Multi-Robot Cooperation through Incremental Plan-Merging

Rachid Alami, Frédéric Robert, Félix Ingrand, Sho'Ji Suzuki

Multi-robot Cooperation through Incremental Plan-Merging*

Rachid ALAMI, Frédéric ROBERT, Félix INGRAND, Sho'ji SUZUKI
LAAS / CNRS
7, Avenue du Colonel Roche
31077 Toulouse - France

Abstract: This paper presents an approach we have recently developed for multi-robot cooperation. It is based on a paradigm where robots incrementally merge their plans into a set of already coordinated plans. This is done through exchange of information about their current state and their future actions. This leads to a generic framework which can be applied to a variety of tasks and applications. The paradigm, called *Plan-Merging Paradigm* is presented and illustrated through its application to planning, execution and control of a large fleet of a autonomous mobile robots for load transport tasks in a structured environment.

1 Introduction

We present, in this paper, an approach we have recently developed for multi-robot cooperation. It is based on a paradigm, called *Plan-Merging Paradigm*, where robots incrementally merge their plans into a set of already coordinated plans. This is done through exchange of information about their current state and their future actions. This paradigm leads to a generic framework which can be applied to a variety of tasks and applications.

In section 2 we define the framework where multi-robot cooperation takes place. In section 3 we briefly present and discuss the proposed paradigm. Section 4 discusses a typical application: a fleet of autonomous mobile robots navigating in a route network. We then discuss implementation issues and related work (section 5).

2 A framework for cooperation

2.1 Problem statement

Let us assume that we have a set of autonomous robots and a central system which, from time to time, assigns and sends goals to robots individually.

Whenever it receives a goal, a robot is assumed to elaborate and execute a plan which achieves it. Goals may be sent asynchronously to the robots even if they are still processing a goal previously sent.

Each robot 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.

However, before executing any plan step, a robot has to ensure that it is valid in the current multi-robot context, i.e. that it is compatible with all the plans currently under execution by the other robots. This will be done *without modifying* the other robots plans, in order to allow the other robots to continue execution.

We call this operation, a *Plan-Merging Operation* (PMO) and its result a *Coordination Plan* (i.e. a plan valid in the current multi-robot context).

Planning, plan merging and execution may run in parallel. In fact, considering the time needed for planning and plan merging operation, and considering the average range of the obtained coordination plans, execution run most often without waiting for a new coordination plan.

2.2 The "global plan" and its properties

Everything works as if there was a *global plan* produced and maintained by the set of robots. In fact, neither the robots nor the central station elaborate, store and maintain such a *global plan*[1].

At any moment, a robot has its own *coordination plan* under execution. Such a *coordination plan* consists of a sequence of actions and events to be signaled to other robots as well as events which are planned to be signaled by other robots. Such events correspond to state changes in the multi-robot context and represent temporal constraints (precedence) between actions involved in different individual coordination plans.

At any moment, the "global plan" is the graph representing the union of all current robot coordination plans. Such a global plan is valid (i.e. it does not contain inconsistent temporal constraint) if it can be represented by a *directed acyclic graph* (*dag*).

The key point here is how to devise a system composed of a set of robots which should, as much as possible, plan independently to achieve their tasks while maintaining such property of the global plan.

*This paper has been submitted to 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan.

[1]Note that the central station maintains a higher level description of the set of missions allocated to the robots. However it does not need to know the plans elaborated by the robots to achieve their missions and how these plans are coordinated.

3 The Plan-Merging Paradigm

Let us assume here that:

1. there exists a mean which allows a robot to get the right to perform a PMO while having the guarantee that it is the only robot doing so. This right should be thought of as a resource allocation[2].
2. there is a mean allowing a robot (which has obtained the right to perform a PMO) to ask for and obtain all the other robots coordination plans.
3. there exists a mean allowing robots to ask or inform one another about the occurrence of an event.

3.1 Performing a *Plan Merging Operation*

Robots are assumed to plan from time to time (whenever it is necessary).

When a robot is not planning and even when it is waiting to obtain the right to perform a PMO, it must be able to send its current coordination plan to another robot (which currently has the right to perform a PMO).

When a robot has to plan, it uses the following protocol which we call the *Plan Merging Protocol* (see Figure 1):

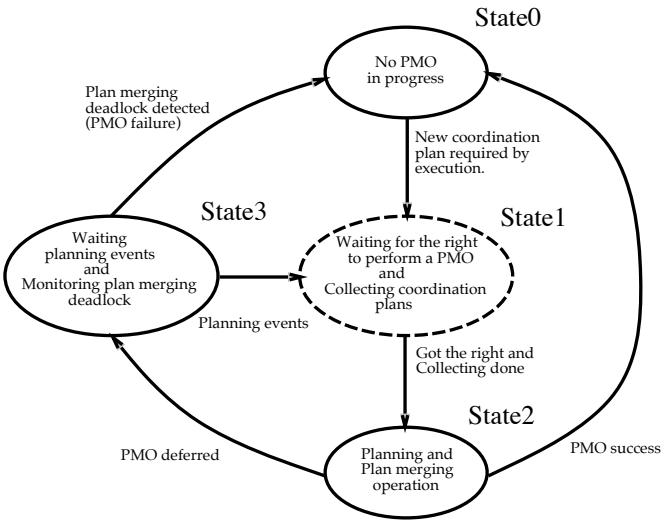

Figure 1: The general protocol state graph

1. It asks for the right to perform a PMO and waits until it obtains it together with the coordination plans of all the other robots.
2. It then builds the *dag* corresponding to the union of all coordination plans (including its own coordination plan)
3. It then tries to produce a new plan which can be inserted in the *dag*, *after* its current coordination plan. The new plan insertion may only add temporal constraints which impose that some of its actions must be executed after some time-points from other robots coordination plans.
   Besides, the insertion must maintain the fact that the obtained global plan is still a *dag*.
4. If it succeeds in producing the desired plan, the robot appends it to its current coordination plan.
5. And finally, it releases the right to perform a PMO.

When a robot executes its coordination plan, if it reaches a step with a temporal constraint linked to another robot time-point, it asks that robot if it has passed that time-point or not. Depending on the answer, the robot will wait until the other robot informs it or will immediately proceed.

3.2 Situations where PMO is deferred or where deadlock is detected

When a robot tries to perform a PMO, it may fail to produce a plan which satisfies the properties discussed earlier.

This may happen in two situations:

1. the goal can never be achieved. This can be detected if the robot cannot produce a plan even if it were alone in the environment. The robot informs the station and waits for a new goal.
2. the robot can generate a plan but this plan cannot be inserted in the global plan. This means that the final state of another robot forbids it to insert its own plan.
   In such situation, the robot can simply abandon the 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, we have introduced two types of events:

1. *execution events:* i.e. events which occur during plan execution and which allow robots to synchronize their execution.
2. *planning events:* i.e. events which occur whenever a robot performs a new PMO. These events can also be awaited for.

Note that, even when a robot fails in its PMO, it leaves the global plan in a correct state (it is still a *dag* and its execution can continue).

In order to detect deadlocks, a robot which finds itself in a situation where it has to wait for a *planning event* from a particular robot, must inform it. Then, it becomes possible for a robot to monitor and detect *deadlock situations* by propagating and updating, the list of robots waiting (directly or by transitivity) for

[2] A simple way to do it is to maintain a "token" through communication; but this is not always desirable nor even possible (see section 4.2).

planning events from itself. Indeed, a deadlock is detected when a robot finds itself in the list of robots waiting for itself.

When a deadlock occurs, it is necessary to take explicitly into account, in a *unique planning operation*, a conjunction of goals (which have been given separately to several robots).

This simply means that the global mission was too constrained to be solved using the Plan-Merging Paradigm. It is then the responsibility of the central station to produce a multi-robot plan.

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.

Note also that, in the case where only a small number of robots are involved in a deadlock, 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.

A detailed discussion on the properties of the Plan-merging paradigm as well as on its ability to cope with execution failures can be found in [2].

3.3 Discussion

The paradigm and the protocol presented so far is generic. We believe that it can be used in numerous applications. Several instances of the general paradigm can be derived, based on different planners: action planners in the stream of STRIPS, as well as more specific task planners or motion planners.

One class of applications which seems particularly well suited is the control of a large number of robots in a route network.

We present in the sequel an application in the case of a fleet (dozens) of autonomous mobile robots. The use of the Plan-Merging paradigm allowed us to deal with several types of conflicts in a general and systematic way.

4 A fleet of autonomous mobile robots

We have applied the Plan-Merging Paradigm in the framework of the MARTHA project[3] which deals with the control of a large fleet of autonomous mobile robots for the transportation of containers in harbors, airports and railway environments.

In such context, the dynamics of the environment, the impossibility to correctly estimate the duration of actions (the robots may be slowed down due to obstacle avoidance, and delays in load and un-load operations, etc..) prevent a central system from elaborating efficient and reliable detailed robot plans.

The Plan-Merging paradigm is well suited to such applications where conflicts are local and involve a limited number of robots. Indeed, its use allowed us to limit the role of the central system to the assignment of tasks and routes to the robots (without specifying any synchronization between robots) taking only into account global traffic constraints.

[3]MARTHA: European ESPRIT Project No 6668. "Multiple Autonomous Robots for Transport and Handling Applications"

4.1 Mission processing

In the MARTHA application, the robots are regularly assigned missions: destination points together with routes and operations to perform when the destination points are reached (docking, loading..).

The environment is a route network: lanes, crossings, open areas. In order to allow efficient and incremental plan merging, we have decomposed the route network into smaller entities called "cells" or "spatial resources" which will be used as a basis for dealing with local conflicts. Basically, the robots navigate through an oriented graph of cells.

When a robot receives a mission, it first refines into an executable plan: a set of trajectories planned autonomously together with the sequence of resources it has to allocate.

Plan-merging will essentially consist in synchronizing the use of such resources by the different robots through inter-robot communication. It takes place from time to time, for a limited number of spatial resources ahead in order to not constrain unnecessarily the other robots.

Mission planning (refinement), plan-merging and execution may run in parallel (see Figure 2), allowing most often the robots not to stop at all (unless they have to effectively wait for a resource which is still occupied by another robot).

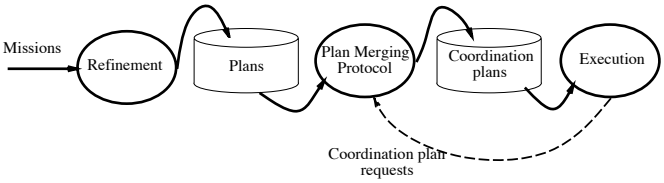

Figure 2: Interactions between mission refinement, plan-merging and execution

4.2 A Plan-Merging Protocol for multi-robot navigation in a route network

We have devised a specific Plan-Merging protocol based on resource allocation. It is an instance of the general protocol described in §3.1, where *State_1* is decomposed into several sub-states (see Figure 3).

To present the *Plan-Merging Protocol* in the particular case of traffic application, let us recall that, in *State_1*(see Figure 3), the robot should obtain the right and the necessary data to perform a *Plan-Merging Operation.*

As, in this context, *Plan-Merging Operation* is done for a limited list of required resources (the cells which will be traversed during the plan to merge), a robot,

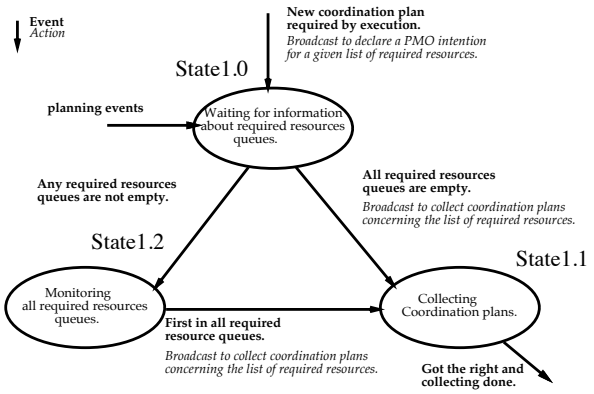

Figure 3: The state1 for traffic application in the protocol state graph

by broadcasting this list, announces its intention to start a PMO:

- No response, means that all required resource queues are empty.
- If another robot is in *State_1.1*, *State_1.2* or *State_2* and if its list of required resources intersect the list contained in the broadcast message, it sends immediately a message *(WAIT-FOR-PMO, robot-name, resource-intersect-list)*.
- If several robots enter simultaneously in *State_1.0* for common resources, the conflict is solved by taking into account the robots priority. Doing so, robots maintain together for each critical resource a queue, without risk of starvation or deadlock.

Due to place limitations, we will not describe in more detail this protocol. A full description may be found in [13].

One of the most interesting attributes 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 .

4.3 When reasoning about cells is not sufficient

While, most of the time, the robots may restrict their cooperation to cell allocation, there are situations where this is not enough. This happen when they have to cross non-structured regions (called "areas") or when an unexpected obstacle, encountered in a lane or in a crossing, forces a set of robot to maneuver simultaneously in a set of cells.

When this happens, 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

1. first, at the cell level, based on resource (cells) allocation
2. then , depending on the context, at trajectory level: motion planning in a set of common cells determined by the first level

This scheme authorizes a "light" cooperation, when possible, and a more detailed one, when necessary, obtained by a further refinement of the *Global Plan* without altering the properties.

4.4 An example of Plan-Merging at a crossing

We shall now illustrate the plan merging paradigm and protocol with a concrete example from the MARTHA application. We have chosen an allocation strategy which makes the robots allocate one cell (at least) ahead when they traverse lanes, while they allocate all the cells necessary to cross and leave a crossing.

The example involves several robots at a crossing divided into four cells with entry and exit cells belonging to four lanes (Figure 4):

(Instant: t0) Robot R_1 is entering c_4 and has a plan to go through c_{11}, c_{12}, and enter c_6. We shall now examine different steps (Figure 4) and show how other robots willing to go through this crossing will coordinate their plans together.

t1: R_1 has already merged its plan to traverse the crossing. R_2 needs to go through c_{12}, c_6 (i.e. a crossing cell and an exit cell); it cannot produce and merge a plan compatible with R_1's plan, since R_1 has not yet merged a plan in which it frees c_6. As a consequence R_2 defers its PMO (switch from *State_2* to *State_3*), and asks R_1 to produce a "planning-event" as soon as it has elaborated a new plan (which supposedly will contain a release operation for c_6). R_2 will remain in *State_3* until R_1 signals a *"planning-event"*. Note that R_2's execution will continue until its current coordination plan is done i.e. until the c_{12}.

t2: R_3 and R_4 attempt to start a PMO at (almost) the same time. Let us assume than R_4 has a lower priority than R_3. R_4 switches to *State_1.2* and R_3 proceeds. It merges a new plan which makes use of c_{10}, c_9 and c_3 without interfering with any other robot coordination plan.

t3: R_4 receives a message from R_3 which is now done with its PMO. R_4 switches from *State_1.2* to *State_1.1*, it broadcasts the list of its required resources (c_{11}, c_{12}, c_{10} and c_2) and gets back coordination plans from R_1 and R_3. It merges a plan with temporal constraints referring to R_1 and R_3 plans.

t4: After a while, R_3 and R_1 need again new PMOs. There is no interference between their respective required resources. Therefore, they can perform their PMOs in parallel.

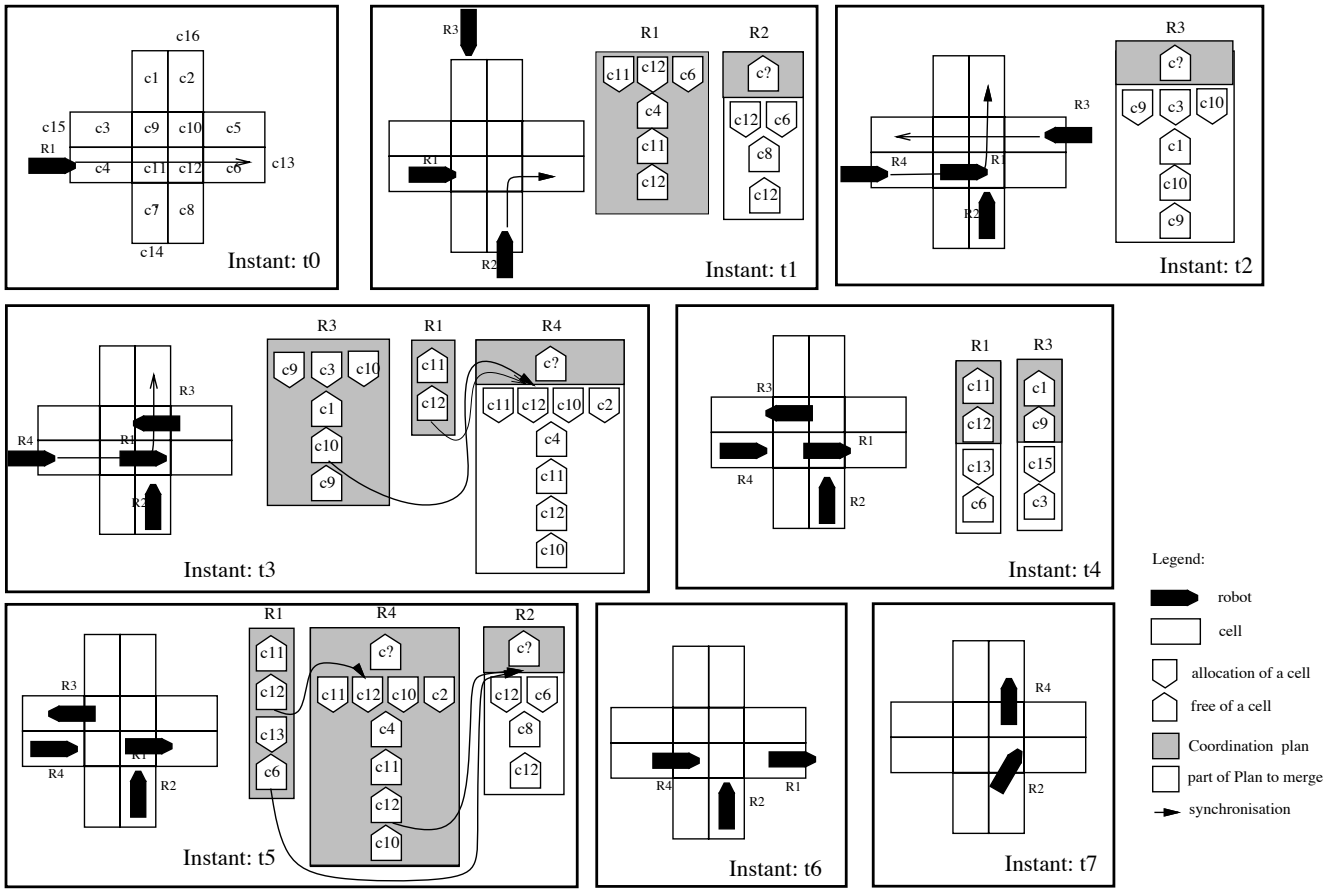

Figure 4: Plan Merging in a Crossing

t5: R_1 has now produced a new plan. It sends a *"planning-event"* to R_2 which can now switch from *State_3* to *State_1.0* and proceed.

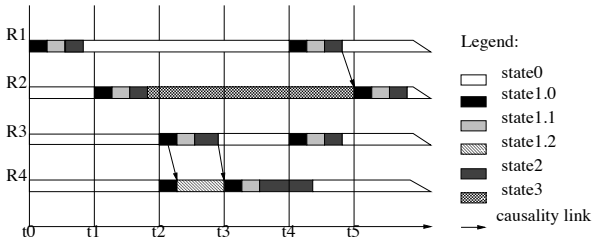

Figure 5: State change chronogram

We can make a number of remarks from this example and its associated state change chronogram (see Figure 5).

- One can see that planning and execution is done in parallel.
- More than one robot can perform a PMO at the same time for disjunctive lists of resources (e.g. R_1 and R_3 at instant **t4**).
- Several robots may use the crossing simultaneously (e.g. R_1 and R_3).
- The example exhibits the two types of synchronization: synchronization based on *execution events* (e.g. R_4 will wait until R_1 leaves c_{11}), and synchronization based on *planning events* (e.g. R_2 waits R_1 has produced and merged a new plan, at instant t5).
- Each robot produces and merges its plans iteratively, and the global plan for the use of the crossing is incrementally built through several PMOs performed by various robots.
- It is not a first arrived first served execution; for example R_2 arrived second, was blocked by R_1, but did not block the crossing and let R_3 and R_4 enter the crossing before it.

4.5 The current implementation

We have developed a complete robot control system which includes all the features described. Its ar-

chitecture is based on generic control architecture for autonomous mobile robots developed at LAAS [4, 1].

It is instantiated in this case by adding an intermediate layer for performing Plan-Merging operations (Figure 2).

The robot supervisor is coded using a C-PRS [7]; it performs Plan-Merging Operations based on a topological planner and on a motion planner. Figure 6 illustrates two trajectories planned and synchronized using a PMO at trajectory level.

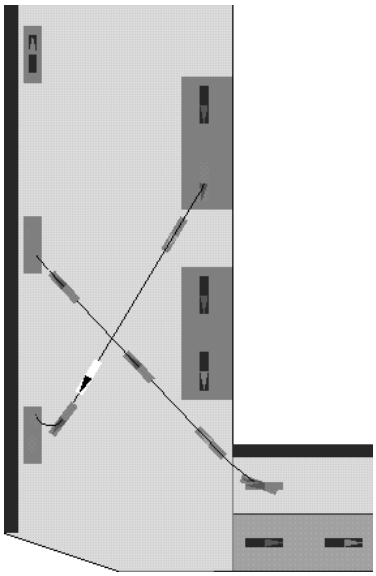

Figure 6: The result of a PMO at trajectory level

For testing and demonstration purposes, it has been linked to a robot simulator.

Experiments have been run successfully on a dozen of workstations (each workstation running a complete robot simulator) communicating through Ethernet. A 3-d graphic server has been built in order to visualize the motions and the load operations performed by all the robots in a route network environment (Figure 7). The simulated robots where able to achieve navigation missions, performing hundreds of PMOs and solving local conflicts. Motion planning and PMOs were sufficiently efficient to allow most often the robots to elaborate and merge their plans without stopping unless necessary.

The software is currently been installed under a real-time multi-processor operating system for future experimentations on real robots.

5 Related work

There are numerous contributions dealing with multi-robot cooperation. However, the term "cooperation" has been used in several contexts with different meanings.

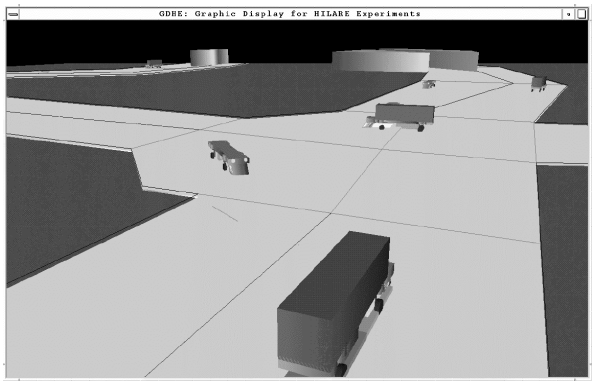

Figure 7: Several simulated robots at a crossing

We will not consider here contributions to cooperation schemes at servo level (e.g. [10]) nor contributions which aim at building an "intelligent group" of simple robots (e.g. [11]). We will limit our analysis to contributions which involve an effective cooperation (at plan or program level) between several robots.

Several approaches have been proposed, such as generation of trajectories without collision (e.g. [5, 14]), traffic rules [6, 8], negotiation for dynamic task allocation [9, 3], and synchronization by programming [12, 16].

Inter-robot communication allows to exchange various information, positions, current status, future actions, etc [3, 16, 15] and to devise effective cooperation schemes.

Traffic rules have been proposed as a way to allow several robots to avoid collision and to synchronize their motion (with limited or even without communication). However, many aspects should be taken into account in order to build the set of traffic rules: the tasks, the environment, the robot features, and so on. This entails that the generated rules are valid only under the considered assumptions. If some of them are changed, the rules have to be modified or sometimes be regenerated completely. Besides, these systems are generally built heuristically and do not provide any guarantee such as deadlock detection.

Negotiation have been used for dynamic task [3] or resource [9] allocation to several robots depending on the situation. One robot or a station allocates tasks to negotiators determined through communication.

Most contributions which make use of synchronization through communication are based on a predefined set of situations or on task dependent properties.

Indeed, most of the methods listed here, deal essentially with collision avoidance or motion coordination and cannot be directly applied to other contexts or tasks.

We claim that our Plan-Merging paradigm is a generic framework which can be applied in different

contexts, using different planners (action planners as well as motion planners). It has some clean properties (and clear limitations) which should allow, depending on the application context, to provide a coherent behavior of the global system without having to encode explicitly all situations that may encoded.

6 Conclusion and future work

The Plan-Merging paradigm we propose has the following properties;

1. It makes possible for each robot to produce a coordination plan which is compatible with all plans executed by other robots.
2. No system is required to maintain the global state and the global plan permanently. Instead, each robot updates it from time to time by executing a PMO.
3. The PMO is safe, because it is robust to plan execution failures and allows to detect deadlocks.

We believe that it can be applied to a large variety if contexts and with different planners (from action planners to task or motion planners), and at different granularities.

Such a multi-robot cooperation scheme "fills the gap" between centralized, very high level planning and distributed execution by a set of autonomous robots in a dynamic environment.

Indeed, it appears to be particularly well suited to the control of a large number of robots navigating in a route network. The application that we have implemented clearly exhibits its main features. It allowed us to make a large number of autonomous robots behave coherently and efficiently without creating a huge activity at the central system.

Besides the demonstration of real robots and the investigation of other classes of applications, our future work will concentrate on developing new cooperation schemes by embedding a multi-robot planning activity inside a PMO.

References

[1] R. Alami, R. Chatila, and B. Espiau Designing an Intelligent Control Architecture for Autonomous Robots. in ICAR'93, Tokyo (Japan), October 1993.

[2] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki. A paradigm for plan-merging and its use for multi-robot cooperation. In *IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Texas (USA), October 1994.

[3] H. Asama, K. Ozaki, et al. Negotiation between multiple mobile robots and an environment manager. In *'91 International Conference on Advanced Robotics (ICAR),Pisa (Italy)*, June 1991.

[4] R. Chatila, R. Alami, B. Degallaix, and H. Laruelle. Integrated planning and execution control of autonomous robot actions. In *IEEE Int. Conf. on Robotics and Automation, Nice, (France)*, 1992.

[5] H. Chu and H.A. EiMaraghy. Real-time multi-robot path planner based on a heuristic approach. In *IEEE International Conference on Robotics and Automation, Nice, (France)*, May 1992.

[6] D.D. Grossman. Traffic control of multiple robot vehicles. *IEEE Journal of Robotics and Automation*, 4(5):491–497, Oct. 1988.

[7] F. F. Ingrand, M. P. Georgeff, and A. S. Rao. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert, Knowledge-Based Diagnosis in Process Engineering*, 7(6), Dec. 1992.

[8] S. Kato, S. Nishiyama, and J. Takeno. Coordinating mobile robots by applying traffic rules. In *IEEE IROS '92, Raleigh (North Carolina, USA)*, July 1992.

[9] C. Le Pape. A combination of centralized and distributed methods for multi-agent planning and scheduling. In *IEEE International Conference on Robotics and Automation, Cincinnati, (USA)*, May 1990.

[10] Z.W. Luo, K. Ito, and M Ito. Multiple robot manipulators cooperative compliant manipulation on dynamical environments. In *IEEE IROS '93, Yokohama, (Japan)*, July 1993.

[11] M. Mataric. Minimizing complexity in controlling a mobile robot population. In *IEEE International Conference on Robotics and Automation, Nice, (France)*, May 1992.

[12] F.R. Noreils. Integrating multi-robot coodination in a mobile-robot control system. In *IEEE IROS '90, Tsuchiura (Japan)*, July 1990.

[13] F. Robert, R. Alami, F. F. Ingrand, and S. Suzuki. A Paradigm for Plan-Merging and its use for Multi-robot Cooperation. Technical Report feb-94, LAAS/CNRS, Toulouse, France, 1994.

[14] T. Tsubouchi and S.Arimoto. Behavior of a mobile robot navigated by an "iterated forecast and planning" scheme in the presence of multiple moving obstacles. In *IEEE Int. Conf. on Robotics and Automation, San Diego, (USA)*, May 1994.

[15] J. Wang. On sign-board based inter-robot communication in distributed robotic systems. In *IEEE International Conference on Robotics and Automation, San Diego, (USA)*, pages 1045–1050, May 1994.

[16] S. Yuta and S.Premvuti. Coordination autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE IROS '92, Raleigh (North Carolina, USA)*, pages 1566–1574, July 1992.