc.navsea.navy.mil/analysis/capabilities.asp (2009)
4. Balch, T., Arkin, R.C.: Communication in reactive multiagent systems. Auton. Robots **1**(1), 1–25 (1994)
5. Botelho, S., Alami, R.: M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (1999)
6. Brucker, P.: Scheduling Algorithms. Springer, New York (2001)
7. Brucker, P.: Scheduling and constraint propagation. Discrete Appl. Math. **123**, 227–256 (2002)
8. Brucker, P., Knust, S.: Complex Scheduling. Springer, New York (2006)
9. CPLEX Manual: ILOG-CPLEX-9.0-UserMan. http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsp/index.html (2009)
10. Dahl, T.S., Mataric, M.J., Sukhatme, G.S.: Emergent robot differentiation for distributed multi-robot task allocation. In: Distributed Autonomous Robotic Systems (DARS) (2004)
11. DEMiR-CF Scenarios: Khepera II DEMiR-CF Videos. http://web.itu.edu.tr/sariel/videos/KheperaII-Movies.html (2009)
12. desJardins, M., Durfee, E., Ortiz, C., Wolverton, M.J.: Survey of research in distributed, continual planning. AI Mag. **20**(4), 13–22 (1999)
13. Dias, M.: Traderbots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments. Phd thesis, Robotics Institute, Carnegie Mellon University (2004)
14. Dias, M., Zinck, M., Zlot, R.M., Stentz, A.: Robust multirobot coordination in dynamic environments. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (2004)
15. Dias, M.B., Zlot, R.M., Kalra, N., Stentz, A.: Market-based multirobot coordination: a survey and analysis. Proc. IEEE **94**(7), 1257–1270 (2006)
16. Finin, T., Labrou, Y., Mayfield, J.: KQML as an Agent Communication Language. MIT Press, Cambridge. Chap Software Agents (1997)
17. FIPA ACL: FIPA ACL Message Structure Specification. http://www.fipa.org/specs/fipa00061/SC00061G.pdf (2002)

18. Gancet, J., Hattanberger, G., Alami, R., Lacroix, S.: Task planning and control for a multi-uav system: Architecture and algorithms. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (2005)
19. Gerkey, B., Mataric, M.J.: Sold!: Auction methods for multirobot coordination. IEEE Trans. Robot. Autom. **18**(5), 758–768 (2002)
20. Gerkey, B., Mataric, M.J.: A formal analysis and taxonomy of task allocation. Int. J. Rob. Res. **23**(9), 939–954 (2004)
21. Horling, B., Lesser, V.: A survey of multi-agent organizational paradigms. Knowl. Eng. Rev. **19**(4), 281–316 (2005)
22. Lemaire, T., Alami, R., Lacroix, S.: A distributed task allocation scheme in multi-uav context. In: IEEE Intl. Conf. on Robotics and Automation (ICRA) (2004)
23. Ossowski, S.: Co-ordination in Artificial Agent Societies, Social Structure and Its Implications for Autonomous Problem-Solving Agents. Springer, New York (1999)
24. Paquet, S.: Distributed Decision-Making and Task Coordination in Dynamic, Uncertain and Real-Time Multiagent Environments. Phd thesis, Laval University, Quebec (2006)
25. Parker, L.E.: Alliance: An architecture for fault tolerant multi-robot cooperation. IEEE Trans. Robot. Autom. **14**(2), 220–240 (1998)
26. Parker, L.E., Tang, F.: Building multi-robot coalitions through automated task solution synthesis. Proc. IEEE (Special Issue on Multi-Robot Systems) **94**(7), 1289–1305 (2006)
27. Pinedo, M.L.: Planning and Scheduling in Manufacturing and Services. Springer, New York (2005)
28. Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer, New York (1994)
29. Sariel, S.: An Integrated Planning, Scheduling and Execution Framework for Multi-Robot Cooperation and Coordination. Phd thesis, Istanbul Technical University, Turkey (2007)
30. Sariel, S., Balch, T., Erdogan, N.: Incremental multi-robot task selection for resource constrained and interrelated tasks. In: IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (2007)
31. Sariel, S., Balch, T., Erdogan, N.: Naval mine countermeasure missions: a distributed, incremental multirobot task selection scheme. IEEE Robot. Autom. Mag. **15**(1), 45–52 (2008)
32. Sariel, S., Balch, T., Erdogan, N.: Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. IEEE/ASME Trans. Mechatron. **14**(2), 198–206 (2009)
33. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**, 165–200 (1998)
34. Smith, R.G.: The contract net protocol: high level communication and control in a distributed problem solver. IEEE Trans. Comput. **29**(12), 1104–1113 (1980)
35. Toth, P., Vigo, D.: The Vehicle Routing Problem. Society for Industrial and Applied Mathematics, Philadelphia (2001)
36. Vig, L., Adams, J.A.: Issues in multi-robot coalition formation. In: Multi-Robot Systems. From Swarms to Intelligent Automata, vol. III, pp. 15–26 (2005)
37. WEBOTS: Webots User Guide (2009)
38. Weglarz, J.: Project Scheduling: Recent Models, Algorithms and Applications. Kluwer, Dordrecht (1999)
39. Zlot, R., Stentz, A.: Market-based multirobot coordination for complex tasks. Int. J. Rob. Res. **25**(1), 73–101 (2006)