Integrating planning and execution for a team of heterogeneous robots with time and communication constraints

Patrick Bechon, Magali Barbier, Christophe Grand, Simon Lacroix, Charles Lesire, Cédric Pralet

Integrating planning and execution for a team of heterogeneous robots with time and communication constraints

Patrick Bechon[1], Magali Barbier[1], Christophe Grand[1], Simon Lacroix[2], Charles Lesire[1] and Cédric Pralet[1]

Abstract— **Field multi-robot missions face numerous unavoidable disturbances, such as delays in executing tasks and intermittent communications. Coping with such disturbances requires to endow the robots with high-level decision skills. We present a distributed decision architecture based first on a hybrid planner that can manage decentralized repairs with partial communication, and secondly on a distributed execution algorithm that efficiently propagates delays. This architecture has been successfully experimented on the field for the achievement of surveillance missions involving eight (8) real autonomous aerial and ground robots.**

I. INTRODUCTION

Teams of robots provide numerous operational benefits in surveillance and search and rescue missions, in which they can provide in real time a clear knowledge of the situation to first responders and safety officers. Such missions span large distance and time ranges. This leads to two main challenges with respect to automated decision-making: *(i)* temporal constraints, that are imposed either by operational objectives (e.g. allowed time to perform a specific observation), technical limitations (e.g. maximal time of flight of a robot) or other constraints (e.g. visibility windows with a communication satellite); and *(ii)* communication requirements, induced by the necessity to regularly report the situation to the operators and to coordinate the various robots tasks.

Satisfying these requirements is all the more challenging that unavoidable disturbances pop up during the mission execution, the most frequent being delays in the execution of tasks and failures to properly communicate. Besides these disturbances, the mission itself may also preclude the mere execution of a pre-defined mission plan: for instance the appearance of an alarm may require confirmation, or the detection of an intruder may trigger a tracking task. Facing these disturbances and events calls for the ability to adapt to complex situations by dynamically monitoring and re-planning the execution of the mission. Furthermore, because of communication constraints and complexity issues, this ability must be distributed among the robots.

In this paper, we describe a multi-robot deliberative architecture applied to a surveillance mission. This architecture is based on previous work in which we developed and benchmarked some algorithms for planning [1], plan repair [2], and multi-agent plan execution [3]. In this paper, we show how we have orchestrated these algorithms within a multi-robot deliberative architecture in order to cope with time constraints, disturbances and partial communications. Moreover, this architecture has been evaluated both in simulation and on the field through experiments involving 8 real robots (Fig. 1).

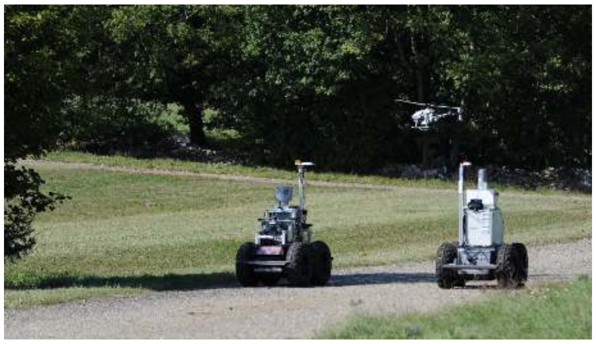


Fig. 1: Two ground robots and an autonomous helicopter involved in a surveillance mission.

Section II presents the related work about multi-robot planning dealing with time and communication constraints. The previously published algorithms on which is based our architecture are briefly summarized in Section III. Section IV then describes the proposed architecture using algorithms that were adapted to comply with the multi-robot context. This architecture has been first evaluated in simulation (Section V) and then deployed on the field (Section VI). We finally conclude this paper (Section VII).

II. RELATED WORK

Autonomous multi-robot systems have received a lot of attention for the last decade. Several architectures and paradigms have been proposed to deal with mission planning and execution in presence of communication limits. A common approach is to consider that each agent performs its own plan computation while some goals can be exchanged between agents within communication windows [4], [5], [6], [7]. In this approach, no global plan exists and the system is almost unpredictable. The same kind of paradigm with auction or market-based approaches have been also proposed in [8], [9], [10], [11]. Other work manage a global plan in a centralized approach [12], limiting the adaptability of its execution.

Another approach consists in having a planner that computes offline a global plan that enforces some synchronizations between robots. This may result in implicitly

[1]P. Bechon, M. Barbier, C. Grand, C. Lesire and C. Pralet are with ONERA – The French Aerospace lab, Toulouse, France `firstname.lastname@onera.fr`

[2]S. Lacroix is with LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France `simon.lacroix@laas.fr`

synchronized plans (e.g. plans are temporally synchronized, but no message is sent between robots, [13], [14], [15]), or explicitly synchronized plans [16], [17], [18]. In the former case, the plan execution is not robust against lateness in tasks execution, and some communication-based back-up strategies must be used [13]. In the latter case, the planning paradigm does not manage temporal constraints nor decentralized repairs when communications are unavailable.

III. BACKGROUND

A. Hybrid planning with HiPOP

The constraints raised by the fielded multi-robot missions have led us to focus on Partial-Order Planning (POP) and Hierarchical Planning (HP). POP [19] is indeed suited to solve temporal problems, with the capability to solve partial problems, which is well suited to repair with partial communication. HP [20] is an efficient planning paradigm where hierarchical information help focusing on specific branches of the search. In previous work [1], we have designed HiPOP, a hybrid POP/HP algorithm.

The main concept of POP is to iteratively build partial plans that may contain *flaws*, i.e. problems that hinder the plan to be complete. Each flaw can be *resolved* in different ways, leading to new candidate plans to be investigated (Alg. 1).

Algorithm 1: HiPOP

```
Input : A problem instance $P = \{L, A, Init, Goal\}$
1  $\mathcal{P} = \{InitialPartialPlan(Init, Goal)\}$ ;
2  while $\mathcal{P} \neq \emptyset$ do
3      $\Pi = PopBestPlan(\mathcal{P})$ ;
4      if $\mathcal{F}(\Pi) = \emptyset$ then
5          return $\Pi$ ;
6      end
7      $f = PopBestFlaw(\mathcal{F}(\Pi))$ ;
8      $\mathcal{P} = \mathcal{P} \cup Resolvers(A, \Pi, f)$ ;
9  end
10 return $\emptyset$
```

The problem P to solve is defined by a set of literals L, a set of available actions A, an initial state $Init$ and a goal state $Goal$. An initial partial plan is built including only the initial and goal states (line 1). The plan is then modified by iteratively adding elements while maintaining a partial order between actions. To add these elements, the best plan Π built so far is chosen, with respect to criteria that can be tuned within the *PopBestPlan* function (see Sec. IV-C for the criterion used in our architecture). If this plan is complete (i.e. its set of flaws $\mathcal{F}(\Pi)$ is empty), the problem is solved. Otherwise, a flaw to resolve is selected (line 7), and we add to the set of candidate plans all the possible resolvers of this flaw (line 8).

In classical POP, possible flaws are either an *open link* (a precondition not satisfied) or a *threat*, i.e. when an action can delete a precondition guaranteed by a causal link. To resolve an open link, we can add a causal link with an existing action in the plan, or add a new action that provides the needed literal and a causal link. To resolve a threat, the threatening action can be placed before or after the threatened link.

In HiPOP, actions can be *abstract*, meaning that we only have a high level description of a set of actions that has not been instantiated yet. A new kind of flaw then arises: *abstract flaws*, when abstract actions in the plan have not been instantiated. To resolve this flaw, we can instantiate this abstract action by any of its associated available methods.

B. Iterative Repair with HiPOP

In [2] we have proposed an Iterative Repair Algorithm based on HiPOP in order to repair plans when some failure or some unexpected event occurs during plan execution. The principle of Iterative Repair (Fig. 2) is to reuse the HiPOP algorithm as it is, iteratively starting a new search from a partial plan instead of an initial empty plan.

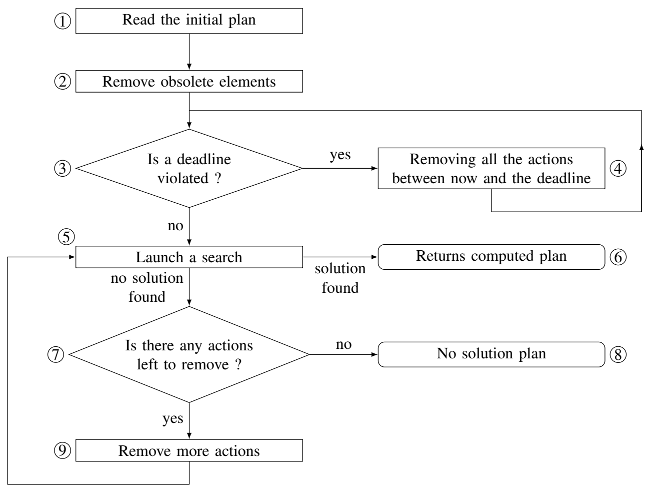


Fig. 2: Iterative Repair Algorithm.

When a repair is triggered, the current plan is loaded (step 1), and obsolete actions (i.e. actions that are no more feasible due to the failure) are removed (step 2). Then, the deadlines (i.e. the temporal constraints) present in the plan are checked (step 3). If a deadline is violated, we remove all the actions between the current time and the deadline (step 4), so that HiPOP will have to rebuild this part of the plan while ensuring the deadline again. Then HiPOP is launched on the resulting initial partial plan (step 5). If a solution is found, the plan is successfully repaired (step 6). Otherwise, the initial partial plan is reduced again by removing more actions (step 9): actions that were linked to previously removed actions are now removed, and HiPOP is launched again on this lighter partial plan (step 5). In case the initial partial plan is empty (checked in step 7), HiPOP has tried to perform a complete replan without success, meaning that the plan cannot be repaired at all (step 8). Note that, as the size of the initial plan considered by HiPOP in step 5 strictly decreses (as we remove more actions), the repair process is guaranteed to finish.

C. HiPOP plan execution

The plan computed by HiPOP takes the form of partially ordered actions. Each action has a starting and an ending timepoint. The plan also contains temporal constraints on these timepoints. This plan can then be modeled as a Simple Temporal Network (STN [21]), which is a common paradigm to represent temporal plans and their execution [22], [23]. STNs are appealing in practice because several problems are solvable in polynomial time, such as determining when each timepoint must be executed in order to satisfy all temporal constraints. After constraints propagation, each timepoint is associated with a time window $[l, u]$: any execution between these bounds will satisfy all temporal constraints and induce a new propagation to compute new time windows for the other points.

Some propagation algorithms, such as found in [24], allow for a quick *relaxation* of constraints. These enable STNs to efficiently handle execution failures that lead to constraint violations: for instance, a robot can take more time than expected to move between two points. In this case, relaxation algorithms can ascertain if these unexpected delays were small enough to be silently absorbed by execution without impacting the plan validity, or if the plan cannot be correctly executed anymore and needs to be repaired.

STNs have been extended to Multi-agent STN (MaSTN [25]) in which each timepoint is associated with an agent, which is the only one that can execute it. In [3], some distributed algorithms have been proposed to effectively minimize the needed communications to efficiently propagate the constraints among the whole STN.

IV. A DISTRIBUTED ARCHITECTURE FOR SURVEILLANCE MISSIONS

A. Mission definition and modeling

In this paper, we consider a generic surveillance mission over a known terrain. The mission is defined as a set of positions to observe and is to be achieved by a team of aerial and ground robots: by integrating complementary motion and observation capacities: teams of heterogeneous robots have indeed interesting characteristics for various large scale field scenarios [26], [27]. Each robot is endowed with the capacity to detect moving ground targets, and upon detection, the robot enters a target tracking phase. The mission has been modeled as a planning problem, with:

- a set of *observations* to perform; each observation is a position in the area to visit; observations points have been randomly generated based on zones of interest;
- a set of available *positions* for each kind of robot; robots being heterogeneous, not all positions can be reached by all robots;
- an *observability function* that indicates for each position p and each position to observe o if o is observable from p; it uses ray tracing and a digital terrain model of the area to determine if a given robot is able to observe o from p;
- a *reachability function* that indicates if a position p is reachable from a position q for a specific robot, and the cost of the motion from q to p;
- a *communication function* that indicates, for a robot r_1 being at position p_1 and a robot r_2 being at position p_2, if r_1 and r_2 can communicate; it is based on ray tracing from p_1 to p_2 and on a given communication range.

B. Architecture overview

We have defined an architecture in order to deploy a team of autonomous robots that could efficiently achieve such a surveillance mission while encompassing disturbances. This architecture settles on the work summarized in section III, which offer the following benefits:

- HiPOP [1] allows to compute flexible temporal plans to allocate observations to the robots, while ensuring temporal constraints;
- the repair process [2] allows to repair plans when disturbances occur, while trying to minimize the computational cost through iterative repair;
- the STN execution mechanism and its distributed extension [3] allow to handle execution delays before trying to repair the plan.

These algorithms have been slightly adapted and extended to be integrated within the same architecture and to allow multi-robot missions with time and communication constraints. The main points addressed are:

- the definition of metrics and set of actions for the planner HiPOP;
- the conditions, related to disturbances, on which a repair must be triggered;
- the management of partial communications (i.e., some robots are unable to communicate with other robots).

The considered disturbances can be gathered in three main categories:

- one (or more) robot becomes unavailable (battery off, robot lost because of navigation default, locomotion system failure, ...)
- communication is not available at a meeting point,
- delays in robot plan execution become too important.

C. Computation of the initial plan

The computation of the initial plan is done by HiPOP. In addition to the problem defined in IV-A, the available actions are **move** actions (i.e., joining two *positions* based on the *reachability function*), **observation** actions (i.e., performing an *observation* from a *position* based on the *observability function*), and **communication** actions. Move and observation actions are allocated to only one robot. Communication actions are defined as joint actions between two robots. These actions are defined in the planning problem to ensure that at some time the robots will be in communication. A deadline is set on these tasks to enforce that regular communication will actually be achieved. The criterion considered to select the best candidate plan in the planning algorithm (function $PopBestPlan$ in Alg. 1) is the total duration of the mission.

The initial plan computed by HiPOP then defines for each robot a series of positions to be reached, observations to be done, and some inter-robot communications rendezvous. This plan takes into account the time spent by robots to reach waypoints, as well as synchronization of multi-robot actions, and tends to minimize the mission duration.

D. Distributed execution of plans

The plan is executed using the distributed propagation algorithm described in [3], that allows to efficiently propagate delays within a team of robots. This algorithm still works even in absence of communication, by making the assumption that no failure invalidating the plan happens on other robots while communications are down. In the missions we consider, this assumption is reasonable: robots have constraints between their tasks only when they need to communicate, in which case they will be physically close, meaning that communication should be available. The consequence of a *missed* communication when a remote delay occurs is that the deliberative architecture will not be able to anticipate on a future missed deadline, and then will not adapt the plan or repair in advance.

E. Repairing process in multi-robot context

In the distributed architecture we propose, plan repair is handled by each robot. The cases that trigger a repair are: (1) a delay that cannot be propagated successfully (the MaSTN becoming inconsistent), (2) a robot is out of service (breakage or empty battery), (3) a robot switched to the target tracking mode (hence not achieving its initial plan), and (4) a human operator request that breaks the execution of a plan. The repairing robot then starts the repairing process by first advertising all its teammates that it will try to repair the plan, making the other robots pause their plan execution. Then the repair is performed according to the process of Fig. 2 until a plan is found. The new plan is finally sent to the other robots so that they can continue the execution.

The Iterative Repair algorithm has been adapted such that all the actions of a robot that become unavailable (out of service or reallocated to a target tracking) are removed from the current plan (step 2 of Fig. 2). Thus, HiPOP will try to reallocate all the observations made by this robot to the other available robots.

F. Repairing with partial communications

In case of partial communication, some of the teammates of the repairing robot may be out of communication range. In that case, the repair is partial: the repairing robot will try to repair the plan without modifying the actions of the unreachable robots. The only consequence of this process is that the actions of the unreachable robots cannot be removed from the current plan in step 9 of Fig. 2.

When an unreachable teammate can communicate again, its local plan and the new repaired plan need to be fused. If the remote robot has only executed actions and propagated delays, this fusion is always possible. Otherwise, a *plan merging* has to be triggered. Plan merging is a native feature of POP algorithms. Possible inconsistencies in the resulting plan are handled like a classical failure: a new repair is launched following the process of Fig. 2. Note that these inconsistencies may only arise when a robot has withdrawn a task of its plan (either a goal to observe, or a rendezvous). A degenerate situation may happen where each robot will repair on its own and that communications are restored one robot after the others. In this case, as the number of tasks that can be withdrawn is finite, the number of needed repairs will also be finite, guaranteeing that the iterative process will end.

Finally, it may occur that a plan cannot be repaired at all, for instance if a time constraint cannot be reached at all, or if an observation cannot be done by any robot. In such a situation, we have decided to give the decision back to the operator, who can decide to withdraw a part of the problem, such as dropping observation points, or removing deadline constraints. This will lead to a new partial plan that will enter the iterative repair process so that the mission can go on.

V. SIMULATION RESULTS

We first evaluated the performance and robustness of our architecture through simulations. The simulated scenario includes 12 robots, 75 positions, and 30 observation points. We ran simulations with disturbing events, including delays in the achievement of motions, robots allocated to the tracking of a target, or robot failures. Figures 3 and 4 show the results of these simulations. Each line represents a scenario of simulation (no disturbance, one broken robot, one broken robot + one robot out of communication, . . .). For each line, we performed 30 runs in which the robot impacted by the disturbance and the date of the disturbance are randomly drawn.

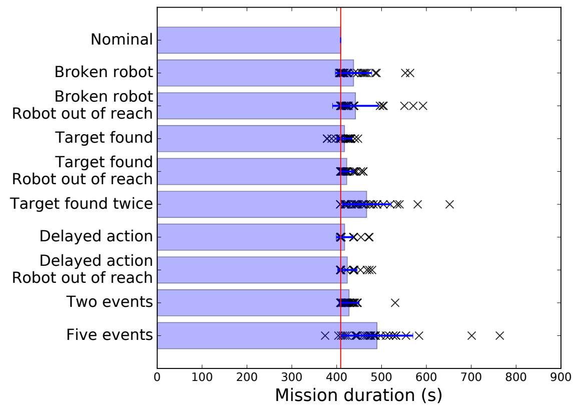


Fig. 3: Duration of the simulated missions. Each cross represents the result of one run. Blue bars represent the average on the 30 runs. The vertical red bar is the average result for the nominal mission.

Fig. 3 shows the duration of the mission. The longest missions are in the last pattern when 5 random events happen. When 2 targets are found, the observation tasks of two robots have to be reallocated, which also significantly increases the mission duration. But this increase is always relatively limited, and less than 20% on average. Moreover, for every situation, the mission has been successfully achieved.

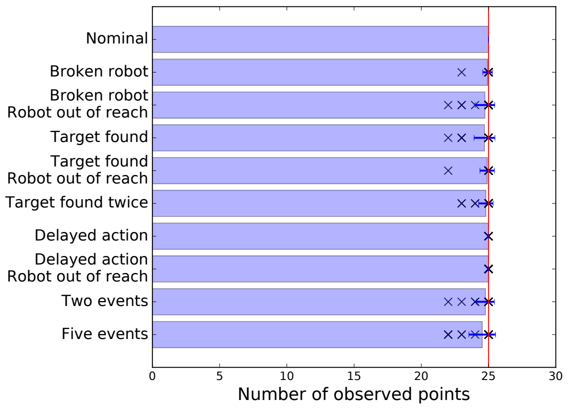


Fig. 4: Number of observations performed during the simulated missions. Each cross represents the result of one run, blue bars represent the average on 30 runs. The vertical red bar is the average result for the nominal mission.

Fig. 4 shows the number of points of interest that are successfully observed during the mission. When a robot breaks down or is assigned to a target, its actions must be reallocated to other robots. In our scenario, there are 3 points of interest that can only be observed by only one robot. So if this robot fails, 1, 2 or 3 of these points can no longer be observed. This explains why the minimum number of observed points is 22 out of 25. Even if all points of interest have not been observed, these missions are considered successful from the architecture point of view: this is the best possible result regarding the failures that occurred.

VI. FIELD RESULTS

The surveillance mission has been experimented with 8 real robots on the field. Two aerial robots and six ground robots with different perception and moving skills were involved. Nine missions were run during an experimental demonstration day on an area of 200m x 200m with scattered buildings. In this section, we present the results obtained for two of these missions. Each mission has 6 communication constraints, 25 points to observe, 93 robot positions. An initial plan has been computed by HiPOP in approximately 8 seconds for each mission. The initial plans vary by the way observations and communications are allocated to the several robots. However, in all the missions, the initial plan contains 104 actions and lasts around 410 seconds.

A. *Mission with delays*

Fig. 5 and 6 respectively present the trajectories and the schedule of a mission achieved in nominal conditions, meaning that no repair was needed. Some delays occured at execution but were successfully propagated in the MaSTN.

Fig. 6a shows the timeline of the initial plan, and Fig. 6b shows the plan actually executed. We can notice in Fig. 5 that the trajectories of the robots have been quite disturbed, especially for aerial robots, due to wind conditions. It results in some delays at execution that can be noticed on the plan of robot *ressac2*. The motions of the ground robots were also underevaluated, which caused a lot of delays at execution (especially robots *effibot1* and *mana*). The robots successfully performed 119 tasks, and they achieved the mission in 449 seconds, exceeding by 9% only the initial plan duration despite the encountered delays.

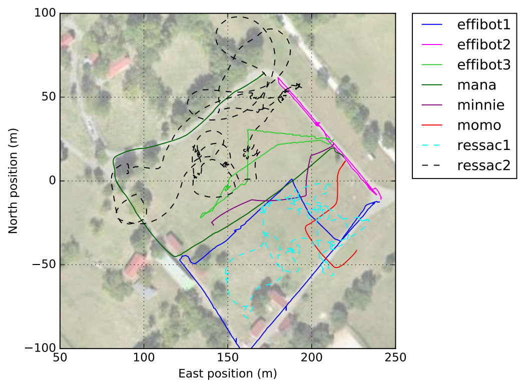


Fig. 5: Trajectory performed by the robots during the mission with only delays (no online repair needed). *ressac*s are aerial robots as shown in Fig. 1, *effibot*s are ground robots that can evolve on paved roads, other robots are all-terrain ground robots as shown in Fig. 1.

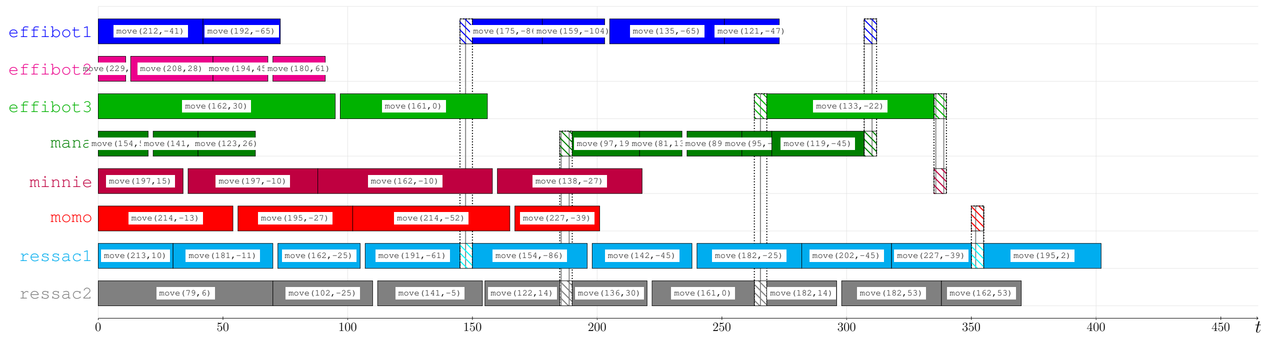
B. *Mission with failures*

Fig. 7 and 8 show the realization of a mission including failures and repairs. Fig. 8a shows the timeline of the initial plan, and Fig. 8b shows the plan actually executed.

In this mission, two robots were declared out-of-service (*effibot1* and *momo*), due to failures in the navigation processes (the robots were unable to reach their next position). These failures are drawn in the timeline with black crosses at the end of a move action. For the robot *momo*, the failure has been detected by the operator who stopped the robot. No repair has been triggered as this information has not been shared with the other robots. Note the *ressac2* was supposed to communicate with *momo*, but withdrawn the communication in order to ensure the deadlines on the rest of its plan. *effibot3* also withdrawn its communication with *ressac2* (that was late) in order to ensure its deadline on its communication with *minnie*.

At time 480, *effibot1* reported a failure as it could not move (blocked by an obstacle). This triggered a repair process, that reallocated to other robots the observations that were not anymore achievable by *effibot1* and *momo*. The *move* actions corresponding to these observations are shown in the timeline of Fig. 8b with a grid pattern for robots *effibot2* and *effibot3*.

Finally, the team of robots performed 99 tasks. The mission has been achieved in 713 seconds (73% more than the initial plan), with all the observation points being observed.


(a) Initial plan

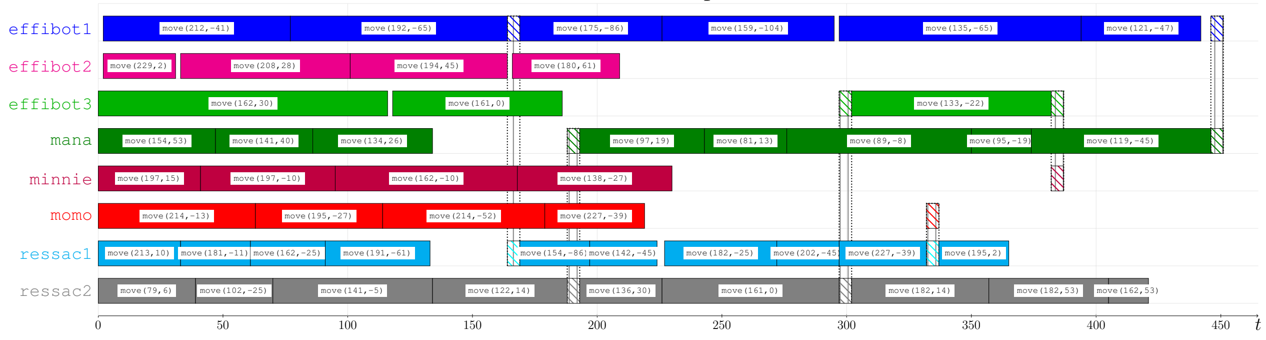

(b) Plan executed by the robots

Fig. 6: Temporal representation of plans (x-axis in seconds). Each line represents the plan of one robot. Communications between two robots are represented by the dashed vertical bars. Observation actions are not visible as their duration is null.

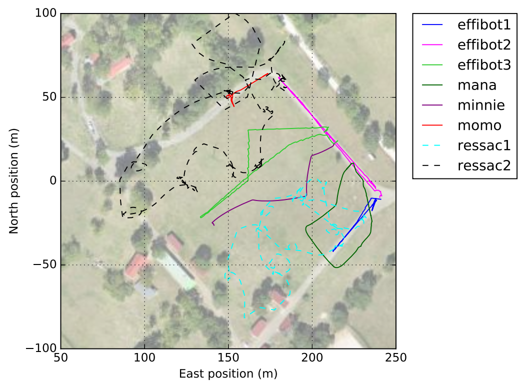
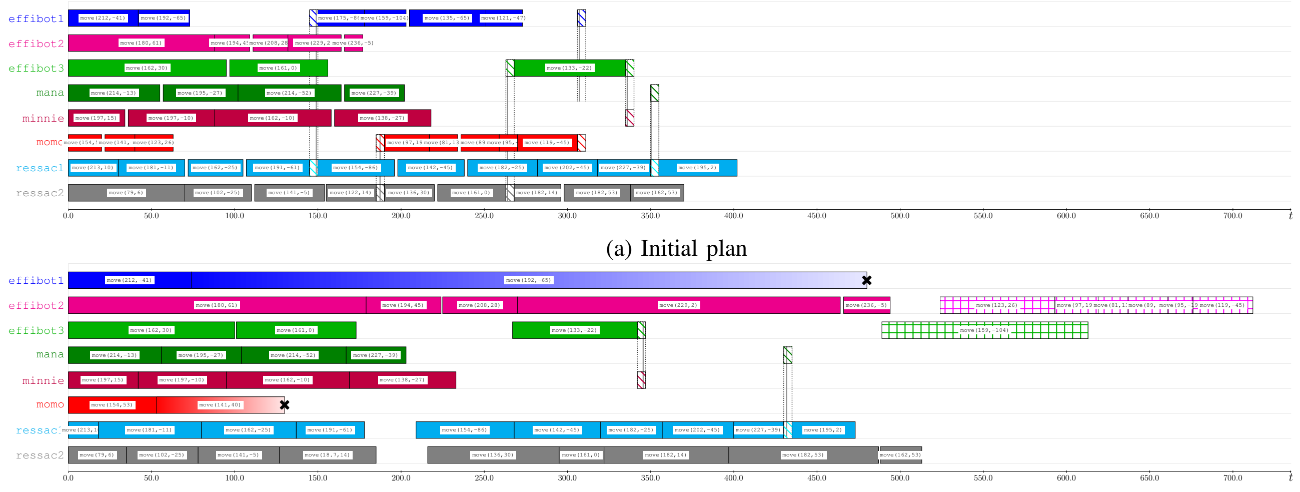


Fig. 7: Trajectory of the robots during a mission with two robots broken.

VII. CONCLUSION

The objective of this research was to implement onboard heterogeneous robots the capability to achieve a realistic surveillance mission on their own, that is with limited communication with operators. This led to the development of a decisional architecture distributed onboard all robots that allows the team to react to disturbing events. HiPOP is first used offline to produce an initial plan with rendezvous to ensure mission consistency as communications are not permanent. Each robot executes its plan and synchronizes with teammates thanks to MaSTN propagation. The plan is temporally flexible, but when it is not valid anymore an iterative repair process running HiPOP allows to compute a new plan, even when some robots are out of communication range. Statistical evaluations have been performed to validate the robustness of the architecture with respect to several disturbances popping up at random times and places during the mission. These evaluations have shown that the architecture ensures the successful achievement of the mission, the robots repairing the plan to perform all the feasible observations, with little impact on the duration of the mission.

Field experiments have been done with 8 robots on an area of 200m x 200m with scattered buildings. Two aerial robots and six ground robots with different perception and moving skills were involved and nine missions were realized. The results for two of them have been detailed, in which the team succeeded in adapting to various disruptive events. The team of robots has spent a total time of 2 hours on the field, performing 18 repairs autonomously. To the best of our knowledge, these experiments were among the rare successful experiments involving up to 8 robots in the field. The robots we have deployed are heavy ones (several tens of kilograms each), and require significant logistic efforts both in terms of infrastructure and people involved (15 persons were involved in the experiments). In such conditions, monitoring the status of each robot and their coordination by human operators is not possible. Despite



(a) Initial plan

(b) Plan executed by the robots

Fig. 8: Temporal representation of plans (x-axis in seconds). Each line represents the plan of one robot. Communications between two robots are represented by the dashed vertical bars. Observation actions are not visible as their duration is null.

these difficulties, the success of each mission, in various non-nominal conditions, is an illustration of the interest, in terms of resilience, of the multi-robot decisional architecture we have developped.

Acknowledgements

This work has been supported by the ACTION project founded by the French Defense Procurement Agency (DGA).

References

[1] P. Bechon, M. Barbier, G. Infantes, C. Lesire, and V. Vidal, "HiPOP : Hierarchical Partial-Order Planning," in *STAIRS*, Prague, Czech Republic, 2014.
[2] ——, "Using hybrid planning for plan reparation," in *ECMR*, Lincoln, UK, 2015.
[3] G. Casanova, C. Pralet, and C. Lesire, "Managing Dynamic Multi-Agent Simple Temporal Network," in *AAMAS*, Istanbul, Turkey, 2015.
[4] B. Coltin and M. Veloso, "Optimizing for Transfers in a Multi-Vehicle Collection and Delivery Problem," in *DARS*, Baltimore, MD, USA, 2012.
[5] C. Sung, N. Ayanian, and D. Rus, "Improving the Performance of Multi-Robot Systems by Task Switching," in *ICRA*, Karlsruhe, Germany, 2013.
[6] V. Raman, "Reactive Switching Protocols for Multi-Robot High-Level Tasks," in *IROS*, Chicago, IL, USA, 2014.
[7] A. Whitbrook, Q. Meng, and P. Chung, "A Novel Distributed Scheduling Algorithm for Time-Critical Multi-Agent Systems," in *IROS*, Hamburg, Germany, 2015.
[8] L. Luo, N. Chakraborty, and K. Sycara, "Distributed Algorithm Design for Multi-Robot Task Assignment with Deadlines for Tasks," in *ICRA*, Karlsruhe, Germany, 2013.
[9] G. Settembre, P. Scerri, A. Farinelli, K. Sycara, and D. Nardi, "A decentralized approach to cooperative situation assessment in multi-robot systems," in *AAMAS*, Estoril, Portugal, 2008.
[10] J. Acevedo, B. Arrue, I. Maza, and A. Ollero, "Cooperative perimeter surveillance with a team of mobile robots under communication constraints," in *IROS*, Tokyo, Japan, 2013.
[11] S.-k. Yun and D. Rus, "Distributed coverage with mobile robots on a graph: locational optimization," in *ICRA*, Saint-Paul, MN, USA, 2012.
[12] D. Mitchell, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot Persistent Coverage with Stochastic Task Costs," in *IROS*, Hamburg, Germany, 2015.
[13] A. Korsah, B. Kannan, B. Browning, A. Stentz, and B. Dias, "xBots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies," in *ICRA*, Saint-Paul, MN, USA, 2012.
[14] Y. Zhang and L. Parker, "Multi-Robot Task Scheduling," in *ICRA*, Karlsruhe, Germany, 2013.
[15] N. Mathew, S. L. Smith, and S. L. Waslander, "A Graph-Based Approach to Multi-Robot Rendezvous for Recharging in Persistent Tasks," in *ICRA*, Karlsruhe, Germany, 2013.
[16] T. Gateau, C. Lesire, and M. Barbier, "HiDDeN: Cooperative Plan Execution and Repair for Heterogeneous Robots in Dynamic Environments," in *IROS*, Tokyo, Japan, 2013.
[17] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated Multi-Robot Exploration Under Communication Constraints Using Decentralized Markov Decision Processes," in *AAAI*, Toronto, Canada, 2012.
[18] M. Spaan and F. Melo, "Interaction-driven Markov games for decentralized multiagent planning under uncertainty," in *AAMAS*, Estoril, Portugal, 2008.
[19] H. Younes and R. Simmons, "VHPOP: Versatile heuristic partial order planner," *JAIR*, vol. 20, pp. 405–430, 2003.
[20] D. S. Nau, T. C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN planning system," *JAIR*, vol. 20, no. 1, pp. 379–404, 2003.
[21] R. Dechter, I. Meiri, and J. Pearl, "Temporal Constraint Networks," *Artificial Intelligence*, vol. 49, no. 1-3, pp. 61–95, 1991.
[22] M. D. Rocco, F. Pecora, and A. Saffiotti, "When robots are late: Configuration planning for multiple robots with dynamic goals," in *IROS*, Tokyo, Japan, 2013.
[23] S. Lemai and F. Ingrand, "Interleaving temporal planning and execution in robotics domains," in *AAAI*, San Jose, CA, USA, 2004.
[24] A. Cesta and A. Oddi, "Gaining Efficiency and Flexibility in the Simple Temporal Problem," in *TIME*, Key West, FL, USA, 1996.
[25] J. C. Boerkoel, L. Planken, R. Wilcox, and J. A. Shah, "Distributed Algorithms for Incrementally Maintaining Multiagent Simple Temporal Networks," in *ICAPS*, Rome, Italy, 2013.
[26] G. D. Cubber, D. Doroftei, Y. Baudoin, D. Serrano, K. Chintamani, R. Sabino, and S. Ourevitch, "ICARUS: An EU-FP7 project Providing Unmanned Search and Rescue Tools," in *IROS 2012 Workshop ROSIN*, Vilamoura-Algarve, Portugal, 2012.
[27] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis, "The SHERPA project: smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *ISRR*, College Station, TX, USA, 2012.