Incremental Multi-Robot Task Selection for Resource Constrained and Interrelated Tasks

Sanem Sariel, Tucker Balch and Nadia Erdogan

***Abstract*— When the tasks of a mission are interrelated and subject to several resource constraints, more efforts are needed to coordinate robots towards achieving the mission than independent tasks. In this work, we formulate the Coordinated Task Selection Problem (CTSP) to form the basis of an efficient dynamic task selection scheme for allocation of interrelated tasks of a complex mission to the members of a multi-robot team. Since processing times of tasks are not exactly known in advance, the incremental task selection scheme for the eligible tasks prevents redundant efforts as, instead of scheduling all of the tasks, they are allocated to robots as needed. This approach results in globally efficient solutions through mechanisms that form priority based rough schedules and select the most suitable tasks from these schedules. Since our method is targeted at real world task execution, communication requirements are kept limited. Empirical evaluations of the proposed approach are performed on the Webots simulator and the real robots. The results validate that the proposed approach is scalable, efficient and suitable to the real world safe mission achievement.**

I. Introduction

Multi-robot task allocation problem is better viewed as a scheduling problem if there are interrelations among tasks, suggesting the use of Operation Research (OR) methods. However, when the problem solving time is limited and/or reallocations are frequently required at runtime, OR methods may not be directly applicable. Our research focuses on distributed task execution to ensure robustness against failures and make the system suitable for environments where reliable teleoperating is not continually possible. Besides the base difficulties, finding a solution with the OR methods in a decentralized setting may need considerably high efforts in both computation and communication. In this case, heuristic methods are preferred to find good solutions in reasonable time.

In this research, our focus is on complex missions with interrelated tasks (project tasks) whose requirements on task execution may vary. These interrelations may correspond to shared resources, producer/consumer, simultaneity and task-subtask dependencies [1]. 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 the delivery tasks. More complicated interrelations may be involved in mission representations. Simultaneous execution requirements imply tightly coupled task execution where the actions implemented by each robot are highly dependent on the actions of others. According to the classification of multi-agent organizations given in [2], coalitions (agent groups) are formed to perform tasks in cooperation. From our perspective, coalitions are suitable to meet the simultaneous resource requirements on executing tasks with sub-teams of robots.

Our earlier experiments on different domains have revealed that incremental assignments eliminate redundant considerations for environments in which the best solution is highly probable to change, and efficient bidding strategies ensure solutions to be efficient with a time-extended view of the problem in a computationally tractable way [3]. In this research, we extend this approach for interrelated complex tasks with resource constraints.

II. Related Work

Task dependency has been analyzed in some earlier multi-robot cooperation schemes. One of the earliest studies on multi-robot coordination presents a generic scheme based on a distributed plan-merging process [4]. M+ scheme [5] combines local planning and negotiations on task allocation for robots having their own local world knowledge. [6] introduces the mechanism concept in the framework M+CTA for the resources used for multi-robot cooperation. 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 adapt their plans in the multi-robot context.

One of the earlier algorithms for coalition formation of cooperative multi agent systems to handle multi-agent (resource) requirements is presented in [7]. Another work on multi-robot coalition formation states the differences of multi-robot and multi-agent coalition formation issues from sensor possessive point of view [8]. Locational sensor capabilities are considered in their work on top of the coalition evaluation step suggested in [7]. Their approach assumes that capabilities are known apriori and coalitions are formed accordingly. ASyMTRe [9], uses reconfigurable schema abstraction for collaborative task execution providing sensor sharing among robots, and connections among the schemas are dynamically formed at runtime. The information labels provide a method for automating the interconnections of schemas, enabling robots to share sensory and perceptual information as needed. Their approach provides a way to form low level coalitions to share robot capabilities.

Given the real-time limitations, instantaneous task assignment becomes profitable as it provides a dynamic solution, allocating tasks to robots whenever resources are available

S. Sariel and N. Erdogan are with Istanbul Technical University, Department of Computer Engineering, TURKEY (email: {sariel,nerdogan}@itu.edu.tr)

T. Balch is with Georgia Institute of Technology, College of Computing USA (email: tucker.balch@cc.gatech.edu)

([10], [11]). However, in this case, 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. [12] models the multi-agent task assignment problem as a scheduling problem for the RoboCupRescue simulation domain. In their approach, each agent locally chooses its best task to accomplish using a scheduling algorithm. Then, the best local task information is exchanged among agents to find the global best task to perform. Earliest Due Date (EDD) algorithm, not taking into account the future considerations and interrelations, is used to schedule the tasks.

Our approach differs from earlier work in that incremental task selection and distributed single item allocation provides cooperation on the global plan to achieve the overall complex mission with interrelated tasks. Coordinated task selection backed up with a realistic world and mission representation meets execution constraints simultaneously. We evaluate our approach on both a realistic simulator and real robots.

III. Problem Statement

Multi-robot task allocation problem may be formulated based on the well known OR problem, Resource Constrained Project Scheduling Problem (RCPSP), which is known to be NP-Hard ([13], [14]). The adapted version of the formulation for our multi-robot task allocation problem 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 $R = \{r_1, ..., r_m\}$. The tasks are interrelated by two types of constraints. First, precedence constraints are defined between activities. These are given by relations $t_i \prec t_j$, where $t_i \prec t_j$ means that task t_j 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. We relax the limitation constraint on $reqno_i$ by allowing it to change during task execution according to new requirements. Consequently, alternative solutions may be found to allocate tasks to robots based on dynamic environmental factors.

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

Since schedules are subject to change, we propose an approach in which tasks are allocated to robots incrementally, without ignoring the overall global solution quality instead of initially scheduling all of the tasks. 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 the execution. Therefore, the allocation problem turns into a selection problem and may be stated as follows: The Coordinated Task Selection Problem (*CTSP*) for each robot in a team is determining the next action 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, $C1$),
- the given precedence conditions (Condition-2, $C2$) 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 itself is appropriate to execute, and
4) Simultaneously respond to contingencies and return to Step 1, when necessary, until the mission is achieved.

IV. Proposed Approach

We propose a distributed, incremental task selection approach as a part of our framework, DEMiR-CF [15], for robots in a multi-robot team where they need to cooperate/coordinate to achieve complex missions including tightly coupled tasks that require diverse capabilities and collective work. Coalitions (C_i) 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 requiring more than one robot. Sizes of coalitions vary according to the minimum number of robots required ($reqno_i$) to execute the tasks. Robots can detect and recover from different types of contingencies by keeping representative models of the system tasks and the states of other robots, details of which are given in [15]. A sample flow of the operations in our approach is summarized as:

1) Mission task definitions are given to the robots (time-extended representation of tasks with precedence constraints to achieve overall mission).
2) Each robot selects the most suitable candidate task to execute by global cost consideration among mission tasks (dynamic task selection/switching).
3) Robots offer execution intentions for the tasks they have selected. During these declarations, inconsistencies and conflicts are resolved.
4) Coalitions are formed for the announced tasks, making sure that each robot is in the most suitable coalition considering the global solution quality.
5) Dynamic task selecting/switching proceeds simultaneously with task execution. This allows switching between tasks if it is more profitable, handling real time contingencies at the same time. Then, corresponding auction and coalition formation procedures (2-4) are applied continually.

A. Mission Representation

A mission, in our approach, 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. Tasks are represented as septuples containing information regarding task execution requirements and task status: $< id, type, reqcap, deplist, reqno, relinfo, precinfo >$.

1) *id*: Each task is assigned a unique task id.
2) *type*: Each task is associated with a description of task type and corresponding action definitions.
3) *reqcap*: Requirements define special sensors and capabilities required to execute the task.
4) *deplist*: Interrelations represent the precedence constraints.
5) *reqno*: Minimum number of robots required to execute the task, either determined before mission execution or during runtime.
6) *relinfo*: Descriptive information regarding task type such as, the latest location, the target location, etc.
7) *precinfo*: Precaution information is used for contingency handling: the task state, the estimated task achievement time and the current execution cost.

Information in a task representation can dynamically be modified during execution. In particular, *relinfo*, *precinfo* and *reqno* are subject to change during execution. Sample mission representations are given in [15].

B. Dynamic Priority-based Task Selection Scheme

In our approach, 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 the next subsection). Therefore, robots in our 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, detailed decisions.

Instead of scheduling all tasks in one step, we propose a Dynamic Priority-based Task Selection Scheme (DPTSS) to allocate tasks 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 needs to be assigned, instead of scheduling all tasks from scratch.

The following definitions are needed to present our formulation to the solution of the *CMAP*. 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. t_i is an *executable task*, if at least $reqno_i$ robots can be assigned for its execution. t_{ie_j} is a *task in execution* by robot r_j or coalition C_j. T_{ie} is a union of tasks in execution. t_{E_j} is an *eligible task*, if it is an *executable task* and is neither in execution (t_{ie}) nor achieved. T_{E_j} is a union of *eligible tasks* for robot r_j. t_ϕ is an *ineligible task*; if it is not an *executable task*, if it is already achieved or if it is not a *suitable task*. T_ϕ is the union of *ineligible tasks*. $P(t_i)$ is defined as the set of all predecessor tasks of the task t_i. t_{A_j} is an *active task* if it is *suitable*, *executable* and tasks in $P(t_{A_j})$ are completed. $T_{A_j} (\subseteq T_{E_j})$ is the union of the *active* tasks for robot r_j. An *inactive task set* $T_{I_j} = T_{E_j} \setminus T_{A_j}$ contains the tasks that are *suitable* but not *executable* yet for robot r_j. A *critical task* t_C is a task that has inflexibility from the point of view of resources and the robot is suitable for that task. L_{C_j} is a prioritized list of *critical tasks* for robot r_j. A *rough schedule* S_{R_j} for robot r_j is a priority queue of mission tasks that r_j assumes it will execute.

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 list (L_{C_j}), the eligible task set (T_{E_j}), the conjunctive arcs (if any) and the requirements. Since each robot r_j has different capabilities, the eligible task sets (T_{E_j}) and the priority queue entries may be different. The critical tasks may be determined either by negotiations or by beliefs. To eliminate intractable communication overhead, we use a rough belief update approach to form the critical tasks. 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.

Algorithm 1 GeneratePriorityList for robot r_j

input: Eligible task set (T_{E_j}), active task set (T_{A_j})
output: Topologically ordered and prioritized schedule list: S_{R_j}

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

Intuitively, robots do not deal with the ineligible tasks (T_ϕ), while forming the rough schedules. The eligible tasks ($T_{E_j} = T \setminus T_\phi$) for robot r_j consists of active and inactive tasks. 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 1. 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 execution of Algorithm 2 where $curcs_j$ represents the remaining capacity of robot r_j and $reqcs(t_i)$ represents the required capacity for task t_i in terms of the consumable resources (e.g fuel).

Algorithm 2 GenerateRoughSchedule for robot r_j

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

$t_s = \phi$; $R = curcs_j$; $achievable = true$;
S_{Rj} = GeneratePriorityList(T_{Ej}, T_{Aj})
/*Determines if the mission is achievable*/
for each $t_i \in L_{Cj}$ **do**
  $R = R - reqcs(t_i)$
  if $R < 0$ **then**
    $achievable = false$
    $R = curcs_j$
    break
  end if
end for
if $S_{Rj} \neq \phi$ and $(top(S_{Rj}) \in L_{Cj} \parallel R - reqcs(top(S_{Rj})) \geq 0)$ **then**
  $t_s = top(S_{Rj})$
end if

In the rough schedule generation algorithm, while forming the rough schedule, 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 itself in case new robots can be deployed. 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 top of the rough schedule that can be executable is the most suitable task to be executed for the robot.

2) DPTSS Algorithm: In our 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 the eligible task set (T_E). Algorithm 3 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: (1) to continue executing the current task (if any), (2) to join a coalition, (3) to form a new coalition to perform a free task, or (4) to stay idle. 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.

Algorithm 3 DPTSS Algorithm for robot r_j

input: Mission (M) task descriptions
output: Action to be performed depending on the selected task

Determine the T_{Ej}, $T_{Aj} \subseteq T_{Ej}$ and $L_{Cj} \subseteq T_{Ej}$
/*GenerateListOfCriticalTasks*/
$L_{Cj} = \phi$
for each $t_i \in T_{Ej}$ **do**
  $P_{ct}(t_i) = \frac{reqno}{\#ofsuitablerobots}$
  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
[S_{Rj}, t_s] = GenerateRoughSchedule (T_{Ej}, T_{Aj}, L_{Cj}, $curcs_j$)
if $t_s \neq \phi$ **then**
  if t_s is the current task **then**
    Continue with the current execution
  else
    Offer an auction to form a new coalition or directly begin execution
  end if
else
  if $t_s \in T_{ie}$ and it is profitable to join the coalition **then**
    Join the coalition
  else
    Stay idle
  end if
end if

C. Distributed Task Allocation Scheme

In our distributed allocation approach, standard auction procedures of CNP [16] are applied to announce the *intentions* of robots on task execution and select the *reqno* number of robots for a coalition in a cost-profitable, scalable and tractable way. Additionally, precaution routines are added to check validity, consistency and coherence in these negotiation steps [15]. Each robot intending to execute a task announces an auction after determining its rough schedule and performing the DPTSS. 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(s) is selected in the auction by considering the cost values. Auction negotiations and the selection of the suitable robots are performed in a completely distributed fashion by the auctioneers. Single task items are auctioned and allocated in auctions. The framework allows multiple auctions to be carried out simultaneously. Validity controls are performed to ensure the system consistency, the details of which are given in [15].

D. Cost/Bid Evaluation and the Tie Breaking Rules

The cost evaluation has a tremendous impact on the solution quality. Each task type as a part of the mission requires a different cost evaluation to efficiently solve the problem. For now, we perform the simplest evaluations for the cost and bid determination and leave a more extended analysis on the cost function design for future work. Cost evaluation is performed by using the corresponding functions

TABLE I

COST EVALUATIONS FOR DIFFERENT TASKS

Task Type(s)	Cost Function
Locate/Pick-up	Estimated time to reach at the location of the object.
Deliver/Push	Estimated time to carry/push the object from the initial location to the final destination.
Clean	Estimated time to cover the whole environment.

given in Table I. If a robot is executing a task when it receives an auction message, it sends the bid value by considering the final destination of the current task as the location of itself. A common situation appears when the auctions are offered at the same time by different robots either for the same task or for different tasks. In our approach, if there are conflicting auctions for the same task, only the one with the smallest cost value continues with the auction negotiation process. 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 (GPR), is used [14].

E. Analysis of the Approach

Our approach offers a polynomial time solution. The critical task list generation takes $O(nlog(n))$ time 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 $O(n(e+log(n)))$. If $(e << n)$, the complexity of the proposed approach reduces to $O(n^2log(n))$.

V. EXPERIMENTS

We have conducted real world experiments and real-time dynamic simulation experiments on Webots, the professional mobile robot simulation software [17]. In our simulation experiments, each environment is represented as a 5m by 5m 3D virtual world where 70mm-size simulated Khepera II robots and objects are located. The environments are randomly generated VRML files containing the robots and the objects. Each Khepera II robot is mainly equipped with a 25MHz MC68331 micro-controller, 512K Flash and 512K RAM memories and 8 infra-red sensors with limited obstacle detection range as it is simulated in Webots. Communication is achieved through wireless links in both simulations and in the real world experiments. Real Kheperas have standard radio turrets mounted on them to communicate through the selected radio frequency.

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. All picked up items are collected in the center of the environment. The items are distributed in the environment at fixed locations for each run. The robot locations are randomly determined. Figure 1 illustrates the mission completion times and the total path length traversed by the robots for sets with different numbers of robots. As expected from the approach, time to complete the overall mission overly reduces with increasing numbers 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 corresponding graph.

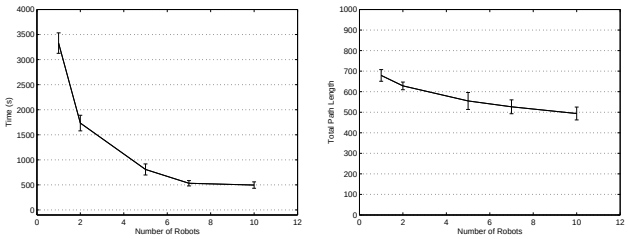

Fig. 1. Left: mission completion time (s); right: total path length (mm) results for the pick-up/delivery mission, with task number fixed at 20.

A sample scenario for a complex mission which includes tasks for pushing boxes and picking-up and delivering items to a desired location is given in Figure 2 with five participating robots. 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 the mission execution. In the second scenario, since the minimum required number of robots to push one of the boxes is two, the two robots form a coalition and push the heavy box synchronously.

Another complex mission allocation scenario which includes tasks for pushing a box, carrying a cylindrical object to a final destination and then inspecting the environment is implemented by three Khepera II robots and the execution scenario is illustrated as overlapped video images in Figure 3 (Web-site reference for the corresponding videos: [18]). There are interrelations between push, carry and inspect tasks respectively as in the graph depicted in Figure 4. While the objects can only be carried by the robots with grippers, the inspection task requires possessing a camera. The box can be pushed with all three robots. However, due

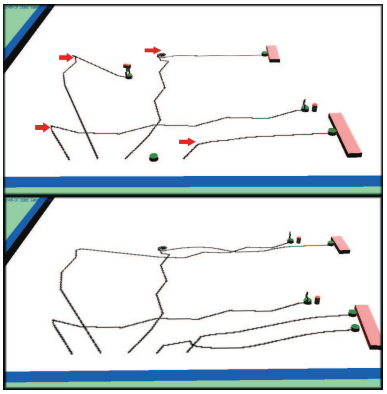

Fig. 2. Scenario 1-2: Robots push and carry boxes to the final destinations.

Fig. 3. Khepera II robots achieve the overall mission of pushing/carrying the objects to the final locations and inspecting the area.

to the cost evaluations and the critical task list consideration, allocations are implemented accordingly. Robots obey the interrelation constraints and each robot involves in a suitable task execution for itself on which the decision is made in a distributed manner.

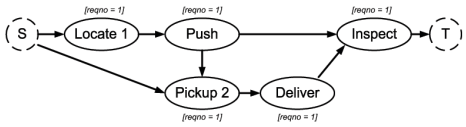

Fig. 4. Real scenario mission graph with interrelated tasks

VI. Conclusion

We have described the details of a new approach that enables the efficient achievement of interrelated, resource constrained tasks of a mission by a multi-robot team. The proposed approach relies on the distribution of the decision mechanism by introducing the CTSP and solves this problem through a scheme that involves the incremental selection and allocation of tasks dynamically deriving the mission execution. The main contributions of this approach are the elimination of the redundant efforts for dealing with the changing structure of the mission due to the uncertain information or the dynamism of the environment, and on the other hand ensuring the time extended consideration of the problem through forming computationally efficient rough schedules, and the applicability of the approach efficiently on real robots with limited computational capacities.

References

[1] S. Ossowski, *Co-ordination in Artificial Agent Societies, Social Structure and Its Implications for Autonomous Problem-Solving Agents*. Springer Verlag, 1999.

[2] B. Horling and V. Lesser, "A survey of multi-agent organizational paradigms," *The Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, 2005.

[3] S. Sariel and T. Balch, "Efficient bids on task allocation for multi-robot exploration," in *The 19th International FLAIRS Conference*, 2006.

[4] R. Alami, F. Ingrand, and S. Qutub, "A scheme for coordinating multi-robot planning activities and plans execution," in *Thirteenth European Conference On Artificial Intellingence*, 1998.

[5] S. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 1999.

[6] R. Alami and S. C. Botelho, "Plan-based multi-robot cooperation," in *Advances in Plan-Based Control of Robotic Agents*, 2001.

[7] O. Shehory and S. Kraus, "Methods for task allocation via agent coalition formation," *Artificial Intelligence*, vol. 101, pp. 165–200, 1998.

[8] L. Vig and J. A. Adams, "Issues in multi-robot coalition formation," in *Multi-Robot Systems. From Swarms to Intelligent Automata. Volume III*, 2005, pp. 15–26.

[9] L. E. Parker and F. Tang, "Building multi-robot coalitions through automated task solution synthesis," *Proceedings of the IEEE, Special Issue on Multi-Robot Systems*, 2006.

[10] L. E. Parker, "Alliance: An architecture for fault tolerant multi-robot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, no. 2, pp. 220–240, 1998.

[11] B. Gerkey and M. J. Mataric, "Sold!: Auction methods for multirobot coordination," *IEEE Trans. Robot. Automat.*, vol. 18, no. 5, pp. 758–768, 2002.

[12] S. Paquet, "Distributed decision-making and task coordination in dynamic, uncertain and real-time multiagent environments," PhD Thesis, Laval University, Quebec, 2006.

[13] J. Weglarz, *Project Scheduling: Recent Models, Algorithms and Applications*. Kluwer, 1999.

[14] P. Brucker and S. Knust, *Complex Scheduling*. Springer Verlag, 2006.

[15] S. Sariel, "An integrated planning, scheduling and execution framework for multi-robot cooperation and coordination," PhD Thesis, Istanbul Technical University, Turkey, 2007.

[16] R. G. Smith, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Transaction on Computers C-*, vol. 29, no. 12, pp. 1104–1113, 1980.

[17] Webots User Guide 5.1.11, 2006.

[18] SarielKh2Movies: http://www2.itu.edu.tr/∼sariel/videos/KheperaII-Movies.html, 2007.