A paradigm for plan-merging and its use for Multi-robot Cooperation

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

Abstract: This paper presents a new approach 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.

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 context where multi-robot cooperation takes place. In section 3 we present the paradigm at an abstract level, and show that it is safe and robust. In section 4 we discuss some key properties of the paradigm. Section 5 reviews related work and section 6 concludes the paper.

2 A context for multi-robot cooperation

2.1 Interleaving planning and execution

As a general assumption, we assume the existence of a central system which sends, from time to time, goals to robots.

Whenever they receive a goal, the robots are assumed to produce 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 action, 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 range of the plan to validate by the PMO, execution may run without ever waiting for a new coordination plan.

In the following, we will first see what are the properties of the plans which should be produced, and then how we can guarantee that the produced plans satisfy such properties.

2.2 Basic Assumptions

In order to explain the general paradigm and how it can be used, we introduce a description of a simplified planning system.

We assume that all robots are equipped with identical planners.

For each robot, the state is described by a set of facts F_i $(i = 1, n)$ which are supposed to be TRUE.

Let us assume that we have p robots: r_j $(j = 1, p)$.

$F_i(r_j)$ denotes that the fact F_i is TRUE for robot r_j.

At any moment a complete state is then described using $F_i(r_j)$.

Let $\mathcal{F}_a$ denote the set of all allowed states. This set takes into account the incompatibility between facts concerning different robots.

Let us assume that planning is based on a set of operators A_k $(k = 1, m)$ which are the same for all robots.

These operators are classically defined using a list a pre-conditions (the facts that must be TRUE or FALSE), an add-list (the facts that will be added by the operator) and a delete-list (the facts that will be negated by the operator)[1].

We assume that the actions executed by one robot modify only its state (i.e. all the facts in the add and delete lists concern only this robot). However, their preconditions may contain facts concerning other robots.

[1] We make use of the closed world assumption

	$F_1(r_j)$	$F_2(r_j)$	$F_3(r_j)$	$F_4(r_j)$	$F_5(r_j)$
$F_1(r_i)$	x				
$F_2(r_i)$					
$F_3(r_i)$			x		x
$F_4(r_i)$				x	
$F_5(r_i)$			x		x

Figure 1: Facts incompatibility table: $F_i(r_j)$ denotes that the fact F_i is TRUE for robot r_j ("x" means "forbidden")

Let us also assume that the corresponding actions have a duration which may vary depending on the execution context and which cannot be computed beforehand. They will be represented in the plans by time intervals denoted $I(A_k)$.

For each fact in the add and delete list of an action A_k there is a time instant included in $I(A_k)$ where the fact is set to TRUE or FALSE. In a general scheme (see for example [5]), these instants are partially ordered (the total order is known only at execution time as a result of a sensing operation for instance). However, in order to simplify the presentation, we will limit ourselves to two identified time points corresponding to the begin and the end of an action.

Consequently, we will assume that all the facts in the add and delete list of action A_k may be TRUE during all the action extent. It is only after the end of the action that we are assured that all the facts of the delete list are set to FALSE and all the facts in the add-list are set to TRUE.

Example: Let us give a simple example.

The state is described using the facts F_1, .. F_5. We have 3 robots.

The set of allowed states $\mathcal{F}_a$ is given implicitly by a table of facts incompatibility (see Figure 1).

There are 9 operators A_1, .. A_9. Figure 2 illustrates the transitions corresponding to these operators for a given robot. From this and from the incompatibility table, we derive the complete sets of pre-conditions, add and delete lists (see Figure 3).

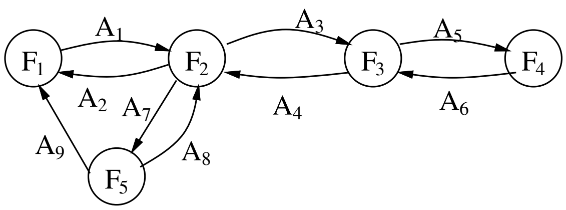

Figure 2: The set of actions

2.3 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 a *global plan*[2].

At any moment, a robot has its own *coordination plan* (i.e. a plan valid in the current multi-robot context and currently executed). This *coordination plan* contains:

- an ordered sequence of time points: the first point corresponds to the current time and the other points corresponds to the beginning and the end of the different (future actions). A list of facts (concerning the robot) is associated to the first point (facts which are true at that moment). A list of transitions (facts being set to TRUE or FALSE) is associated to each of the other time points.
- a set of *temporal constraints* (precedence) between time points corresponding to the beginning of actions in the plan and time points corresponding to transitions in other robots coordination plans. These temporal constraints are introduced in order to synchronize different robot execution plans.

Example: An example of *coordination plan* for robot r_2 can be (see Figure 4):

initial time point: $t_{l,2}$ with initial state = $(F_5(r_2))$

time points	add-list	delete-list	comment
$t_{l+1,2}$	$(F_2(r_2))$	()	begin A_8
$t_{l+2,2}$	()	$(F_5(r_2))$	end A_8
$t_{l+3,2}$	$(F_1(r_2))$	()	begin A_2
$t_{l+4,2}$	()	$(F_2(r_2))$	end A_2

Figure 4: An example of coordination plan

such that[3]: $(t_{l,2} < t_{l+1,2} < t_{l+2,2} < t_{l+3,2} < t_{l+4,2})$ and: $(t_{k,1} < t_{l+3,2})$, meaning that robot r_2 has to wait until robot r_1 has passed $t_{k,1}$.

When a robot is staying still and has completely executed its planned actions, its *coordination plan* is represented by only one time-point.

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 temporal constraints impossible to satisfy) if it can be represented by a *directed acyclic graph* (*dag*).

[2] 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 synchronized.

[3] "<" is a temporal relation between two time points.

	delete-list	pre-conditions list	add-list
$A_1(r_i)$	$F_1(r_i)$		$F_2(r_i)$
$A_2(r_i)$	$F_2(r_i)$	$\neg F_1(r_j)$ $(j \neq i)$	$F_1(r_i)$
$A_3(r_i)$	$F_2(r_i)$	$\neg F_3(r_j)$, $\neg F_5(r_j)$ $(j \neq i)$	$F_3(r_i)$
$A_4(r_i)$	$F_3(r_i)$		$F_2(r_i)$
$A_5(r_i)$	$F_3(r_i)$	$\neg F_4(r_j)$ $(j \neq i)$	$F_4(r_i)$
$A_6(r_i)$	$F_4(r_i)$	$\neg F_3(r_j)$, $\neg F_5(r_j)$ $(j \neq i)$	$F_3(r_i)$
$A_7(r_i)$	$F_2(r_i)$	$\neg F_3(r_j)$, $\neg F_5(r_j)$ $(j \neq i)$	$F_5(r_i)$
$A_8(r_i)$	$F_5(r_i)$		$F_2(r_i)$
$A_9(r_i)$	$F_5(r_i)$	$\neg F_1(r_j)$ $(j \neq i)$	$F_1(r_i)$

Figure 3: Pre-conditions, add and delete lists

Example: Figure 5 illustrates a global plan for three robots. Robot r_3 is staying still.

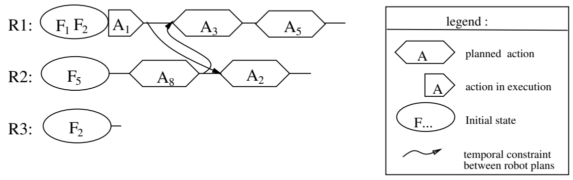

Figure 5: A global plan for three robots

The key point here is how to implement a system where robots are planning individually while maintaining such a property of the global plan.

3 The Plan-Merging Paradigm

Let us assume here that:

- 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 thought of as a resource allocation[4].
- there is a mean allowing a robot (which has got the right to perform a PMO) to ask for and obtain all the other robots coordination plans.
- there exists a mean allowing robots to ask or inform one another about the occurrence of an event (represented by a time point in an individual robot plan).

3.1 Performing a *Plan Merging Operation*

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

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

When a robot has to plan:

[4] A simple way to do it is to maintain a "token" through communication.

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 gives up the right to perform a PMO.

When a robot executes its coordination plan, if it reaches a time point 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.

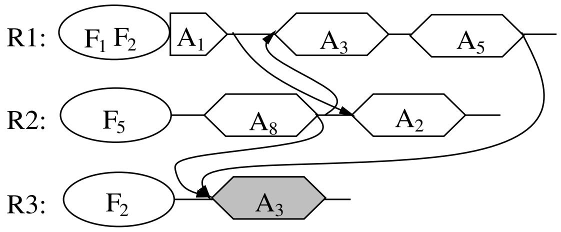

Figure 6: The result of a plan merging operation

Example: Figure 6 illustrates a Plan-Merging Operation. Robot r_3 is staying still, its current state is: $(F_2(r_3))$. It wants to plan in order to reach the goal $(F_3(r_3))$.

It collects the coordination plans of robots r_1 and r_2. The global plan is represented in figure 5. It succeeds in producing a plan consisting of only one action A_3 with two temporal constraints: the action can be executed only when r_1 will finish its action A_5 and r_2 will finish action A_8.

Note that the second temporal relation can be removed from the dag as the first one entails a stronger constraint on r_3.

3.2 Detecting deadlock or situation where a PMO is deferred

When a robot tries to perform a PMO, it may fail to produce a plan which verifies 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 is 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 current or the future 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 forbidding it to insert its plan.

This is why we require to add in robots *coordination plan* the list of the robots which should perform a PMO before they can proceed.

Doing this, we have now two types of events which may occur:

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 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).

By adding in the *coordination plans*, the list of awaited *planning events* (represented by $wait\_pmo(r_i)$), it becomes possible to detect *deadlocks*.

A *deadlock* is detected by a robot r_j when it fails in a PMO and finds itself in a situation where it has to wait for a robot r_k which is already waiting (directly or by transitivity) for r_j to perform a new PMO.

A deadlock will occur whenever it is necessary to take explicitly into account, in a unique planning operation, a conjunction of goals (which have been given separately to two independent robots).

This simply means that the global mission was too constrained to be solved using the Plan-Merging operation. 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. It has been specially devised for multi-robot cooperation and it can be used in a large class of applications particularly in traffic for a great number of robots, freeing the central station from the burden of solving all local conflicts.

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 of a great number of robots, one can decide when one robot detects a deadlock, to allow it to plan for all the concerned robots (if their number is small, which is generally the case). The Plan-Merging paradigm remains then applicable: the inserted plan will then concern several robots at a time.

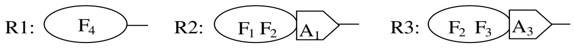

Figure 7: A situation where a deadlock may occur

Example: Figure 7 illustrates a Plan-Merging Operation which leads a robot to decide to wait for a PMO event.

Robot r_1 is staying still, its current state is: $(F_4(r_1))$. It wants to plan in order to reach the goal $(F_3(r_1))$. Action A_6 can be used to reach the desired state.

However, an action A_6 for r_1 cannot be inserted because the last planned state of r_3 is $F_3(r_3)$. r_1 abandons its PMO and decides to wait until r_3 performs a PMO.

If, after a while, r_3 performs a PMO in order to insert a plan which will lead it to state $F_2(r_3)$, this will allow a new and successful PMO for r_1.

But if, r_3 performs a PMO in order to insert a plan which will lead it to state $F_4(r_3)$, it will finds itself in a situation where it should wait for r_1 and a deadlock will be detected.

4 Discussion

4.1 Incremental construction and correctness of the "global plan"

Note that no system (neither the central station, nor the robots) maintain permanently the global state or the global plan.

It is only when a robot performs a PMO that it builds momentarily an image of the current global plan. However, the global plan is guaranteed to be permanently correct because:

- there is only one robot at a time which have the right to perform a PMO i.e. to add new plan steps and new temporal constraints.
- when a robot obtains the other robots coordination plans, it is guaranteed that they contain the last modifications introduced by all previous PMOs.

Thus, the whole process is incremental. A robot may "enter" into coordination process concerning several robots, and "leave" it after a while, without the need to maintain a unique representation of the global plan. Its construction as well as its execution are performed in a distributed and synchronized manner.

4.2 Planning before or during a PMO

A PMO may embed a planning operation (i.e. the elaboration of a plan taking into account the current global plan), or simply a merging operation of an already elaborated robot plan into the global plan. In the second case, the PMO is limited to a synchronization with the other robot plans.

The choice between this two possibilities depends mainly on the application and on the extent of plans which have to be merged.

Note also that merging plans consisting in long sequences of actions may induce a great number of constraints for the future PMOs. This is again application dependent. For example, in traffic applications, it is certainly better to limit the range of the inserted plan in order to allow a smooth traffic.

4.3 Satisfying real-time constraints

Note that the paradigm we propose do not impose any constraints on the time necessary for planning, performing a PMO or executing an action.

Indeed, in the general case, planning time cannot be bounded. Besides, the execution continues while planning or PMO is performed.

This is why robot synchronization is based on events as perceived and produced by robots along their execution and not on a numerical estimation of the duration of actions of other operations performed by robots.

4.4 Accounting for execution failures

The Plan-Merging paradigm is also robust to plan execution failures. Indeed, as execution is synchronized through event produced by the robots, when a robot fails in the execution of one of its actions, it is able to inform robots which ask for the occurrence of events it is supposed to produce, that such events will never occur.

This information may cause other robot plans to fail. All robots which have a "broken" coordination plan will rebuilt their state and try a PMO again.

Depending on the constraints imposed by an event which will not occur, a cascade of plan failures may occur. This may cause a brutal increase of PMO activities with several robots trying to perform a PMO at almost the same time, but the system will be maintained safe thanks to the properties discussed earlier (guarantee of always having a valid global plan and of detecting deadlocks or situations where a PMO should be deferred).

4.5 Asking for events or signaling them

When a robot inserts its plan in a set of already coordinated plans, it determines the set of events which will occur as a result of other robot actions and which constrain its own plan.

In order to be informed on the events, it may ask the concerned robots to signal to it asynchronously that they have occurred or have been cancelled because of plan failure. This will authorize, in numerous cases, to have robots being informed that a temporal constraint has been satisfied before they reach the concerned time-point, allowing them to proceed without suspending their execution.

5 Related work

The term "multi-robot cooperation" is used in several contexts with different meanings. Our interest is to implement an effective cooperation at planning or programming level, not at servo level (e.g. [9]) nor at a level of an "intelligent group" of simple robots (e.g. [10]).

Several approaches have been proposed, such as the generation of trajectories without collision (e.g.[12, 4]), traffic rules[7], negotiation for dynamic task allocation[8, 2], and synchronization by programming[11] or communication[13].

The methods listed above essentially deal with collision avoidance or motion coordination. Each method depends on the analysis of a set of pre-defined situations, on task dependent properties, or on domain-dependent heuristics. They do not seem to be directly applicable to other contexts or tasks.

We claim that our Plan-Merging paradigm is a generic framework which can be applied to 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 be encountered.

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. Real-time constraints are satisfied. The plan is executed while planning or a PMO is performed.
4. The PMO is safe, because it is robust to plan execution failures and allows to detect deadlocks.

The Plan-Merging paradigm and the PMO can be applied to a large class of problems.

It appears to be particularly well suited to the control of a large number of robots in a route network.

We are currently implementing this paradigm on a set of mobile robots (Hilare family). The on-board Control System will involve a layered plan-based architecture [3, 1] for mission refinement and execution. The Plan-Merging paradigm is encoded as set of situation-driven procedures within a Robot Supervisor written in C-PRS[6].

Furthermore, we make use of this paradigm in an application involving a large number (dozens) of highly autonomous robots for container transport in harbours, airports and railway stations (MARTHA ESPRIT Project).

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] 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)*, pages 533–538, June 1991.

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

[4] 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)*, pages 475–480, May 1992.

[5] M. Ghallab, R. Alami, and R. Chatila. Dealing with Time in Planning and Execution Monitoring. In R. Bolles, editor, *Robotics Research: The Fourth International Symposium*. MIT Press, Mass., 1988.

[6] Ingrand, F. F., Georgeff, M. P., and Rao, A. S. An Architecture for Real-Time Reasoning and System Control. *IEEE Expert, Knowledge-Based Diagnosis in Process Engineering*, 7(6):34–44, December 1992.

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

[8] 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)*, pages 488–493, May 1990.

[9] Z.W. Luo, K. Ito, and M Ito. Multiple robot manipulators cooperative compliant manipulation on dynamical environments. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '93), Yokohama, (Japan)*, pages 1927–1934, July 1993.

[10] M. Mataric. Minimizing complexity in controlling a mobile robot population. In *IEEE International Conference on Robotics and Automation, Nice, (France)*, pages 830–835, May 1992.

[11] F.R. Noreils. Integrating multi-robot coodination in a mobile-robot control system. In *IEEE International Workshop on Intelligent Robots and Systems (IROS '90), Tsuchiura (Japan)*, pages 43–49, July 1990.

[12] C.W. Warren. Multiple robot path coordination using artificial potential fields. In *IEEE International Conference on Robotics and Automation, Cincinnati, (USA)*, pages 500–505, 1990.

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