Multi-robot cooperation through the common use of "mechanisms"

Silvia Silva da Costa Botelho, Rachid Alami

Multi-robot cooperation through the common use of "mechanisms"

S.C. Botelho, R. Alami
FURG
Av. Italia Km8, 96201-000 Rio Grande/RS - Brazil
LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France

Abstract

In this paper we propose a new contribution to treat a class of cooperative issues in the multi-robot context. These issues are associated with the common use of some entities, called mechanisms*, by several robots. A* mechanism *can be seen as a generalization of the notion of resource. The robots can modify its state directly or through requests. The robots can also share its utilization. Multi-robot cooperation will be expressed as a distributed decisional process that tends to solve detect and treat resource conflict situations as well as sources of inefficiency. We discuss these issued and illustrate them through a simulated system, which allows a number of autonomous robots to plan and perform cooperatively a set of servicing tasks in a hospital environment.*

1 Introduction

Starting from the **Plan-Merging** Paradigm [1] for coordinated resource utilization - and the **M+ Negotiation for Task Allocation - M+NTA** [4, 3] for distributed task allocation, we have developed a generic architecture for multi-robot cooperation [5, 2]. This architecture is based on a combination of local individual planning and coordinated decision for incremental plan adaptation to the multi-robot context. In this paper we present a method developed to detect and treat conflict and cooperative situations due to common utilization of some entities. The *mechanism* concept allows a set of autonomous agents not only to perform their tasks in a coherent and non-conflict manner but also to cooperatively enhance their task achievement performance.

After a brief analysis of related work, we present a general architecture for multi-robot cooperation. We introduce the *mechanisms* and focus on the cooperative plan enhancement issues. Finally, we present an implemented system which illustrates, in simulation, the key aspects of our contribution.

2 Related work

The field of multi-robot systems has become a large field[9]. We restrict our analysis here to contributions proposing cooperative schemes at the architectural and/or decisional level. In such stream, *behavior-based* and similar approaches [17], [16], propose to build sophisticated multi-robot cooperation through the combination of simple (but robust) interaction behaviors. ALLIANCE [18] is a distributed behavior based architecture, which uses mathematically modeled motivations that enable/inhibit behaviors, resulting in tasks (re)allocation and (re)decomposition.

AI-based cooperative systems have proposed to provide models for the agents interaction which are domain independent. For example, Brafman [7]/Ephrati [14] enrich the STRIPS formalism, aiming to build centralized/decentralized conflict-free plans. Clement [10] develops specialized agents which are responsible for HTN individual plans coordination.

Several generic approaches have been proposed concerning goal decomposition, task allocation and negotiation [12]. PGP [13] (and later GPGP [11]) is a specialized mission representation that allows exchanges of plans among the agents. DIPART [19] is a scheme for task (re)allocation based on load balancing. Cooperation has also been treated through negotiation strategies [20] like CNP-based protocols[22], or BDI approaches where agents interaction is based on their commitment to achieve individual/collective goals ([15],[23]). Another perspective is based on the elaboration of conventions and/or rules. Shoham [21] proposed "social behaviors" as a way to program multi-agent systems. In STEAM [24], coordination rules are designed in order to facilitate the cohesion of the group.

Cooperation for achieving independent goals has been mostly addressed in the framework of application-specific techniques such as multi-robot cooperative

navigation [26, 8].

3 Cooperation for Plan Enhancement

In the context of autonomous multi-robot systems, we identify three main steps that can often be treated separately: the *decomposition* of a mission into tasks (mission planning), the *allocation* of tasks among the available robots and the *tasks achievement* in a multi-robot context (Fig. 1). In this paper, we limit ourselves to this last aspect i.e. the concurrent achievement of a set of tasks. Indeed, we assume a set of autonomous robots which have been given a set of partially ordered tasks. This could be the output of a central planner [25], or the result of a collaborative planning and task allocation process [4]. One can consider this plan elaboration process is finished when the obtained tasks have a sufficient range and are sufficiently independent to allow a substantial "selfish" robot activity.

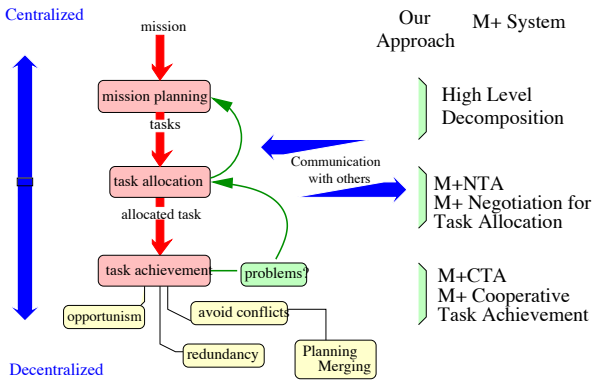

Figure 1: Our architecture for multi-robot cooperation

However, and this is a key aspect in robotics, the allocated tasks cannot be directly "executed" but require further refinement taking into account the execution context. Since each robot synthesizes its own detailed plan, we identify two classes of problems related to the distributed nature of the system: 1. coordination to avoid and/or solve conflicts and 2. cooperation to enhance the efficiency of the system. The first class has been often treated in the literature. The second class is newer and raises some interesting cooperative issues linked to the improvement of the global performance by detecting possible enhancements.

We have developed a distributed cooperative scheme[6] called *M+ cooperative task achievement* - `M+CTA`. It is based on the *mechanism* concept and provides a framework for dealing with issues such as:

1. **opportunistic action re-allocation**: one robot can opportunistically detect that it will be beneficial for the global performance if it could perform an action that was originally planned by another robot;
2. **detection and suppression of redundancy**: it may happen that various robots have planned actions which achieve the same world state. This feature provides the reasoning capabilities that allow them to decide when and which robot will achieve them, avoiding redundant executions;
3. **incremental/additive actions**: the robots detect that an action originally planned by one robot can be incrementally achieved by several robots with a "cumulative" effect and that this could be beneficial to the global performance.

4 The *mechanism* concept

In `M+CTA`. the task achievement level is based on an incremental plan validation process. Starting from a task that has been allocated to it, a robot, R_p, plans its own sequence of actions, called *individual plan*. This plan is produced without taking into account the other robots' plan. After this planning step, R_p negotiates with the other robots in order to incrementally adapt its plan in the multi-robot context.

A number of conflict/cooperative situation problems are raised when a group of agents share the common use of some entities or devices in the environment.

The *mechanisms* provide a suitable framework for robot cooperation. Indeed, there are numerous applications and particularly for servicing tasks, where the robots often need to operate or to interact with automatic machines or passive devices in order to reach their goals or to satisfy some intermediate sub-goals that allow them to finally reach their main goals.

For example, a robot has to open a door in order to enter a room, or heat the oven to a given temperature before cooking a cake, etc..

The *mechanism* can be seen as an extension of the concept of resource token: a robot not only allocates and frees a mechanism, it not only consumes or produces it, it can also explicitly manipulate it or act on it, directly or through requests to a controller attached to the mechanism.

The simplest entity that will be dealt with through a *mechanism* is a spatial resource that can be used by only one robot at a time: a place where to park. A door is a little more sophisticated. It may have several states, it may been open or closed, or open to a certain extent. A door can be automatic or manual. Besides, depending on the context, a door should be maintained closed as much as possible or not. Note also that there often exist procedures to operate some machines with several steps and rules to share the use of a machine. An interesting example is the elevator.

The *mechanisms* will allow: 1. to identify the entities of common use, 2. to fix rules to guarantee correct and coherent cooperative *utilization* of such entities and 3. to negotiate their common use among the agents.

4.1 A scenario of cooperation

A *mechanism* is a data structure that defines how to use a device or a machine. It defines, somehow, the instructions (or directions) for use: the possible sequences of operations, in what conditions it can be shared or used simultaneously by several users, etc..

A mechanism is a data structure that defines how to use a device or a machine. It defines, somehow, the instructions (or directions) for use: the possible sequences of operations, in what conditions it can be shared or used simultaneously by several users, etc..

In the current version of our system, this knowledge is represented by:

- known initial and final states,
- a set of alternative *paths*; each path is partially instantiated and represents a valid sequence of actions and state changes of the associated entity.
- a set of *social rules*.

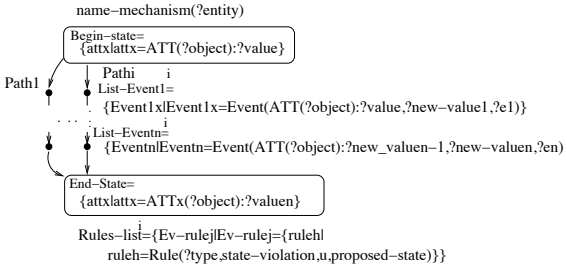

Figure 2: A generic mechanism M

Social Rules impose constraints that must be taken into account during the *mechanisms* use. They have been introduced in order to allow the robots produce easily *merge-able* plans. Social rules are always associated with some mechanism states, which, in particular situations, are not allowed. Social rules specify forbidden or undesirable states and propose states that satisfy the rules. This field is used by the planner in order to avoid the violation of the rule. Thus, social rules have the following generic description:

RULE($type, violation\_state, s, proposed\_state$)

Social rules are domain dependent; the current version of our system deals with three types of constraints:

1. `amount`: where the "resource" $violation\_state = (att(?object) : v)$ represented by an attribute att and a value v is limited to a maximum number of s agents. Note that such rules allow to describe the resource constraints of the system. For instance *a limitation of 2 robots at desk D1* can be represented by RULE(`amount`,$pos-robot(?r)$: $D1$,2,$pos-robot(?r)$:`OPEN_AREA`), where it is proposed to send the robot to an `OPEN_AREA`, in order to satisfy the rule.
2. `end`: where $proposed\_state$ must be satisfied at the end of each robot utilization of the resource. This class guarantees a known final state, allowing the planner to predict the state of an attribute (initial state for the next plan).
3. `time`: where $violated\_state$ can be maintained true only during a given amount of time s.

The use of social rules in the planning phase: We associate to the social rules a scalar value called *obligation level*. Whenever a robot plans, it considers the *proposed states* of the rules as mandatory goals that will be added to its current list of goals. However, depending on the obligation level, goals will be posted 1. as a conjunction with the current robot goals or 2. as additional goals that the robot will try to satisfy in subsequent planning steps. In such case, the planner will produce *additional plans* that will achieve each low-level obligation social rule.

During the execution of a plan, the robot may or may not execute these additional plans, thus neglecting temporarily the *proposed state*. Note that if another agent asks the robot to fulfill the *rule proposed state*, it will then (and only then) perform the associated

additional plan. The *obligation level* may also change depending on the context[1].

4.2 *Mechanisms* and *Jobs*

Whenever a robot R_p detects that its plan uses an entity associated with a mechanism M, it builds a *job* M_j^p. A *job* is a dynamic structure, which results from the instantiation of a *path* of a given mechanism by the current robot plan. A job is composed of *steps*. Each *step* has a set of information associated with it: for instance, the agent that effectively executes the action, the other plan actions that depend on it (`successors`), etc. *Jobs* are used as structure and language of negotiation allowing R_p and other agents to decide about the common utilization of an entity. Figure 3 shows a plan produced the robot R_p that uses a furnace. R_p builds a job M_j^p and that will be negotiated. This *job* ends when the final state of the associated mechanism is reached.

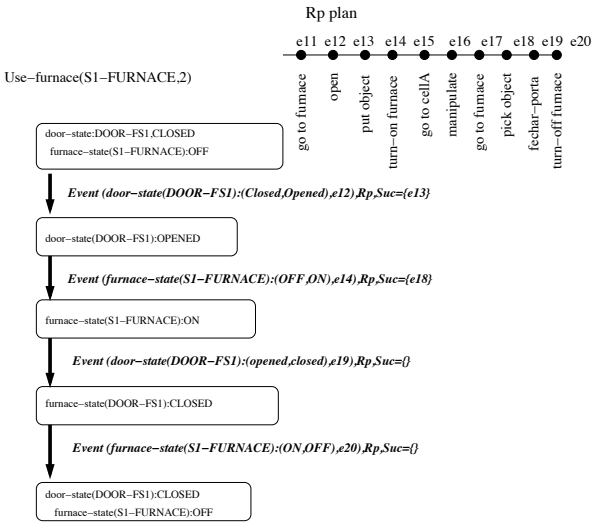

Figure 3: A *job* corresponding to the use of a furnace by *Rp*

[1]Note that this notion of social rules is different, or even complementary, from the social behaviors proposed by [21]. While *social behaviors* are explicitly coded in its reactive task execution, the *social rules* are used at the robot decision level as constraints in its planning, negotiation and execution activities.

5 Cooperation based on *mechanisms*

The `M+CTA` level involves three activities that correspond to different temporal horizons and may run in parallel: 1. task planning which produces an individual robot plan; 2. the plan negotiation activity which adapts the plan to the multi-robot context; and 3. the effective plan execution.

From time to time, depending on higher level requirements, the robot invokes its own planner and it incrementally appends new sequences of actions to its current individual plan. This is a standard task planning activity; however, the obtained plan satisfies the social rules and is consequently easily *merge-able*.

Incremental plan negotiation

Let us assume that R_p has an individual plan composed of a set of actions A_i^p which manipulate mechanisms. It performs an incremental negotiation process in order to introduce each action A_i^p in the multi-robots context. This operation is "protected" by a mutual exclusion mechanism[2]. The result is a coherent plan which includes all the necessary coordinations and some cooperative actions. It is default free and can be directly executed. However, it remains "negotiable" (other robots can propose a plan modification) until it is incrementally "frozen" in order to be executed. We analyze here below the different steps involved in this negotiation process.

The negotiation steps:

The negotiation process comprises two steps: the **announcement** and the **deliberation**. During this process, a robot negotiates a set of *jobs* of its current plan, which are not yet announced[3].

Step 1: the announcement. Whenever a robot,R_p needs to validate an action A_i^p (belonging to *job* M_j^p. R_p corresponding to the use of a mechanism M), in the multi-robot context, it announces its will to negotiate a job involving M. It obtains the current list of jobs involving M.

Step 2: R_p deliberates. Having the current job list, R_p has two alternatives associated with its job M_j^p and each member list M_j^q, see figure 4:

[2]We assume that the robots are equipped with a reliable inter-robot communication device.

[3]We treat together all jobs "entwined" to avoid deadlock situations.

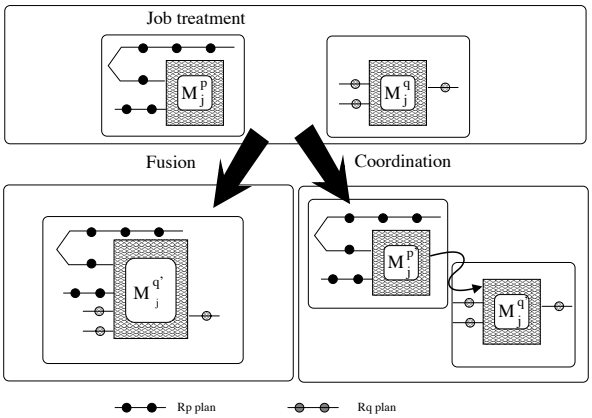

Figure 4: Job treatment possibilities: fusion or coordination.

Fusion: since our robots are cooperative, the aim is to enhance as much as possible the overall performance. Thus, the robot always try to merge his *job* with the current (already negotiated) *jobs* M_j^q. This is done by trying to detect and suppress redundant transitions. The result is a new job $M_j^{q\prime}$ whose actions may be distributed between the different robots.

However, the constraints imposed by *social rules* may prevent a fusion between two jobs. The only remaining solution is to coordinate them in order to avoid conflicts.

Coordination: in this situation R_p can use a mechanism M only after its released by the agents associated with M_j^q. In other words, M_j^p has to be coordinated with M_j^q by adding temporal constraints to the jobs.

After each deliberation process, the robots to adapt their plans to the job modification. We have defined the following operations:
`insert_message_wait` that introduces a temporal order constraint between two actions belonging two robots, and `insert/delete`, when an action is reassigned to another robot or when an action execution is neglected.

Note that such a negotiation 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.

Job execution process:

Before executing an action A_i^p, the robot **validates** the transition associated to A_i^p. Indeed, a transition remains "negotiable" until its **validation**. Once validated, it is "frozen" and the other robots can only perform insertions after a validated transition. Action execution causes the evolution of the system, resulting in events that will entail new planning, negotiation and execution steps for the robot itself and for the other robots.

6 Illustration and future work

We have implemented a first version of the overall system and run it on simulation. We describe here below some of the obtained results. The application domain that we have chosen is a set of mobile robots in a hospital environment. Servicing tasks are items delivery to beds as well as bed cleaning and room preparation[4]. Fig. 5 shows the simulated environment and 14 partially ordered tasks: `T0,...T13` and the initial world state description[5]. Each robot is equipped with a STRIPS-based task planner and a motion planner.

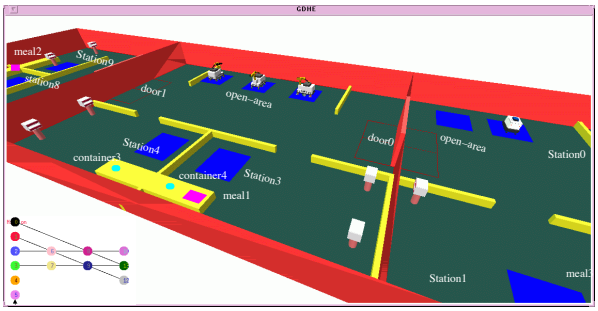

Figure 5: Example 1: Transfer object and clean beds in a hospital area

The robots must negotiate the use of the following *mechanisms*, see figure 6: 1. `clean-room` that allows cleaning actions with cumulative effects when executed several times or by several robots; 2) door-manipulation with `open/close` actions, which can be potentially redundant; and 3) a mechanism that controls the use the dock station by the robots. This mechanism has an `amount` rule (with low *obligation level*) that limits the number of robots near a station to one.

The set of tasks is transmitted to five robots. After a first phase (not described here [4]), they plan

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

[5]Due to the lack of space, we exhibit here a simplified world state representation.

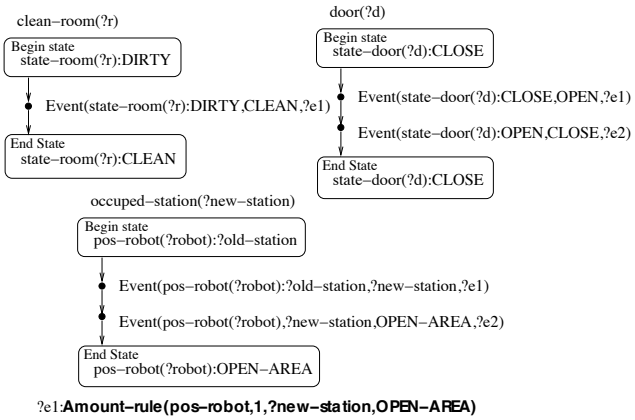

Figure 6: *Mechanisms* to negotiate.

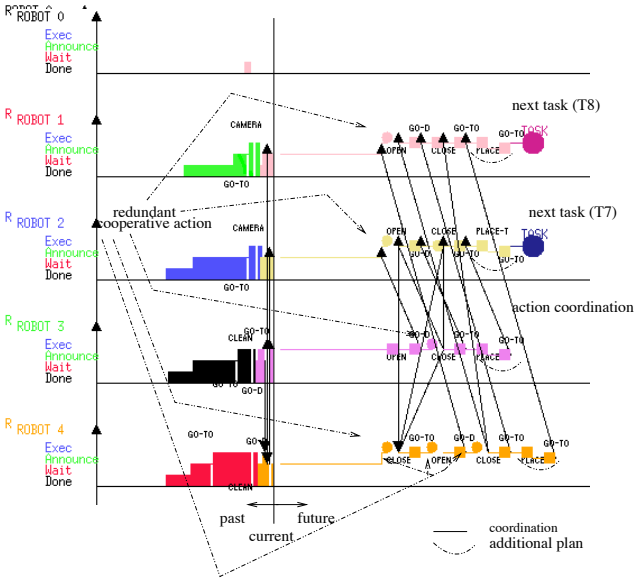

Figure 7: M+ task achievement process.

and incrementally allocate the tasks using *M+ Cooperative Task Allocation*. Fig. 7 shows the individual plans after a number of negotiation processes. Note that `r0` has allocated `T6` in a first step. However it has lost it because `r1` has found a better cost to achieve it. `r1` is achieving `T3`. It has elaborated a plan with six actions in order to achieve its main goal `state-room(S1):CLEAN` and to satisfy the social rule requiring `state-door(D0):CLOSED` with a high obligation level. Besides, it has also produced an *additional plan* that satisfies rule 1 (with a low obligation level) by introducing a `go-to(OPEN-AREA)` action. After several *jobs* negotiation processes, `r1` deletes its `open` action, which will be accomplished by `r3`. This robot will open a first time the door and after all robots take in advantage of this event. Afterwards, `r1` will close the door for everybody. We can see also the incremental allocation process: while the robots are achieving their current tasks, they try to allocate their future task, for instance: `r1-T6` and `r2-T9`. The arrows between robot plans illustrate the temporal constraints induced by the coordination between jobs.

The overall process continues; the tasks are incrementally planned, negotiated and executed. Fig. 8 shows the final result of this run. One can notice, that the robots have satisfied the *social rule* associated to the robot position near the stations. Indeed, some robots delete redundant actions (open/close door), accomplished opportunistically by others. Besides, some robots also helped the others to clean rooms.

Table 9 shows the time sharing among execution and deliberation activities. Deliberation activities are decomposed into task allocation and *mechanisms* negotiation. All activities run in parallel. Note that execution activities are more expensive, however `r0` has a high task allocation activity due to the mission nature and to its proper context: the tasks order limits their execution in parallel and `r0` spends a lot of time searching for a task to perform.

We have run the system several times with different parameter values. These parameters are associated with two aspects: the type of cooperation and the number of robots. We have run the system with three different cooperation strategies: 1. `COOP-TOTAL`: treating redundancy and opportunistic incremental help between *jobs* ; 2. `NO-INC`: only treating redundant cases with no incremental help; and 3. `NO-COOP`: the system allows only coordination between *jobs*.

On the whole, `COOP-TOTAL` enhance the system performance: better cost and less actions (see Figs; 10 and 11).

When we change the number of robots, we can see, table 12 that the number of achieved actions with 5 and 3 robots is smaller than with 2 robots. Note that there is no difference between 3 and 5 robots tests; this is due to the nature of mission. The partial order of tasks prevents an optimum deployment of more than 3 robots.

The number of robots vs. the workload is presented in table 13. We can see that when we have 5 robots, one of them (`r0`) is almost idle. This fact explains the

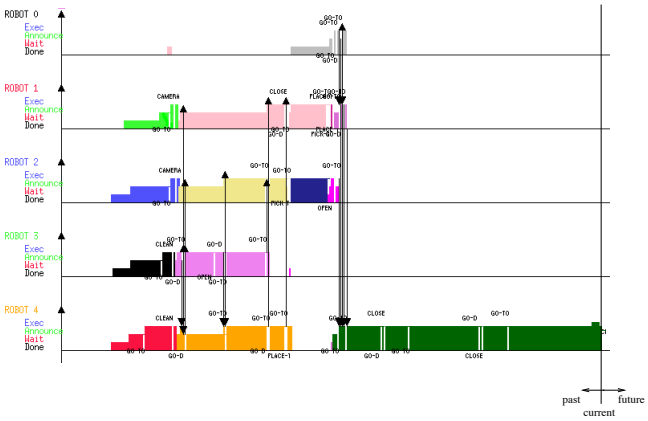

Figure 8: The final result of the run.

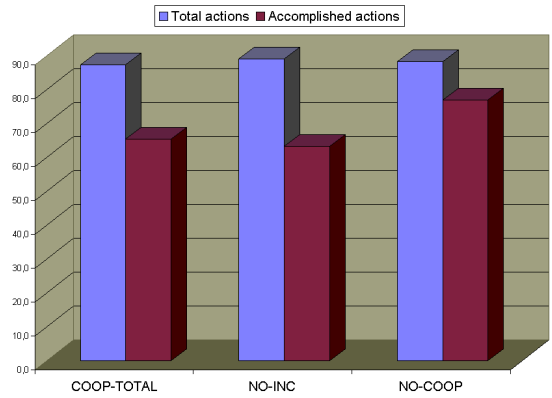

Figure 10: Planned and achieved action by the robots.

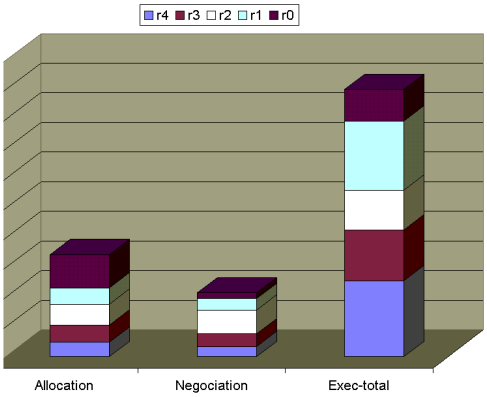

Figure 9: Time Sharing.

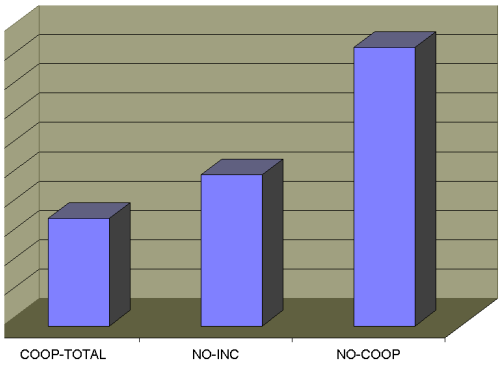

Figure 11: The costs.

similar results between 3 or 5 robots tests. However, note that our system has found a very good balance when only three robots are involved.

Conclusion:

We have proposed and discussed a scheme for cooperative multi-robot task achievement based on *mechanism*. This scheme is a key component of a general architecture for multi-robot cooperation. Its main originality comes from its ability to allow the robots to detect and treat - in a distributed and cooperative manner - resource conflict situations as well as sources of inefficiency among the robots. We have presented its main ingredients and operators, and illustrated its use through a simulated system.

We intend to validate our approach through a number of significant different application domains. Besides, we would like to extend and further formalize the overall system and its representational and algorithmic ingredients, taking into account cost and time issues to help planning and negotiation activities.

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. Botelho. *Une architecture dcisionnele pour la coopration multi-robots*. PhD thesis, LAAS-CNRS, 2000.

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

[4] 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.

[5] S. S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and

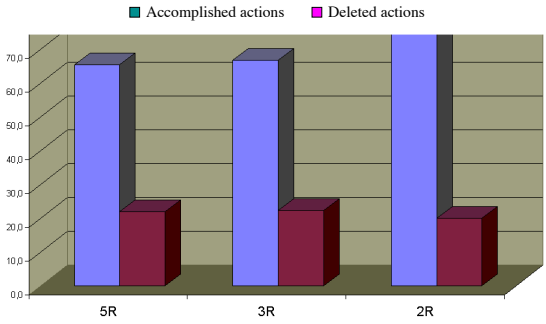

Figure 12: Number of robots vs. planned and achieved actions.

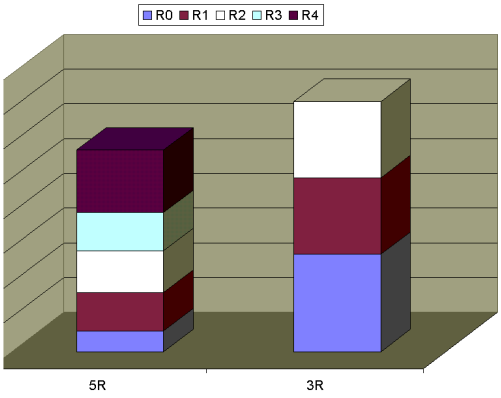

Figure 13: Workload for each robot.

achievement. In *IEEE Int. Conf. on Robotics and Automation (ICRA'2000)*, 2000.

[6] S. S. C. Botelho and R. Alami. Robots that cooperatively enhance their plans. In *Distributed Autonomous Robotic Systems - DARS2000*, 2000.

[7] C. Boutilier and Brafman R. Planning with concurrent interaction actions. In *AAAI'97*, 1997.

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

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

[10] B. Clement and E. Durfee. Top-down search for coordinating the hierarchical plans of multiple agents. In *Third International Conference on Autonomous Agents*, pages 252–259. Association of Computing Machinery, 1999.

[11] K. Decker and V. Lesser. Generalizing the partial global planning algorithm. In *Int Journal of Cooperative Information Systems 92*, 1992.

[12] M. DesJardins, E. Durfee, Ortiz C., and M. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, pages 13–22, 1999.

[13] E. Durfee and V. Lesser. Using partial global plans to coordinate distributed problem solvers. In *IJCAI87*, 1987.

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

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

[16] D. Mackenzie and R. Arkin. Multiagent mission and execution. *Autonomous Robots*, 4:29–52, 1997.

[17] M. Mataric. *Interaction and Intelligent Behavior*. PhD thesis, Massachusetts Institute of Technology, 1994.

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

[19] M.E. Pollack. Planning in dynamic enviroments: the dipart system. *Advanced Planning Technology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative*, 1996.

[20] J. S. Rosenschein and G Zlotkin. Rules of and encounter: Designing convention for automated negotiation among computers. *Artificial Intelligence - MIT press*, 1994.

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

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

[23] G Sullivan, A. Glass, B. Grosz, and S. Kraus. Intention reconciliation in the context of teamwork: an initial empirical investigation. *Cooperative Information Agents III, Lecture Notes in Artificial Intelligence*, 1652:138–151, 1999.

[24] M. Tambe. Agent architectures for flexible, practical teamwork. In *First International Conference on Autonomous Agents*, 1998.

[25] T. Vidal, M. Ghallab, and R. Alami. Incremental mission allocation to a large team of robots. In *IEEE International Conference on Robotics and Automation*, April 1996.

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