Next Generation of Airborne Platforms: From Architecture Design to Sensors Scheduling

Ludovic Grivault, Amal El Fallah-Seghrouchni, Raphaël Girard-Claudon

Preliminary Version

Next Generation of Airborne Platforms From Architecture Design to Sensors Scheduling

Ludovic Grivault
Laboratoire d'Informatique de Paris 6
Thales Airborne Systems
Paris, France
ludovic.grivault@lip6.fr

Amal El Fallah-Seghrouchni
Laboratoire d'Informatique de Paris 6
Paris, France
amal.elfallah@lip6.fr

Raphaël Girard-Claudon
Thales Airborne Systems
Elancourt, France
raphael.girard-claudon@fr.thalesgroup.com

***Abstract*—Airborne platforms such as Remote Piloted Aircraft Systems (RPAS) are operating in highly critical contexts. The next generation of RPAS will be endowed with multifunction sensors (i.e. each sensor offers a large panel of functions to the platform's manager during the mission). As a platform, RPAS carry out a wide collection of complex tasks, thanks to the interleaving of the various services of sensors. The sensors are in charge of collecting data from the environment. In this paper, we aim to design a system as a software medium layer between the platform manager and the hardware resources on board of the airborne platform (i.e. multifunction sensors). Today, the requirements of the platform in terms of autonomy, modularity, robustness and reactivity as well as the industrial constraints call for the design of a new multifunction system architecture. Such a design may rely on multi-agent paradigm since it is modular by design and the agents naturally bring autonomy and proactivity to the system. This paper presents new and original contributions: (1) an original agentification of the system in the form of a multi-agent architecture that captures the dynamic of the environment by creating tactical objects (i.e. agents) that may appear in the mission theater; (2) agents that generates a plan of tasks according to the resources (e.g. the sensors) he needs; (3) a scheduler that handles the plans of tasks issued by the agents in order to provide an efficient scheduling of sensors.**

***Index Terms*—Sensor Suite; Autonomous System; RPAS; Multi-agent Systems; Multi-function; Agent.**

I. The context

Nowadays, airborne platforms are used worldwide as a strategic asset during different kinds of operations including conflicts, surveillance and rescue. These operations occur in highly dynamic environments with a low predictability under scenarios combining up to a thousand entities. The involved entities all have their own behaviors, speeds and trajectories. In this context, onboard instruments (i.e. sensors) allow the platform, hence the mission manager, to collect knowledge from the field.

Throughout the years, sensors have become complex systems, multinational, able to share data, communicate and, since recently, collaborate. Sensors are all specific to various physical dimensions (electromagnetic at different wavelengths, optics, infrared, etc.) and different range (few meters to hundreds of kilometers, shallow to wide angles, etc.). Because of this variety, collaboration between sensors allows to deduce new data concerning the environment by overlapping outputs coming from many sensors. In this article, we will study the management of resources onboard Remote Piloted Aircraft Systems (RPAS). Our approach aims to design a suitable architecture to deal with resources, i.e. various sensors in our target application. We adopt the multi-agent paradigm by using an agent-based architecture for the multi-sensor and multi-function system [1], [2]. We will show in this article how the sensors' cooperation and coordination can by ensured by the multi-agent architecture and a temporal scheduling.

The evolution of battlefields due to many factors, including new technologies and conflicts' transformation, leads to emerging needs [3]. These needs directly impact the development of airborne platforms and thus of Multi-Sensor Systems (MSS). On the one hand, new operating conditions imply the use of autonomous platforms with advanced flexibility and multirole capabilities [4]. On the other hand, the technologies' fast evolution together with the cost reduction objective entail industries to develop more reliable and durable systems [5]. Sensors carried by RPAS are now able to perform a large panel of functions such as image acquisition, spectrum analysis, and object tracking [6]. All these sensors play a major role in operation and their optimization has become essential.

From a MSS point of view, the orthogonal constraints cited earlier (i.e. low cost versus high autonomy) lead to look for a new architecture able to enhance the MSS' autonomy and resilience while optimizing the sensors' use [1].

This paper presents new and original contributions: (1) a multi-agent architecture that captures the dynamic of the environment by creating tactical agents that may appear in the mission theater which corresponds to an original agentification of the system; (2) each agent generates a plan of tasks according to the resources he needs, namely the sensors; (3) our scheduler that handles the plans of tasks issued from the agents in order to provide an efficient scheduling of sensors. The coordination of the sensors is then supported by a scheduling mechanism in order to satisfy the requirements of the mission and the platform in a hardly-constrained environment.

Our paper goes on to present a realistic scenario (corresponding to real systems conditions) and shows, through simulations, how the multi-agent system evolves and how our scheduling manages the agents' plans of tasks in a realistic mission theater.

This paper is organized as follows: section 2 presents the re-

lated work and emphasizes the originality of our contributions. Section 3 presents our framework including the multi-agent architecture we propose for the design of the next generation of airborne platforms; section 4 details the scheduling mechanism; section 5 provides our experimental results based on the scenario given by our industrial partner. Finally section 6 concludes this paper and presents our perspectives.

II. Related work

Agent-based online architectures are currently used within many Airports' Air Traffic Controllers (ATC) [7], [8]. These agent ATC architectures demonstrated the advantages brought by agents in term of autonomy. Objectives of ATC are about to control the traffic in geographical areas [9]. This task is usually done by a human operator who can be potentially overburdened depending on area attendance [10]. In this context, agents can be used to follow the location of aircraft in a geographical area and assist/alert the operator along different situations. In ATC, agents are mainly used as secondary operators assisting the main system's user with automatic treatment, discharging the operator from a certain workload. ATCs have many constraints in common with a MSS, especially complex visualization of the field, data overloads, high criticality and low delays. The fundamental difference between ATC and MSS lies in the presence of the sensors. Sensors in this kind of airborne platforms are highly complex instruments, continuously expecting precise requests to work (time, orientation, duration, power, tracking movements, etc.). Furthermore, all requests, treatments and products should be treated in a real-time manner, leading to a highly responsive and predictive sensors' behaviors.

Driving sensors through a multi-agent system was studied before in the context of sensor-mission assignment [11]. In this previous architecture, sensors were agentified and sharing missions which were given by a mission manager. In our system, the MSS is also generating sensors plans by analyzing the data coming from the field and making sensors plans in consequence. This feature leads the MSS to support low-level sensors' requirements as well as high-level autonomy goals simultaneously. From a scheduling point of view, our scheduler manages plans of tasks feasible in a particular time window. Each task is specified by precedence and duration constraints. The plans are weighted by a priority coefficient operationally determined and the industrial need requires to take principally this coefficient in input. In our architecture, the objective is not to balance the use of resources since each task is dedicated to one precise resource but to have all priority plans scheduled at the end of the scheduling process. This approach is quite different from the literature of scheduling which is principally centered on sharing divisible tasks with dynamic priority in order to distribute them on resources in an optimized way.

III. The MSS framework

At first sight, the MSS acts as an interface between the Mission Manager and the sensors' apertures set. The MSS helps to provide high-autonomy features as well as an accurate control of sensors and efficient use of limited available resources (sensors' apertures, power, cooling, computing power, etc.) [12]. To realize this MSS, we will resort to a multi-agent architecture since the agents are suitable to bring the flexibility and the autonomy required by the MSS. The following section will describe our proposed architecture given in Fig. 1b as well as the inputs and outputs of our MSS architecture.

Between high-level decisions and sensor management, agents play an important role in the MSS' architecture.

A. Agent Design

One of the most important contributions of our architecture is the conceptual meaning of the agents. Indeed, in the MSS, agents are not additional software mission managers but virtual instances of the field objects. Agents have a unique objective: collect as much data as possible about a unique field object through the use of sensors in order to fulfill high-level orders. To achieve his goal, an agent will try to select and execute one of the available functions on the NGAP. In practice, a function will rely on a pre-compiled plan of tasks while the local scheduler is in charge of time instantiation (tasks durations, deadlines, etc.). In our framework each task is associated with a resource and sensors are assimilated to material resources.

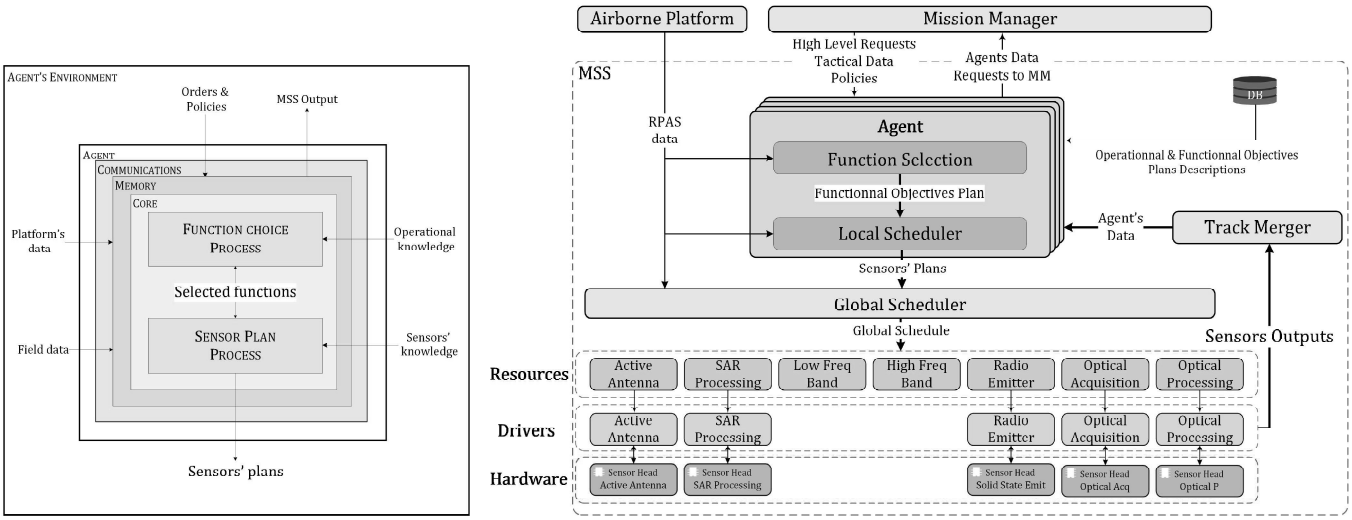
Please refer to the article [1] for more details about agent's design and MSS architecture. The agent's design is shown on fig. 1a. Tactical agents are equipped with communication modules, memory and a core. They have a double role: creating high-level sensors' objectives and generating sensors plans.

B. Generic Tactical Agents

Agents are generic when created, meaning that they are all able to instantiate various available plans of tasks at the system birth. Agents become specialized along the platform flight after receiving data about the corresponding field object. It should be specified that the MSS can detect an object without knowing neither which kinds of object it is nor the object's position. Therefore, it should also be specified that all sensors cannot be used with all kind of object. In case the agent is not specified because of a weak data feeding, available functions for this agent would refer to a very large set of sensors. When knowledge about this object expands, functions become more specific and more precise to this kind of agent.

Each agent is the mirror of an actual object from the field. This approach creates a complete matching between the tactical situation and the agents' group. This connection between the agents and field objects brings many advantages:

- A natural virtual embedded vision of the field with a network of active objects.
- An easy access to behavior analysis and learning functions versus in-field unexpected events.
- A strong modularity of development.
- A high autonomy of the MSS provided by the agents' proactiveness.
- An easy modeling of an open system with objects that appear or disappear dynamically.



(a) Agent's Design

(b) SMS Agent-Based Architecture

Fig. 1: The Agent's Design and Agent-based MSS Architecture.

- A first step for a full decentralized tactical situation architecture.

Representing the tactical situation by agents brought us to call them the *tactical agents*.

From the operational point of view, all field's objects possess a specific degree of interest for sensors. For instance, a highly critical or dangerous object on the field would naturally lead to a proportional use of sensors to gather knowledge about this object. The closed control loop achieves this autonomy objective with the implication of agents. The objects' degree of interest is described by the priority level of the agent. The agent's priority is a score based on operational rules. The plan priority is set when created by an agent and equals the priority of this agent. This means each plan received by the scheduler have a priority linked to the operational interest of real objects.

C. The Multi-Agent System Evolution

At the system's start-up, the set of agents is almost empty. Only a special kind of agent is present in the Multi-Agent System (MAS) and represents the RPAS. This agent has access to functions such as watching (i.e. vigil mode) and objects search. In fact, this agent acts as a bootstrap for the initial set of agents: the RPAS agent plans functions which lead to the discovery of new objects, hence to the creation of new agents. A freshly created agent has a potential depending on the data already collected. During detection of the object and so creation of the agent, enough data can be already seized and completely specify the agent (e.g. if the agent was created after taking a high-resolution picture).

IV. Scheduling

A. MSS Efficiency

The efficiency of the MSS relies on the consistency of achieved tasks according to environment's parameters:

- Events from the field (e.g. weather changes).
- Platform condition (e.g. platform's speed and attitude).
- MSS state (e.g. sensor failure).
- Field objects' behaviors (e.g. object's appearance or attitude changes).
- Operators' instructions (e.g. specific operating policies given by different operators).

The highest efficiency is reached if the MSS collected the maximum volume of significant information about the field regarding all the previous parameters.

To answer these constraints, one solution is to attribute to each agent a priority level. The agent's priority reflects the potential interest of the field object from an operational point of view and hence allows a proportional access to sensors.

The great number of objects present on the field implies a large quantity of sensors' plans created by the agents. Many of these plans can be insignificant from an operational point of view. As an example, we can imagine a scenario in which the platform is tracking an important object on the field through the radar sensor, the importance of the object implies a high level of priority. After sorting by priority order, all the requests will not be achievable by the same sensor. A part would be achieved by another one (e.g. camera sensor) while the other part would be simply unachieved. In spite of a partial realization of agents' requests the resulting efficiency is optimal in the given situation.

The determination of the agents' priority level is an important point of the scheduling consistency.

B. Plan of Tasks

Year after year, the number of functions (e.g. *take a picture* or *listen to signals on M-band*) achievable by an MSS multiplied. Today, sensors allow to realize many different functions. Each function is achieved through a specific plan of tasks.

A task is an indivisible action achieved by a resource. A task can be identified as T_k, of duration D_k and scheduled on the timeline of a resource r_j. The task is starting at t_s and finishing at t_s+D_k. A plan of tasks is an ordered set of tasks to achieve a sensor function (e.g. *Take a picture* requires the use of two resources: a *Optical camera* and an *Optical image processing unit*).

Fig. 2 represents a plan composed of 3 tasks, each of them needing a distinct resource. This figure shows the asynchronous and indivisible features of resources' occupations. In fact, the tasks 1 and 2 are starting and ending at the same time while the task 3 is starting before the previous tasks' end. The resources are fully allocated during the tasks. The plan weight is determined by agents and reflects the importance to execute the plan at an operational level. Plan tasks and tasks directly inherits the agent's priority.

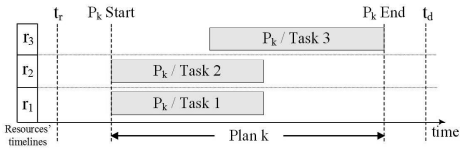


Fig. 2: A plan of tasks P_k on 3 resources.

The plan P_k is defined such that $P_k = \{\gamma, T_r, T_d, C, T\}$ where γ is the plan's priority with $\gamma \in \mathbb{N}$, T_r the release time of the plan, T_d the plan's execution deadline and C the set of constraints which specifies the order of the set of tasks $\mathcal{T}_k = \{T_1, T_2, T_3\}$.

C. Scheduler

The scheduler takes in input the plans issued by the agents and the plans already scheduled on the timelines, their priorities and define a global schedule. After sorting all the plans by priority, the scheduler's algorithm is calculating the start time for each task contained within the plans. The result is a global schedule constituted of interleaved tasks. This scheduling is achieved for a temporal horizon T_H. The plans which were not accepted within the temporal horizon are not scheduled and will be processed later when the average priority of all the plans will be lower. If a plan is not scheduled, the agent is advised about the failure and is able to submit a new plan on less busy resources. The scheduling algorithm is described by the Algorithm 1.

Because our algorithm is scheduling plans one by one and the industrial context requires to schedule the plans of the highest priority in spite of the low-priority plans, the plans of highest priority are scheduled in the first place.

The scheduling starts by sorting all the plans by priority. The algorithm finds the starting date of each task of each plan by checking the compatibility of the task with the tasks within the global schedule. If the task is not compatible, the algorithm shift the task's start-time until the task is compatible. When a start-time is found, the algorithm proceed the same way with the next task of the same plan. If one of the tasks is not compatible, the plan's release time is shifted. If after shifting the plan the release time is higher than the plan's deadline, the plan is abandoned and the owner (i.e. agent) is advised. When all the tasks of the plan are scheduled and compatible, the plan is sent to the global schedule and the next plan of lower priority is entering the scheduler.

Algorithm 1: Scheduling Algorithm

Input: $\mathcal{P}$ the set of the whole plans of tasks
Output: t_s: the start time of each task T_n for each plan P_k
Data: T_H: Temporal horizon

```
1  P ⟵ sortPlansByPriority(P);
2  foreach Plan Pk of P do
3      while ∃Tn unscheduled ∈ Tk ∧ Pk.tr < Pk.td do
4          foreach Task Tn of Tk do
5              Tn.ts ← precedence(Cn);
6              while ¬isCompatible(Tn) ∧ Tn.ts < TH do
7                  if matchingConstraints(Tn) then
8                      nextTaskOfThePlan(Pk);
9                  else
10                     shift(Tn.ts);      /* increment ts */
11             if ¬isCompatible(Tn) then
12                 shift(Pk.tr);          /* increment tr */
13     if ∃Tn unscheduled ∈ Tk then
14         abandon(Pk); adviseAgent("unplanned",ownersOf(Pk));
15     else
16         addToGlobalSchedule(Pk);
```

V. Experimental Results

A. Scenario

Since test in real situations is complex and very expensive to be achieved with this kind of platform and MSS, we implemented this architecture and its environment in simulation. Hence, we developed a special test scenario, able to show the main decisions an operator takes during a mission. This scenario gathers up to 10 steps where the platform is deployed in different contexts with different criticality. Thanks to this scenario we can now evaluate the behavior and the decisions taken by our MSS architecture by simulation in operational situations. Fig. 3 is the visualization of the main window of the simulation engine.

The upper-left graphics show the simulation progression, platform situation and agents population statistics. The downer-left graphics display the real-time tactical situation generated by the simulation. This tactical situation is RPAS subjective and shows only objects and data known by the RPAS. Here, the tactical agent 4 (TA-4) is selected. On the right part of the window, the real-time knowledge of the selected agent is displayed. Each line represents knowledge contained in the agent's memory. An instant picture of this knowledge map shows the precise progression of the agent in the tactical context. In this simulation, knowledge is identified by a *key* and a *value*. On the left part of the column are the keys and on the left the values (e.g. *PRIORITY* is a key, 3 the value).

The keys starting by *TACT* are knowledge specific to tactical agents. On this knowledge map we can see with keys *TACT_GROUND*, *TACT_MOVING* and *TACT_EM_EMITTER* that the TA-4 is representing a static ground object emitting an electromagnetic signal. The *CURRENT_OO* (operational objective) indicates the agent is trying to locate and identify the matching object. The knowledge *CURRENT_TACT_STATE* describes the current state of the tactical finite state machine and shows that the agent is now waiting the end of execution of the accepted plans pointed by the key *ACCEPTED_PLANS*.

The previous states of the visible tactical situation were realized fully automatically and no human manipulations were made. All the necessary data were embedded in the algorithms, plans' descriptions and tactical rules, as for a real platform during flight preparation. The simulation starts by the creation of the RPAS agent, which is a particular derivation of the tactical agent class. The RPAS goal is to build plans able to collect data from the field in order to detect objects and instantiate their matching agents. Once the RPAS created, many watching plans were built and sent to the scheduler. The sensors worked according to the plans and the produced data were sent to the track merger. The track merger, after an analysis of the tactical situation, created 3 agents corresponding to the sensors data. The 3 tactical agents then generated sensors plans in order to complete their tactical knowledge.

After many tasks, an objective is given to the platform: search the object TST (i.e. Time Sensitive Target) in a particular area. After a while and many sensors tasks, the TST was found as expected without human control on the MSS' sensors. Some functions were implemented to enhance the robustness of the MSS including agent death and replication for avoiding blocking agents situations. The MSS' global behavior matched our expectations during simulations and sensors' tasks were correctly scheduled. Work should be done to refine choices' models concerning agents' plans, sensors' behaviors, and objects' behaviors. Though, the modularity of the MSS is improved by this architecture and the agent nature allows to specify architecture characteristics block by block. Concerning the system autonomy, the simulation showed the ability the MSS has for managing high-level objectives depending on its own observations, without other interventions from the operator than specifying policies.

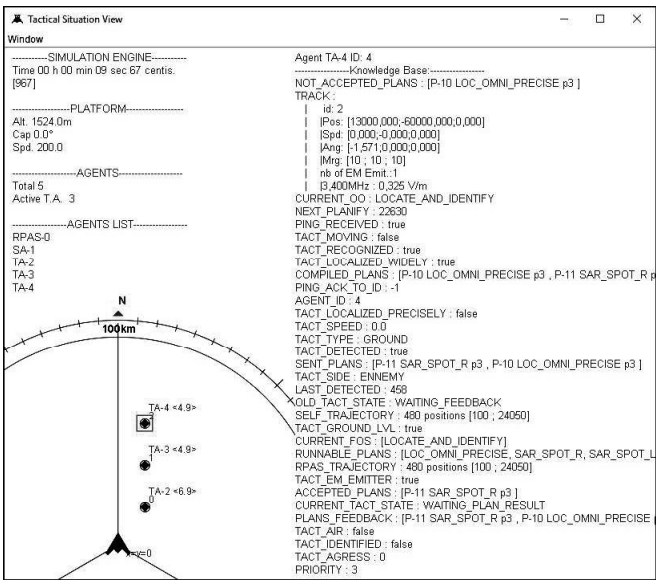


Fig. 3: Visualization of simulator's main frames.

B. Experimentation

In this work we have conducted a set of experiments. The plans to be scheduled came from the scenario described in section V-A. The set of available resources are: ANT_L: Left active antenna, ANT_R: Right active antenna, HF: High-frequency band, LF: Low-frequency band, POW: Power, RDR_IMG: Radar image process unit, OPTRO: Optical camera, OPT_IMG: Optical image process unit, JAM: Jammer and DECOY: Chaffs and flares. Agents have submitted 100 plans to the global scheduler (each agent has a local scheduler that generates pre-compiled plans). As done by our algorithm, the scheduling results are given in fig. 5 and fig 6.

Fig. 5 shows that the number of scheduled plans increases with the size of the temporal horizon. In our simulation, whatever the priority of the plan, its deadline coincides with the temporal horizon. In case the plans are quite temporally constrained then giving more time to the scheduler is not useful to increase the number of scheduled plans. Fig. 5 also shows the scheduling time depends on the temporal horizon. The larger is the horizon, the less reactive is the scheduler. From an operational point of view, a scheduling time above 660ms is not acceptable: the temporal horizon should be under 120s to keep a scheduling time under 100ms . Fig. 6 shows that the plans with the highest priority are scheduled as soon as possible even in a narrow window (temporal horizon). In addition, the average priority of scheduled plans converges to the average priority of all the plans. The operational requirements are met by the scheduling we propose since most of the time the MSS faces situations that need short time scheduling with few plans with high priorities. Fig. 4 shows the experimental output of the scheduler with tasks incoming at different times of the scenario.

In this dynamic instantiation of the scheduler, the global schedule is redefined each time a plan with a priority higher than the lowest priority of the scheduled plans is received from agents. To avoid started plans to be stopped before they finish, they are isolated from the schedule's queue. Started plans are stopped only if higher priority plans cannot successfully be scheduled because of their time window constraints (i.e. release/deadline times).

Conclusion

Our study aims to deal with new scheduling problems in the context of RPAS. We are interested in the scheduling of plans of tasks instead of the classical scheduling of tasks. This implies several differences with existing algorithms. For

instance, removing unfeasible plan of tasks releases a set of resources which strongly impacts the ongoing scheduling. We also have to deal with a flow of requests from the agents. This can be roughly viewed as an online scheduling, but at this stage we have no information about the probabilities of agents requests. From the architecture point of view, our design of multi-agent system allows to consider dynamic and open theater. All the new objects from the field are taken in charge by tactical agents at the runtime. The dynamic of the architecture, its flexibility and the first results of our scheduling mechanism provide promising solution for the next generation of airborne platforms. Indeed, the multifunction and multi-sensor features of this platform are fully exploited by the multi-agent system.

Our perspectives are various, ranging from the improvement of the scheduling by using learning approach in order to anticipate the law of arrival of demands from the agents to the decentralization of the architecture and multi-platform cooperation.

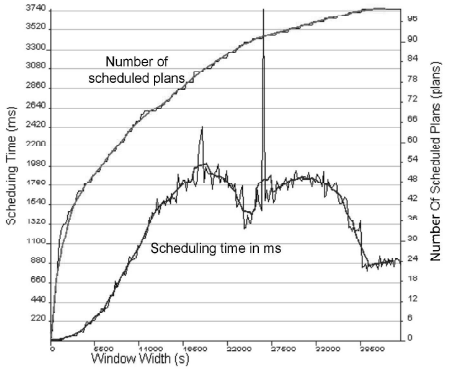


Fig. 5: Number of scheduled plans and scheduling time depending on the temporal horizon.

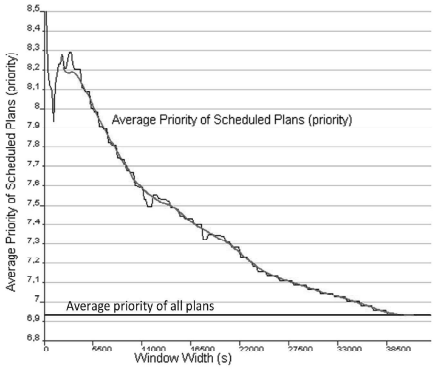


Fig. 6: Average priority of scheduled plans depending on temporal horizon

REFERENCES

[1] L. Grivault, A. El Fallah-Seghrouchni, and R. Girard-Claudon, "Agent-Based Architecture for Multi-sensors System Deployed on Airborne Platform," in *2016 IEEE International Conference on Agents (ICA)*, Sep. 2016, pp. 86–89.

[2] L. Grivault, A. El Fallah Seghrouchni, and R. Girard-Claudon, "Coordination Of Sensors Deployed On Airborne Platform: A Scheduling Approach," in *EUMAS-AT2016*, Valencia, Spain, Dec. 2016.

[3] S. Kemkemian, M. Nouvel, P. Cornic, P. Le Bihan, and P. Garrec, "Radar systems for sense and avoid on UAV," *International Radar Conference*, Oct. 2009.

[4] A. Schulte, D. Donath, and F. Honecker, "Human-system interaction analysis for military pilot activity and mental workload determination," *IEEE International Conference on Systems, Man, and Cybernetics*, 2015.

[5] L. Chabod and E. Chamouard, "Low cost moving target tracking and fire control," *International Radar Conference*, Oct. 2009.

[6] S. Kemkemian and M. Nouvel-Fiani, "Toward common radar & EW multifunction active arrays," in *2010 IEEE International Symposium on Phased Array Systems and Technology*, Oct. 2010, pp. 777–784.

[7] M. Nguyen-Duc, Z. Guessoum, O. Marin, J.-F. Perrot, and J.-P. Briot, "A multi-agent approach to reliable air traffic control," in *2nd International Symposium on Agent Based Modeling and Simulation (ABModSim'08)*, Vienna, Austria, Mar. 2008.

[8] T. J. Callantine, "CATS-based Air Traffic Controller Agents," *San Jose State University*, 2002.

[9] K. Tumer and A. Agogino, "Distributed Agent-Based Air Traffic Flow Management," *The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems - AAMAS*, 2007.

[10] Y. Ibrahim, P. Higgins, and P. Bruce, "Evaluation of a collision avoidance display to support pilots' mental workload in a free flight environment," *IEEE International Conference on Industrial Engineering and Engineering Management*, 2013.

[11] Thao Le, Timothy J. Norman, and Wamberto Vasconcelos, "Agent-based Sensor-Mission Assignment for Tasks Sharing Assets," *IFAAMA*, 2009.

[12] L. Chabod and P. Galaup, "Shared Resources for Airborne Multifunction Sensor Systems," *IET International Conference on Radar Systems*, 2014.

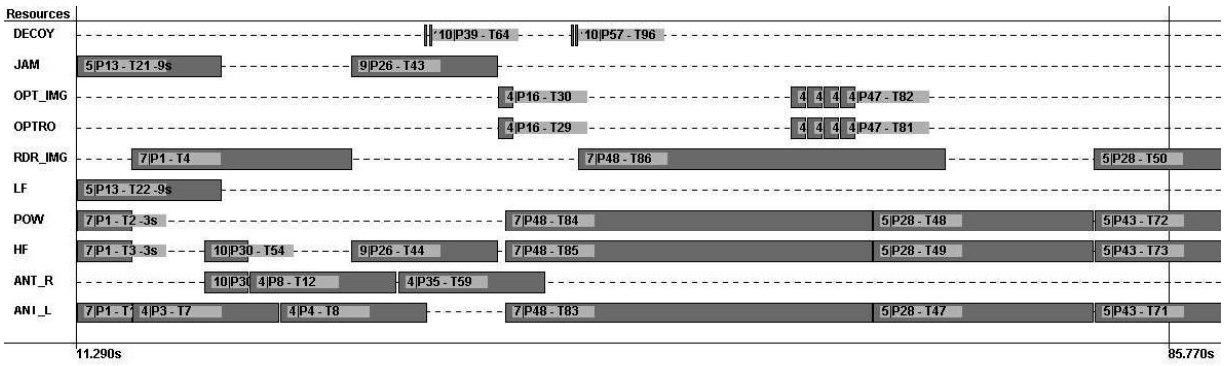


Fig. 4: Example of scheduling of a set of plans on all considered resources of the platform.