A multi-robot cooperative task achievement system

Silvia Silva da Costa Botelho, Rachid Alami

A multi-robot cooperative task achievement system

S.C. Botelho*, R. Alami
LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France

Abstract

This paper discusses a general architecture where various schemes for multi robot task achievement can be integrated called "M+ Cooperative task achievement". The main originality comes from its ability to allow the robots to detect - in distributed and cooperative manner - resource conflict situations as well as sub-optimalities. Different decisions are performed by the robots such as actions re-scheduling, suppression of redundancies and opportunistic enhancement of the "global plan". Finally, we illustrate its use through a simulated system, which allows a number of robots to plan and perform cooperatively a set of tasks in a hospital environment.

1 Introduction

Our long-term target is the development of a comprehensive set of schemes, mechanisms and tools for multi-robot cooperation. The first step towards this target was the development of the Plan-Merging Paradigm [1]. A second main step was the development of a distributed cooperative task allocation scheme called M+ Protocol [3, 2]. This paper presents **M+ task achievement**, which is a decentralized multi-robot scheme based on an on-line combination of local individual planning and multi-robot plan validation for coordinated and cooperative behavior. The robots plan/refine their respective missions, taking into account the other robots' plans and *social rules* as planning/refinement constraints, and thus produce validated multi-robot plans containing coordinated and cooperative actions.

In the last decade, various studies have been made concerning the field of multi-robot systems [5]. We restrict our analysis here to contributions proposing cooperative schemes at the decision level. Indeed, several generic approaches have been proposed concerning goal decomposition and task allocation (Contract Nets [15], Partial Global Planning [7]), negotiation [9, 13], motivational behaviors [11, 8]. Cooperation for achieving independent goals has been mostly addressed in the framework of application-specific techniques such as multi-robot cooperative navigation [16, 4].

In the multi-robot context, we distinguish three main issues that can, very often, be treated separately and in a hierarchical manner: *decomposition* of a global mission into tasks, *allocation* of the tasks among the robots and *task achievement*. Figure 1 illustrates our view of the problem. Each issue can be treated in a centralized and/or decentralized way by its own planning module.

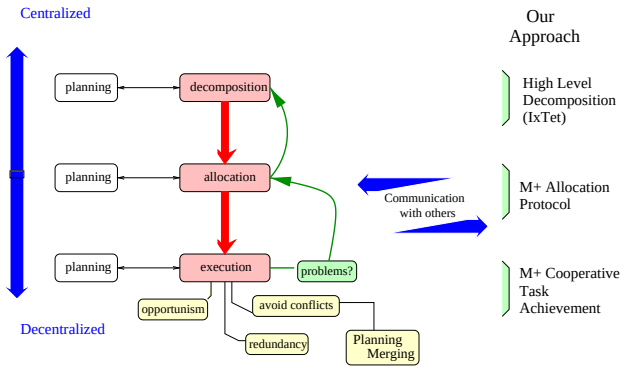

Figure 1: Cooperative Multi-robot systems: our approach

Mission Decomposition can be executed by a very high level central planner, in our case **IxTeT** [10]. For task allocation, we have developed an incremental cooperative mechanism, called **M+ protocol** [3]. It is based on a negotiation mechanism that consists of a combination of task planning and cost estimation activity. *Task achievement* corresponds to the effective execution of the task. During this stage the robot interacts directly with the environment and perceives modifications caused by the other agents.

Since each robot synthesizes its own plan, we have defined three issues induced by the distributed nature of the system and which may need cooperative decision. These issues are:

*On leave from FURG, Brazil

avoid conflict: although *mission decomposition* guarantees a conflict-free set of goals, it may occur that individual plans elaborated by the different robots to achieve these goals contain resource conflict situations. In this paper, we propose a cooperative scheme for multi-robot coordination which is an extension of the *Plan-Merging* Paradigm [1].
redundancy: it can occur that some plans embed redundant actions, whose effects can be (or is already) accomplished by another robot. We distinguish two kinds of redundant actions: exclusive and cumulative actions. The former occurs when only one agent can accomplish the task, and the robots must cooperate and decide which robot will be in charge of it. In the other case, the task can be accomplished simultaneously by a set of robots. They can decide to achieve the task together, thus increasing the performance of the system;
opportunism: during task execution, one robot can decide to execute an action, that was not originally planned by it, that enhances the global performance of the system.

These issues are treated by the **M+ cooperative task achievement**. One key aspect is the introduction in the world description of *social rules* that help to detect and to treat situations that need cooperative decision or may benefit from it. The next section gives a more formal presentation of the world description and the *social rules*. We then discuss the *mechanisms* for plan validation (section 3) and how they are used in the cooperative negotiation processes (section 4). Section 5 presents a multi-robot application on which we have tested and validated our approach.

2 The World State Description

In this section, we present the world model we use. It has been specially devised to allow reasoning on the cooperative issues mentioned above while maintaining a STRIPS-like representation in order to allow the robots to use efficient practical planners. In our case, we use a system called PROPICE-PLAN [6], which combines the IPP planner [?] which a Procedural Reasoning system.

2.1 A state description

A world state, in our system, is described through a set of predicates ATTRIBUTE(ENTITY, VALUE). There are predicates that can only be changed by a given robot. We call them *exclusive* predicates. For instance, the state of the gripper of a robot is an *exclusive* predicate. Conversely, predicates that can be changed by several robots are called *non-exclusive* and have to be manipulated as possible contingent effects. For instance, if there are several robots able to open a door, `STATE-DOOR(D1 OPEN)` will be considered as a *non-exclusive* predicate. *Exclusive and non-exclusive* predicates are obtained off-line from the possible actions of each entity.

Some predicates have an associated *achievement* level which is interpreted a threshold that entails a change in its value. Such attribute is called *incremental*. For example, STATE_CLEAN(LANE1,OK) is an *incremental* predicate with an achievement level equal to 0.8. These predicates will allow us to deal with actions which can be performed in several steps by one robot or which can be performed simultaneously by several robots (actions with cumulative effects).

2.2 Social Rules

The cooperative activity is based on the common satisfaction of a set of constraints that we express in terms of what we have called *social rules*. *Social rules* have two main aims: to help the plan coordination, allowing coherence among the plans; and to enhance the expressive power of the world and action models. Social rules will impose constraints that will be taken into account during the planning and also during the plan validation and negotiation process[1].

We define three classes of rules:

- `time`: $(TIME\text{-}RULE\ pred, u, s)$, where *pred* predicate can be maintained true (without change) only during a specific `time` u. The s field is the *proposed state* to avoid the violation of the rule. For instance the rule: *X1 x-ray machine must be on at most 10 minutes* can be represented as $(TIME\text{-}RULE\ (STATE\_RAY(X1,ON)),10,(STATE\_RAY(X1,OFF)))$, where it is proposed to turn off $X1$ to avoid the rule violation.
- `amount`: $(AMOUNT\text{-}RULE\ (a : v), u, (s_a : s_v))$ where a attribute can assume v value only for u entities. As in the previous class, s field (attribute and value) is the *proposed state* to avoid violation of the rule. Note that with this kind of rule, it is possible to describe the resource constraints of the system. For instance *we can have only 2 robots on the L1 lift* can be represented as $(AMOUNT\text{-}RULE\ (POS\text{-}ROBOT:L1),2,(POS\text{-}ROBOT:OPEN\text{-}AREA)$, where it is proposed to send the robot to an OPEN-AREA, to respect the

[1] Note that this notion is somewhat different from the idea of social behaviors proposed by [14]. While the *social behaviors* are directly used by the robot, because they are explicitly coded in its reactive task execution, the *social rules* will be used at the robot decision level as constraints in the planning and cooperation activity.

rule.

• `end`: (*END–RULE pred*), where *pred* predicate must be achieved at the end of a plan execution. This class is used to set constraints that allows the robot planners to predict the end state of the attribute (initial state of the next planning). For example, the social rule: *DT1 drinks trolley must be in the refectory after use* can be represented as (*END–RULE* (*POS*(*DT*1,*REFECTORY*))).

The use of the social rules. The robots will always try to build plans that respect the rules by setting the *proposed states* as a component of their list of goals. However, there could be various levels of constraints. The rules may have an *obligation level* and, depending on it, the *proposed state* can be posted as a goal added to the current list of goals; or posted as a supplementary goal that should be satisfied in a second step. In such case, the robot will produce a main plan plus a set of *additional-plans* that satisfy the *social rules* with low obligation level. During the plan execution, the robot may or may defer, or even remove, these additional-plans, depending on the current state of the environment and on the requests of the other robots.

3 M+ Plan Mechanisms

Let P^k the current plan (a partially ordered set of actions $\mathcal{A}_i^p$) of a given robot R_p. Two time-points are associated to each action: $start(A_i^p)$ and $end(A_i^p)$ corresponding to the beginning and to the end of the action execution. Let us assume that P^k was the result of a previous validation process of robot R_p. This plan will be modified whenever R_p wants to add new actions (obtained after a call to its own planner) or whenever R_p receives cooperation requests (see section 4) from other robots R_q. We have defined the following *mechanisms* for plan modification:

• `insert_wait`: this mechanism adds a temporal order constraint to a plan. For example, $P^{k+1} = INSERT\_WAIT(A_i^p, A_j^q, R_q, P^k)$ imposes to R_p to execute A_i^p after R_q has performed A_j^q.

• `insert`: inserts a new action A_s into a plan after a given action A_i, $P^{k+1} = INSERT(A_s, A_i, P^k)$. It is used when a robot decides to add a unplanned action to its current plan (an opportunistic action or a new goal).

• `delete`: deletes A_d of the plan P^k, $P^{k+1} = DELETE(A_d, P^k)$. We use this mechanism when it is decided that another agent will be in charge of the action. The same mechanism is also used when an action execution is first deferred and finally deleted due to a low *obligation level* of a rule.

• `simult_exec`: it establishes a "communication channel" between two or several robots when they decide to execute simultaneously an action with cumulative effects. For example, when a robot R_p decide to execute A_i^p simultaneously with A_j^q performed by R_q, $P^{k+1} = SIMULT\_EXEC(A_i^p, A_j^q, P^k)$.

• `re-plan`: from state W and a goal G, it calls its planner and synthesizes a new plan, $P^{k+1} = REPLAN(W, G)$.

Moreover, we introduce two notions that are used by the robots in their cooperative activities. We define: 1. *interference predicates* as being all the predicates whose modifications can interfere in other robots' plan. *Interference predicates* are composed of *non-exclusive* predicates and of all predicates that belong to *social rules*. 2. *Contingent block* as a sub-plan which begins with an action that changes an *interference* predicate and which finishes with an action that changes again the value of the same predicate. By considering these mechanisms, the robots are able to change their plans, taking into account cooperative issues, and validating their actions in a multi-robot context.

4 The deliberation Process

In order to accomplish the deliberation process, let us assume that each robot processes the goals that it receives sequentially, taking as its initial state the final state of its current plan. By doing so, it incrementally appends new sequences of actions to its current plan. During task achievement, the robot tries to validate its next action. From the proposed semantics, the robot extracts relevant multi-robot context information from its plan. It negotiates the validation of the blocks of its action, aiming to respect previously validated plans, social rules and its task goal. This operation is "protected" by a mutual exclusion mechanism[2]. The result of this validation process is a new plan validated, and hence ready to be executed, in the multi-robot context.

4.1 Coordination and Cooperation

The deliberation process will allow the robots to *coordinate their plans in order to avoid resource conflict situations and also to* cooperate in order to enhance the global system performance.

Coordination Scenario. A robot R_q has to solve a coordination scenario when it has to perform a block that is *incompatible* with a block planned to be performed by another robot R_p. Two blocks are *incom-*

[2]Let us assume that we have a set of autonomous robots equipped with a reliable inter-robot communication device.

patible in two cases: 1. when they involve the same attribute and the same entity, or 2. when they may violate a *social rule*.

Cooperation Scenario. The search for possible cooperation that we propose involves, in the current version of the system, the detection of redundancies as well as opportunistic action allocation.

`redundancies`: this may happen when two actions (A_i^p, A_j^q) planned for two robots have the same *non-exclusive effects*. Depending on the action types, the actions can be allocated to only one robot (**redundant** actions) or to both robots (**cumulative** actions with incremental effects).

`opportunism`: this happens when a robot R_p can easily `insert` into its current plan an action A_j^q planned to be performed by another robot R_q.

4.2 The deliberation steps

The deliberation process comprises three steps: the **announcement**, the **offers analysis** and the **validation**.

Step 1: the announcement. Whenever a robot (e.g R_p) wants to validate an action A_i^p in the multi-robot context, it announces it by providing the block of actions that is associated with A_i^p.

After having received an action announcement, the other robots the other robots search for possible coordination and cooperation scenarios in their announced/validated plans, and send back their offers to R_p.

Step 2: R_p analyzes the offers. R_p brings together all received coordination and cooperation offers.

Coordination scenarios. This analysis is directly derived from the Plan-merging paradigm[1]. Action insertion is performed incrementally by adding temporal constraints to R_p plan[3].

Cooperation scenarios. In a cooperative scenario, R_p verifies which candidates are able to execute the cooperative blocks (composed of *Abc begin*, *Aec end* and *Acl causal link* actions). The robot builds a *cooperative final block*, choosing the agent(s) that will achieve *Abc* and *Aec*, and which *Acl* causal link actions will participate in the cooperation. Due to the need of respecting the *social rules*, it may occur that some robots can not participate in the *cooperative final block*.

Step 3: Validation. R_p informs the other about the result of the announcement process. The robots use the plan modification mechanisms to adapt their plans to the deliberation result. For plan coordination, the robots use only `insert-wait`. Concerning the cooperative interaction, the robots have two possibilities: 1. when R_q has an *accepted* offer, it uses the `insert-wait`,`delete` and `insert` mechanisms to adapt its plan to the cooperation, otherwise, 2. the robot has a *rejected* offer, so it must use `insert-wait` to coordinate its plan with the *cooperative final block*.

A block is "negotiable" until its **validation**. Once validated, block modifications are forbidden and the robot is ready to execute the action. The other robots can only perform insertions after a validated block.

Note that such a deliberation process involves only communication and computation and concerns future (short term) robot actions. It can run in parallel with execution of the current coordination plan.

5 Example

A first version of the scheme has been implemented. We describe an illustrative example of its use. The robots are in a hospital environment composed of open areas connected by doors, beds and unloading points.

`r0`, `r1` and `r3` are simulated mobile robots equipped with manipulators[4]. They must transfer objects and clean beds. Figure 2 shows the goals and the initial world state description. The example shows *M+ cooperative task achievement* where, from individual plans, each agent analyzes its actions searching for conflicts, redundancies and opportunistic situations that can be negotiated with the others.

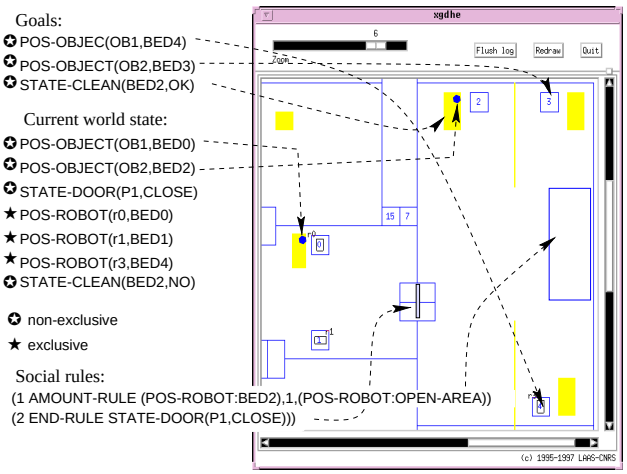

Figure 2: Example 1: Move objects and clean beds

[3]We use the mechanisms described in [12] for detecting and treating in a distributed way the deadlocks that may occur

[4]Each robot control system runs on an independent Sun workstation which communicates with the other workstations through TCP/IP.

The robots must respect the following *social rules*: 1. `amount` type rule: maximum one robot beside a bed, staying in an `OPEN-AREA` as a *proposed state* to respect the rule (low *obligation level*); 2. `end` type rule: the door must be closed (high *obligation level*).

After a first phase not described here, robots plan and allocate each goal using *M+ protocol* [3]. The final result of this phase is: `r0` allocates `(POS-OBJECT OB1 BED4)` as its main goal, `r1` allocates `(POS-OBJECT OB2 BED3)` and `r3` is in charge of `(STATE-CLEAN BED2 OK)`. Figure 3 shows each individual plan. Note that the plans consist of actions to achieve each goal and also planned actions that satisfy the *social rules* and which can be possibly neglected during the action execution. For instance, `r3` has two actions (`goto` and `clean`) to achieve `STATE-CLEAN(BED2,OK)` and a additional-plan (`goto OPEN-AREA`) to comply with rule 1. On the other hand, as rule 2 (door close) has a high obligation level, it is posted as a goal to be achieved together with the main goals. In this example we have the following coordination and cooperation issues: 1. to execute `pick-place, clean` actions a robot needs to go to a bed and to park beside it, but only one robot can dock at a time (rule 1) 2. `open/close` door is a cooperative redundant action (only *non-exclusive* effect: `STATE-DOOR(<door>, OPEN)`) and 3. `clean bed` is a cooperative incremental action (only *non-exclusive/incremental* effect: `STATE-CLEAN(<bed>,OK)`) that allows simultaneous execution by several robots.

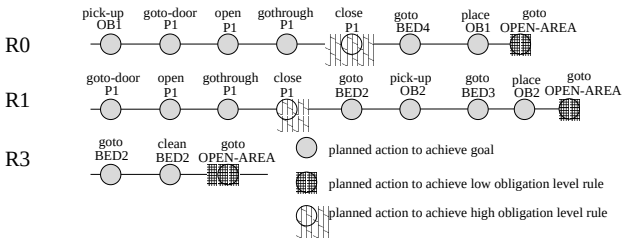

Figure 3: *M+ protocol* three mono plans.

During task achievement, each robot tries to validate its next action. `r0` begins the deliberation process of its first action `pick-up OB1`. We assume that this action has an *interference predicate* related to the position of `OB1`. It broadcasts its announcement, but as no robot has a concerned actions, `r0` does not receive any offer, it can then validate this action directly.

Similarly, `r1` tries to validate `goto-door`; this action does not involve any *interference predicates*. Thus, `r1` can validate it directly.

It is now `r3`'s turn to validate its first action. It verifies that `goto BED2` belongs to an *interference* block, due to the *social rule* associated to the bed dock limit. Thus, it builds up an announcement. `r1` presents a block in conflict (fifth action of its plan) but this block has not yet been announced, so `r1` does make any offer. Without any offer, `r3` gets its `goto BED2` announced.

Following the incremental deliberation process `r1` validates directly its `goto P1` action (it does not have *interference predicates*), and it takes its next action: `open-door deliberation`. Due to the absence of negotiated/validated plans, `r1` does not receive any offer.

`r0` begins soon afterwards its `open-door deliberation`. Concerning *social rule* 2, we can see that `open` action imposes an *end type* rule, where the door must be closed at the end of a plan. Thus, `open P1` and `close P1` compose a block that must be validated. Moreover, the *open/close* door actions have only *non-exclusive effects*, being cooperative actions. `r0` announces it, and the other robots analyze their current plans. `r1` has this block in its current negotiable plan, with a cooperative block too. `r1` formulates a *cooperative block offer*. From its block and the `r1` offer, `r0` finds the best combination between *begin/end* causal link actions (see section 4.2). The figure 4 shows the result of this deliberation process:

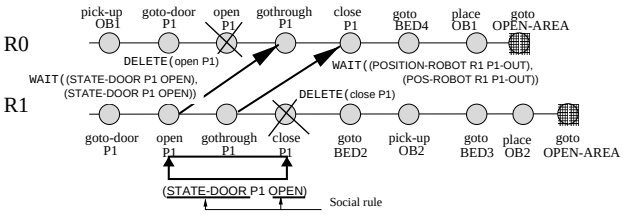

Figure 4: Redundant and cooperative actions.

Continuing the incremental deliberation, it is time for `r3` to validate its `clean-bed` action. Like `open`, `clean` is a cooperative action. Moreover it is an incremental action, since `STATE-CLEAN` is an incremental predicate. `r3` announces this action, but none of the other robots has this effect in their current actions. However, they try to cooperate, inserting this action into their plans. Since `r1` will be next to `BED2` in a future plan execution, it uses the *insert* mechanism, and it adds a `clean-bed` action after its arrival in the `BED2`. From this, `r1` sends an opportunistic offer to `r3`. `r3` analyzes the offer, taking into account that it is an *incremental* action, each robot executes a part of the action according to this delay level.

The robots validate their actions successively. It is now `r1` turn to announce its `goto BED2` action. As the `goto OPEN-AREA` action of `r3` finishes before the

end contingent block action of `r1` (`goto BED3`), and nobody has yet validated its actions, `r3` will therefore execute this block before `r1`.

Incrementally, robots validate their actions. Figure 5 shows the final result of the *M+ task achievement*. Plans are executed coherently; resources conflicts are treated correctly. Besides, the robots exhibit some cooperative behaviors: `r1` opens the door for `r0` and it also helps `r3` to clean the bed. `r0` closes the door for both of them. `r1` illustrates another coordinated behavior: once it waits until `r3` leaves `BED2`. Note that `r0` and `r1` have neglected the execution of their `goto OPEN-AREA` *end contingent block* action, since no robot has requested them to do so.

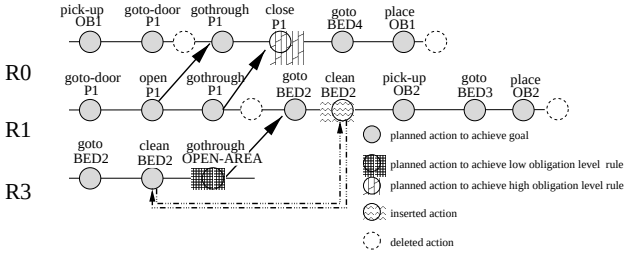

Figure 5: Coordinated and cooperative final plans

6 Conclusion

We have proposed a cooperative multi-robot task achievement scheme. The robots detect and solve various coordination issues. Besides, they exhibit effective cooperation abilities that allow them to enhance the global system performance. We have developed semantics for action and plan description; as well as *mechanisms* that allow the robots to incrementally adapt their plans to the multi-robot context while preserving a coherent behavior of the global system. A first version of the system has already been implemented and tested in simulation. Our future work will be to improve its implementation, to increase its reasoning capabilities which take into account other classes of agents; and to incorporate it on real robots.

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

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] S. S. C. Botelho. A distributed scheme for task planning and negotiation in multi-robot systems. In *ECAI'98*, 1998.

[3] S. S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE ICRA'99*, 1999.

[4] A. Brumitt, B. Stentz. Dynamic mission planning for multiple mobile robots. In *IEEE ICRA'96*, 1996.

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

[6] O. Despouys and F Ingrand. Propice-plan:toward a unified framework for planning and execution. In *ECP'99*, 1999.

[7] E. Durfee and V. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Trans. on Sys., Man and Cybernetics*, 21(5), 1991.

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

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

[10] P. Laborie. *IxTeT: une approche integrée pour la Gestion de Ressources et la Synthèse de Plans.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1995.

[11] L. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2):220–239, 1998.

[12] S. Qutub, R. Alami, and F Ingrand. How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *IEEE IROS'97*, 1997.

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

[14] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, (75):231–252, 1995.

[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] S. Yuta and S. Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE IROS'92*, 1992.