Planning Coordination and Execution in Multi-robots Environment *

Rachid Alami and **Félix Ingrand** and **Samer Qutub** †
LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse Cedex 04
E-mail: {rachid,felix,sam}@laas.fr

Abstract

We present and discuss a generic cooperative scheme for multi-robot cooperation based on an incremental and distributed plan-merging process. Each robot, autonomously and incrementally builds and executes its own plans taking into account the multi-robot context. The robots are assumed to be able to collect the other robots current plans and goals and to produce plans which satisfy a set of constraints that will be discussed.

We discuss the properties of this cooperative scheme (coherence, detection of dead-lock situations) as well as the class of applications for which it is well suited. We show how this paradigm can be used in a hierarchical manner, and in contexts where planning is performed in parallel with plan execution. We also discuss the possibility to negotiate goals within this framework and how this paradigm "fills the gap" between centralised planning and distributed execution.

We finally illustrate this scheme through an implemented system which allows a fleet of autonomous mobile robots to perform load transfer tasks in a route network environment with a very limited centralised activity and important gains in system flexibility and robustness to execution contingencies.

1 Introduction

In the field of multi-robot cooperation, we claim that it is useful to make a distinction between two main issues:

- **C1:** the first issue involves goal/task decomposition and allocation to various robots
- **C2:** the second issue involves the simultaneous operation of several autonomous robots, each one seeking to achieve its own task or goal.

While several contributions have concentrated more particularly on one issue or the other, we claim that in numerous multi-robot applications, both issues appear and even "invoke" one another "recursively".

This is particularly true for autonomous multi-robot applications and, more generally, when the allocated tasks or goals cannot be directly "executed" but require further refinement, because the robots act in a same physical environment and because of the multiplicity of uncertainties.

Let us assume a set of autonomous robots, which have been given a set of (partially ordered) tasks or goals. This could be the output of a central planner, or the result of a collaborative planning process. One can consider this plan elaboration process finished when the obtained tasks or goals have a sufficient range and are sufficiently independent to cause a substantial "selfish" robot activity. However, each robot, while seeking to achieve its task will have to compete for resources, to comply with other robots activities. Hence, several robots may find themselves in situations where they need to solve a new goal/task interaction leading to a new goal/task allocation scheme.

In such context, planning and plans coordination can be classified along different strategies or choices.

Global versus local. When one plans actions and motions for a number of robots, such as a fleet of mobile robots, one can consider the whole fleet or limit the scope of planning to the robots "involved" in the considered resources. Indeed, it seems to be rather inefficient to take into account all the robots present on the field for any decision which only involves a subset of them. However, this global versus local tradeoff is only possible when dealing with a properly sized environment. If the number of critical exclusive resources (such as spatial resource: lane cells or crossing cells) is more or less equal to the number of robots, conflict resolution will, by propagation, involve the whole fleet. On the other hand, if the environment is properly sized, conflicts remain local, and the solutions are negotiated locally without disturbing the unconcerned robots.

Complete versus incremental. Similarly, one can limit the scope of the planning and plan coordination in time. When a mission (i.e. a number of goals) is sent to a robot, it can plan (or try to plan) and coordinate the whole mission. But considering the execution hazards, and the inaccuracies with which one can forecast at what time such and such contingent actions will end, it seems to be inefficient (not to say a waste of time and resource) to plan too far ahead. The plan coordination should be done continuously, to guarantee a fluid operation, and slightly ahead, to avoid to over constrain the others robot plans and to break the coordinated plans too often.

Centralised versus distributed. This last aspect of the planning and plan coordination problem is where the planning and plan coordination should take place, on a centralised com-

*This paper has been published in the proceedings of IEEE ICAR97,Monterey, California (USA).

†Authors list in alphabetical order.

puter or on board the robots. Centralised versus distributed does not change the computing complexity of the treatment. However, in a centralised approach, all the data (which are mostly local) need to be sent to the central station, and therefore require a more reliable communication link with a higher bandwidth between the robot and the central station. Moreover, we will see that the proposed protocol can be implemented with a local communication (i.e. only with the robots in the vicinity).

The approach we have chosen can be classified as *local*, *incremental* and *distributed* [Alami *et al.*, 1994; 1995]. However, when the situation imposes it, our paradigm may "evolve" dynamically toward a more centralised, global form of planning [Qutub *et al.*, 1997].

After this introduction which makes a general presentation of our approach, and situates it in the general multi-robot planning debate, we present the related work in section 2. We shall introduce a more formal presentation of the Plan Merging Paradigm (PMP) and its operators in section 3. Section 4 briefly presents the multi-robot application on which we tested and validated the PMP.

2 Related work

While several generic approaches have been proposed in the literature concerning task or goal decomposition and allocation (Contract Nets [Smith, 1994], Partial Global Planning [Durfee and Lesser, 1991], distributed search [Durfee and Montgomery, 1991], negotiation [Jennings, 1995; Ferguson and Allen, 1994; Rosenschein and Zlotkin, 1994], motivational behaviours [Parker, 1994; Ephrati *et al.*, 1994]), cooperation for achieving independent goals have been mostly treated using task-specific or application-specific techniques [Le Pape, 1990; Yuta and S.Premvuti, 1992]

We argue that there is also a need for generic approaches to **C2**. One can of course make the robots respect a set of rules (e.g. traffic rules), or more generally "social behaviours" [Shoham and Tennenholtz, 1995], which are specially devised to avoid as much as possible conflicts and to provide predefined solutions to various situations. However, this cannot be a general answer applicable to various domains.

Our scheme provides and guarantees a coherent behaviour of the robots in all situations (including the avalanche of situations which may occur after an execution failure) and a reliable detection of situations which call for a new task distribution process.

3 Presentation of the PMP

An robot which wants to elaborate a coordinated plan need by some means to get some of the other robots plans. This is performed through request broadcasted by the planning robot and responses (plans or goal) sent by the other robots. Some communication means are thus required and should provide reliable messages and broadcasts delivery and should guarantee messages reception order.

Part of the robot computation need to be made in mutual exclusion over some resources. In our case, the resource in mutual exclusion is the physical resource "planning/coordination" operation and not the use of the physical resource itself. For this we use a distributed mutual exclusion (DMEX) protocol based on path reversal [Naimi *et al.*, 1996].

3.1 World Model using State attributes

We use a formalism borrowed from the IxTeT temporal planner [Laborie and Ghallab, 1995]. Time is explicitly dealt with on the basis of time-points. The usual symbolic constraints of time-points algebra (i.e before, simultaneous, after) are handled.

The world state is described through a set of *multi-valued state attributes*. Each state attribute describes a particular feature of the world; it is a k-ary mapping from some finite domains into a finite range called the *value* of the attribute.

We rely on a reified logic formalism [Shoham and Tennenholtz, 1995] where state attributes are temporally qualified by the predicates *hold* and *event*:

$hold(att(x_1, ...) : v, (t_1, t_2))$ asserts the persistence of the value of attribute $att(x_1, ...)$ to v for each t: $t_1 < t < t_2$. This function sets a *unique* value at a given time point to each possible instance of each attribute.

$event(att(x_1, ...) : (v_1, v_2), t)$ states that an instantaneous change of value of $att(x_1, ...)$ from $v1$ to $v2$ occurred (or will occur) at time t. The predicate *event* can be defined by:

$$event(att(x_1, ..., x_n) : (V_{\texttt{before}}, V_{\texttt{after}}, t) \iff$$

$$\begin{aligned} &\exists t_1, \exists t_2 / (t_1 < t < t_2) \\ &\wedge\ hold(att(x_1, ..., x_n) : V_{before}, (t_1, t)) \\ &\wedge\ hold(att(x_1, ..., x_n) : V_{after}, (t, t_2)) \end{aligned}$$

In order to exhibit more clearly the possible conflicts which may arise between robots, we provide an explicit description of resource uses. A resource can be a single item with a unit capacity (an *unsharable resource*), or an aggregate resource that can be shared simultaneously between different actions seeing that its maximal capacity is not exceeded.

The resource availability profiles and their uses by different operators are described by means of three predicates [Laborie and Ghallab, 1995]. The PMP only requires the operator $use(r : q, (t_1, t_2))$ which asserts that an integer quantity q of resource r is used between time-points t_1 and t_2.

Such a formalism is well suited to describe a "scenario" in a dynamic domain. At a given instant, the current scenario is represented by the initial values of the state attributes and the expected changes. A scenario $S = (A, E, U, T, L)$ where:

- A is a set of assertions
- E is a set of events
- U is a set of resource availability profiles and their uses
- T is the set of of all the time-points referenced in A , U and E
- L is a lattice composed of time-points linked by temporal precedence relations $(t_i < t_j)$.

Such a description is maintained by each robot and serves as an input to planning operations. It must be always *consistent*, i.e. events on the same domain attribute should be totally ordered with values compatible with this order.

3.2 Tasks models

Robot actions are represented by "tasks" which will be used as plan operators. A task Tsk is described by:

• A set of time-points including at least $start(Tsk)$ and $end(Tsk)$ corresponding to the beginning and to the end of the task execution;
• A set of events describing the changes of the world induced by the task
• A set of resources uses,
• A set of assertions on state attributes to express the conditions required to allow the use of the task (insertion conditions) and the persistence of some facts between two task events,
• A set of temporal constraints specifying a partial order between the time points of the task.

Each robot planner will be equipped with a set of tasks[1] which correspond to its own functional capabilities. Whenever, the planner will be invoked, it will be given an initial scenario and a goal. If it succeeds to produce a plan, its output will be a new scenario where new tasks are inserted.

3.3 Plan Merging Operators

Mono-robot context. In order to establish our notations, let us assume that we have only one robot R_i . It processes sequentially the goals it receives, taking as initial state the final state of its current plan. Doing so, it incrementally appends new sequences of actions to its current plan.

Assume that an robot R_i has already a plan under execution $S_i^k = (A_i^k, E_i^k, U_i^k, T_i^k, L_i^k)$. When it receives a new goal G_i^{k+1}, it tries to produce a new plan to achieve it. As S_i^k corresponds to a plan which is under execution, it is considered as contingent and thus cannot be modified by the planner.

Let $S_i^{k+1} = (A_i^{k+1}, E_i^{k+1}, U_i^{k+1}, T_i^{k+1}, L_i^{k+1})$ be the obtained plan:
$S_i^{k+1} = \mathsf{Plan}(S_i^k, G_i^{k+1})$.

Besides all elements contained in S_i^k, S_i^{k+1} contains new actions, events, used resources as well new temporal relations: $A_i^k \subset A_i^{k+1}$ .. $L_i^k \subset L_i^{k+1}$.

However, because the planner is not allowed to modify S_i^k, the set of new temporal relations L_i^{k+1}/L_i^k([2]) do not contain any temporal relations which adds a new constraint to time-points referenced in T_i^k.

$L_i^{k+1}/L_i^k = \{(t_m < t_n) | t_m \in T_i^{k+1}, t_n \in T_i^{k+1}/T_i^k\}$

Multi-robot context. The scenario for R_i is slightly different. It has to avoid resource conflicts with the other robots. This is done by synchronising its plan with other robots plans.

The global process is the following. We have n robots performing tasks in parallel in the environment. At any moment each robot R_i has a current plan CP_i under execution which is valid in the multi-robot context. It contains all the necessary synchronisations with other robots current plans which prevent any conflict to occur.

A current coordinated plan for an robot R_i can then be specified as a scenario $CP_i = (A_i, E_i, U_i, T_i, L_i, W_i)$; W_i is a set of temporal relations $(t_m < t_n)$ where t_m belong to a scenario of another robot $((t_m \in T_j)$ while $t_n \in T_i)$[3].

The system is dynamic; the robots execute their current coordinated plans and update them; the temporal constraints expressed in the different W_i result in message-based synchronisation between robots.

The global system is assumed to be coherent, i.e. the union of all CP_i $i \in [1..n]$, is such that the union of all the lattices L_i and $W_i (i \in [1..n])$ is a lattice[4].

We are interested in operators and mechanisms which allow any of these robots to add a new actions to its own plans while maintaining this property.

3.4 Plan then Insert

For the first operator, we separate plan generation and plan coordination as two phases. Whenever, an robot R_i receives a new goal G_i^{k+1}, it first elaborates a plan $S_i^{k+1} = \mathsf{Plan}(CP_i, G_i^{k+1})$ where CP_i is the result of the previous plan-merging step updated by the execution.

Then, it has to ensure that S_i^{k+1} is valid in the multi-robot context. This is done:

- by obtaining, through communication, all the current plans, under execution, CP_j which may interfere with S_i^{k+1} (i.e. not all the robots plans, but only those which make use of resources referenced in U_i^{k+1}/U_i^k).
- it then builds the union of the obtained current plans $MP_i = \bigcup_j CP_j$.
- and finally, it tries to insert S_i^{k+1} into MP_i; this is done by scheduling the new tasks contained in S_i^{k+1}. Because, MP_i is currently under execution, the only latitude is to add new constraints[5] W_i^{k+1}. If it succeeds, it updates its current plan $CP_i \leftarrow (A_i^{k+1}, E_i^{k+1}, U_i^{k+1}, T_i^{k+1}, L_i^{k+1}, W_i^{k+1})$.

Such operation, called Plan-merging Operation results in the modification of CP_i based on the latest state of the current plans that may be concerned by this modification (fig 1). In order to guarantee a coherent update, it is sufficient to perform it in a critical section which locks any update of $MP_i = \bigcup CP_j$ for the set of resources defined by U_i^{k+1}/U_i^k. Note however, that the planning operation itself is not performed under a critical section.

Note also that several Plan-merging Operations may be performed in parallel if they involves disjunctive resource sets.

3.5 Searching for an Insertable Plan

This alternative operator is more powerful. Indeed, it takes into account all the other robot current plans in the planning operation itself.

$CP_i^{k+1} = \mathsf{PlanMerge}(MP_i, G_i^{k+1})$

where MP_i is the union of all current plans which may interact with CP_i^{k+1}.

The difficulty here is to compute the set MP_i. This in general impossible (unless, we take into account all the robots in the environment). However, it can be possible in some situations. For example if the environment is hierarchical, and the MP_i only involves robots in a specific part of this environment.

---

[1] We assume that all tasks effects are given without ramifications.

[2] A/B denotes the set difference $A - B$.

[3] t_n clearly corresponds to the starting time-point of a R_i task.

[4] However, no robot stores nor manages is as a whole.

[5] This will have as a consequence, a synchronisation of the start events of the new tasks contained in S_i^{k+1} with execution events produced by other robots.

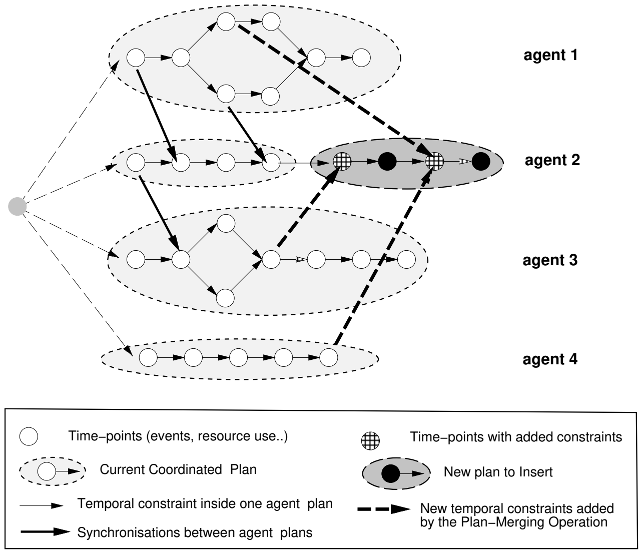

Figure 1: Robot 2 performs a Plan-Merging Operation.

3.6 Situations where PMO is deferred or where a "deadlock" is detected

When a robot R_i tries to perform a PMO, it may fail to produce a plan which can be inserted into MP_i. This means that the final state of at least another robot R_j (as it is specified in its current plan CP_j) forbids R_i to insert its own plan.

In such situation, R_i can simply abandon its PMO and decide to wait until the robots, that it has identified, have performed a new PMO which may possibly make them change the states preventing it to insert its plan.

Hence, besides *execution events* — i.e. events specified in W_i and which allow robots to synchronise their execution —, we introduce *planning events* — i.e. events which occur whenever a robot performs a new PMO. These events can also be awaited for. They establish a temporal order relation between robots plan-merging activities, noted $Wait\_for\_PMO(R_i, R_j)$.

When a R_i concludes that it has to wait for R_j to perform a successful PMO, it informs R_j. Robots maintain and propagate a graph of robots waiting for them (directly or by transitivity) to perform a PMO.

When a robot R_i succeeds in performing a PMO, it informs its immediate successors (if any) and discards its graph. But if it fails, it has to determine the set of robots from which it has to wait *planning event*. A "deadlock" occur if one of these robots is already waiting (directly or by transitivity) for R_i to perform a PMO.

When a deadlock occurs, it is necessary to take explicitly into account, in a *unique planning operation*, the conjunction of goals of all the robots involved in the cycle. One can decide to allow the robot which detected the deadlock, to plan for all the concerned robots. The Plan-Merging paradigm remains then applicable: the inserted plan will then concern several robots at a time. Again a PMO for several robots at a time, instead of one, may fail leading, in very intricate situations, to more and more aggregation until one reaches a completely centralised system (see [Qutub *et al.*, 1997] for a detailed discussion). At this point, if the last planning robot cannot find a solution, it means that the problem is infeasible.

Here we must recall that we do not claim that the Plan-Merging paradigm can solve or help to solve multi-robot planning problems. The main point here is that the Plan-Merging paradigm is *safe* as it includes the detection of the deadlocks i.e. situations where a cooperation scheme of type **C1** should take place.

4 Application to a fleet of mobile robots

For the case of a number of mobile robots in a route network environment, we have devised a specific *Plan-Merging Protocol* based on spatial resource allocation (see [Alami *et al.*, 1995]). It is an instance of the general protocol described above, but in this context, *Plan-Merging Operation* is done for a limited list of required spatial resources: a set of cells. The robot broadcasts the set of required cells and receives back only the set of coordinated plans from other robots which have already planned to use some of the mentioned cells.

One of the most interesting property of this protocol is that it allows several PMOs to be performed simultaneously if they involve disjunctive resource sets. This is particularly useful when there are several local conflicts at the same time as it is often the case in a route network like environment.

Plan-Merging for cell occupation: In most situations, robot navigation and the associated Plan-merging procedure are performed by trying to maintain each cell of the environment occupied by at most one robot. This allows the robots to plan their trajectories independently, to compute the set of cells they will cross and to perform Plan-Merging at cell allocation level.

In order to optimise cells resource allocation and to minimise crossing obstruction, the allocation strategy is to allocate one cell ahead when the robot moves along lanes, while it allocates all the cells necessary to traverse and leave the crossing.

When reasoning about cells is not sufficient: While, most of the time, the robots may restrict their cooperation to cells allocation, there are situations where this is not enough. This happens when they have to cross large (non-structured) areas or when an unexpected obstacle, encountered in a lane or in a crossing, forces a set of robots to manoeuvre simultaneously in a set of cells. In such situations, a more detailed cooperation (using the same protocol but a different planner: the motion planner) takes place allowing robots to coordinate their actions at trajectory level. Thus, we have a hierarchy of PMOs: first, at the cell level. Then, depending on the context, at trajectory level: motion planning in a set of common cells determined by the first level. This hierarchy authorises a "light" cooperation, when possible, and a more detailed one, when needed.

Examples of Plan Merging Operation

We shall now present two examples to illustrate the use of the Plan-merging paradigm. The first one is a sequence of PMOs at Cell Level.

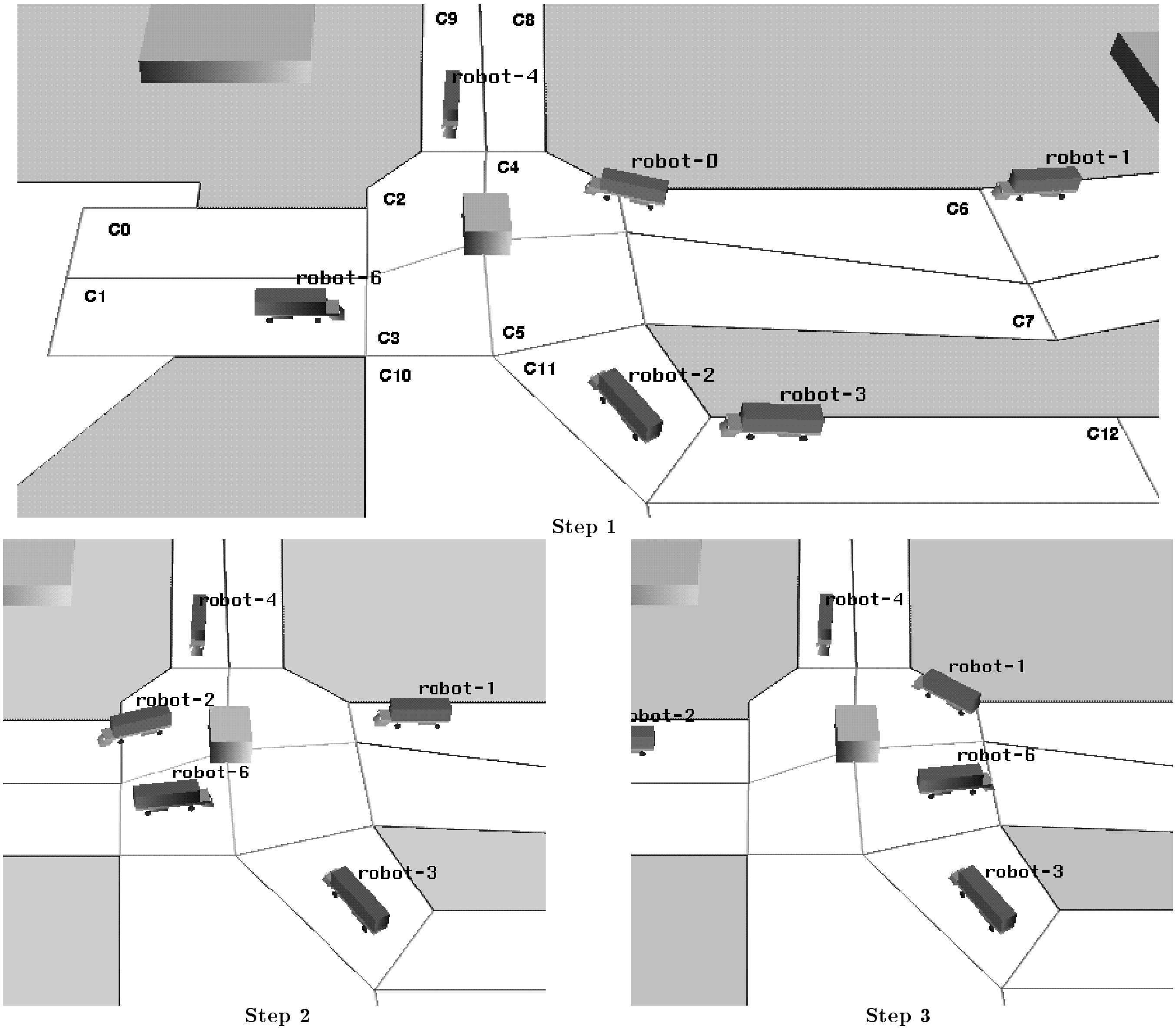

Figure 2: Plan-merging at the cell level.

Example 1: Coordination at cell level (Figure 2).

- **Step 1:** This snapshot shows the involved cells of the environment.
  The robot destinations are the followings:
  - Robots 0 and 1 on the right go to cell $C8$ above the crossing using cell $C4$.
  - Robots 2 and 3 at the bottom right traverse the crossing to reach the left cell $C0$ using cells $C5, C4$ and $C2$.
  - Robot 6 goes from left to the right cell $C7$ using cells $C3$ and $C5$.
  - Robot 4 goes from up to the lower cell $C10$ using cells $C2$ and $C3$.

  The PMOs have occurred in the following order: robot-0 then robot-2 and then robot-6 in parallel with robot-1 (because robot-6 and robot-1 have disjunctive lists of resources) and finally robot-4.

- **Step 2:** The following synchronisations have been planned: robot-2 on robot-0 (which frees $C4$), robot-6 on robot-2 (which frees $C5$) and robot-1 on robot-2 (which frees $C4$), robot-4 on robot-2 (which frees c2) and robot-6 (which frees $C3$)

- **Step 3:** One should note that at this stage, robot-3 PMO fails because robot-2 has not yet planned an action to free the cell $C0$.

Example 2: Trajectory level (Figure 3) The second example illustrates PMO at trajectories level in a large open area with two obstacles in the middle, and 10 docking/undocking stations. In such an environment, there are no cell allocations (the robots are all in the same cell), all synchronisations are made at trajectory level.

Figure 3 shows a situation where all the robots have planned and coordinated a complete trajectory. The trajectories displayed on the figure are the one which have been sent by the robots for execution display.

• The robot destinations are: $r0$ goes to station 9, $r4$ to station 5, $r1$ to station 1, $r28$ to station 7, $r27$ to station 0, $r20$ to station 4.

• PMOs were done in the following order: $r1$, $r4$, $r27$, $r20$, $r28$ and $r0$.

• One can see that synchronization hold for the following robots: $r4$ on $r1$, $r27$ on $r1$ and $r4$, $r20$ on $r1$ $r27$ and $r4$, $r28$ on $r27$ $r20$ and $r1$, $r0$ on $r1$ $r4$ $r28$ and $r27$.

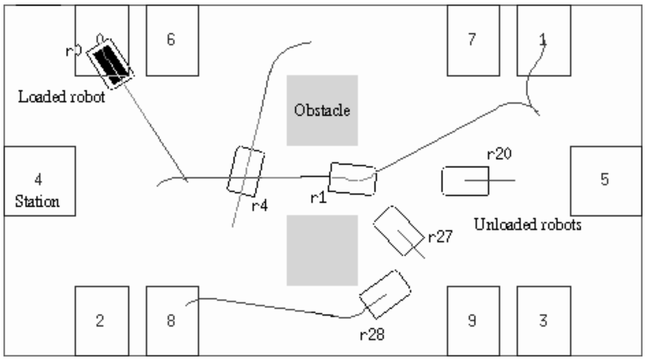

Figure 3: **Plan-merging at the trajectory level.**

5 Conclusion

We have argued that, in the field of multi-robot cooperation, it is useful to make a distinction between two main issues: **C1** goal/task decomposition and allocation, and **C2** cooperation while seeking to achieve loosely coupled goals. We have even claimed that in numerous multi-robot applications, both issues appear and even "invoke" one another "recursively".

We have then proposed a generic approach called *Plan-Merging Paradigm* which deals with **C2** issues and clearly establishes a link with **C1** issues.

The proposed scheme may be considered a priori to be too restricted (limited): it is clearly insufficient in intricate puzzle-like situations where it is necessary to take into account the conjunction of goals explicitly in one planning step.

However, it appears to be well suited and sufficient in numerous applications, particularly when the number of robots becomes important.

Besides, we use such a scheme in a framework of a paradigm which:

• guarantees the global coherence the entire system (i.e. the set of robots)

• detects the situation where it it not applicable (we call theses situations "Planning Deadlock situations")

• provides the mechanisms which allow to build up a cooperative scheme which starts from a completely decentralised system and which allows robots to progressively aggregate when then face situations that need so.

Acknowledgements: This work was partially supported by the MARTHA (ESPRIT III) Project, the CNRS, the Région Midi-Pyrénées and the ECLA ROCOMI Project.

It is the fruit of a very intensive collaboration between numerous researchers: F. Robert, S. Fleury, M. Herrb. We would like also to acknowledge the help and the effective involvement of L. Aguilar, H. Bullata, B. Dacre-Wright, M. Devy, P. Gaborit, M. Ghallab, M. Khatib, J. Perret, T. Siméon, S. Suzuki,

References

[Alami *et al.*, 1994] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki. A paradigm for plan-merging and its use for multi-robot cooperation. In *IEEE ICSMC*, 1994.

[Alami *et al.*, 1995] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *IEEE ICRA*, 1995.

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

[Durfee and Montgomery, 1991] E.H. Durfee and T. A. Montgomery. Coordination as distributed search in a hierarchical behavior spac. *IEEE Transactions on Systems, Man and Cybernetics*, 21(6), 1991.

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

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

[Jennings, 1995] N.R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intention. *Artificial Intelligence*, 73, 1995.

[Laborie and Ghallab, 1995] P. Laborie and M. Ghallab. Planning with sharable resource constraints. In *IJCAI*, 1995.

[Le Pape, 1990] C. Le Pape. A combination of centralized and distributed methods for multi-agent planning and scheduling. In *IEEE ICRA*, 1990.

[Naimi *et al.*, 1996] Mohamed Naimi, Michel Trehel, and André Arnold. A log(n) distributed mutual exclusion algorithm based on path reversal. *Journal of Parallel and Distributed Computing*, 34, 1996.

[Parker, 1994] L.E. Parker. Heterogeneous multi-robot cooperation. Technical Report AITR-1465, MIT, 1994.

[Qutub *et al.*, 1997] S. Qutub, R. Alami, and F. Ingrand. How to Solve Deadlock Situations within the Plan-Merging Paradigm for Multi-robot Cooperation. Technical Report 97-012, LAAS/CNRS, 1997.

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

[Shoham and Tennenholtz, 1995] Y. Shoham and M. Tennenholtz. On social laws for artificial societies: Off-line design. *Artificial Intelligence*, 734, 1995.

[Smith, 1994] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1994.

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