How to Solve Deadlock Situations within the Plan-Merging Paradigm for Multi-robot Cooperation*

S. Qutub, R. Alami, F. Ingrand
LAAS-CNRS
7, Avenue du Colonel Roche, 31077 Toulouse CEDEX 04
E-mail: {sam, rachid, felix}@laas.fr

Abstract

Our motivation in proposing the Plan-Merging Paradigm as a cooperation scheme was to allow an efficient distribution of the decisions for a better reactivity to contingencies for multi-robot applications with loosely coupled tasks.

The paradigm proved to be quite efficient because it exploits the fact that most conflicts can be solved locally and because it allows a finer overlapping between plan refinement, plan coordination and execution.

However, we would like to have a scheme which distributes as much as possible the decision processes for planning and coordination while maintaining two key features:

- the coherence of the global system 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.

We develop in this paper the ingredients which guarantee such features as well as their consequences in terms of requirements for the task planners involved.

1 Introduction

We have already presented and discussed the Plan-Merging Paradigm (PMP), a generic scheme for multi-robot cooperation [3, 4, 1]. It is based on an incremental and distributed plan-merging process.

We have applied the PMP to multi-robot applications [2] with loosely coupled tasks, where each robot has a local view and a partial knowledge of the other robots activities. We showed that the PMP is quite efficient because it exploits the fact that most conflicts can be solved locally and because it allows a finer overlapping between plan refinement, plan coordination and execution.

We have also discussed the key features of this distributed cooperative scheme related to the coherence of the global system and its ability to detect situations where it is not applicable i.e. situations where it is necessary to take into account a conjunction of goals. We call such situations "Planning deadlock situations".

We present here a set of extended operators and complementary mechanisms which permit a localized management of the planning and coordination processes as well as a progressive transition to more global schemes which may even "degrade" to a unique and centralized planning activity. The result is a generic multi-robot cooperative scheme which is well suited for loosely coupled tasks but is able to treat any conflicting situation.

In section 2, we present a short discussion of related work. Section 3 describes briefly the *PMP* and explains how tasks dependencies are detected and propagated (through communication) in order to maintain the global system coherence and to detect planning deadlock situations. Section 4 introduces the notion of *Local Multi-robot Planning* that allows us to modify dynamically the distributed nature of the system in order to deal with the deadlock situations. Section 5 describes an implemented system which illustrates our approach and describes the behavior of a set of mobile robots in a very constrained environment.

2 Related Work

We limit our discussion here to multi-robot issues which involve the simultaneous operation of several autonomous agents, each one seeking to achieve its own task or goal. The conflicts proceed from the fact that the robots intend to use common resources simultaneously (narrow passages, crossings, devices, etc).

Many approaches have been proposed to deal with this problem especially *centralized* approaches where a central system determines a set of non-conflicting plans that solves the conflicts[11]. These approaches suffer from deficiencies in realistic applications when the number of robots becomes important. Other approaches are based on predefined *traffic rules*, which are only applicable to "route networks" modeled environment. In such cases, it is very difficult to find a set of free-deadlock rules for all the possible situations [7, 9]. Some *reactive systems* have been proposed

*This paper has been published in the proceedings of IEEE IROS 97, Grenoble, France.

where the robots actions are the direct consequences of the information collected by the robots sensors [6] or through communication [8]. While the results of such approaches may be inefficient due to the local decision making based on sensory information, the main limitation here is that there is no guarantee of a global coherence of the system. In [5] the authors propose an idea mixing *sensory* information with a world discrete model to solve conflicts for a small number of robots. Finally, a *master-slave* approaches have been proposed where a robot becomes the master of the blocked robots during the conflict, resolves the conflicts and distributes the solutions [12]. In this paper, we refine this master-slave approach by mixing it with the *Plan-Merging Paradigm* to generate, during a deadlock situation, a set of plans that will be validated in the global context (through a Plan-Merging Operation) before the execution phase.

3 The Plan-Merging Paradigm

Due to space limitations, we give a short presentation of the Plan-Merging Paradigm and we insist only on the situations where it does not apply. The interested reader may refer to previous papers [3, 4, 1] for a more detailed presentation of this cooperation scheme and its applications.

Let us assume that we have a set of autonomous robots and a central station which, from time to time, sends goals to robots individually. Whenever a robot R_i receives a new goal G_i^j, it elaborates an *Individual Plan* (IP_i^j) which achieves it. Each robot processes sequentially the received goals. Doing so, it incrementally appends new actions to its current plan.

However, before executing any plan step, a robot must ensure that it is valid in the multi-robot context, i.e. that there exists no other plan of another robot which may conflict with it. We call this operation *Plan Merging Operation* (PMO) and the resulting plan a *Coordinated plan* (i.e. plan valid in the current multi-robot context). Such a *Coordinated Plan* (CP_i) consists of a sequence of actions and *execution events* to be signaled to other robots as well as *execution events* that are planned to be signaled by the other robots. Such *execution events* correspond to temporal constraints between actions involved in different coordinated plans.

At any moment, the temporal constraints between all the actions included in the union of all the coordinated plans ($GP = \bigcup_k CP_k$) must constitute a *directed acyclic graph* [3, 4]. GP is a snapshot knowledge of the current global situation and its already planned evolution.

3.1 The PMO and its results

When R_i receives its j-th goal G_i^j, it elaborates a plan IP_i^j which achieves it; then it performs a PMO in a *critical section*: it collects the coordinated plans CP_k of the robots which may interfere with IP_i^j, and builds their union $GP = \bigcup_k CP_k$. The insertion of IP_i^j in the global plan GP, if it succeeds, adds temporal order constraints to actions in IP_i^j and transforms it into a coordinated plan CP_i. The out-coming CP_i is feasible in the current context, and does not introduce any cycle in the resulting GP.

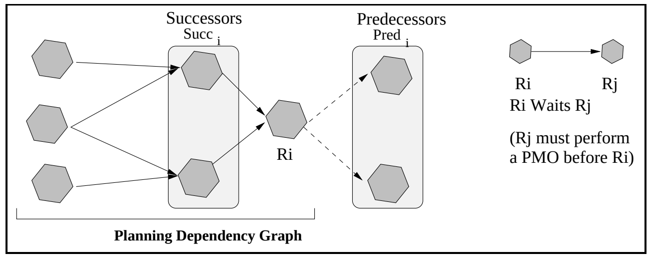

Figure 1: *R_i Planning Dependency Graph PDG_i and $Pred_i$.*

The PMO is protected by a critical section in order to prevent other robots to perform simultaneously a modification of GP.

However a PMO may fail because the final state of at least another robot (as specified in GP) forbids R_i to insert its own plan. Let us call $Pred_i$ (predecessors) the set of all such robots. In this case, R_i defers its PMO and waits until one of the robots in $Pred_i$ has performed a new successful PMO which may possibly change the states preventing R_i to insert its plan. Hence, we introduce temporal order relations between robots plan-merging activities.

In addition to *execution events* — i.e. events elaborated by the PMO and which allow agents to synchronize their execution —, we define *planning events* — i.e. events which occur whenever a robot performs a new successful PMO. The *planning events* can also be awaited for. The temporal relations between robots plan-merging activities are maintained by each robot in an additional data structure called *Planning Dependency Graph PDG_i* (Figure 1).

The *Planning Dependency Graph* serves to manage PMOs order (when necessary) as well as to detect and prevent any robot to enter a *waiting cycle*, where it would wait for itself by transitivity. We call such a situation a "Planning Deadlock Problem". The detection of deadlocks at the planning/coordinating phase permits to anticipate and avoid deadlocks during the execution phase where "backtracks" are not always possible or induce inefficient maneuvers.

The "planning deadlock problem" emphasizes the fact that the PMO is unable to take into account a set of goals, sent to different robots, with strong interdependencies. This limitation leads us to elaborate an extension to the PMP that allows the use of planning from a distributed to a more centralized scheme and from a local to a more global resolution.

3.2 Dependency Graph Construction

This section focuses on the incremental construction of the Planning Dependency Graph PDG_i and its constraints propagation mechanism.

Each robot R_i maintains a list $Pred_i$ and a graph PDG_i. $Pred_i$ is the list of all of robots that block its plan-merging activity. PDG_i specifies [1] all the robots that depends on R_i, directly or by transitivity, for their plan-merging activities.

We call $Succ_i$ (successors) the set of robots that are directly waiting for a *planning event* from R_i (Figure 1).

PDG_i is maintained through the following procedure:

⋄ When a robot R_i starts a PMO, $Pred_i$ is set to the empty list. After the PMO:

- if the PMO has succeeded: R_i signals a *planning event* to all robots in $Succ_i$ and clears its current graph PDG_i.
- if the PMO has failed: R_i determines $Pred_i$ and checks if it induces planning dependencies which produce a cycle in PDG_i:
  - in such a case, a *deadlock situation* is detected which means that the given goals are interdependent, they cannot be treated simply by insertion, but need to be handled by a planner that takes into account the *conjunction* of goals of all the robots involved in the cycle (see §4).
  - If the newly established *planning dependencies* do not introduce any cycle in PDG_i, R_i transmits PDG_i to all robots in $Pred_i$.

⋄ When a robot R_k receives PDG_i from a robot R_i, R_k adds it to its own Dependency Graph PDG_k and propagates this information to all robots in $Pred_k$. R_k is sure that the received PDG_i can be added to PDG_k without creating any cycle[2].

4 Deadlock Resolution Strategy

The deadlock resolution strategy that we present is based on cooperative (not competitive) robots behavior. We assume that all robots are equipped with a multi-robot planner which can be used, when necessary, for an arbitrary number of robots.

4.1 General presentation

Let us call DL_i the set of robots involved in a cycle detected by R_i. When detecting such a cycle, R_i has the necessary information in PDG_i to elaborate and validate a plan for all the robots in DL_i. Note that the blocked robots are unable to add any new executable action to their current coordinated plans CP_k. Therefore, if nothing is done, they will come to a complete stop when their plans CP_k have completed.

[1] A node in PDG_i represents a set of robots, their current states and their goals.

[2] If such cycle existed, R_i would have discovered it.

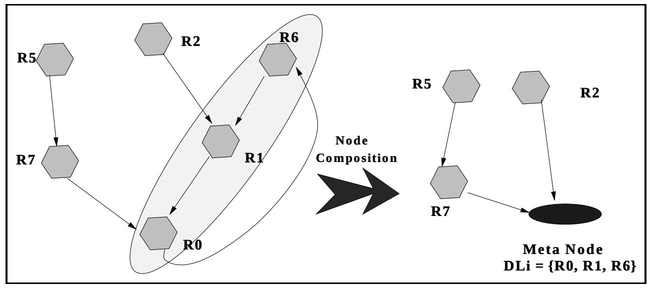

Figure 2: *Node Composition in R_0 Dependency Graph (PDG_0).*

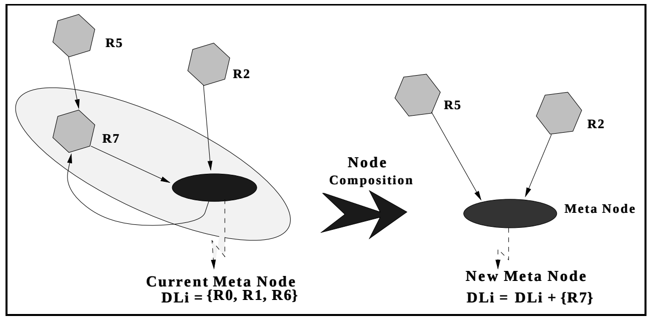

Figure 3: *Forming a new Meta-Node in PDG_0 by unifying the current Meta-Node and the newly detected deadlock.*

To solve the deadlock, the robot R_i becomes (temporarily) the local coordinator (noted R_i^{LC}) for all robots in DL_i. To do so, it makes use of its *Local Multi-robot Planner* that will take explicitly, in one planning operation, the conjunction of goals of all robots in DL_i. This fact will be represented in its Dependency Graph PDG_i^{LC} as a *Meta-Node* (Figure 2) which includes all robots in DL_i.

R_i becomes a local coordinator R_i^{LC} whose responsibility is to:

1. Find a multi-robot solution (Sol_i^{LC}), if it exists, to the conjunction of goals. This solution is represented by a *lattice* whose nodes are high level actions to be performed to break the cycle and whose arcs are "deadlock synchronization events" between these actions.
2. Try to insert Sol_i^{LC} in the current multi-robot context, i.e. the set of current plans CP_k of the robots which are not involved in DL_i:
   2.1. If the insertion succeeds, R_i^{LC} sends to each robot in DL_i its plan and waits for an acknowledgment. If all the blocked robots accept these plans, the coordinator R_i^{LC} gives the permission to start the execution. The deadlock cycle is broken. Each robot in DL_i recovers its "planning and plan-merging" autonomy.
   2.2. If the insertion fails, this means that the final state of at least one robot (not included in

DL_i) forbids R_i^{LC} to insert Sol_i^{LC}. R_i^{LC} determines the set of such robots $Pred_i^{LC}$. It then verifies that $Pred_i^{LC}$ does not create a cycle when inserted PDG_i^{LC}:

2.2.1 if no cycle is created, R_i^{LC} defers its PMO, transmits PDG_i^{LC} to all robots in $Pred_i^{LC}$ and waits until one of them has performed a new successful PMO;

2.2.2 if a new deadlock DL_i^{LC} is detected, R_i^{LC} generates a new *Meta-Node* containing the union of DL_i and DL_i^{LC}. It then restarts the same process (Figure 3), acting as a coordinator of a greater set of robots.

4.2 Deadlock Automata

We describe here the finite state automata (Figure 4, 5) that defines the behavior of the robots during a deadlock situation.

Figure 4 describes the behavior of a robot[3] R_i when it detects a deadlock DL_i. R_i sends a message *(Deadlock-give-info R_i)* to all robots in DL_i (state 0) and waits for replies. If it receives*(Cycle(DL_k))* from $R_k \in DL_i$ (state 3), this means that R_k is the coordinator of another deadlock DL_k; a new *Meta-node* is created containing the union of $DL_i \Leftarrow DL_i \cup DL_k$. When all the expected replies are received, R_i becomes the local coordinator R_i^{LC} of all robots in DL_i. It invokes its Local Multi-robot Planner in order to find a plan Sol_i^{LC} for the conjunction of goals of DL_i (state 2); then it distributes Sol_i^{LC} to the robots in DL_i and waits for an acknowledgment (state 4).

Figure 5 describes the behavior of a R_j, involved in a cycle DL_i when it receives a message *(Deadlock-give-info R_i)* from a coordinator R_i. There are three possible cases:

1. R_j is the coordinator of another deadlock DL_j (state 4) : it transmits the message *(Cycle(DL_j))* to R_i; This message contains all the necessary information concerning DL_j. DL_j will be merged with DL_i (see also state 3 in Figure 4).
2. R_j participates in another cycle DL_k whose coordinator is R_k ($R_j \neq R_k$)(state 3) : it transmits the message *(Deadlock-give-info R_i)* to R_k.
3. R_j does not participate to any cycle (state 2) : it sends its current state and goal to R_i and waits for a plan from it.

4.3 Discussion

The problem discussed in this paper is a typical problem in distributed multi-robot applications where each robot does not have a global view of the world. To solve it, we have accepted to reduce momentarily the "distribution level" of a part of the system at some very particular instants to increase its ability to treat intricate situations. When a subset of robots enters in a deadlock cycle, the robot that detects the cycle becomes the *cycle coordinator* whose responsibility is to find a solution to the problem. If the coordinator finds a solution, it tries to validate it in the global plan GP, constructed from the coordinated plans of the non-blocked robots, by a PMO. If the insertion succeeds, the coordinator distributes the solution to the concerned robots and the system returns to its initial distributed state (Figure 6).

[3] R_i acts here for itself or as a local coordinator for a set of robots determined in a former step

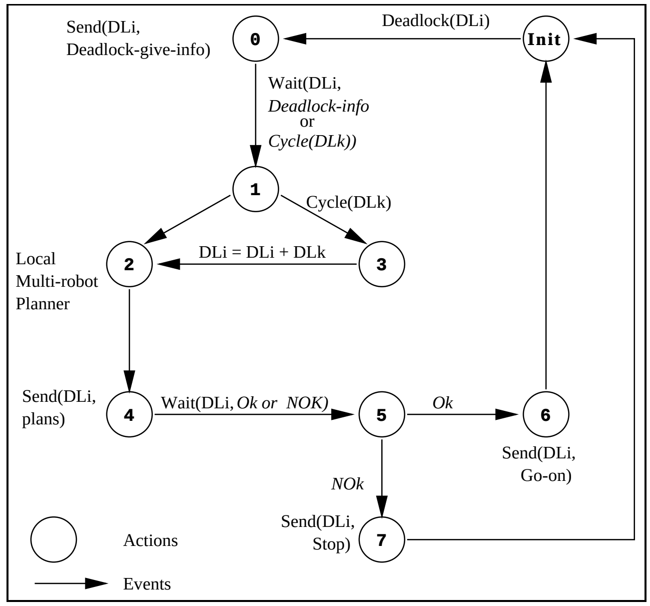

Figure 4: *The coordinator finite state automate.*

If the insertion fails and produces a new cycle, the coordinator recursively applies the same algorithm to the current *Meta-Node* and to the detected cycle to create a new *Meta-Node*. So, we may imagine some very complicated situations where the *Meta-Node* starts to grow up and does not stop until the inclusion of the whole system (all robots). In such situation, our completely distributed system tends to a completely centralized system (Figure 6).

Note also that we may have, in parallel, many deadlocks which do not interfere and which are solved independently. At the same time, we may have other cycles that group and un-group dynamically depending on the context.

5 Examples

In order to illustrate our approach, we have implemented a generalized PMP that takes into account a conjunction of goals characterizing a deadlock situation. The application involves a large fleet of autonomous mobile robots [4, 1]. While the overall system allows to operate a large number of robots in a route-network environment, we will limit ourselves here to intricate situations that may happen from time

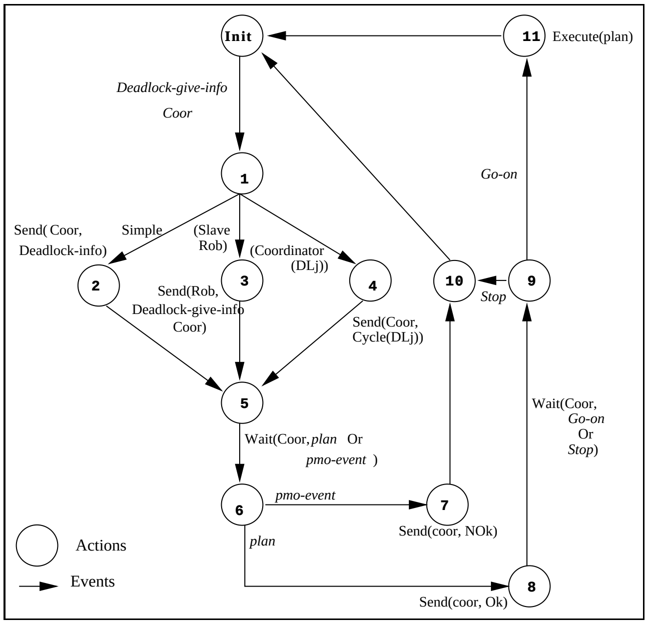

Figure 5: *The other robots finite state automate.*

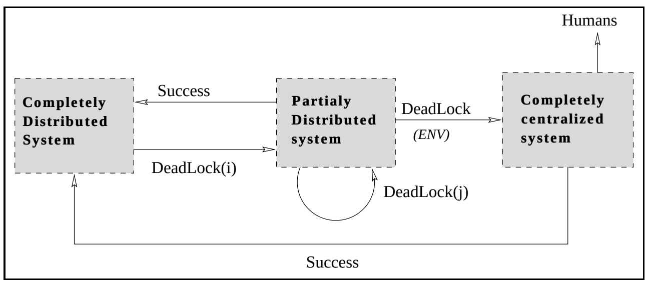

Figure 6: *The evolution of the global system to a more or less distributed system is function of the situations complexity.*

to time.

Each robot R_i is equipped with a multi-robot planner which can be used for an arbitrary number of robots. Such a planner allows to plan and synchronize paths in an environment described as a graph of spatial entities (cells & stations) using an **A*** algorithm.

Note that we could have used also a multi-robot motion planner. However, even though it would allow to solve intricate situations without a pre-structuring of the environment into a discrete set of places, such planners can hardly be used when the number of robots is greater than 3[10].

5.1 A Simple Deadlock Situation

This example treats a simple deadlock DL_3 involving two robots (R_3, R_6) where the goal of each robot is the initial position of the other one (Figure 7).

Robot	Initial-State	Goal-State
R_6	$Station_1$	$Station_0$
R_3	$Station_0$	$Station_1$

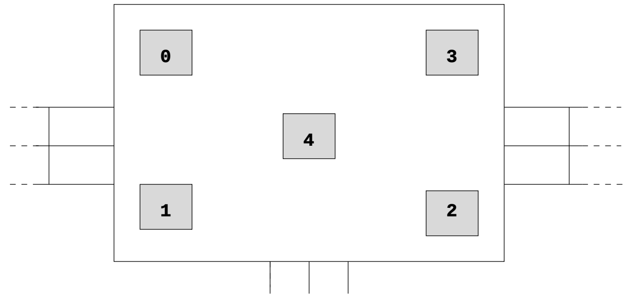

Figure 7: *A part of the testbed environment: an area with 5 stations.*

R_6 fails in its PMO and decides to wait for a *planning event* from $Pred_6 = \{R_3\}$. It propagates this information to R_3 which adds R_6 to PDG_3. R_3 performs a PMO, detects a cycle and becomes the coordinator for $DL_3 = \{R_3, R_6\}$. R_3 invokes its local multi-robot planner and finds a solution to the given conflict. The solution (trivially) uses $station_4$ as a buffer. finally, R_3 performs a PMO in order to insert such a plan. No other robot is present in the area; the PMO succeeds.

5.2 Two Independent Deadlock Cycles

To increase the complexity of the situation, let us create a second deadlock $DL_1 = \{R_1, R_0\}$ and assume that the coordinator R_1 elaborates a plan for R_1 and R_0 which uses the same station ($Station_4$) as a buffer. (Figure 7).

Robot	Initial-State	Goal-State
R_0	$Station_3$	$Station_2$
R_1	$Station_2$	$Station_3$

Two independent cycles $DL_3 = \{R_3, R_6\}$ and $DL_1 = \{R_1, R_0\}$ are created and use the same buffer station ($Station_4$)[4], two coordinators work these two deadlocks and two independent "lattice solutions" are found. The resultant lattices are coordinated with each other and with the other robots' coordinated plans by a PMO.

The validation of these two lattices in the global context imposes a new synchronization event between R_1 and R_6 concerning the occupation of $Station_4$ (Figure 8).

5.3 Two Incompatible Deadlock Cycles

This example treats the case where the system switches to a centralized system when many deadlocks emerge requiring one global centralized planning activity.

The initial and final states are given below (Figure 7) :

[4]This station minimizes the distance criteria given to the planner.

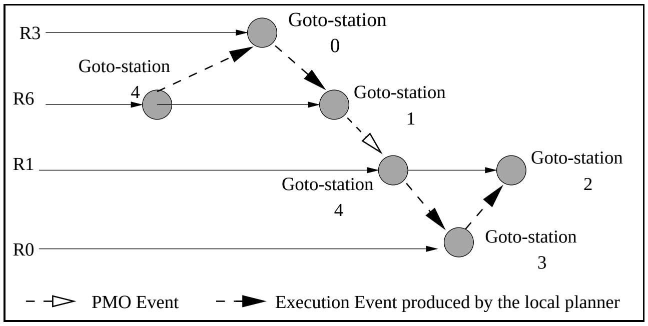

Figure 8: *The set of the totally ordered actions that solves DL_1 and DL_3.*

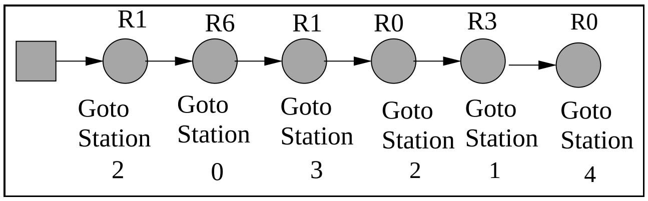

Figure 9: *The set of synchronized actions that solves the conflict after the aggregation of two deadlocks $DL_1 = \{R_1, R_6\}$ and $DL_3 = \{R_3, R_0\}$.*

Robot	Initial-State	Goal-State
R_6	$Station_3$	$Station_0$
R_3	$Station_4$	$Station_1$
R_0	$Station_1$	$Station_4$
R_1	$Station_0$	$Station_3$

Two coordinators R_3 and R_1 try to break independently two detected deadlocks ($DL_1 = \{R_1, R_6\}$ & $DL_3 = \{R_3, R_0\}$). Each coordinator produces a solution Sol_i^{LC} that resolves locally the given deadlock. However, Sol_3^{LC} and Sol_1^{LC} cannot be merged together without introducing a cycle in GP. Therefore, R_3^{LC} takes the "control" of the situation, informs the blocked robots (R_0, R_1, R_6) that it became the new coordinator for this new deadlock ($DL_3 \Leftarrow DL_3 \cup DL_1$). The two *Meta-nodes* are unified to produce one *Meta-node* representing the four conflicting robots. Finally, R_3^{LC} produces a solution plan for the overall task (Figure 9).

Note that, in this example, the whole distributed system switched to a totally centralized one where the generated solution is thus valid in the multi-robot context without a PMO.

6 Conclusion

The effectiveness of the Plan-merging paradigm has already been discussed and illustrated through the implementation of a system involving up to 30 simulated mobile robots. It has also been implemented on a set of 3 real robots in a laboratory environment.

The Plan-merging paradigm is a well suited paradigm to multi-robot applications with loosely-coupled tasks. However, even if an application is designed to ease robots interaction, one cannot guarantee in the general case that tightly-coupled tasks will never happen. For example, the robots may find themselves in intricate situations simply because of an unknown obstacle placed in a critical place. This is why it is important to design a system which is able to efficiently exploit the tasks decoupling, but which is also able to detect and solve transient "puzzle-like" situations.

We have presented here a set of extended operators and associated mechanisms which allow not only to detect but also to solve situations where the robots goals are tightly coupled. This extension is done for the sake of completeness. The operators permit a coherent management of the distributed planning and coordination processes as well as a progressive transition to more global schemes which may even "degrade" to a unique and centralized planning activity.

References

[1] Luis Aguilar, Rachid Alami, Sara Fleury, Matthieu Herrb, Félix Ingrand, and Frédéric Robert. Ten autonomous mobile robots (and even more) in a route network like environment. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '95), Pittsburgh, (Pennsylvania USA)*, 1995.

[2] R. Alami, S. Fleury, M. Herrb, F. Ingrand, and F. Robert. Multi Robot Cooperation in the Martha Project. *IEEE Robotics and Automation Magazine, Robotics and Automation in Europe : Projects funded by the Commission of the European Union*, ?(?), ? 1997. To be published in 1997. Also available as LAAS/CNRS Technical Note 96392.

[3] 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)*, 1994.

[4] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *IEEE International Conference on Robotics and Automation, Nagoaya, (Japan)*, 1995.

[5] K. Azarm and G. Schmidt. A decentralized approach for the conflict-free motion of multiple mobile robots. *iros-96*, 3, November 1996.

[6] Harinaraya and Lumelsky. Sensored-based motion planning for multiple mobile robots in an uncertain environment. In *IEEE International Workshop on Intelligent Robots and Systems*

(IROS '94), Munich, (Germany), pages 1485–1492, 1994.

[7] S. Kato, S. Nishiyama, and J. Takeno. Coordinating mobile robots by applying traffic rules. In *IEEE International Conference on Intelligent Robots and Systems (IROS '92), Raleigh (North Carolina, USA)*, pages 1535–1541, July 1992.

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

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

[10] P. Svestka and M.H. Overmars. Coordinated Motion Planning for Multiple Car-Like Robots using Probabilistic Roadmaps. In *IEEE International Conference on Robotics and Automation, Nagoaya, (Japan)*, 1995.

[11] Warren. Multiple-robots path coordination using artificial potentiel fields. In *IEEE International Conference on Robotics and Automation, Cincinnati, (USA)*, pages 500–505, 1990.

[12] S. Yuta and S.Premvuti. Coordinating autonomous and centralized decision making to achieve cooperative behaviors between multiple mobile robots. In *IEEE International Conference on Intelligent Robots and Systems (IROS '92), Raleigh (North Carolina, USA)*, pages 1566–1574, July 1992.