Robots that cooperatively enhance their plans

Silvia Silva da Costa Botelho, Rachid Alami

Robots that cooperatively enhance their plans

Silvia Botelho and Rachid Alami

LAAS-CNRS - 7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4 - France

Abstract. This paper discusses briefly our general architecture for multi-robot cooperation and then focuses on a scheme called "M+ Cooperative task achievement". 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 present its main ingredients and mechanisms, and illustrate its use through a simulated system, which allows a number of robots to plan and perform cooperatively a set of servicing tasks in a hospital environment.

1 Introduction

In previous contributions, we have treated a number of problems related to multi-robot operation and cooperation. Starting from the **Plan-Merging** Paradigm [1] - and its implementation for coordinated resource utilization - and the **M+ protocol** [3,2] for distributed task allocation, we have developed a generic architecture for multi-robot cooperation. This architecture involves a task achievement scheme which is essentially based on on-line combination of local individual planning and coordinated decision for incremental plan adaptation to the multi-robot context.

We present and discuss here a set of cooperation issues which allow a set of autonomous robots not only to perform their tasks in a coherent and non-conflict manner but also to cooperatively enhance their task performance.

We begin with a brief analysis of related work. Section 3 discusses briefly our general architecture for multi-robot cooperation and defines informally our cooperative task achievement scheme. Section 4 describes the main ingredients that we use in order to perform cooperative plan enhancements. In section 5, we describe the task achievement process and focus on its negotiation component. Finally, section 6 describes an implemented system which illustrates, in simulation, the key aspects of our contribution.

2 Related work

In the last decade, several studies have been done concerning the field of multi-robot systems [6]. We restrict our analysis here to contributions proposing cooperative schemes at the architectural and/or decisional level.

We can cite the *behavior-based* and similar approaches[17], [16], that 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 modelled 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 [4]/Ephrati [12] enrich the STRIPS formalism, aiming to build centralized/decentralized conflict-free plans. Clement [7] develops specialized agents which are responsible for HTN individual plans coordination.

Several generic approaches have been proposed concerning goal decomposition, task allocation and negotiation [9]. PGP [11] (and later GPGP [8]) 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 [21] like CNP-based protocols[23], or BDI approaches where agents compromise to achieve the individual/collective goals ([13],[14],[24]).

Another perspective is based on the elaboration of conventions and/or rules. Shoham [22] proposed "social behaviors" as a way to program multi-agent systems. In STEAM [25], 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 [27,5].

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 the obtained tasks among the available robots and the *tasks achievement* in a multi-robot context (Figure 1).

In this paper, we limit ourselves to this last aspect i.e. the concurrent achievement of a set of tasks by a number of robots. Indeed, we will 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 [26], or the result of a collaborative planning and task allocation process [3]. One can consider this plan elaboration process finishes when the obtained tasks have a sufficient range and are sufficiently independent to cause a substantial "selfish" robot activity.

However, and this is a key aspect in robotics, the allocated tasks cannot be directly "executed" but require further refinement, because the robots act

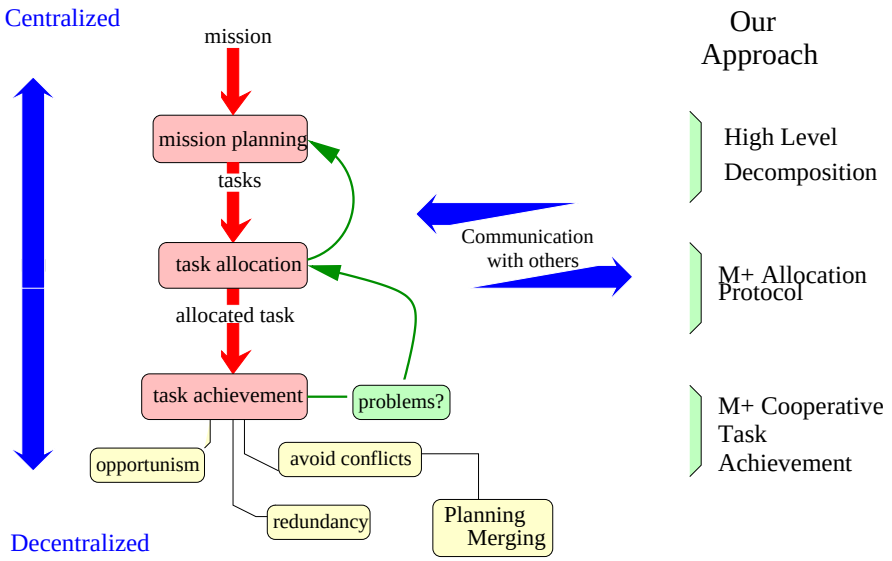

Fig. 1. An Architecture for multi-robot cooperation

in the same physical environment and because of the multiplicity of uncertainties. Since each robot synthesizes its own detailed plan for achieving its allocated task, 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 scheme, called *M+ cooperative task achievement*, which partially answers these questions by considering three features:

- **opportunistic action re-allocation**: during its own task execution, 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;
- **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 the robots to decide when and which robot will achieve them, avoiding redundant executions;
- **incremental/additive actions**: this feature allows the robots to detect that an action originally planned by one robot can be incrementally achieved by several robots and that this could be beneficial to the global performance.

In the next section, we describe its main ingredients: a world description, a set of *social rules*, and their use in a cooperative decisional process based on incremental planning as well as on a set of mechanisms for plan adaptation.

4 Mechanisms for Plan Enhancement

The world model we use 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 invoke efficient practical planners (in our implementation, we use PROPICE-PLAN [10] with a STRIPS-like IPP [15] planner).

4.1 A state description

The world state is described through a set of predicates. We have two kinds of predicates: 1) *stable* predicates which represent constant environment features (e.g. `CONNECTED(A1):A2`, `CONNECTED(A1):A3`) and 2) *evolutive* predicates which represent features that can be changed and whose modification can be planned (e.g. `GRIPPER(R1): EMPTY`). Besides, for a given robot, there is a subset of *evolutive* predicates - called *exclusive* predicates - that can only be changed by the robot itself (e.g. `POSITION-ROBOT`).

4.2 The incremental validation process

The *M+ task achievement* scheme is based on independent planning capabilities together with a set of cooperative mechanisms. Starting from the task that have been allocated to it, a robot produces its own plans called *individual plans*. It must then negotiate with the other robots in order to incrementally adapt its actions in the multi-robot context.

4.3 Social Rules

This cooperative activity is based on the common satisfaction of a set of constraints expressed in terms of what we have called *social rules*. *Social rules* have been introduced in order to produce easily *merge-able* plans. Besides, they allow to enrich the features description of the environment. They impose constraints that must be taken into account during the planning and also during the validation process[1].

We define three classes of rules[2]:

[1] Note that this notion is different, or even complementary, from the social behaviors proposed by [22]. 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 and negotiation activity.

[2] It is possible to have other classes of rules related to the application domain.

- `time`: $(TIME\text{–}RULE\ pred, u, s)$, where *pred* predicate can be maintained true only during a given amount of time `time` u. The s field is a *proposed state* which can be used by the planner in order to avoid the violation of the rule. For instance the rule: *Machine M1 must be ON at most 10 minutes* can be represented as $(TIME\text{–}RULE\ (STATE(M1,ON)),10,(STATE(M1,OFF)))$, where it is proposed to turn off $M1$ to avoid the rule violation.
- `amount`: $(AMOUNT\text{–}RULE\ (a:v), u, (s_a:s_v))$ where the "resource" $(a:v)$ represented by an attribute a and a value v is limited to a maximum of u entities. As in the previous class, the s field (attribute and value) is the *proposed state* to avoid the rule violation. 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 $(AMOUNT\text{–}RULE\ (POS\text{–}ROBOT{:}D1),2,(POS\text{–}ROBOT{:}OPEN\text{–}AREA)$, where it is proposed to send the robot to an $OPEN\text{–}AREA$, in order to satisfy the rule.
- `end`: $(END\text{–}RULE\ pred)$, where *pred* predicate must be satisfied at the end of each robot activity. This class guarantees a known end state, allowing the planner to predict the final state of an attribute (initial state of the next planning). For example, the social rule: *D1 door must be closed* can be represented as $(END\text{–}RULE\ (STATE\text{–}DOOR(D1,CLOSED)))$

The use of social rules in the planning phase: We associate to the social rules a scalar called *obligation level*. This parameter helps to distinguish between rules that must be systematically respected in order to obtain merge-able plans while the satisfaction of some other rules can be deferred i.e planned but not necessarily executed.

Whenever a robot plans, it considers all the *proposed final states* of the rules as mandatory goals that will be added to its list of current goals. However, depending on the rules obligation level, their proposed state can be posted 1. as a conjunction with the current robot goals or 2. as additional goals that the robot will try to satisfy in a 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 remove these additional plans, thus neglecting the *proposed state* and "violating" a social rule. Note that if another agent asks the robot to fulfill the *rule proposed state*, it will then (an only then) perform the associated additional plan. The *obligation level* may change depending on the context.

4.4 Operations on plans

Let P_k^p be the plan which was the result of the last validation process of robot R_p. P_k^p consists of a set of partially ordered actions $\mathcal{A}_k^p$. 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 from another robot R_q.

We have defined the following *mechanisms* for plan modification:
`insert_message_wait`: this mechanism introduces a new temporal order constraint between two actions belonging two robots
`insert`: inserts a new action As in the current plan.
`delete`: deletes an action, $A_d \in \mathcal{A}_k^p$, of P_k^p plan. We use this mechanism when an action is re-assigned to another robot or when an action execution is neglected, due to a low *obligation level* of a rule;
`replan`: from state W and a goal G, it calls its planner and finds a new plan.

Moreover, we introduce new 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 with the other robots plans. *Interference predicates* are composed of *non-exclusive* predicates and of all predicates that belong to *social rules*. 2. *Block of actions* 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 concepts and mechanisms, the robots are able to change their plans, taking into account cooperative issues, and validating their actions in a multi-robot context.

5 The task achievement process

The M+ task achievement process involves three activities (Figure 2): 1. the task planning which produces a mono robot "merge-able" plan; 2. the plan negotiation activity which adapts the plan to the multi-robot context; and 3. the effective plan execution.

These three activities correspond to different temporal horizons and may run in parallel (Figure 3). While task planning is a purely internal activity, the other activities are performed in a critical section in order to ensure a coherent distributed multi-robot plan management and execution.

5.1 The task planning activity

This is a standard task planning activity. The robot invokes its own planner. It takes as initial state the final state of its current plan. By doing so, it incrementally appends new sequences of actions to its current plan. This new plan does not consider explicitly the other robots' plans. We call it a *mono-plan*. However, the obtained plan satisfies the social rules (see §4.3) and is consequently easily *merge-able*.

5.2 The plan negotiation

The negotiation process allows the robots to *coordinate* their plans in order to avoid resource conflict situations and also to *cooperate* in order to enhance the

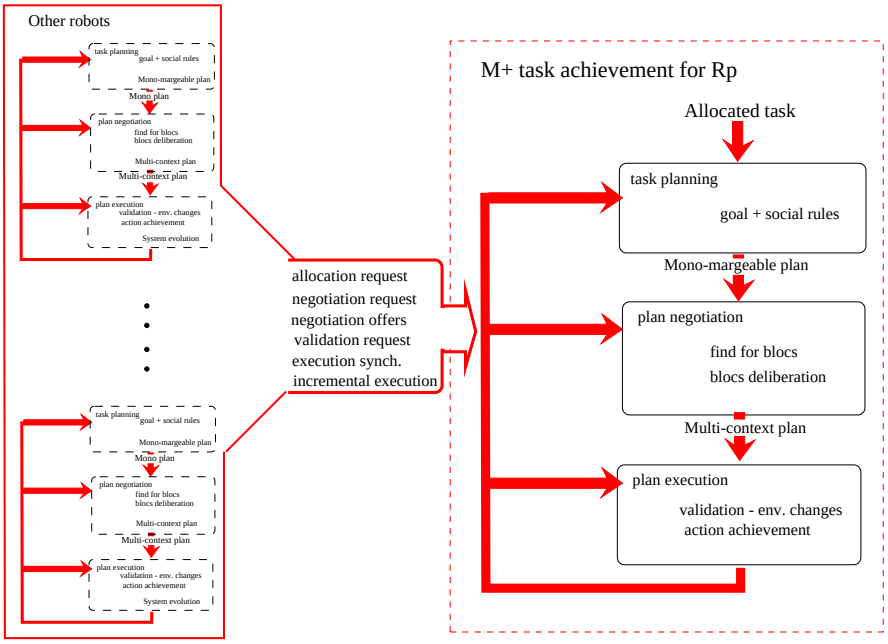

Fig. 2. The M+ task achievement process

global system performance. Figure 4 shows an instantaneous state of three robots plans.

Let us assume that R_1 begins a negotiation process in order to introduce its action A_i in the multi-robot context. This operation is "protected" by a mutual exclusion mechanism[3]. The result is a new set of negotiated actions. It is a coherent plan which includes all the necessary coordinations and some cooperative actions. Such a plan is default free and can be directly executed. However, it remains "negotiable" (other robots can perform a negotiation and propose a plan modification) until it is incrementally "frozen" in order to be sent to the plan execution activity.

5.3 The negotiation steps

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

During this process, a robot finds and negotiates all the *blocks of actions* of its current plan which are not yet announced. A block can be classified in *cooperative* or *coordinated*. A block is cooperative when its begin and end action have only *non-exclusive* effects. Therefore these actions can be used/passed

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

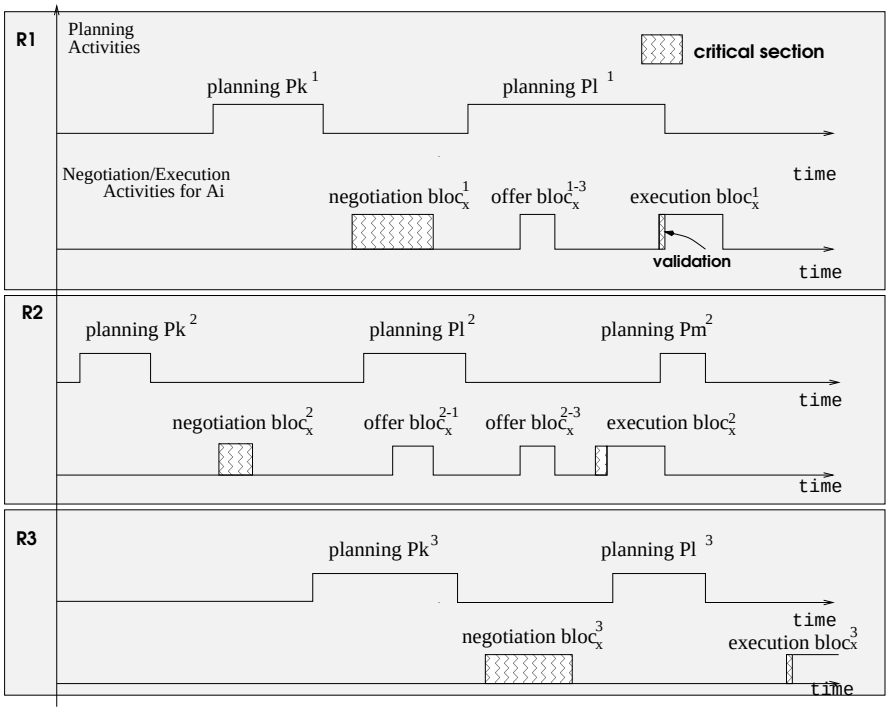

Fig. 3. The task achievement process and its protected activities

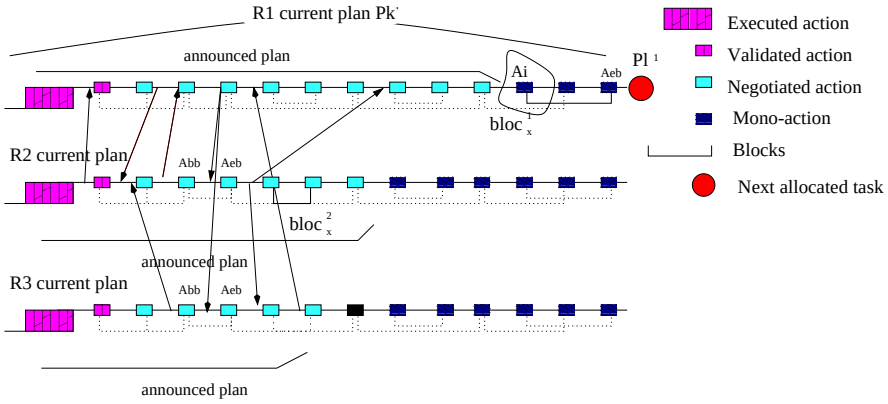

Fig. 4. The M+ plans: negotiation for A_i

by/to other robots. The blocks must be coordinated when they are *incompatible*. Two blocks are *incompatible* in two cases: 1. when they involve the same attribute and the same entities, or 2. when they may violate a *social rule*.

Step 1: the announcement. Whenever a robot (e.g R_p) wants to negotiate 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 search for possible coordinated and cooperative blocks 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.

Coordinated blocks: this analysis is directly derived from the Plan-merging paradigm[1]. Block insertion is performed incrementally by adding temporal constraints to the robots plans[4].

Cooperative blocks: 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: Deliberation. 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 block coordination, the robots use only `insert_message_wait`. Concerning the cooperative interaction, the robots have two possibilities: 1. when R_q has an *accepted* offer, it uses the `insert_message_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_message_wait` to coordinate its block with the *cooperative final block*.

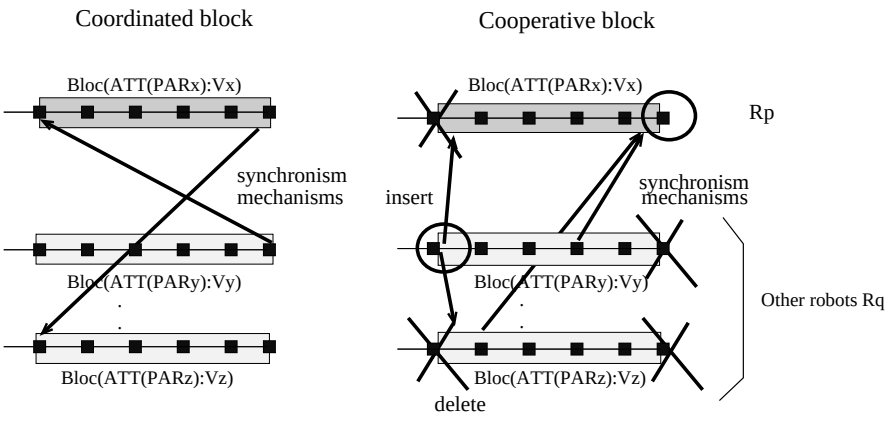

Fig. 5. Plan Modification mechanisms

[4] We use the mechanisms described in [20] for detecting and treating in a distributed way the deadlocks that may occur

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.

5.4 Execution process

Before executing an action, the robot **validates** the blocks pf actions associated with it. Indeed, a block is "negotiable" until its **validation**. Once validated, the block is "frozen", its modifications are forbidden and the robot is ready to execute the action. The other robots can only perform insertions after a validated block. 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 Example

A first version of the scheme has been implemented. We describe here below an illustrative example of its use. The robots are in a hospital environment composed of open areas connected by doors. Servicing tasks are items delivery to beds as well as bed cleaning. There are three mobile manipulator robots `r0`, `r1` and `r2`[5].

Example 1.

Figure 6 shows the tasks goals (there are 5 partially ordered tasks:`T0,...T4`) and the initial world state description[6].

The robots must respect the following *social rules*: 1. an `amount` rule (with low *obligation level*) that limits the number of robots near a bed to one, $(AMOUNT\text{-}RULE\ (POS\text{-}ROBOT\ BED1),1,(POS\text{-}ROBOT\ OPEN\text{-}AREA)$ and 2. an `end` rule (with high *obligation level*) $(END\text{-}RULE\ (STATE\text{-}DOOR(D1,CLOSED)))$ that requires to close the door. Besides, there are potentially the following coordination and cooperation issues:
1. coordination for resource conflict near the beds (rule 1)
2. `open/close` door is a cooperative (with potentially redundant effects) action (with only *non-exclusive* effects: `STATE-DOOR(<door>): OPEN/CLOSE`) and 3. `clean bed` is a cooperative incremental action (only *non-exclusive/incremental* effects: `STATE-CLEAN(<bed>):OK`) that allows cumulative effects when executed several times or by several robots.

The set of tasks is transmitted to the three robots. After a first phase (not described here), the robots plan and incrementally allocate each tasks using *M+ protocol* [3]. The allocation is incremental; in a first step, `r0` allocates `T2` (i.e. `POS-OBJECT(OB3): BED1`), `r1` allocates the cleaning task `T1` as its current task, and it also allocates `T3` as its next task. `r2` is in charge of `T0`.

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

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

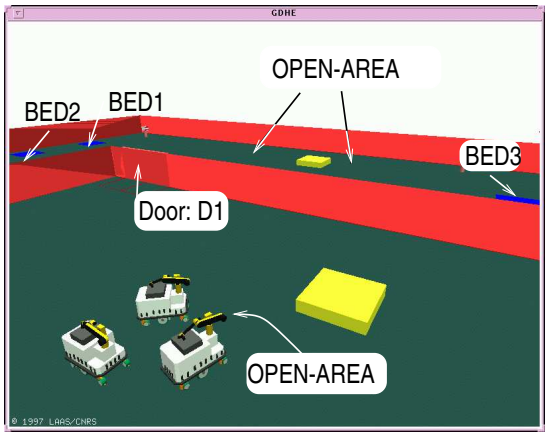

Fig. 6. Example 1: Transfer object and clean beds in a hospital area

Since there is a temporal order which imposes `T4` to executed after the end of `T3`, `T4` has not been allocated yet.

Figures 7 shows the individual plans before any negotiated task achievement while Figure 8 shows their state after a number of negotiation processes. Note that `r1` has elaborated a plan with six actions in order to achieve its main goal `STATE-CLEAN(BED):OK` and to satisfy the social rule requiring `(STATE-DOOR(D1,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.

The robots engage a negotiation process. They negotiate their `open-door` blocks. Indeed, the *social rule* 2 imposes to close the door at the end of any plan which opens it. Thus, `open(D1)` and `close(D1)` compose a *block* that must be negotiated. After a number of negotiation processes where each robot announces its blocks and the other robots formulate their *cooperative offers* [7]. Finally, there will be only one `open-close` sequence instead of three. The robots will decide who will open (`r0`) and who will close (`r1`) the door and how this will constrain temporally their plans. Figures 8 shows the result of this negotiation with the deleted actions: `open` of `r1` and `r2`, and `close` for `r0` and `r2`, represented as circles.

One can also notice, that the robots have satisfied *social rule* associated to the robot position near the beds. Indeed, they negotiated the `go-to(<bed>)` actions and inserted synchronization constraints to avoid conflicts (`insert_message_wait` represented by arrows in Figure 8).

[7] Note that a robot elaborates its offers only on the basis of its already negotiated and/or validated blocks

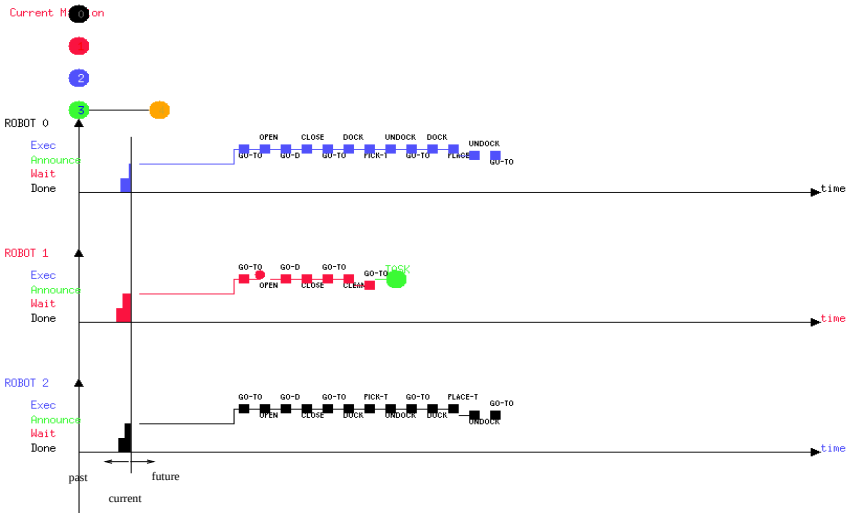

Fig. 7. M+ Allocation result: Individual plans before negotiation process

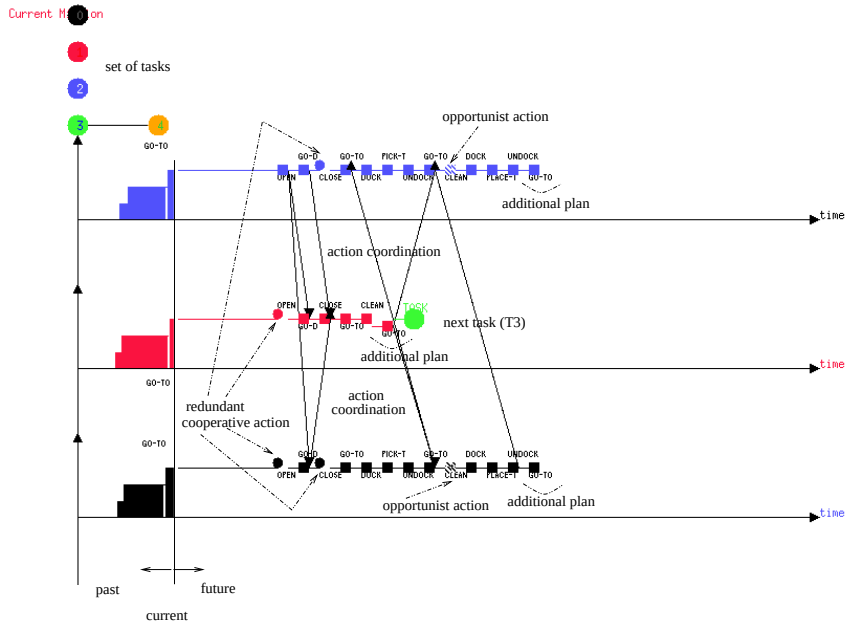

Fig. 8. M+ negotiation

Continuing the incremental negotiation, it is time for `r1` to negotiate its `clean` action. Like `open`, `clean` is a cooperative action. Moreover it is an *incremental* action. `r1` announces this action, but no robot is a priori concerned by it (no effect in the current plans). However, as `r0` and `r2` have planned to be next to `BED1` in a near future, they use the *insert* mechanism, and add a `clean` action after their arrival near to `BED1` (for delivering an object). `r0` and `r2` send their opportunistic offers to `r1`. `r1` analyzes the offers, taking into account that it is an *incremental* action. It decides that each robot will execute part of the action. The added `clean` actions to `r0` and `r2` plans are represented by a different filling pattern in Figure 8.

The overall process continues; the tasks are incrementally planned, negotiated and executed. Figure 9 shows the final result of this run. One can observe that the tasks have been achieved without conflicts and that the robots have coordinated their actions: `r0` and `r2` wait until `r1` leaves `BED1` (1), `r0` waits until `r2` leaves `BED3` (2) and `BED1` (3).
Moreover, they have also exhibited several cooperative interactions. Indeed, `r0` opens the door (4) and *opportunistically* `r1` and `r2` take advantage of this, deleting their `open` action from their current plans (5). Besides, `r0` and `r2` also help `r1` to clean a bed (6). Finally, `r1` closes the door for all (7) (`r0` and `r2` delete their `close` action (8)).

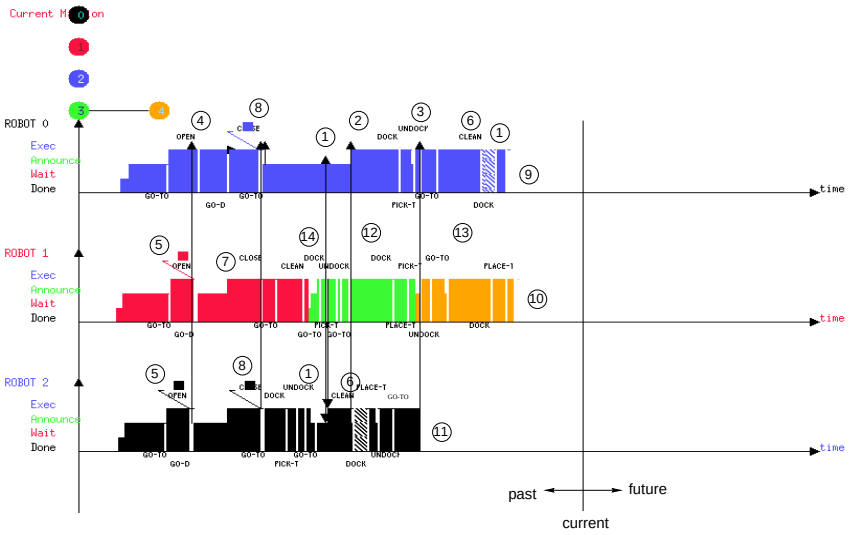

Fig. 9. Example 1: Final result of M+ task achievement

Note also that `r0` (9) and `r1` (10) have neglected the execute their additional plans that make them go to an `OPEN-AREA` because no robot has requested them to leave the beds. This was not the case for `r2` because `r0` has requested it to free `BED1` (11).

We can see that `r1` has achieved tasks `T3` (12) and `T4` (13). Note also, that `r1` has been able to avoid to execute its first additional plan which appeared at the end of `T1` (which included a `go-to(OPEN-AREA)` action) and to directly switch to the achievement of `T3` (14) which included a `go-to(BED3)` action.

Example 2 Due to lack of space we do not give details here on another run of the same example but with different time conditions. In this second example `r0` and `r2` are slower. They decide not to help (see Figure 10) `r1` to clean `BED1`. Thus `r1` achieves its `clean` action alone.

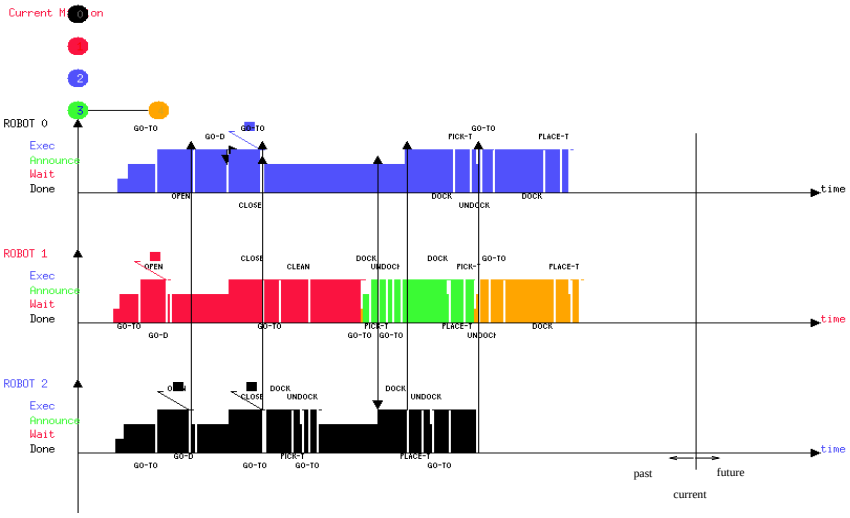

Fig. 10. Example 2: Final result of M+ task achievement

Example 3 In this example, we have inhibited the cooperative mechanisms and allowed only coordination. The final result is a set of coordinated plans. Each robot coordinated its open-door block execution with the others, waiting the end of `close(D1)` (of the other robots) to begin its `open(D1)` action (see Figure 11). The global performance clearly suffers from this.

7 Conclusion

We have proposed and discusses a scheme for cooperative multi-robot task achievement. 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 mechanisms, and illustrated its use through a simulated system.

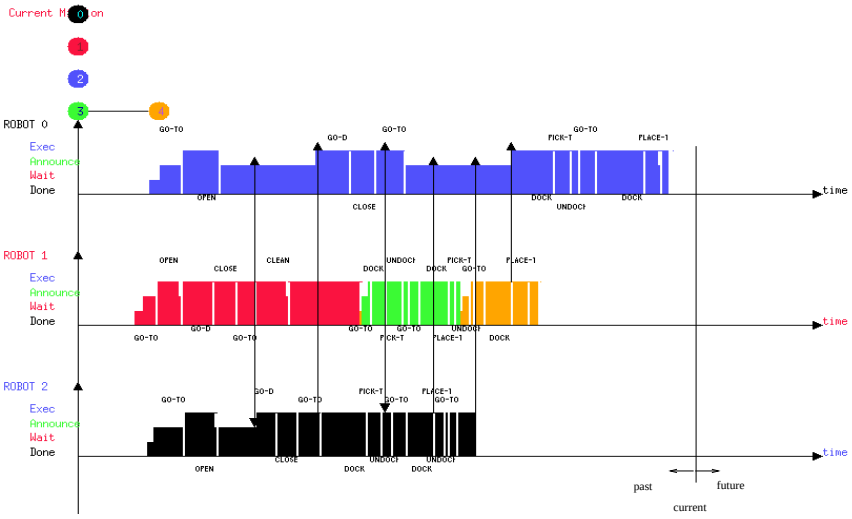

Fig. 11. Example 3: Final result of M+ task achievement

Our future work is twofold. First, We envisage to validate our approach through a number of significant application domains and to implement it on real laboratory robots. Besides, we would like to extend and further formalize the overall system and its representational and algorithmic ingredients.

Another interesting aspect is the fact that such multi-robot architectures, raise new complementary issues and constraints which are not correctly treated by the existing task planners.

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. C. Boutilier and Brafman R. Planning with concurrent interaction actions. In *AAAI'97*, 1997.
5. A. Brumitt, B. Stentz. Dynamic mission planning for multiple mobile robots. In *IEEE ICRA'96*, 1996.
6. Y. Cao, A. Fukuna, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.
7. 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.

8. K. Decker and V. Lesser. Generalizing the partial global planning algorithm. In *Int Journal of Cooperative Information Systems 92*, 1992.
9. M. DesJardins, E. Durfee, Ortiz C., and M. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, pages 13–22, 1999.
10. O. Despouys and F Ingrand. Propice-plan:toward a unified framework for planning and execution. In *ECP'99*, 1999.
11. E. Durfee and V. Lesser. Using partial global plans to coordinate distributed problem solvers. In *IJCAI87*, 1987.
12. E. Ephrati, M. Perry, and J. S. Rosenschein. Plan execution motivation in multi-agent systems. In *AIPS*, 1994.
13. T. Fukuda, Y. Kawauchi, M. Buss, and H. Asama. A study on dynamically reconfigurable robotic systems. JSME *International Journal*, 34(2):295–302, 1991.
14. N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.
15. J. Koehler, B. Nebel, J. Hoffmann, and Dimopoulos Y. Extending planning graphs to an adl subset. In *ECP97*, 1997.
16. R. Mackenzie, D. Arkin. Multiagent mission and execution. *Autonomous Robots*, 4:29–52, 1997.
17. M. Mataric. *Interaction and Intelligent Behaviour*. 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 Thechnology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative*, 1996.
20. 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.
21. J. S. Rosenschein and G Zlotkin. Rules of and encounter: Designing convention for automated negotiation among computers. *Artificial Intelligence - MIT press*, 1994.
22. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, (75):231–252, 1995.
23. R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, c-29(12), 1980.
24. 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.
25. M. Tambe. Agent architectures for flexible, practical teamwork. In *First International Conference on Autonomous Agents*, 1998.
26. 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.
27. S. Yuta and S. Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE IROS'92*, 1992.