Local Plan Execution and Repair in a Hierarchical Structure of Sub-Teams of Heterogeneous Autonomous Vehicles

Gateau T., Barbier M., and Lesire C.
{firstname.lastname}@onera.fr
ONERA Centre de Toulouse – DCSD/CD
2 avenue Édouard Belin, B.P. 4025
F-31055 TOULOUSE CEDEX 4

Abstract

Robots have already reached a high level of autonomy. A robust cooperation in a team of robots would be the next step. In this paper, we consider a mission involving a team of heterogeneous autonomous vehicles, and we propose to make plan-repair as local as possible, in particular to manage the uncertainty on communication availability. For that, we are leaning upon a hierarchical structure of the plan and we have defined an algorithm to automatically extract the team hierarchical structure. This is intended to increase reactivity and efficiency of the team in front of disturbing events at execution time, particularly concerning communication constraints.

Keywords

Multi-Robot Cooperation, Team Structure, Plan Execution, Local Repair.

1 INTRODUCTION

A team of autonomous vehicles must achieve a mission. The team is here composed of AAVs (Autonomous Aerial Vehicles) and AGVs (Autonomous Ground Vehicles), but this work is generic regarding to the type of robots. Each vehicle has already a high level of autonomy, thanks to its own embedded architecture. In consequence, each vehicle has hardware and software means (data fusion and decision algorithms). Many architectures are indeed really mature nowadays and well adapted to the ability of the robot they are controlling [6, 1, 2, 10, 4]. In this work, the robots are all able to move in obstructed environments by following a computed path, identifying an obstacle and locally planning a new path if necessary. The robots can also execute autonomously specific tasks, such as detecting and following a target.

Communication is a key issue for the coordination and information sharing between vehicles. Moreover, when a robot fails in achieving a task, he may need the other robots help, or at least, may want to inform them of the situation to make the team plan change. In this work, all the robots have communication capabilities. However, restriction concerning communications during the mission could be strong. All vehicles are far from being able to communicate with others at all time. Thus, it is recommended to anticipate rendez-vous in order to maximize chances of communication.

1.1 Team Mission Example

The heterogeneous team is formed of two AAVs (noted as aav1 and aav2) and two AGVs (noted as agv1 and agv2) with their own embedded architecture. The operation area is composed of two buildings, trees and obstacles not well localized. The mission of the team is to go from waypoint A to waypoint B and to map zones F and G (Figure 1).

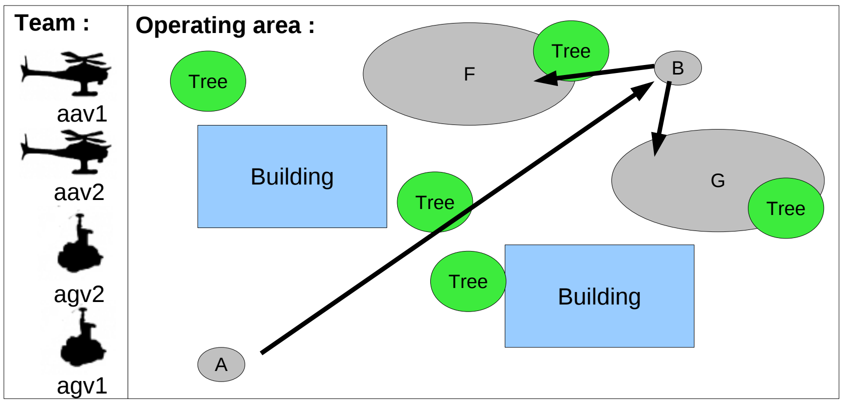

Figure 1: Operation Area

All vehicles are equipped with a video camera and GPS means. Each AAV is a rotorcraft and can fly above ground obstacles, so they have no moving restriction. AGVs moves are limited by buildings, trees' trunks and other terrestrial obstacles. The AAVs can produce a low resolution map of the area overflown. We will call this task *AerialRecon* and AAV map can be used by the AGVs to navigate. AAVs are able to communicate at any moment but communications between an AGV and an AAV or between two AGVs depend on their relative location.

Trees may obstruct AAVs' camera field of view, so they will require help from the AGVs to produce the complete map. The two AGVs are able to take pictures under trees, thus to complete the gaps of the AAV's map.

All vehicles are parked at waypoint A. The initial plan computed off line gives the following tasks: the four vehicles must go to waypoint B and have a rendez-vous at date d1. Then aav1 and agv1 (resp. aav2 and agv2) must map zone F (task *ExploreZone(F)*), (resp. G, *ExploreZone(G)*).

To achieve the task *ExploreZone(F)* (resp. *ExploreZone(G)*), aav1 (resp. aav2) must make an *AerialRecon*, then transmit the map to agv1 (resp. agv2). Eventually agv1 (resp. agv2) must complete the map by a *TerrestrialRecon(F)* (resp. *TerrestrialRecon (G)*).

1.2 Problem

The problem we want to solve is the on-board implementation of the reaction to disturbing events that always occur in such operational missions and invalidates the current plan. Two examples of disturbing events are: (1) during waypoint B rallying phase, the two AGVs are blocked by an unknown obstacle between the two buildings (under trees for example), (2) during the exploration of zone F, aav1 fails achieving its local resolution map, due to poor visibility conditions. The difficulty is therefore to implement the reaction at the adequate level: Which vehicles are concerned by the event ? Which vehicles must be informed of the event ? Which vehicles must compute a new plan ? Could the new plan be only a local repair of the old one ?

We also propose to reuse local robot architectures and to take advantage of a well constructed team structure to execute the mission as efficiently as possible.

1.3 State of the Art

Controlling the execution of a multi-robot system is not a trivial task. Many architectures dealing with this problem have been proposed.

[3] propose a very advanced study in dynamic structures of sub-teams. They notably define the notion of *agenticity* that comes to define a team as an *agenticity hierarchy*, on what we will lean upon. Plans are represented as hierarchical Petri nets. Besides, partial plan repair is explicitly studied. However, the communication issue is not really discussed, and formalism in the planning and the executive phase differs.

Communication is a key issue for a multi-robot system. [14] points out that "the more the team is flexible and robust, the higher communication load is required". Nevertheless, we want our robots to support temporary lack of communication. The mission is constructed around the teamwork core, and not conversely. However, in the same way as [13] and [9], these architectures are based on the BDI framework, which brings huge coherency issues, even more if communications are interrupted. For the moment, it seems difficult to overcome that with this framework.

[7] want to allow plan execution including in situation where communication is not permanent. However, like [11], the plan is distributed to the agents, but there is no local or global repairing. For [9], robots lose their local repairing capabilities when communication is down.

Partial replanning is another key point we are interested in. The Retsina architecture [13] allows it in an original way, based on HTN planning and a dynamic list of objectives stored as a priority queue, but there is no explicit global team plan.

Our contribution is to allow a robot to react to a disturbing event and try to make a plan repair as local as possible, based on a hierarchical organization of a global mission plan, and a well built hierarchical and dynamical team structure. We assume that communication is unpredictable. Contrary to [8] who define a team depending on communication contact, we define a team depending on the vehicles which are involved into the achievement of a task. We also want to keep an explicit view of the sub-teams plans. And last but not least, we want to reuse local architectures that already exist and are mature for local control of a robot.

2 Architecture Overview

In order to increase the repairing capabilities and to organize the partial repair, we propose to model the mission tasks in a hierarchically structured plan.

The team is divided into sub-teams, which may be divided again and again, until the elementary vehicles. This creates a Hierarchical Structure of Sub-Teams (HSST), described further. Each sub-team would be responsible for achieving some specific tasks, and it seems natural to choose a hierarchical plan to make it easily organized. The global supervisor knows which level of the plan needs to be recomputed if a disturbing event occurs, and is able to isolate it easily thanks to the structure.

2.1 HTN Formalism

In this work the task structure of the mission is modeled thanks to HTN (Hierarchical Task Network) formalism [5].

A HTN is a hierarchical set of abstract and elementary tasks. One or several methods are assigned to an abstract task and describe the way to achieve it, using the other abstract or elementary tasks. The selection of a method is constrained by the fulfillment of preconditions. Theoretically, if the recipe provided by a method, picked out amongst the possible methods of the task, is followed, then the objectives of the task will be achieved. Figure 2 shows some classical structures: sequential execution of tasks, choice between two methods and parallel execution of tasks.

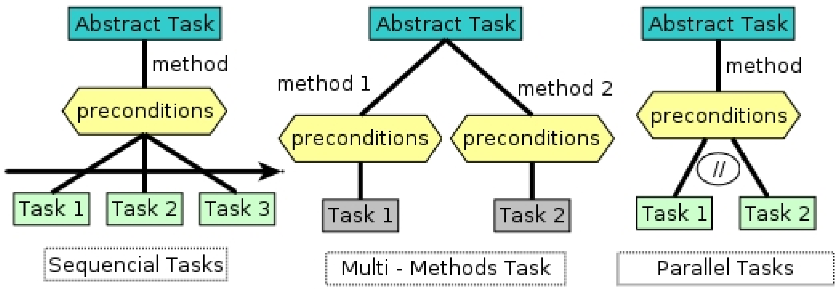

Figure 2: Examples of HTN formalism

2.2 Distribution of the Plan

Before the mission execution, a hierarchical planner, such as JSHOP2 [12], provides an off-line *instantiated HTN*. Classical HTN planners provide a list of completely instantiated elementary tasks that must be executed to achieve the mission. In our work, the tasks are still instantiated but the hierarchical structure of the plan is kept. Each robot must have a copy of the instantiated HTN (Figure 3) , and is going to supervise the execution of the plan: elementary tasks in the instantiated HTN lead to a call to the robot functionalities through its local architecture. The middleware which interfaces the instantiated HTN and the local architecture will be called the *local supervisor*.

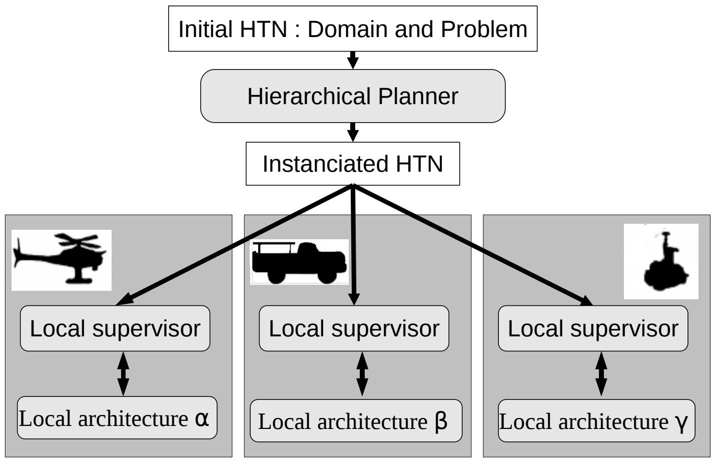

Figure 3: Distribution of the initial HTN plan

3 Plan Execution and Repair

Figure 4 shows an (simplified) instantiated HTN which describes the mission example.We introduce another method for exploration (in dotted line) based on the task *JoinRecon*: the AAV and the AGV explore jointly the zone with the AGV moving according to the AAV directives. This method requires a permanent communication and should last longer.

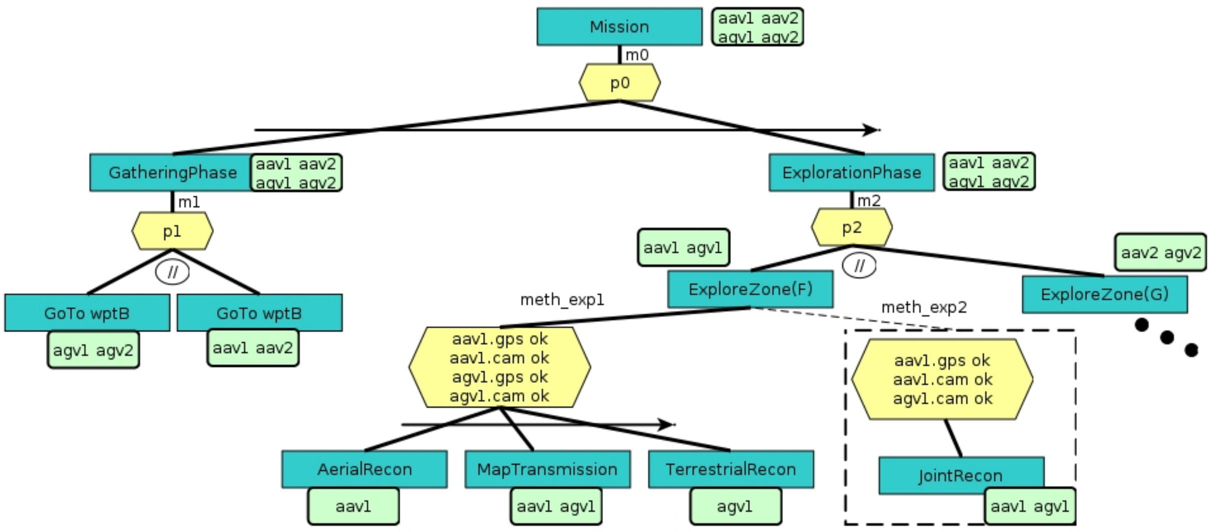

Figure 4: Instantiated HTN mission plan

3.1 Hierarchical Structure of Sub-Teams

At any moment each vehicle must be aware of the sub-team it belongs, especially to allow the repair at the sub-team level. The HSST (Hierarchical Structure of Sub-Teams) must then be extracted from the instantiated HTN.

The HSST is extracted applying Algorithm 1 to the root task of the instantiated HTN (*Mission* task in Figure 4).

Algorithm 1 $Extract(T)$

```
Require: $T$: a HTN task
 1: $A = \emptyset$
 2: if $T$ elementary then
 3:     return Agents(T)
 4: end if
 5: for all $T_i \in$ children($T$) do
 6:     $A_i = Extract(T_i)$
 7:     if $T$ sequential then
 8:         $link(A, A_i, sequence)$
 9:     else if $T$ parallel then
10:         $link(A, A_i, parallel)$
11:     else if $T$ multi-method then
12:         $link(A, A_i, \text{xor})$
13:     end if
14: end for
15: return $A$
```

The $link(A, B, \mathcal{R})$ function adds B hierarchy in A's one, based on relation $\mathcal{R}$. Agents(T) designates the set of vehicles involved in achieving task T.

A result example of Algorithm 1 is shown on Figure 5.

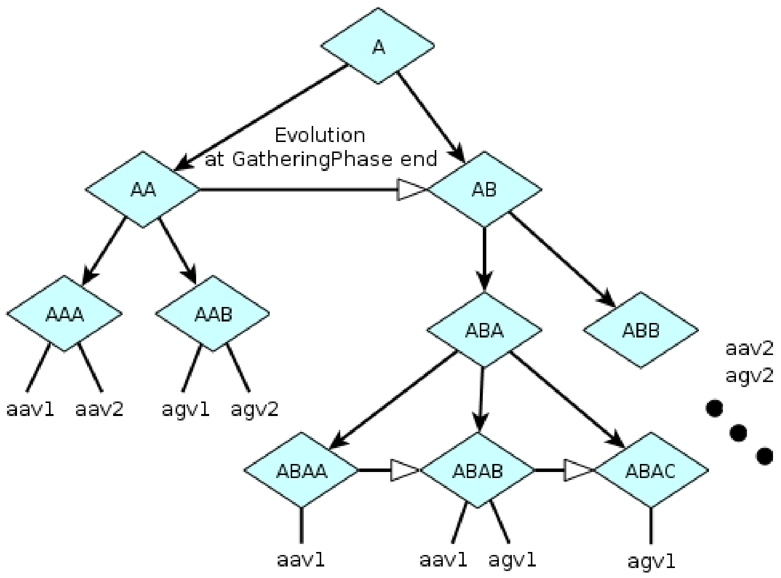

Figure 5: Hierarchical Structure of Sub-Teams corresponding to the instantiated HTN of Figure 4

3.2 Local Execution and Repair

Vehicles taking part in a task are explicitly marked next to the task representation. The local supervisor of each vehicle covers and executes the plan: it transmits the order for executing the elementary tasks to the local architecture when the vehicle is concerned. Then, it waits for the execution result. For example, in order to achieve the *ExploreZone(F)* task, aav1 is responsible for the *AerialRecon*, then agv1 joins aav1 to execute a *MapTransmission* communication task, and eventually agv1 has to execute a *TerrestrialRecon* task. If the tasks are all completed without any disturbing event, the execution of *ExploreZone(F)* is completed.

If an execution fails at the vehicle level, the local supervisor must decide the procedure to apply: it knows which vehicles are concerned thanks to the HSST (where communication links have to be established) and what part of the plan must be recomputed thanks to the HTN structure.

Back to the example, we suppose now that aav1 fails in achieving task *AerialRecon*, so agv1 will not receive a map of F in the joint task *MapTransmission*, which will also fail. Therefore, the sub-team ABAA (aav1) is first concerned about repairing its plan. If aav1 manages to recompute a plan with another way to achieve task *AerialRecon*, it can execute its new plan and the mission can continue. Else, if aav1 does not manage it, its local supervisor must look up in the tasks hierarchy: *ExploreZone(F)* task is concerned by the failing of task *AerialRecon*, so sub-team ABA (i.e aav1 and agv1) will have to replan *ExploreZone(F)*. If such a plan is available, the mission execution goes on with this new plan. For example, method *meth_exp2* (Figure 4) provides a new way (task *JointRecon*) to achieve task *ExploreZone(F)*. This repair is *local*: sub-team ABB has not been concerned by replanning, so a communication link has not been required. Sub-team ABB still executes *ExploreZone(G)*.

If no local repair had been possible at this level, the local supervisor would look up in the task hierarchy, and repeat the process until the root task is reached.

4 Future Work

In this paper, we have presented how to construct a HSST from an instantiated HTN team plan, and the way it could be used to repair the plan as locally as possible. However two key points have to be deeply studied:

In our previous example, ABA had to repair its local plan, but we did not take into account the possibility for the teammates to communicate. In the future works, we will have to define communication recovery procedures in cases when communication is essential to replan at a sub-team level.

In the same example, ABB was not concerned by the dynamic repair of ABA. Therefore, this points out a consistency issue: in ABB's plan, ABA still executes *meth_exp1*. We propose to overcome this by distributing only the relevant part of the off-line instantiated HTN to the vehicles' local supervisor. Hence, ABB would know that ABA is executing *Explorezone(F)*, but not the way it's doing it.

That will be integrated on-board autonomous vehicles and will be evaluated through several experiments.

Acknowledgments. This work is part of the Action Projet (`http://action.onera.fr/`), supported by the French Defence Agency DGA.

References

[1] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand, 'An architecture for autonomy', *IJRR*, **17**(4), (1998).

[2] Magali Barbier, Jean-François Gabard, Dominique Vizcaino, and Olivier Bonnet-Torrès, 'ProCoSA: a software package for autonomous system supervision', in *CAR*, Montpellier, France, (2006).

[3] Olivier Bonnet-Torrès and Catherine Tessier, 'From teamplan to individual plans: a Petri net-based approach', in *AAMAS*, Utrecht, The Netherlands, (2005).

[4] P. Doherty, J. Kvarnström, and F. Heintz, 'A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems', *JAAMAS*, **19**(3), (2009).

[5] K. Erol, J. Hendler, and D. Nau, 'HTN planning: complexity and expressivity', in *AAAI*, (1994).

[6] Guillaume Infantes, Charles Lesire, Henry de Plinval, and Florent Teichteil, 'An Orocos-based decisional architecture for the Ressac missions', in *CAR*, Toulouse, France, (2009).

[7] Sylvain Joyeux, Rachid Alami, and Simon Lacroix, 'A plan manager for multi-robot systems', in *FSRS*, Chamonix, France, (2007).

[8] François Legras and Catherine Tessier, 'LOTTO: group formation by overhearing in large teams', in *AAMAS*, Melbourne, Australia, (2003).

[9] Martin Magnusson, David Landén, and Patrick Doherty, 'Planning, executing, and monitoring communication in a logic-based multi-agent system', in *ECAI*, Patras, Greece, (2008).

[10] C. McGann, F. Py, K. Rajan, H. Thomas, R. Henthorn, and R. McEwen, 'A deliberative architecture for AUV control', in *ICRA*, (2008).

[11] Roberto Micalizio and Pietro Torasso, 'Supervision and diagnosis of joint actions in multi-agent plans', in *AAMAS*, Estoril, Portugal, (2008).

[12] Dana Nau, H. Muñoz-Avila, Y. Cao, A. Lotem, and S. Mitchell, 'Total-order planning with partially ordered subtasks', in *IJCAI*, Seattle, WA, USA, (2001).

[13] Massimo Paolucci, Onn Shehory, and Katia Sycara, 'Interleaving planning and execution in a multiagent team planning environment', *Linköping Elec. Articles in CIS*, **5**(18), (2000).

[14] Milind Tambe, 'Towards flexible teamwork', *JAIR*, **7**, (1997).