An extension of the plan-merging paradigm for multi-robot coordination

Fabien Gravot, Rachid Alami

An extension of the Plan-Merging Paradigm for multi-robot coordination

Fabien Gravot and Rachid Alami
LAAS-CNRS, 7, Avenue du Colonel Roche,
31077 Toulouse CEDEX 04, France.
E-mail: {fgravot,rachid}@laas.fr

Abstract

This paper reports on our recent efforts to extend the Plan-Merging Paradigm, in order to improve its ability to adapt to various execution contexts. They concern the algorithmic as well as the architectural issues.

The scheme can now be parameterized, allowing to choose and even to update on-line robot priorities, and to take into account various constraints imposed by the application. This leads to different coordination policies.

The architectural improvements allowed a better distribution of the computation load, leading to more complex applications as, for instance, the coordination of multiple mobile manipulators.

The result is a more flexible scheme that should widen its application fields. We describe the main ingredients of this new scheme and illustrate its implementation through two examples.

1 Introduction

In the multi-agent cooperation field, we claim that agents must be able to plan/refine their respective missions, taking into account other agents plans as constraints, and thus to produce plans with coordinated action to ensure their proper execution.

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

While several generic approaches have been proposed in the literature concerning goal decomposition and allocation (Contract Nets [16], Partial Global Planning [8], distributed search [9], negotiation [4, 11, 15], motivational behaviors [13, 10]), cooperation for achieving independent goals have been mostly treated using task-specific or application-specific techniques [5, 7, 6].

We argue that there is also a need for generic approaches to perform plan coordination. We have proposed, several years ago, a coordination scheme called "Plan-Merging Paradigm" (PMP). PMP proved to be quite efficient and allowed us to exhibit several examples where robots were operated with very limited central activity, while preserving essential properties like global coherence, detection of dead-lock situations,...

We present, in the following, a number of important extensions to the "basic PMP". They have been developed in order to improve its ability to adapt to various execution contexts, and concern the algorithmic as well as the architectural issues.

The scheme can now be parameterized, allowing to choose and even to update on-line robot priorities, and to take into account various constraints imposed by the application.

The new architecture allows the use of heterogeneous robots as well as a better distribution of the computation load. The result is a more flexible scheme that should widen its application fields.

In the next section, we briefly remind the main features of the *basic PMP*. Then, we present a new plan-merging algorithm which integrates priorities (section 3), and progressive plan-merging through a hierarchical resource description and the use of colored tokens (section 4). Section 5 presents the new architecture. Finally, these new features are illustrated through implemented examples involving intricate plan coordinations for heterogeneous robots (section 6).

2 The Plan Merging Protocol

Let us assume that we have a set of autonomous robots and a central station which sends goals to robots. Whenever a robot receives a new goal, it elaborates an *Individual Plan* which takes as initial state the final state of the current plan.

However, before executing this plan, a robot must ensure that it is valid in the multi-robot context, i.e. all potential conflicts with the other robots plans are considered. We call this operation *Plan Merging Operation* (PMO) and the resulting plan a *Coordinated plan*. 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 the different coordinated plans.

At any moment, the temporal constraints between all the actions included in the union of all the coordinated plans constitute a *directed acyclic graph* [3] which is a snapshot of the current situation and its already planned evolution (Fig. 1).

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 (fig. 2 state 2) under mutual exclusion (fig. 2 state 1): it collects the coordinated plans CP_k of the robots which may interfere with IP_i^j, and builds their union $GP = \bigcup_k CP_k$. Then, it tries to insert IP_i^j in GP. If it succeeds, it adds temporal order constraints to actions in IP_i^j and transforms it into a coordinated plan CP_i. The resulting CP_i is feasible in the current context, and does not introduce any cycle in the resulting GP.

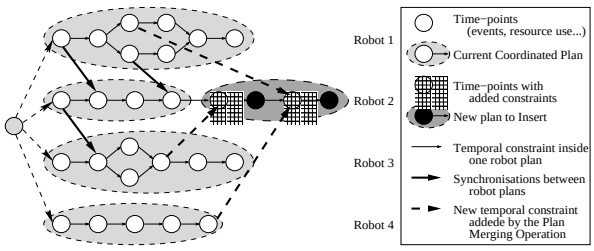

Figure 1: Robot 2 performs a PMO.

However a PMO performed by R_i may fail because the final state of at least another robot R_k (as specified in GP) forbids R_i to insert its own plan IP_i^j in GP. R_i defers its PMO and waits (fig. 2 state 3) until at least one of the "blocking" robots has performed a new successful PMO. Hence, we introduce, when necessary, temporal order relations between the different plan-merging activities. This data structure allows to detect dead-lock situations. Depending on the context, the coordination processes allow a progressive transition to more global schemes which may even "degrade" to a unique and centralized planning activity[14].

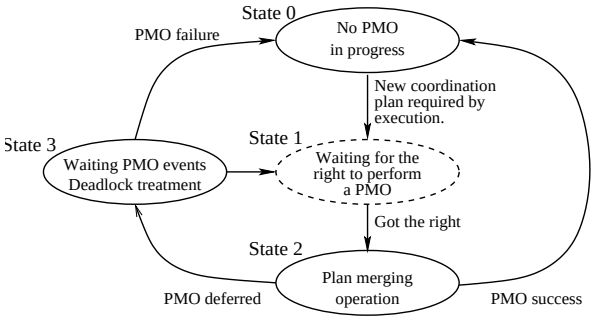

Figure 2: The Plan Merging Protocol.

PMP proved to be quite efficient and allowed us to exhibit several examples where a number of robots (up to 30, in simulation) were operated with very limited central activity, while preserving essential properties like global coherence, detection of dead-lock situations,...

However, PMP suffered from several limitations. For example, all robots were assumed to be identical and were considered at the same level of priority. This somehow limits the coordination to an insertion of new actions after the existing plans which are considered as contingent events. As a consequence, the robots were limited to short terms plan insertion. The next sections describe and illustrate the improvements that solve these limitations.

3 Dealing with Priorities

The main improvement over the "basic PMP" is the introduction of priorities. They are given to each resource needed by the robot. They can change dynamically during the merging operation or during the plan execution. To ensure coherence along the plans we use three priority evolution rules.

3.1 Priority evolution rules

The first one (**r 1**) is the rule of priority decrease along a plan:

$$\forall \mathcal{U}_1, \mathcal{U}_2, \mathcal{A}(\mathcal{U}_1) \prec \mathcal{A}(\mathcal{U}_2) \Rightarrow \mathcal{U}_1.p \geq \mathcal{U}_2.p$$

Where $\mathcal{U}$ represents the use of a resource, $\mathcal{U}.p$ its associated priority, and $\mathcal{A}(\mathcal{U})$ its allocation event.

The earlier resources have higher priority.

The two other rules are applied during the merging operation. When two robots have the same priority, we decide to give the advantage to the robot which is subjected to the merging operation.

The second rule is the priority inheritance:
We assume to have two robots R_1 and R_2 in conflict for the use of resources $\mathcal{U}_{R_1}(i)$ and $\mathcal{U}_{R_2}(j)$.

- **r 2.1:** If $\mathcal{U}_{R_1}(i).p > \mathcal{U}_{R_2}(j).p$ and R_1 is the robot doing the merging operation then
$$\forall \mathcal{U}_{R_1}(k) : \begin{array}{l} \mathcal{A}(\mathcal{U}_{R_1}(k)) \prec \mathcal{F}(\mathcal{U}_{R_1}(i)) \\ \Rightarrow \mathcal{U}_{R_1}(k).p \geq \mathcal{U}_{R_2}(j).p + p_{inc} \end{array}$$
- **r 2.2:** If $\mathcal{U}_{R_1}(i).p \geq \mathcal{U}_{R_2}(j).p$ and R_1 is not the robot doing the merging operation then
$$\forall \mathcal{U}_{R_1}(k) : \begin{array}{l} \mathcal{A}(\mathcal{U}_{R_1}(k)) \prec \mathcal{F}(\mathcal{U}_{R_1}(i)) \\ \Rightarrow \mathcal{U}_{R_1}(k).p \geq \mathcal{U}_{R_2}(j).p \end{array}$$

Where $\mathcal{F}(\mathcal{U})$ represents the release event of $\mathcal{U}$, and p_{inc} the minimal increment. This rule allows the robot which has the priority on a resource $\mathcal{U}$ to have a higher priority on resources needed to be able to free $\mathcal{U}$ than other robots which want to take $\mathcal{U}$.

The third rule is for "strong links" use. Such links correspond to external constraints on the merging operation: conservative insertion, incremental constraints, hierarchical constraints, other planner constraints... However, these links should not create a cycle in the graph of events.
If we have a strong link from $\mathcal{U}_{R_2}(j)$ to $\mathcal{U}_{R_1}(i)$:

- **r 3.1:** If R_2 is not the robot doing the merging operation then $\mathcal{U}_{R_1}(i).p \geq \mathcal{U}_{R_2}(j).p + p_{inc}$
- **r 3.2:** If R_2 is the robot doing the merging operation then $\mathcal{U}_{R_1}(i).p \geq \mathcal{U}_{R_2}(j).p$

Those constraints can be dynamically added or removed to modify the robots behavior. For example, the conservative insertion allows to forbid a robot R_2 with a higher priority to break the plan of R_1, when it performs a PMO. We create links to not freed resources of R_1 from resources of R_2 which are in conflict. If we don't use this conservative insertion, R_1 would be stopped to let R_2 go, and need another PMO to go on.

The incremental constraints are links created during the last merging operation. To avoid any change in a previously negociated synchronization, we represent them by strong links.

3.2 Priority evolution procedure

To comply with the previous rules, the algorithm increases resources priorities which don't satisfy these rules. We also choose to do minimal increment. So, the priorities are not going to dramatically change unless it is necessary.

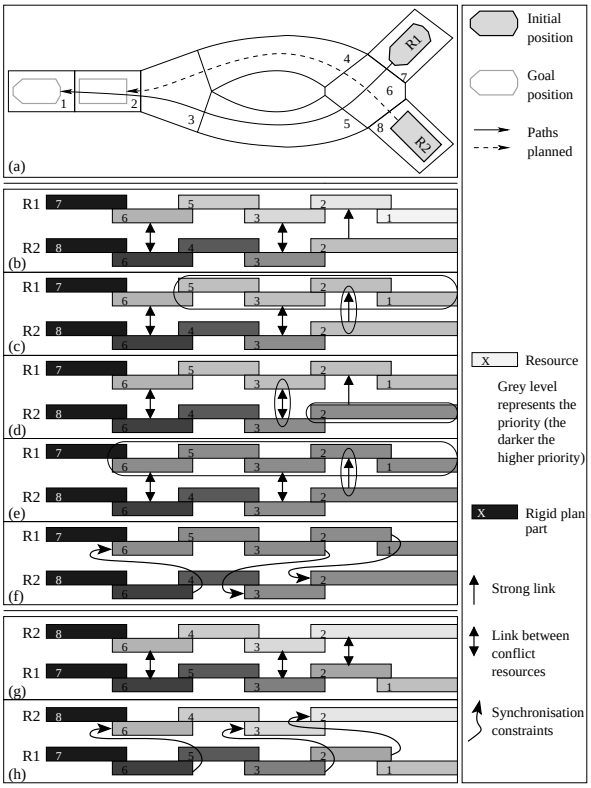

Figure 3: Examples of evolution of priority rules. (a) Coordination of R_1 and R_2. (b), (c), (d), (e), (f) show different steps in the merging of R_2 plan when it has the highest priority. (g), (h) show PMO steps of R_1 plan when it has the highest priority.

We introduce a special (very high) priority to protect a part of plan that we do not want to be temporally constrained and that we call the "rigid part" of the plan. The robots are allowed to execute the rigid parts of their plans during the insertion protocol.

We have two possible failures for this merging operation: creation of a wait event from the rigid part of a plan, or creation of a wait event associated to a not freed resource. In this case we create a dependency which is solved by the third state of the "Plan Merging Protocol" (fig. 2 state 3).

Figure 3) shows an example where the three rules are used. Note that resource uses are partially ordered: a robot allocates the next resources before it

leaves the ones it occupies. Hence, boxes partially overlap.

First (fig. 3 (b),(c),(d),(e),(f)), we assume that R_1 has already merged its plan, and R_2 wants to do a PMO. R_2 has priority on R_1 (its resources have higher priority). Figure 3(b) shows the effect of the first rule with the decrease of priority along plans. It also indicates conflicts between resources ($\mathcal{U}(2)$, $\mathcal{U}(3)$ and $\mathcal{U}(6)$), and a strong link, due to the conservative insertion. If R_2 allocates $\mathcal{U}(2)$ before R_1, R_1 will not be able to achieve its plan. So, the conservative insertion creates this strong link.

Figure 3(c) shows the evolution of priorities in R_1's plan ($\mathcal{U}_{R_1}(1)$, $\mathcal{U}_{R_1}(2)$, $\mathcal{U}_{R_1}(3)$ and $\mathcal{U}_{R_1}(5)$):

Rule **r 3.2**: R_2 is performing the PMO and there is a strong link from $\mathcal{U}_{R_2}(2)$ to $\mathcal{U}_{R_1}(2)$, so the priority of $\mathcal{U}_{R_1}(2)$ changes to $\mathcal{U}_{R_2}(2).p$.

Then rule **r 1** is applied to $\mathcal{U}_{R_1}(3)$ and $\mathcal{U}_{R_1}(5)$: we have $\mathcal{U}_{R_1}(5).p = \mathcal{U}_{R_1}(3).p = \mathcal{U}_{R_1}(2).p = \mathcal{U}_{R_2}(2).p$.

And finally rule **r 2.2**: $\mathcal{U}_{R_1}(2).p \geq \mathcal{U}_{R_2}(2).p$, and R_2 is doing the PMO, so $\mathcal{U}_{R_1}(1).p = \mathcal{U}_{R_2}(2).p$.

Figure 3(d) shows the evolution of priority in R_2 plan ($\mathcal{U}_{R_2}(2)$) due to the rules **r 2.1**. Then with figure 3 (e), we have a similar evolution in R_1 plan to fig.3(c). After that, all the rules are satisfied, so we can establish a synchronization between the robots (fig. 3(f)). R_2 goes before R_1 for $\mathcal{U}(6)$, but waits for R_1 releasing resources $\mathcal{U}(2)$ and $\mathcal{U}(3)$, allowing R_1 to achieve its own plan.

In figure 3(g),(h), we assume that R_2 has already merged its plan, and that R_1 wants to do a PMO. Figure 3(g) shows the initial state, which satisfies the three rules, hence, the synchronization (fig. 3(h)) where R_1 goes always before R_2.

Thanks to these rules, one can decide to set priorities which decrease along the plan proportionally to a forecast time of resource allocation. This is a way to introduce a treatment of time in plans that could decrease waiting delays. One can also introduce robots with greater priority on all their resources to have ambulance robots.

This capacity of robots to pass before or after other robots during the PMO allows not only to have robots for emergency tasks, but also to merge plans much longer than before.

4 Hierarchical plan merging

We introduce the possibility to add information to the environment description such as a hierarchical description, or a limited access to resources through colored tokens. With these two notions we have added more flexibility to plan refinement. Besides, this hierarchical description allows to decrease the resource intersection computation time.

4.1 Hierarchical resources

Hierarchical plan description allows to deal with partially structured environments and to reduce computation time. For each resource used, we can associate a sub-plan. For example, we are able to describe a navigation plan in terms of corridors and rooms, then give a geometrical description of robot movements inside each room. This allows to limit the computation burden; when there is no conflict between resources at a given level, there is no need to compute possible intersection of their sub-resources.

4.2 Colored tokens

We have also added colored tokens. A set of tokens can be associated to each resource. The resource usage is then defined as the allocation by a robot of a given number of tokens of a specific color. With this mechanism we can limit the number of robots which could access to a resource. The color token is used to add information on sub-resources. For example, in an area with several parking places, we can associate to each one a color. When the robot wants to enter the area, it can know what are the free parking places and then create a sub-plan with less conflicts.

5 The new architecture

While the previous version only used a two-dimensional planner and one fixed robot type (rectangular), we have increased the capacity of the new implementation by allowing a 3D geometrical description for environment or robots with any robots type (holonomic mobile robot, nonholonomic mobile manipulator...). The underlying planner is Move3D [12]. One consequence, is the increase of time required to compute plan intersections. To reduce this problem, we have chosen to distribute this calculation.

Figure 4 shows the changes in the global architecture. The merging procedure and the 3D plan intersection computation are now distributed. However, we still need a "central" stage to deal with problems that occur with more than two robots. To do this we collect all synchronizations, and inverse

links which cause deadlock. Note also that there is only one broadcast: the plan to merge. Hence, we also decrease substantially the amount of transmitted data.

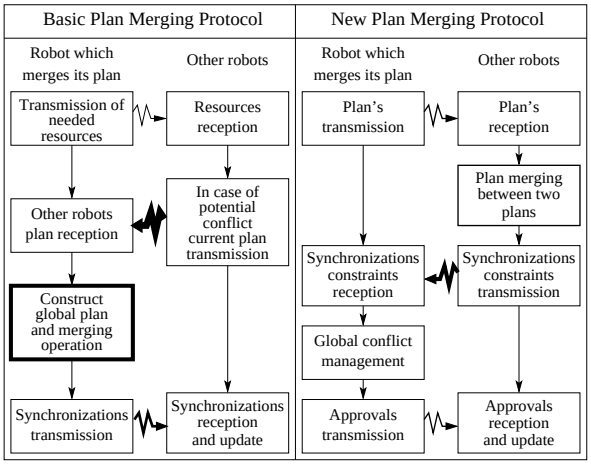

Figure 4: Plan-Merging protocol architecture. The new implementation allows a distributed computation of resources intersection. The amount of transmitted data is represented by the width of arrows.

With this implementation we drastically reduce the merging computing latency time. In fact, this latency time tends to be independent of the number of robots (the longest stage is the distributed one). But, even if we reduce the communication time, it is still linear in the number of robots. For a great number of robots, this will be the limitation.

6 Implementation

We describe two illustrative examples: the first one (fig. 5) shows the ability to deal with a 3D environment with several kinds of robots. The second one (fig. 6) is an application of long plans insertion.

The first example involves several robots of various types in an indoor 3D environment. There are three types of mobile robots as used in our laboratory: Hilare, a nonholonomic robot with a a 6 degrees of freedom manipulator, Diligent (XR400 Nomadic), a holonomic robot, Scout (Nomadic), a nonhomolomic robot. One can observe how the robots can insert their plans before or after other robots. For instance Diligent has to wait for the Scout (fig 5(c)) although it has inserted its plan first. The Scout has the highest priority and goes before Diligent, but after both Hilare robots because their initial positions are in the rigid part of plan. We also, observe the great number of synchronizations between the two Hilare because they follow each other. Scout, which is smaller than Diligent, is able to go under Hilare arms (fig 5(c)) before Diligent which has to wait (fig 5(d)) until the two Hilare robots have almost completely raised their arms...

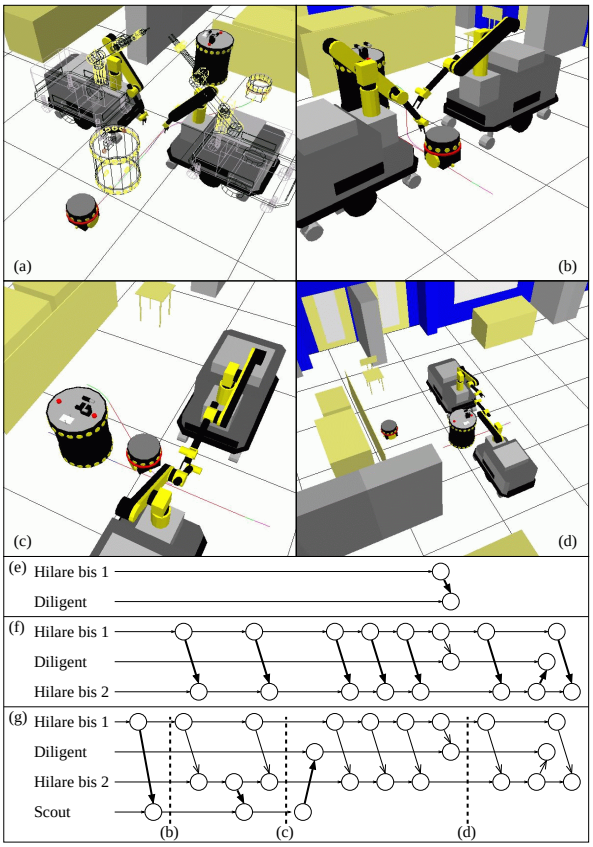

Figure 5: 3D Plan-Merging. (a) Initial and goal positions. There are four robots: from left to right, Hilare 2, Scout, Diligent, Hilare 1. Scout has the highest priority. (b), (c), (d) represent synchronized positions. (e), (f), (g) show a simplified global plan evolution. The dashed lines in the global plan correspond to the situations (b), (c), (d).

The second example in an airport environment with 7 "robots": 2 trucks, 2 buses, 2 planes and 1 car. It shows the benefit of decreasing priority along plans, when a bus goes before other vehicles in the beginning of its plan, and after at the end (fig. 6(a)). It also shows the use of robots with high priority. For instance, we have given the highest

priority to planes: consequently, a plane is able to forbid other vehicles to go through its way, even if it does a PMO after the others (fig. 6 (b),(c)).

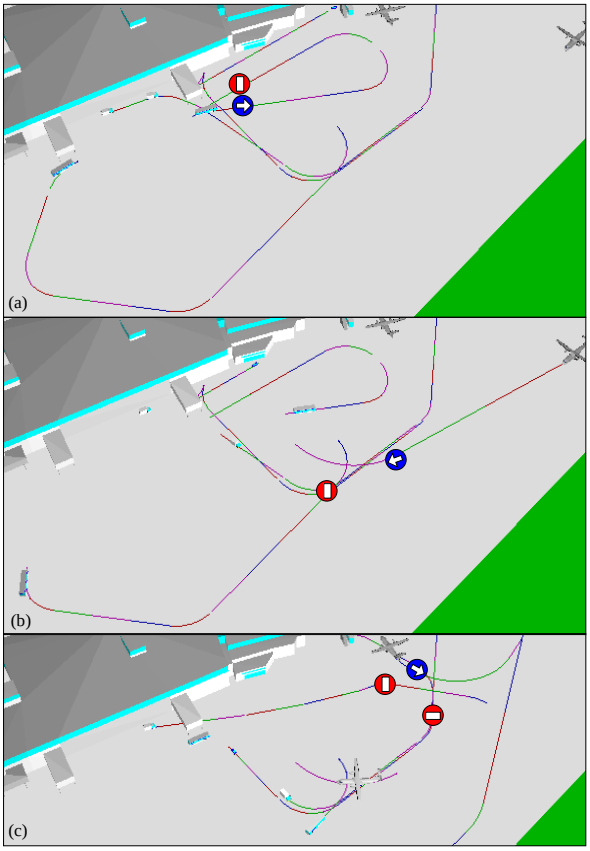

Figure 6: Coordination of two planes, two buses, two trucks and one car in an airport environment. Both planes have the highest priority. Curves on the ground represent the merged plans.

7 Conclusion

We have presented and illustrated some key extensions to the Plan-Merging Paradigm. They concern algorithmic as well as architectural issues.

The new version is more flexible and more general. It allows to implement various coordination schemes. For instance, with a constant priority for all resources, the system behaves like the "basic PMP". With a decreasing priority along plans we have a merging policy which favours short terms actions. With different priority levels by robots we have the notion of emergency robots and tasks. Besides, the protocol deals with priorities that change during plan execution, allowing robots to merge again their plans when they need or want to change their coordinations. Besides, the new implementation accepts heterogeneous mobile manipulators in a 3D environment.

References

[1] R. Alami, F. Ingrand, and S. Qutub, 'A Scheme for Coordinating Multi-robot Planning Activities and Plans Execution', in *ECAI*, (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*, **5**(1), (1998).

[3] R. Alami, F. Robert, F. F. Ingrand, and S. Suzuki, 'Multi-robot cooperation through incremental plan-merging', in *IEEE ICRA*, (1995).

[4] Asama, H. and Ozaki, K. (1991). Negotiation between multiple mobile robots and an environment manager. In *International Conference on Advanced Robotics*, pages 533 – 538.

[5] Azarm, K. and Schmidt, G. (1997). A decentralized approach for the conflict-free motion of multiple mobile robots. *Advanced Robotics*, 11(4):323 – 340.

[6] Cao, Y., Fukunaga, A., and Kahng, A. (1997). Cooperative mobile robotics: antecedents and directions. *Autonomous Robots*, (4):1 – 23.

[7] Dudek, G., Jenkin, M., Milios, E., and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots*, (3):375 – 397.

[8] 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).

[9] E.H. Durfee and T. A. Montgomery, 'Coordination as distributed search in a hierarchical behavior space', *IEEE Transactions on Systems, Man and Cybernetics*, **21**(6), (1991).

[10] E. Ephrati, M. Perry, and J.S. Rosenschein, 'Plan execution motivation in multi-agent systems', in *AIPS*, (1994).

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

[12] C. Nissoux , T. Simeon , J. P. Laumond , 'Visibility Based Probabilistic Roadmaps', in *IEEE IROS*, (1999)

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

[14] S. Qutub, R. Alami, and F. Ingrand, 'How to Solve Deadlock Situations within the Plan-Merging Paradigm for Multi-robot Cooperation', in *IEEE IROS*, (1997).

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

[16] 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).