Plan-Based Multi-robot Cooperation

Rachid Alami, Silvia Silva da Costa Botelho

Plan-based multi-robot cooperation

Rachid ALAMI*,
Silvia Silva da Costa BOTELHO**

LAAS/CNRS, Toulouse (France)
FURG, Rio Grande, RS (Brazil)

Abstract. Several issues arise if one wants to operate a team of autonomous robots to achieve complex missions. The problems range from mission planning taking into account the different robots capabilities to conflict free execution.
This paper presents a general architecture for multi-robot cooperation whose interest stems from its ability to provide a framework for cooperative decisional processes at different levels: mission decomposition and high level plan synthesis, task allocation and task achievement.
This architecture serves as a framework for two cooperative schemes that we have developed: M+NTA for Negotiation for Task Allocation, and M+CTA for Cooperative Task Achievement.
The overall system has been completely implemented and run on various realistic examples. It showed effective ability to endow the robots with adaptative auto-organization at different levels. A number of performance measures have been performed on simulation runs to quantify the relevance of the different cooperative skills that have been proposed.

1 Introduction

Starting from the **Plan-Merging** Paradigm [3] for coordinated resource utilization - and the **M+ Negotiation for Task Allocation Protocol** [8, 7] for distributed task allocation, we have developed a generic architecture for multi-robot cooperation [9, 6].

This architecture is based on a combination of local individual planning and coordinated decision for incremental plan adaptation to the multi-robot context. It has been designed to cover issues ranging from mission planning for several robots, to effective conflict free execution in a dynamic environment. It is aimed not only to integrate our past contributions but also to allow to investigate new cooperation and coordination schemes.

* R. Alami is with LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse Cedex 4,France. E-mail: Rachid.Alami@laas.fr
** S.C. Botelho is with FURG, Av. Italia Km8, 96201-000 Rio Grande/RS, Brazil. E-mail: silviacb@ee.furg.br

The goal of this paper, after a brief analysis of related work, is to present an overview of the architecture and to discuss our current instantiation. We will successively address (1) a distributed task allocation protocol and (2) a cooperative task achievement scheme that detects and treats resource conflict situations as well as sources of inefficiency. Finally, we present an implemented system which illustrates, in simulation, the key aspects of our contribution.

The overall system allows 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 achievement performance taking into account the robots capabilities as well as their execution context.

2 Related work

The field of multi-robot systems covers today a large spectrum of topics[18, 13, 28]. We here restrict our analysis to contributions proposing cooperative schemes at the architectural and/or decisional level. In such stream, *behavior-based* and similar approaches [26, 25], propose to build sophisticated multi-robot cooperation through the combination of simple (but robust) interaction behaviors. ALLIANCE [27] 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 domain independent models for agents interaction. For example, [11] and [20] enrich the STRIPS formalism, aiming to build centralized/decentralized conflict-free plans, while [14] develops specialized agents which are responsible for individual plans coordination.

Several generic approaches have been proposed concerning goal decomposition, task allocation and negotiation [4, 16]. PGP [19] (and later GPGP [15]) is a specialized mission representation that allows exchanges of plans among the agents. DIPART [29] is a scheme for task (re)allocation based on load balancing. Cooperation has also been treated through negotiation strategies [31] like CNP-based protocols [33], or BDI approaches where agents interaction is based on their commitment to achieve individual/collective goals [22, 34]. Another perspective is based on the elaboration of conventions and/or

rules. For instance, "social behaviors" [32] have been proposed as a way to program multi-agent systems. In STEAM [35], 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 [36, 12, 5].

3 A multi-robot architecture for incremental plan enhancement

The generic architecture that we propose covers issues ranging from mission planning for several autonomous robots, to effective conflict free execution in a dynamic environment.

This architecture is based on a combination of local individual planning and coordinated decision for incremental plan adaptation to the multi-robot context. 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 tasks, 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.

It remains to determine what are the relevant decisional problems that should be addressed. The architecture we propose is precisely an answer to this question. It provides a framework where multi-robot decisional issues can be treated at three different levels: 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 (Figure 1).

Indeed, we claim that it is often possible (and useful) to treat these three issues separately. As we will see, this levels deal with problems of different nature, leading to specific representations, algorithms and protocols.

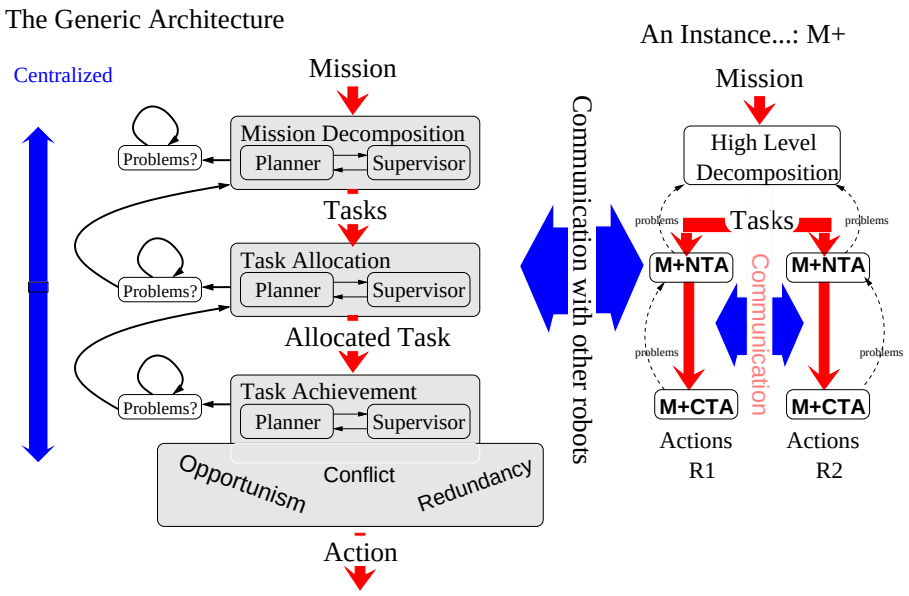

Fig. 1. Our architecture for multi-robot cooperation

This architecture is directly derived from the LAAS[1] architecture [1]. It involves a hierarchy of three decisional levels having different temporal constraints and manipulating different data representations. Each level has a reactive (supervisor) and a deliberative component (planner, plan-merger...).

Communication between robots can take place at a different levels. For a given level, both components communicate with their corresponding component. The reactive components exchange *signals* and run *protocols*; the deliberative components exchange plans, goals and data.

Let us examine the three levels with more detail.

3.1 Mission Decomposition: planning and supervision

This is a pure plan synthesis problem. It consists in decomposing a mission, expressed at a very high level, into a set of partially ordered tasks that can be performed by a given team of robots. One can con-

[1] LAAS: LAAS' Architecture for Autonomous Systems.

sider that 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.

We assume that there is no need at this level to know precisely the current robots states. It should be enough to know the types of available robots, their number, their high level features.

An example of such a mission could be transporting and assembling a superstructure in a construction site. It may require to synthesize a sophisticated plan composed of numerous partially ordered tasks to be performed by various robot types with different capabilities: transport of heavy loads, maneuvers in cluttered environment, manipulation...

Mission decomposition is a purely deliberative. It is at this level that there are less needs of context dependent information. It can be done in a central way. It is essentially a one thread process.

Of course it can benefit from several CPUs but this is a distribution of computing load, which is different in nature from problems calling for cooperative decision-making based on independent goals, on various robot capabilities and contexts.

In our current implementation, mission planning is produced by a central high level planner, for instance IxTeT [24], or the mission is provided directly by the user as a set of partially ordered tasks.

3.2 Task allocation among the robots

A mission is a set of partially ordered tasks, where each task (T_i) is defined as a set of goals to be achieved. The tasks are allocated to the robots based on their capabilities and on their execution context.

This level is not necessarily distributed. However, its distribution is clearly preferred since task allocation is essentially based on proper or local information. Indeed, the tasks may be allocated (and re-allocated when necessary) incrementally through a negotiation process between robot candidates. 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.

We have implemented this level through M+NTA[2]. This system has all necessary protocols and algorithms for cooperative task allocations (see §4).

3.3 Task achievement in a multi-robot context

The allocated tasks, and this is a key aspect in robotics, cannot be directly "executed" but require further refinement taking into account the execution context [1].

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 sources of inefficiency and proposing possible enhancements.

Coordination to avoid conflicts Each robot, while seeking to achieve its goal will have to compete for resources, to comply with other robots activities. Indeed, the higher levels, even if they produce valid mission decomposition, do not consider all possible conflicts that may appear at task execution level. We have already treated resource conflict situations as well as coordinated navigation [2, 21]. We will see, in the sequel, that the Plan-Merging Paradigm can be extended to more general conflicts.

Cooperation to enhance the system performance We have identified several cooperative issues based on local interactions:

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 lead to the same

[2] NTA: Negotiation for task achievement

world state. There should be some reasoning capabilities to allow them to decide when and which robot will perform actions that lead to the desired state while 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.

In our current instantiation of the architecture, M+CTA[3] implements this incremental task achievement level.

3.4 Cooperative reaction to contingencies

The architecture provides hierarchical reaction to contingencies. When a failure (or an unexpected event) occurs at a level, it is first treated at this level and if no solution is found, the higher level is invoked. In the framework of our multi-robot cooperative architecture, this process allows to re-consider the previous allocation or decomposition choices. This should allow a multi-robot team to adapt to its execution context and to *auto-organize* itself in order to perform complex missions in presence of uncertainty.

3.5 Discussion

In the following we discuss some design issues relative to our architecture. Architectural choices may often be considered somehow as arbitrary. Our design is partially intuitive and partially based on our own observations and on the main domains in the literature where multi-robot cooperation has been applied.

For instance, in the great majority of multi-robot systems described in the literature, only one aspect or the other is addressed. But this is only possible if the other aspects are simplified. At the highest level, the mission is often given already decomposed or with a small number of (trivial) decompositions. For example: transferring a bunch of n objects is trivially decomposed in n transfer tasks of individual objects. Numerous other possibilities (perhaps more efficient) may exist depending on the types of objects, the robot capabilities and their current state...

[3] CTA: Cooperative Task Achievement

In numerous multi-mobile robot systems, elaborated motion coordination - which clearly belongs to the the task achievement level - is neglected or ignored. Such simplification is acceptable only for non constrained environments where local non-coordinated obstacle avoidance schemes are sufficient.

One, two or three levels It may happen that for some applications, it is impossible to separate the mission decomposition and the task allocation aspects because they are too tightly linked. This is the case when the mission decomposition depends heavily not only on the types of robots available in the environment but also on their number and their current situation. In such case, the two levels should be merged in a one step planning process.

The frontier between levels that corresponds to a real qualitative change is between the task allocation and the task achievement levels. But, of course, it is still possible to devise intricate examples that challenge any architectural decomposition.

Cooperative skills Not all levels are activated or even present on all robots in a given application. For instance, one can imagine, in a hospital environment, the operation of several teams of mobile robots: a cleaning robots team, a meals and linen delivery team, and a set autonomous wheel-chairs (some of them do not even belong to the hospital)

The cleaning team may cooperate at mission level. The meals and linen delivery team may cooperate at task allocation level. All robots need to cooperate at resource conflict level.

Global coherence and efficiency While the architecture may be considered as satisfactory in terms of identification of the relevant levels of abstractions and their articulation, this is not a guarantee of global coherence nor of efficient operation of the robots.

Indeed, such properties depend primarily on the cooperative schemes and the algorithms that are implemented *inside* each level. For example, the Plan-Merging Paradigm has been devised to provide quite efficient local solutions to most resource conflicts while maintaining two key features [30]:

- the coherence of the global scheme and the ability to detect the situations where it is not applicable
- a localized management of the planning and coordination processes with, in particularly intricate situations, a progressive transition to more global schemes which may "degrade" to a unique and centralized planning activity.

The following sections present successively M+NTA and M+CTA.

4 M+NTA: Negotiation for 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 antecedent tasks are already achieved or under execution. Robots are informed whenever a task is started or finished.

The M+NTA task allocation process allows the robots to incrementally choose a task among the current *executable tasks*. We use an adapted version of the *Contract Net Protocol* [33] for the negotiation. 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 criterion will be the costs of the plans elaborated by different robots depending on their capabilities and situations.

Figure 2 shows M+NTA task allocation state diagram. There are 5 possible states: *planning*, *eval-cost*, *candidate*, *best-candidate* and *idle*.

A robot R_p enters the *eval-cost* state whenever there is an update in the set of *executable tasks* and it is ready to negotiate a new task (1). It invokes its planner (2) in order to synthesize plans for the executable tasks and to estimate their costs (3). Then, R_p selects the task for which it can propose a better cost than the cost announced by other robots, if any.

If there is no such task, R_p enters the *idle* state (8).

If a task T_k is selected, R_p enters the *candidate* state (4). R_p sends its offer to current *best candidate* for this task and waits for an answer. If the answer is positive, meaning that the current *best candidate* accepts to transfer T_k to R_p. R_p enters the *best-candidate*

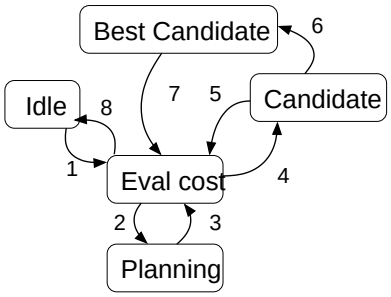

Fig. 2. State diagram for the task allocation protocol

state (6). However, the answer can be negative, meaning that the cost of R_p is not better than the cost of the current *best candidate*. R_p then abandons T_k and enters the *eval-cost* state in order to select a new task (5).

When a robot R_p enters the *best-candidate* state for a given task T_k, it holds in this state until either it begins T_k execution or until it receives a better offer from another robot, or until it abandons T_k due to a failure or to a cooperative reaction (7). It then enters the *eval-cost* state in order to select a new task.

M+ Cooperative Reaction

The `M+NTA` scheme also provides a treatment for cooperative robot behavior in case of execution failure. When a problem occurs that prevents a robot R_p 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_p does not find a new plan, it sends relevant information with a *request for help* to the other robots and waits. If several robots propose their help, R_p selects the best offer. R_p abandons T_k only if it receives no help offer.

5 M+ Cooperative task achievement (M+CTA) and 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: 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 their utilization. An interesting example is the elevator.

The *mechanisms* 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.

5.1 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.

In the current version of our system, this knowledge is represented by (see figure 3):

- 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*.

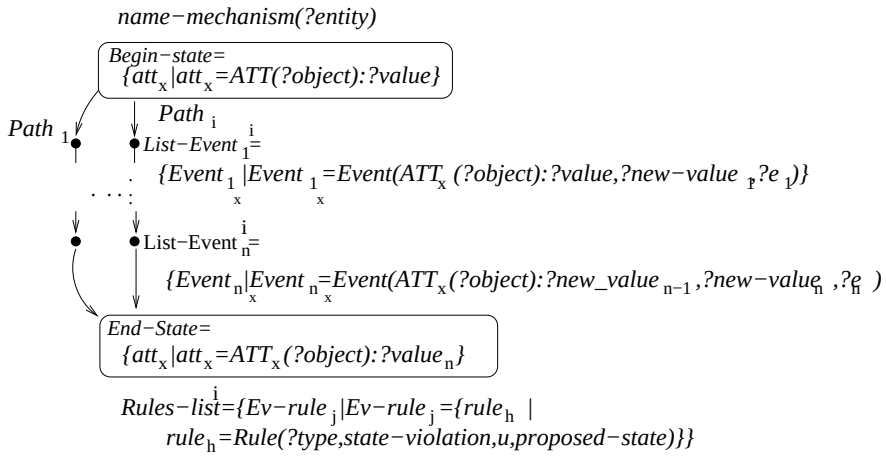

Fig. 3. A generic mechanism M

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 specify forbidden or undesirable states and propose some desired states. This information is used by the planner in order to avoid the violation of the rule. Thus, social rules have the following generic description:

$$\textsc{Rule}(type, violation\_state, s, proposed\_state)$$

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

1. `amount`: where $violation\_state = (att(?object) : v)$ represents a resource that should 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 $\text{RULE}(\texttt{amount}, (pos\_robot(?r) : D1), 2, \texttt{OPEN\_AREA})$, 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 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 *violation_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 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[4].

[4] Note that this notion of social rules is different, or even complementary, from the social behaviors proposed by [32]. 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.

5.2 *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 in 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 4 shows a plan produced by robot R_p that uses a furnace. R_p builds a job M_j^p that may be negotiated. This *job* ends when the final state of the associated mechanism is reached.

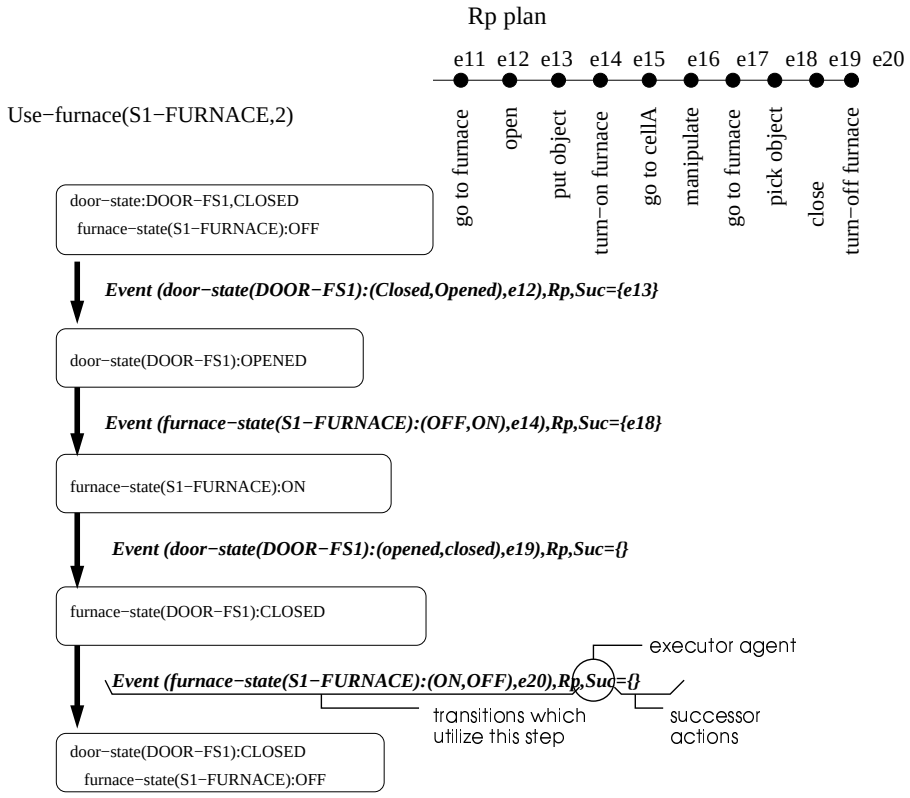

Fig. 4. A *job* corresponding to the use of a furnace by *Rp*

6 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*.

6.1 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[5]. 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 in the following the different steps involved in this negotiation process.

6.2 The negotiation steps:

The negotiation process consists of two steps: the **announcement** and the **deliberation**. During this process, a robot negotiates a set of *jobs* of its current plan[6].

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.

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

[6] We treat together, in one step, all "interleaved" jobs to avoid deadlock situations.

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 5:

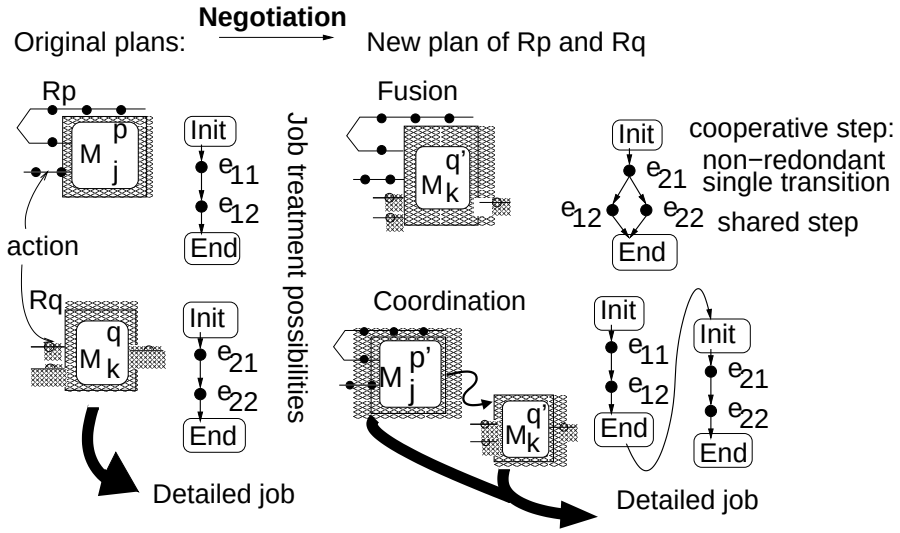

Fig. 5. Job treatment possibilities: fusion or coordination.

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 treat redundant and shared 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 release by the agents involved in 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 adapt their plans to the jobs modification. We have defined the following operations:

`insert_message_wait` that introduces a temporal order constraint between two actions belonging to two robots, and `insert/delete`, when an action is re-assigned to another robot.

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.

7 Illustration

M+ is a complete multi-robot decisional system. It is an instance of the general architecture described in §3. In this implementation, we use PROPICE-PLAN [17] together with a STRIPS-like planner called IPP[23] and a motion planner. Each robot control system runs on an independent workstation which communicates with the other workstations through TCP/IP. For a given application, the environment topology, the robot features, the list of mechanisms and the set of STRIPS actions are input parameters for M+.

We have implemented a first version of the overall system and run it on the realistic simulation platform that was initially developed for the Martha project [2]. Below we describe some of the obtained results.

Our objective here is to measure and compare gains that are obtained when the robots are equipped with the cooperative and coordinations skills that we propose. The interested reader may refer to [10] for a set of documented runs where we examine the different negotiation and cooperation steps.

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. Figures 6 and 7 show the simulated environment and 14 partially ordered tasks: `T0,...T13` and the initial world state description[7].

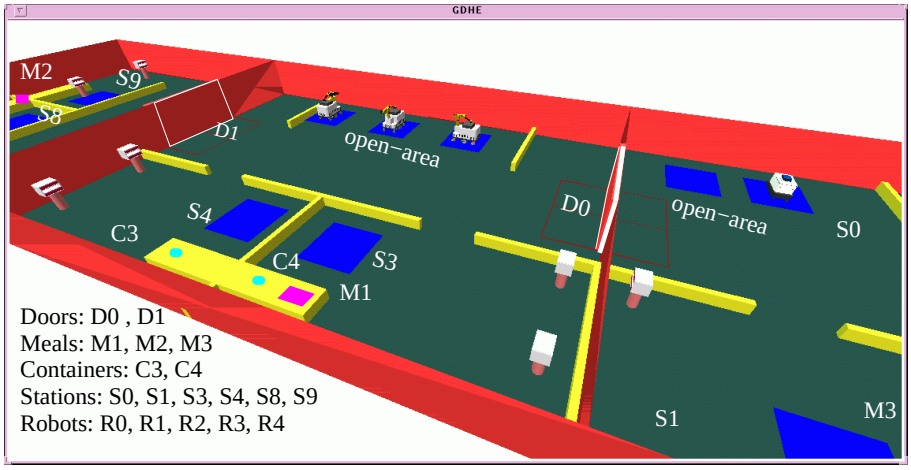

Fig. 6. Example 1: Transfer objects and clean beds in a hospital area

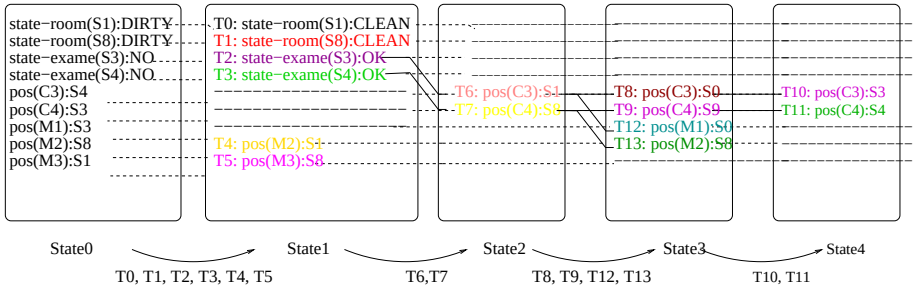

Fig. 7. Example 1: The decomposed mission: 14 individual partially ordered tasks.

The robots must negotiate the use of the following *mechanisms*: (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

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

redundant; and (3) a mechanism that controls the use of 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 (described in [8]), they plan and incrementally allocate the tasks using *M+ Cooperative Task Allocation*. Figure 8 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. Indeed, `r1` is achieving `T6`. It has elaborated a plan with six actions in order to achieve its main goal `pos(C3):S1` 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 is accomplished by `r3`. This robot opens a first time the door and all robots take 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 tasks. For instance: `r1-T6` and `r2-T9`.

The overall process continues; the tasks are incrementally planned, negotiated and executed. Figure 9 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 detected and deleted redundant actions (open/close door) accomplished opportunistically by others. Besides, some robots also helped the others to clean rooms.

Figure 10 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

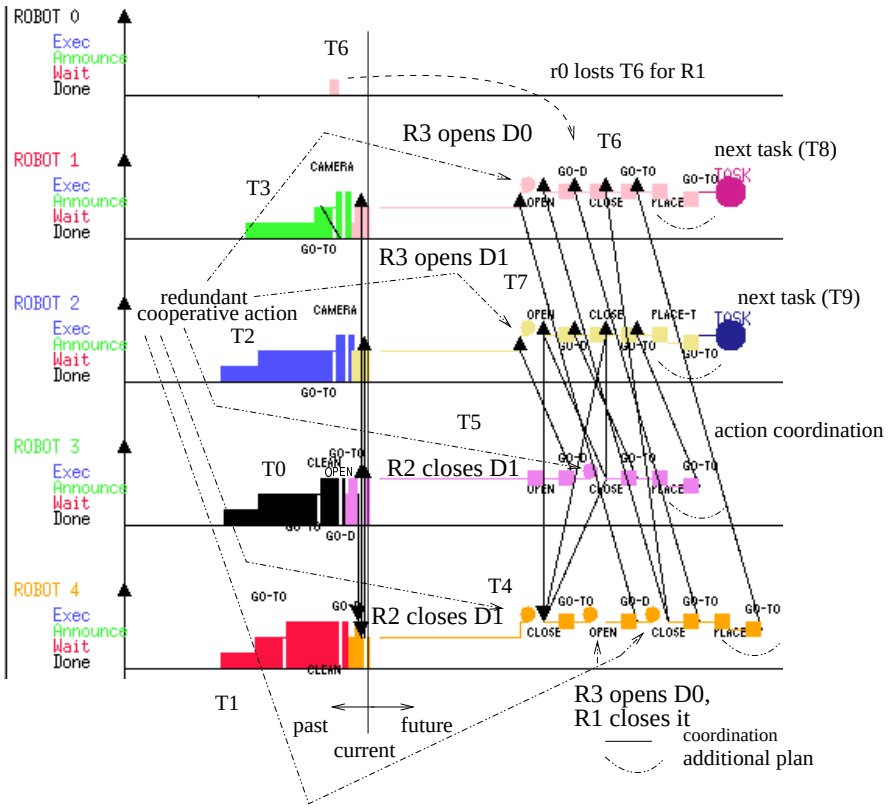

Fig. 8. M+ task achievement process: 5 execution streams corresponding to 5 robots that plan, negotiate and coordinate incrementally their activities. The arrows between robot plans illustrate the temporal constraints induced by the coordination between jobs.

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` enhances the system performance with better costs and less actions (see Figures 11).

When we change the number of robots, we observe (Figure 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

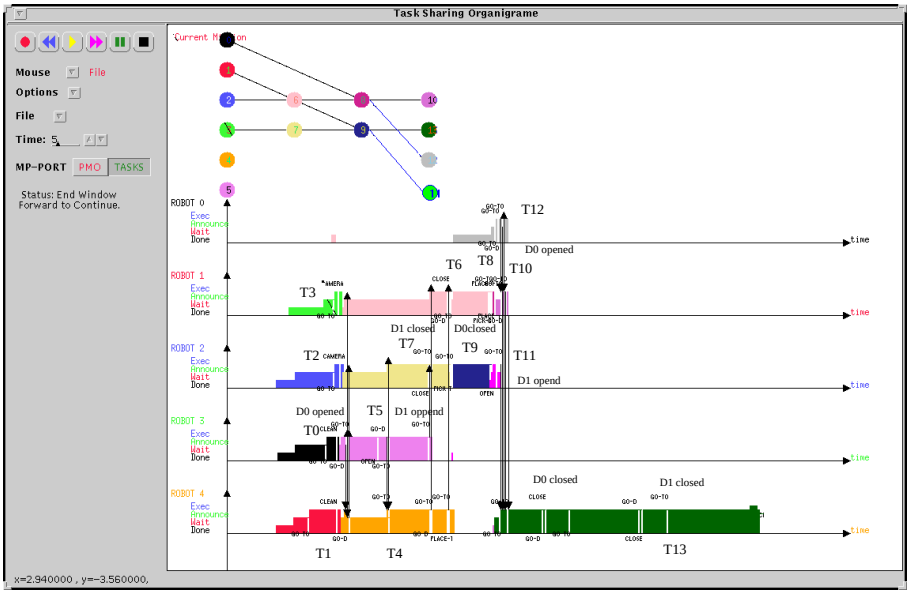

Fig. 9. The final result of the run. All streams are now finished. The top part of the figures shows the partial order between tasks

robots tests; this is due to the nature of mission. The partial order of tasks prevents an optimum deployment of more than 3 robots.

Concerning the the workload, we can see that when we have 5 robots, one of them (`r0`) is almost idle (Figure 12). This fact explains the 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.

8 Conclusion

We have proposed a generic architecture for multi-robot cooperation. Its interest stems from its ability to provide a framework for cooperative decisional processes at different levels: mission decomposition and high level plan synthesis, task allocation and task achievement.

We have built an instance of this architecture with negotiation for task allocation and cooperative plan coordination and enhancement at the task achievement level.

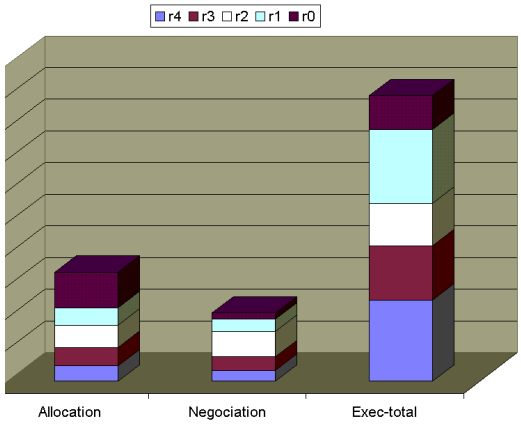

Fig. 10. Time spent in decisional (allocation, negotiation) and execution activities by the different robots.

We have also discussed a scheme for cooperative multi-robot task achievement, called *mechanism*. This scheme is a key component of our 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.

This architecture has been completely implemented and run on various realistic examples. It showed effective ability to endow the robots with adaptative auto-organization at different levels. We have

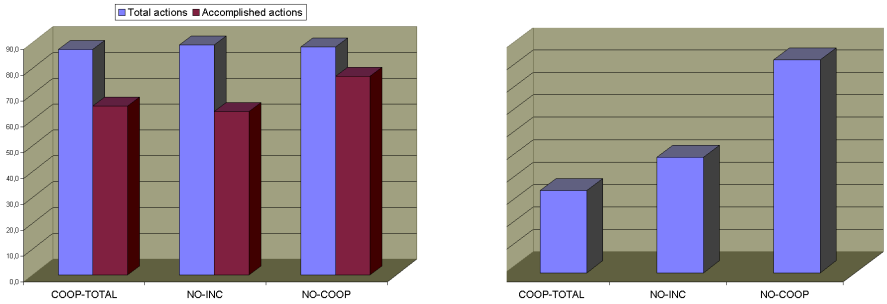

Fig. 11. Relevance of the proposed cooperative schemes. When the robots use all the proposed schemes (COOP-TOTAL), they perform less actions and the global cost is lower.

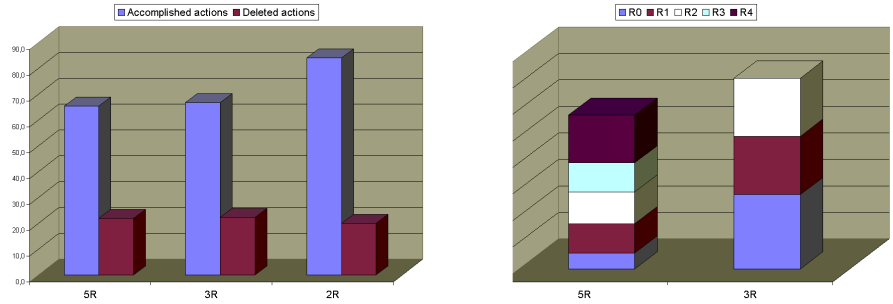

Fig. 12. Illustration of the influence of the number of robots for the same mission. Left:Number of robots vs. planned and achieved actions. Right:Workload for each robot

made some preliminary measures that allow to verify and to quantify the relevance of the different cooperative skills that have been proposed.

It remains to validate this 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.

Besides, it is interesting to observe that this study has raised several issues that deserve further investigations: 1) the opportunistic help for global performance improvement, 2) a class of cooperative issues that can be translated into operations on plans, 3) the integration of a behavior model like the social rules at the planning level in order to synthesize easily merge-able plans.

References

1. R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An Architecture for Autonomy. *International Journal of Robotics Research*, 17(4):315–337, April 1998.
2. 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.
3. R. Alami, F. Ingrand, and S. Qutub. A scheme for coordinating multi-robot planning activities and plans execution. In *ECAI'98*, 1998.
4. H. Asama and K. Ozaki. Negotiation between multiple mobile robots and an environment manager. In *IEEE ICRA'91*, pages 533–5382, 1991.
5. K. Azarm and G. Schmidt. A decentralized approach for the conflict-free motion of multiple mobile robots. *Advanced Robotics*, 11(4):323–340, 1997.

6. S. Botelho. *Une architecture dcisionnele pour la coopration multi-robots.* PhD thesis, LAAS-CNRS, 2000.
7. S. S. C. Botelho. A distributed scheme for task planning and negotiation in multi-robot systems. In *ECAI'98*, 1998.
8. 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.
9. S. S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE Int. Conf. on Robotics and Automation (ICRA'2000)*, 2000.
10. S. S. C. Botelho and R. Alami. Robots that cooperatively enhance their plans. In *Distributed Autonomous Robotic Systems 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer*, 2000.
11. C. Boutilier and Brafman R. Planning with concurrent interaction actions. In *AAAI'97*, 1997.
12. A. Brumitt, B. Stentz. Dynamic mission planning for multiple mobile robots. In *IEEE ICRA'96*, 1996.
13. Y. Cao, A. Fukuna, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.
14. 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.
15. K. Decker and V. Lesser. Generalizing the partial global planning algorithm. In *Int Journal of Cooperative Information Systems 92*, 1992.
16. M. DesJardins, E. Durfee, Ortiz C., and M. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, pages 13–22, 1999.
17. O. Despouys and F Ingrand. Propice-plan:toward a unified framework for planning and execution. In *ECP'99*, 1999.
18. G Dudek. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3:375–397, 1997.
19. E. Durfee and V. Lesser. Using partial global plans to coordinate distributed problem solvers. In *IJCAI87*, 1987.
20. E. Ephrati, M. Perry, and J. S. Rosenschein. Plan execution motivation in multi-agent systems. In *AIPS*, 1994.
21. F. Gravot and R. Alami. An extension of the plan-merging paradigm for multi-robot coordination. In *IEEE International Conference on Robotics and Automation, Seoul, Korea*, May 2001.
22. N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75, 1995.
23. J. Koehler, B. Nebel, J. Hoffmann, and Dimopoulos Y. Extending planning graphs to an adl subset. In *ECP97*, 1997.
24. P. Laborie. *IxTeT: une approche integrée pour la Gestion de Ressources et la Synthèse de Plans.* PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1995.
25. D. Mackenzie and R. Arkin. Multiagent mission and execution. *Autonomous Robots*, 4:29–52, 1997.
26. M. Mataric. *Interaction and Intelligent Behavior.* PhD thesis, Massachusetts Institute of Technology, 1994.
27. L. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Trans. on Robotics and Automation*, 14(2):220–239, 1998.

28. L. Parker. Current state of the art in distributed robot systems. In *Distributed Autonomous Robotic Systems 4, Lynne E. Parker, George Bekey, and Jacob Barhen (eds.), Springer*, pages 3–12, 2000.
29. M.E. Pollack. Planning in dynamic enviroments: the dipart system. *Advanced Planning Technology: Technology Achievements of the ARPA/Rome Laboratory Planning Initiative*, 1996.
30. 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.
31. J. S. Rosenschein and G Zlotkin. Rules of and encounter: Designing convention for automated negotiation among computers. *Artificial Intelligence - MIT press*, 1994.
32. Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 0(75):231–252, 1995.
33. R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, c-29(12), 1980.
34. 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.
35. M. Tambe. Agent architectures for flexible, practical teamwork. In *First International Conference on Autonomous Agents*, 1998.
36. S. Yuta and S. Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE IROS'92*, 1992.