M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement

Silvia .C. Botelho, Rachid Alami

M+ : a scheme for multi-robot cooperation through negotiated task allocation and achievement

S.C. Botelho*, R. Alami
LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France
e-mail : {silviacb,alami}@laas.fr

Abstract

In this paper, we present and discuss a distributed scheme for multi-robot cooperation. It integrates mission planning and task refinement as well as cooperative mechanisms adapted from the Contract Net Protocol framework. We discuss its role and how it can be integrated as a component of a complete robot control system. We also discuss how it handles distributed task allocation and achievement as well as cooperative reaction to contingencies. Finally, we illustrate its use through a simulated system, which allows a number of robots to perform load transfer tasks in a route network environment.

1 Introduction

This paper presents a new multi-robot cooperative scheme based on a combination of local planning and negotiation for task allocation and cooperative reaction to contingencies. It is built on the assumption that, in a complex system composed of several autonomous robots equipped with their own sensors and effectors, the ability of a given robot, to achieve a given task in a given situation can be best computed using a planner. Indeed, we claim that the robots must be able to plan/refine their respective missions, taking into account the other robots' plans as planning/refinement constraints, and thus producing plans containing coordinated and cooperative actions that ensure their proper execution and will serve as a basis for negotiation.

In the last decade, various studies have been made concerning the field of multi-robot systems [4]. We restrict our analysis here to contributions proposing cooperative schemes at task level. Indeed, several generic approaches have been proposed concerning goal decomposition and task allocation (Contract Nets [15], Partial Global Planning [5]), distributed search [6], negotiation [10, 14, 8], motivational behaviors [12, 7]. Cooperation for achieving independent goals has been mostly addressed in the framework of application-specific techniques such as multi-robot cooperative navigation [17, 3].

We argue that as the robotic systems become more sophisticated, so the range of possible tasks increases. Thus we need more flexible and generic approaches to describe and perform task planning, decomposition and allocation in multi-robot environments. In this paper, we present a decentralized system, called M+ protocol. This protocol, in allied to the LAAS architecture [1], can be used in a wide range of multi-robot applications. The scheme allows the achievement of a cooperative mission. It has the capacity for on line task decomposition, (re)planning and (re)allocation. Each robot has reasoning, decision and reactive capabilities.

In our scheme, every robot receives the same mission. A mission is a set of partially ordered tasks, where each task (T_i) is defined as a set of goals to be achieved. Each robot has its own local knowledge of the world. In accordance with its context and capabilities it plans and decomposes the tasks into a set of actions (A_i). The execution of these actions achieves the goals of the corresponding tasks. Figure 1 shows a simple mission and the planned actions (by the different robots) that achieve one of its tasks.

The tasks are allocated (and re-allocated when necessary) incrementally by the robots, through a negotiation process. This negotiation is combined with a task planning and cost estimation activity which allows each robot to decide its future actions taking into account its current context and task, its own capacities as well as the capacities of the other robots. In order to take into account the current and short-term context, anticipation is limited to one task ahead for each robot. This allows, whenever possible, an overlapping between the execution of the current task for

*On leave from FURG, Brazil

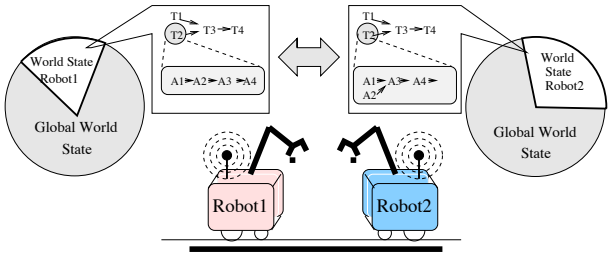

Figure 1: The mission and the set of actions that each robot plans to achieve task T_2

a robot and the planning and task allocation of the next task.

The proposed scheme is also designed to provide, when necessary, a cooperative reaction to contingencies. Whenever the execution of a task fails due to some unpredictable problems that prevent a robot from achieving the task, it first tries to find another set of actions that can achieve the current goal(s). If it fails, it requests help from the other robots. When a robot receives a request for help, it tries to find plans of cooperative actions in order to achieve its current goal(s) in conjunction with those of the failed robot.

Note that M+ is devised to be utilized at a level where one can assume that the robots should be able to achieve the tasks by their own. However, and this is a key aspect in robotics applications, we do not need the robot planners to avoid resource conflicts. Another cooperation scheme can be used in conjunction with M+ at a lower level: the Plan Merging paradigm [13].

We shall introduce a more formal presentation of the method for describing the world state and the actions in section 2. The developed architecture, and the M+ protocol are presented in section 3 and 4, respectively. Section 5 presents the multi-robot application on which we tested and validated M+.

2 The World State Description

The state of the world is described through *attributes*. An *attribute* has the description: $(ATTRIBUTE\ a) : (x, v)$, where a is the name of the attribute, x is the name of the *object* on which the attribute is applied, and v is the *value* of the attribute.

The state of the world can be described by a state vector composed of instantiated attributes. The system maintains three data structures: (1) S_w, the *current state* vector (2) IAS_w, the *intentional action state* vector corresponding to the expected state of the world at the end of the current action, and (3) ITS_w the *intentional task state* vector corresponding to the expected state at the end of the current task.

We point out that the robots can have different world states, since each robot has its own local knowledge.[1]

3 The Architecture

The M+ cooperative system was developed to be integrated using the LAAS Architecture [1] where the *decisional system* can be composed of several layers. Each layer comprises two entities: a planner and a supervisor. The *planner* produces the sequence of actions necessary to reach a given goal. It is used as a resource by the *supervisor* which actually interacts with the next layer, controls the execution of the plan and reacts to incoming events. This paradigm consists in guaranteeing a bounded reaction time for a response to an event (possibly after the reflex reaction already taken by the functional and execution control levels).

We have utilized the M+ cooperative system as a *Task Layer* on top of a *action layer* (see Figure 2) for autonomous robots. The action coordination layer is capable of planning and executing cooperative navigation actions[1] as well as simple manipulation actions. It also includes a cooperative mechanism of a different nature, specially designed for solving resource conflicts: the Plan-Merging Paradigm [1].

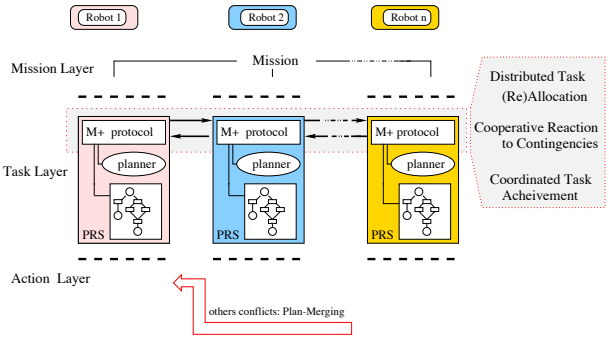

Figure 2: A cooperative task layer

The new Task Layer is installed on all robots and is in charge of the negotiation mechanisms described in this paper. It receives a list of tasks (a set of goal attributes), from a higher level called *Mission layer*. A mission can be generated by a very high level central planner, in our case IxTeT [11], or it can be sent directly by one or several users.

[1] In fact, every world description is associated with a specific robot. Thus, we have $S_w^{R_i}$, $S_w^{R_j}$, where $S_w^{R_i} \neq S_w^{R_j}$. In this paper we omit the robot index, in order to simplify the presentation.

This layer is composed of two entities: a *task planner* and *task supervisor* implementing the *M+ protocol*. Its has been implemented in *PRS* [9]; its different components run as tasks inside *PRS* and communicate through its database.

From time to time, the *task supervisor* will submit plan requests to the *task planner*. *M+* will then use the obtained plan to compute an associated cost that will serve as a basis for negotiation.

In the current version we use GraphPlan [2] as the standard planner. The main reason why we have chosen GraphPlan is that it is sufficiently fast to allow on-line planning activities. Moreover, the fact that GraphPlan finds the plan with the smallest number of steps allows us to have a reasonably good idea of the estimated cost of a plan.

In the next section we will describe the *M+ protocol* in more details.

4 The M+ Protocol

The M+ Protocol embeds three activities: *M+ task allocation*, *M+ cooperative reaction* to contingencies and *M+ task execution*. The *M+ task allocation* is in charge of task refinement and allocation. It embeds the negotiation mechanisms which allow a robot to choose incrementally the best task to be executed taking into account the current context. We use an adapted version of the *Contract Net Protocol* [15] for the negotiation.

The *M+ cooperative reaction* activity is invoked when a failure occurs during task execution; it updates the world state, manages the exchange of information between the robots and controls the (re)planning and requests for help.

Finally the *M+ execution* is in charge of task execution control as well as distributed synchronization between robots tasks and actions.

The following sections present the three activities in more details.

4.1 M+ Task Allocation

Each robot receives the same mission description, i.e. the same set of partially ordered tasks. At any moment a task is said to be *executable* if all its antecedents are already achieved or under execution. As we will see below (§4.3), the robots are informed whenever a robot announces a first offer to perform a task, and whenever a task is started or finished.

The M+ Allocation Task mechanism allows the robots to incrementally choose a task to perform among the current *executable tasks*. We limit the negotiation and planning to the set of executable tasks because, in general, a plan to perform a task may depend on the state of the world resulting form the previous tasks. The choice criteria will be the costs of the plans elaborated by different robots depending on their capabilities and situations.

Figure 3 shows the *M+ task allocation* state diagram. There are 5 possible states: *plan*, *eval-sharing*, *candidate*, *best-candidate* and *idle*.

A robot R_i enters the *eval-sharing* state whenever there is an update in the set of *executable tasks* and it is ready to negotiate a new task. It invokes the planner (1) in order to elaborate plans for the executable tasks and to estimate their costs (2). Then, R_i compares its costs to offers announced by other robots. It then selects the task of the lowest cost that it can perform and whose cost is better that the cost announced by other robots, if any.

If there is no such task, R_i enters the *idle* state (7).

If a task T_k is selected, R_i enters the *candidate* state (3). There are two possible cases:

1. if T_k has not yet been selected by another robot (no robot has announced a first offer to perform it) R_i robot broadcasts[2] a "first offer" announcement message to inform the other robots of its ability to perform T_k at a given cost C_k^i. Besides, R_i will become responsible for all messages routing concerning T_k. It then enter the *best-candidate* state (5).

2. T_k has already been announced by another robot (let R_j be such robot). R_i has still the possibility propose to R_j a best offer. This will true until R_j begins to execute the task. R_i sends its offer to R_j and waits for an answer. The answer can be positive, meaning that R_j accepts to transfer T_k to R_i. R_i will then enter the *best-candidate* state (5). Note however that R_j will stay in charge of all messages routing concerning T_k. The answer can also be negative, meaning that another robot R_p has already made a better offer to R_j or that the execution of T_k has already been started. R_i will then abandon T_k and enters the *eval-sharing* state in order to select a new task (4).

When a robot R_i enters the *best-candidate* state for a given task T_k , it stays in this state until either it begins T_k execution or until it receives a best offer from another robot, or until it abandons T_k due to a failure or to a cooperative reaction (6). It then enters the *eval-sharing* state in order to select a new task.

[2]This message, called "first offer" is broadcasted in a critical section, in order to guaranty that there is only one robot performing such transition. See §4.4.

When a robot is in the *idle* state, it stays in this state until there is a change in the set of executable tasks (8); this can be the result of a message signaling that a task has been started, or abandoned or the reception of a new mission.

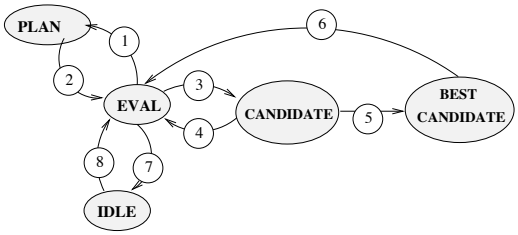

Figure 3: The task allocation state diagram

4.2 M+ Cooperative Reaction

The M+ scheme provides also for cooperative robot behavior in case of execution failure. When a problem occurs that prevents a robot R_i from achieving a task T_k, it first tries to re-plan in order to find another set of actions to achieve T_k starting from the new state resulting from the failure. But if R_i does not find a new plan, it sends relevant information with an *request for help* to the other robots and waits. If several robots propose their help, R_i will select the best offer. R_i will abandon T_k only if it receives no help offer.

The "help context": In nominal situations, a robot maintains only the information associated with itself and with the goals corresponding to each task executed by the other robots.

Thus, whenever a robot R_j decides to give help to robot R_i, it needs more information about R_i current state in order to elaborate a plan that achieves the conjunction of goals of R_i and R_j.

Let us assume that R_i current task is T_k while R_j current task is T_l.

In order to select the relevant information that should be sent together with a request for help, R_i selects from its current world state all attributes that have a relation with the goal expressed by T_k. If $(ATTRIBUTE;\ name) : (object; value)$ is included in goal expressed by T_k, the M+ protocol will select from R_i current world state all attributes that have *object* in the object or value field, as well as the attributes that have R_i itself as object or value. It will then issue a *request for help* message (see Figure 4).

When R_j decides to process a *request for help* message issued by R_i, it tries to elaborate a new plan which satisfies the conjunction of goals expressed by

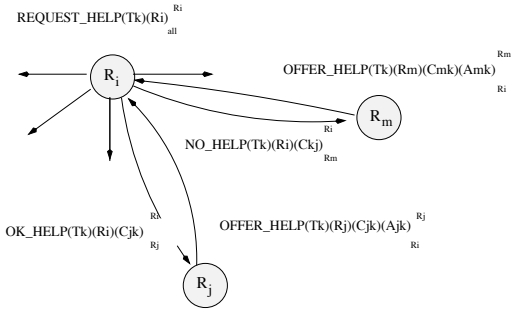

Figure 4: The request for help after failure

T_k and T_l. If R_j succeeds in elaborating such a plan, its estimates cost and sends an offer of help to R_i. Note that this plan may contain intricate actions involving R_i and R_j.

After a timeout, R_i collects the offers of help (if any) and chooses the less expensive. It then informs the robot that has been chosen. If no offer has been provided, R_i abandons its task.

4.3 M+ Execution

M+ Execution is in charge of task execution control as well as distributed synchronization between robots tasks and actions.

In "nominal situations", there is only a need for synchronization at task level: this is achieved using two message types. The M+ task controller of a given robot broadcasts a message to all robots whenever it starts or finishes a task.

The start messages will be used by the different robots to update their sets of executable tasks. The messages signaling the achievement of a task T_k will allow the other robots to start tasks that are successors of T_k in the mission description.

M+ Execution is involved in action synchronization only when a cooperative reactive to contingencies between two robots is running; synchronization messages are the exchanged only between the two robots involved in the cooperation.

4.4 System Coherence

The system coherence is obtained through the following premises:

- Only one robot is allowed to issue a "first offer" broadcast for a given task. This is made possible because the robots broadcast such message in a critical section, protected by a distributed implementation of a global semaphore [16].
- For each task T_k, the robot which has issued the first offer, becomes responsible for all messages

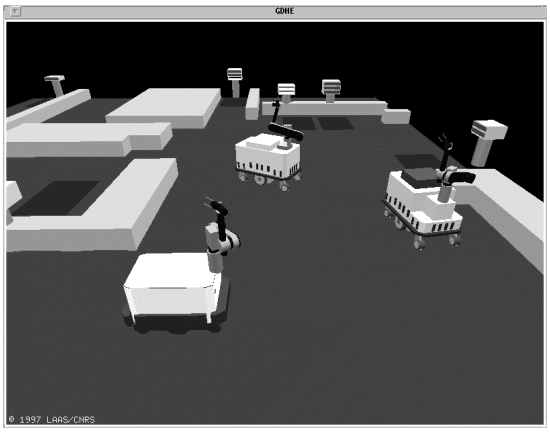

Figure 5: An example of the simulated environment.

routing concerning T_k. This allows the maintenance of coherent information concerning that task during the negotiation process without the need of broadcast messages or semaphores.

- Each robot maintains its own coherence; for example, at the level of each robot, there is only one planning activity at a time.

5 An Implemented Example

A first version of the M+ protocol has been implemented. We describe below an illustrative example of its use. The system involves three simulated mobile robots mobile manipulators (R_0, R_1 and R_2) that achieve load transfer tasks. Each robot control system runs on an independent Sun workstation which communicates with the other workstations through TCP/IP.

The environment is composed of open areas connected by lanes. The open areas contain "stations" where the robot have to dock in order to pick-up or put-down containers (see Figure 5).

The available actions are: `pick-up-from-station`, `put-down-on-station`, `pick-up-from-robot`, `put-down-on-robot` and `go-to`. An instance of a task is to transfer a container C_x from station S_y to station S_z.

Figure 6 shows an overview of a result of a run involving three simulated robots. When the system starts, the 3 robots receive a mission composed of a set of 10 partially ordered tasks.

The left side of the figure shows a directed acyclic graph which gives the partial order between tasks: $T_0 \prec T_3$, $T_1 \prec T_3$, $T_0 \prec T_9$, $T_1 \prec T_9$, $T_3 \prec T_5$, and $T_4 \prec T_5$ where '$\prec$' means 'before'.

The right side of the figure shows the temporal evolution of the tasks. One can see the evolution of the 10 tasks during time, their state as well as the role of the different robots.

Each robot has a specific color: black (R_0), gray (R_1) and dark gray (R_2). The task are represented in thee different states: `Done` (when the task has not yet been chosen by any robot or is finished), `Wait` (when the task has already been allocated by a robot), and `Exec` (when it is under execution).

After a while corresponding to an intensive planning and negotiation activity: R_0 chooses and begins to execute T_6, R_1 chooses T_2 and R_2 chooses T_1.

After having started execution, the robots start, by anticipation to negotiate their future actions: for example, R_1 chooses T_8 while executing T_2.

One can observe the result of two re-allocation negotiations (shown by ellipses) for the tasks T_9 and T_5.

Let us comment the first one: "R_1 finishes T_2, begins T_8 and allocates T_9; this is possible because T_0 and T_1 have already been achieved. While executing T_3, R_2 finds itself in a situation when it can make a better offer for T_9. R_1 accepts the offer and start a search for a new task".

We can also observe the result of a cooperative reaction (emphasized by a rectangle): "R_1 detects a failure while executing T_2; it cannot find a corrective plan by its own; it issues a request for help and receives an offer from R_0. T_2 is successfully achieved.".

6 Conclusion

We have proposed a decentralized multi-robot system scheme, called M+, for loosely coupled tasks planning and negotiation. It provides for distributed task allocation and re-allocation (before execution) as well as (partial or total) task re-allocation, planning and cooperative task achievement after an execution failure.

We have utilized the M+ scheme to implement a *Task Layer* of an autonomous Robot Control System. This layer is composed of two entities: a *task planner* and *task supervisor* implementing the *M+ protocol*.

A first version of the system has been implemented and already tested in simulation. Our future work will be to improve its implementation, to augment its reasoning capabilities by using more elaborate task planners and to implement it on real robots. A second step would be to focus on the treatment of joint goals and on on-line cooperative plan revision.

Acknowledgments: This work was partially supported by CNPq (Brazil) under grants to the first au-

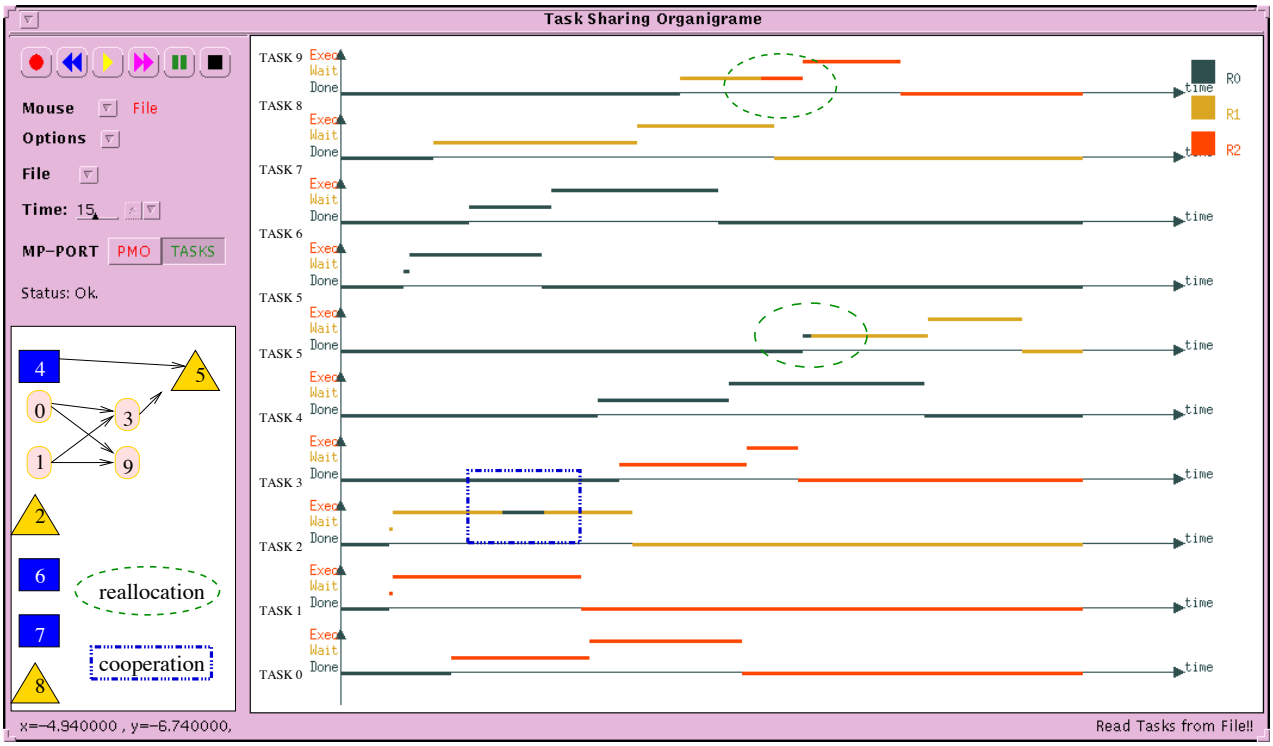

Figure 6: General overview of the simulation: mission with 10 tasks and 3 robots.

thor.

References

[1] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi-robot cooperation in the martha project. *IEEE Robotics and Automation Magazine, Special Issues: Robotics and Automation in Europe*, 1997.

[2] A. Blum and M.. Furst. Fast planning through planning graph analysis. In *IJCAI'95*, 1995.

[3] A. Brumitt, B. Stentz. Dynamic mission planning for multiple mobile robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA'96)*, 1996.

[4] Y. Cao, A. Fukuna, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. autonomous robots. *Autonomous Robots*, 4:7–27, 1997.

[5] E. Durfee and V. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *TCiber*, 21(5), 1991.

[6] E.H Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior space. In *IEEE Transactions on Systems, Man and Cybernetics*, 1991.

[7] E. Ephrati, M. Perry, and J. S. Rosenschein. Plan execution motivation in multi-agent systems. In *AIPS*, 1994.

[8] G. Ferguson and J.F. Allen. Arguing about plans: plan representation and reasoning for mixed-initiative planning. In *AIPS*, 1994.

[9] F.F. Ingrand, M.P. Georgeff, and A.S. Rao. An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6), 1992.

[10] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.

[11] P. Laborie. *IxTeT: une approche integree pour la Gestion de Ressources et la Synthese de Plans*. PhD thesis, Ecole Nationale Superieure des Telecommunications, 1995.

[12] L. Parker. Cooperative robotic for multi-target observation. In *to appear in Intelligent Automation and Soft Computing, special issue on Robotics Research at Oak Ridge National Laboratory*, 1998.

[13] S. Qutub, R. Alami, and F Ingrand. How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *IEEE Int. Conf. on Intel. Robots and Sys. (IROS'97)*, 1997.

[14] J. S. Rosenschein and G Zlotkin. Designing conventions for automated negotiation. *AI Magazine*, 15, 1994.

[15] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, c-29(12), 1980.

[16] M. Trehel and M. Naimi. Un algorithme distribué d'exclusion mutuelle en log(n). *Technique et Sciences Informatiques*, 6(2), 1987.

[17] S. Yuta and S. Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE Int. Conf. on Intel. Robots and Sys. (IROS'92)*, 1992.