A. El Fallah Seghrouchni
(Sorbonne Université)

L. Grivault
(Thales Defense Mission Systems)

E-mail: amal.elfallah@lip6.fr

DOI: 10.12762/2020.AL15-03

Multi-Agent Paradigm to Design the Next Generation of Airborne Platforms

Airborne platforms such as Remote Piloted Aircraft Systems (RPAS) operate 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.
Our main goal is to design a system as a software medium layer between the platform manager and the hardware resources on board 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 a multi-agent paradigm since it is modular by nature and the agents naturally bring autonomy and pro-activity 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 dynamics of the environment by creating agents depending on objects that may appear in the mission theater; (2) agents that generate a task plan (a task is an action that will require a sensor to be achieved) according to the resources (e.g., the sensors) needed; (3) a scheduler that handles the task plans issued by the agents in order to provide efficient sensor scheduling.

The context

Nowadays, airborne platforms are used worldwide for air superiority, as a strategic asset during various 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, multifunction, 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 ranges (few meters to hundreds of kilometers, shallow to wide angles, etc.). Due to this variety, collaboration between sensors allows new data to de deduced concerning the environment by overlapping outputs coming from many sensors.

The sensor scopes and ranges are not limitless, the function set is expanding, and with a maximum of about a thousand entities in the field, the global sensor capacity is the main limit for the enhancement of the MSS.

As a result of sensor limits in terms of range and scopes, the platform's localization is one of the main requisites for sensor efficiency. This requirement implies that the MSS has to be fully aware of the platform trajectory and speed.

Due to the criticality of the context and the mission's objectives, operators are expecting a certain determinism from the decisions proposed

by the MSS. The MSS will be following clearly defined rules specifying sensor actions and which tasks will be accepted by the scheduler.

The evolution of battlefields due to many factors, including new technologies and conflict transformation, leads to emerging needs [4]. These needs directly affect the development of airborne platforms and thus of Multi-Sensor Systems (MSS). On the one hand, new operating conditions entail the use of autonomous platforms with advanced flexibility and multirole capabilities [9]. On the other hand, the rapid evolution of the technologies together with the cost reduction objective are leading industries to develop more reliable and durable systems [2]. Sensors carried by RPAS are now able to perform a large panel of functions, such as image acquisition, spectrum analysis, and object tracking [5]. All of these sensors play a major role in operation and their optimization has become essential.

In this article, we will study the management of resources onboard Remote Piloted Aircraft Systems (RPAS). Our approach is aimed at designing 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. This review presents new and original contributions: (1) a multi-agent architecture that captures the environment dynamics by creating agents watching the mission theater, which corresponds to an original agentification of the system; (2) each agent generates a task plan according to the resources that it needs, namely the sensors; (3) our scheduler, which handles the task plans issued by the agents in order to provide efficient sensor scheduling. 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 and shows, through simulations, how the multi-agent system evolves and how our scheduler manages the agents' task plans in a realistic mission theater.

This paper is organized as follows: Section 2 briefly presents the multi-agent paradigm and related work, and emphasizes the originality of our contributions. Section 3 presents our framework, including the multi-agent architecture that 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.

Related Work

In artificial intelligence, an intelligent agent (IA) refers to an autonomous and goal driven entity called agent that acts in order to achieve goals. An agent is usually embedded in an environment that can perceive through sensors and modify consequent actuators. It may be simple (such as a reactive agent) or complex (such as a BDI agent, or cognitive agent) depending on the modelling requirements. In all cases, intelligent agents may be endowed with skills to achieve their goals. Cognitive agents may use knowledge and intelligent skills, such as methodic, functional, procedural approaches, algorithmic search or reinforcement learning. A multi-agent system (MAS) is composed of an organization of multiple interacting intelligent agents. A multi-agent paradigm can solve problems that are difficult or impossible for an individual agent or a monolithic system to solve [11].

Agent-based online architectures are currently used within the Air Traffic Controllers (ATC) [7,1] of many Airports. These agent ATC architectures demonstrated the advantages brought by agents in terms of autonomy. The objectives of ATC are to control the traffic in geographical areas [10]. This task is usually done by a human operator, who can be potentially overburdened depending on area attendance [3]. In this context, agents can be used to follow the location of aircraft in a geographical area, and assist/alert the operator in various situations.

In ATC, agents are mainly used as secondary operators assisting the main system's user with automatic treatment, freeing the operator from some of the 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, movement tracking, etc.). Furthermore, all requests, treatments and products should be processed in a real-time manner, leading to highly responsive and predictive sensor behaviors.

Driving sensors through a multi-agent system has been studied previously in the context of sensor-mission assignment [6]. In this previous architecture, sensors were *agentified* and shared missions, which were given by a mission manager. In our system, the MSS also generates sensor plans by analyzing the data coming from the field and making sensor plans in consequence. This feature leads the MSS to support low-level sensor requirements, as well as high-level autonomy goals simultaneously.

From a scheduling point of view, our scheduler manages task plans (tasks are complex actions that require resources such as sensors to be achieved) that are feasible within a particular time window. Each task is specified by precedence and duration constraints. The plans are weighted by an operationally determined priority coefficient, and the industrial need requires mainly this coefficient to be taken as input. In our architecture, the objective is not to balance the use of resources, since each task is dedicated to one precise resource, but rather to have all priority plans scheduled at the end of the scheduling process. This approach is quite different from those described in the scheduling literature, which is mainly centered on sharing divisible tasks with dynamic priority, in order to distribute them among resources in an optimized way.

The MSS framework

At first sight, the MSS acts as an interface between the Mission Manager and the sensors' aperture set. The MSS helps to provide high-autonomy features, as well as an accurate control of sensors and efficient use of limited available resources (sensor apertures, power, cooling, computing power, etc.) [2]. To build this MSS, we will resort to a multi-agent architecture since the agents are suitable for bringing the flexibility and the autonomy required by the MSS. The following section will describe our proposed architecture, given in the figure, as well as the inputs and outputs of our MSS architecture.

MSS Architecture description

High-Level Orders and Policies

Policies and high-level orders are the only commands available to the operator for regulating the MSS behavior. They are sent dynamically (according to the changes that may occur during the flight) to the Mission Manager for effective control of the airborne platform.

High-level orders are defined as objectives to be achieved by the mission as a whole, while the policies are defined as a set of rules to be followed by the MSS. Policies impact the behavior of the multi-agent system that implements the MSS, and thus act at various levels. This may imply some restrictions on the sensors, or on the autonomy of the agents' behavior. Indeed, the policies are transmitted to every agent when updated by the mission manager.

Knowledge Base

The knowledge base gathers all knowledge needed by the MSS agents to plan for the use of required sensors. These data specify the generic characteristics of the field objects, the generic agents' behaviors and data acquisition procedures on the high-level decision side. On the low-level side, available knowledge is about sensor delays, scope specifications, speed requirements, running times and all kinds of data required for the evaluation of sensor operations. They are loaded offline to the MSS.

Platform Data

The platform's data communication provides all essential data for high and low level decisions. In fact, the air temperature, platform altitude, speed, position or even weather are required data to plan sensor operations, since they affect sensor operations. Since the current platform position is needed for real-time sensor requests, upcoming positions are also required in order to plan next sensor actions. For this reason, the platform flight plan is available to the MSS. This flight plan is dynamic and can be adapted by the operator during the mission; however, the actual position can differ from it; e.g., in case of an unplanned maneuver.

Global Scheduler

The global scheduler receives all of the plans from the agents, to schedule them accurately on sensor timelines. Due to the number of objects present in the field (*i.e.,* a large number of agents in the architecture), the scheduling is an important process in our system.

Resource Managers

When one of the resource managers receives the plan produced by the scheduler, the resource accurately plays the content of its timeline as specified in the global plan. The manager reads all of the timeline tasks present and, in the case of material resources, sends the corresponding orders to the sensors.

The Track Merger

After the sensors have accomplished their tasks, the results are sent to the track merger. The track merger merges all data coming from sensors and delivering data to the agents. The merging is a complex operation, due to the scattered data recovered from the field. The partial observation of the field leads to a lack of object data continuity.

MSS Outputs

This architecture assists the Mission Manager during decision making, so the purpose of the MSS is to share all of the data gathered by the sensors. The MSS and the Mission Manager are both counterparts exchanging information about the tactical situation (*i.e.,* the theater's body of knowledge). In addition, the Mission Manager is able to control the MSS manually during operation and to bypass the MSS' decisions.

Closed Loop Sensor Control

This four-step process of planning, scheduling, sensing and merging operations constitutes a fast sensor closed control loop.

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

Agent Design

Agents have a unique objective: to collect as much data as possible about field objects through the use of sensors in order to fulfill high-level orders. To achieve its 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 task plan 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.

Agents are equipped with communication modules, memory and a core. They have a double role: creating high-level sensor objectives and generating sensors plans.

To increase the architecture's potential, we consider three classes of resource: a) sensors (e.g., an antenna); b) any type of equipment that can be reserved for the functioning of the sensors (e.g., an image processing unit); and c) any physical magnitude necessary for the proper functioning of the sensors (e.g., frequency).

This allows the agents to make task plans involving sensors, as well as the resources needed by sensors. This classification of resources has proved to be useful for the exclusive use of sensors when they need access to the same non-material or material resource. With this classification, the task dependencies are reduced, and the allocation process is faster due to fewer exchanges between the scheduler and the agents.

In our architecture, all of the resources are considered as artifacts [8]. In this context, an agent has a double role: creating high-level sensor objectives and generating, for a given function, a feasible task plan with an accurate allocation of resources.

Communication features are needed for exchanging data with an agent's environment while the memory feature provides necessary variables for the agent's operations and rational behavior. The core is hosting all running algorithms supported by the two previous features, in order to exchange and to store computed data.

An agent has in its memory a map of all variables standing for real objects, such as speed, altitude, position, attitude and vital signs. This map is empty at the creation of the agent, and is filled over time with collected data, to be aggregated in order to provide knowledge about objects.

Two main processes are at the core of the decision of the agent and determine which, how and when sensor functions should be activated.

The first main process determines which high-level functions (*i.e.,* operational functions) should be achieved in order to acquire data.

For this purpose, groups of algorithms need many inputs, such as platform data, operational knowledge (e.g., rules of actions depending on object characteristics), group orders, policies, agent orders, and specific field object variables.

The second main process considers the previous result and evaluates the precise timing of function execution. Firstly, it considers the sensor knowledge suitable for the function, such as sensor scopes, resource needs, timing constraints, etc.

Then, the global platform context, such as the flight plan, temperatures, weather and all platform variables likely to influence the sensors' work. Finally, all of these data modify the chosen function to a specific sensor plan placed along the platform's trajectory.

Feedback between these two processes is important to prevent the unfeasibility of the high-level function. In fact, the agent's main benefit is autonomy. This benefit is also brought by the adaptability of decisions taken by agents and the ability that it has to propose new solutions if a particular sensor is not available (sensor failure, weather incompatibility or other sensor conflict).

Agents roles

Agents are generic when created, meaning that they are all able to instantiate various available task plans at the system birth. Agents become specialized along the platform flight after receiving field data. It should be specified that the MSS can detect an object without knowing either what kind of object it is or the object's position.

Therefore, it should also be specified that not all sensors can be used with all kinds of object. If the agent is not specified because of a weak data feed, available functions for this agent would refer to a very large set of sensors. The connection between the agents and field objects brings many advantages:

- A natural virtual embedded vision of the field with a network of active objects.
- Easy access to behavior analysis and learning functions when faced with in-field unexpected events.
- Strong modularity of development.
- High autonomy of the MSS provided by the agents' proactiveness.
- Easy modeling of an open system, with objects that appear or disappear dynamically.
- A first step for a fully decentralized tactical situation architecture.

From the operational point of view, all field objects possess a specific degree of interest for sensors. For instance, a highly critical or dangerous object in 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.

Scheduling

MSS Efficiency

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

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

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

The great number of objects present in the field implies a large quantity of sensor 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 in the field through the radar sensor; the importance of the object implies a high level of priority.

For instance, if the platform is approaching a highway used by 300 vehicles with low operational interest, agents will send sensor plans corresponding to an identification procedure.

After sorting by priority order, not all of the requests will be achievable by the same sensor. A part of these would be achieved by another one (e.g., a camera sensor) while the other part would be simply unachieved. Despite a partial realization of agent 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.

Task plan

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

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 starts at t_s and finishes at $t_s + D_k$.

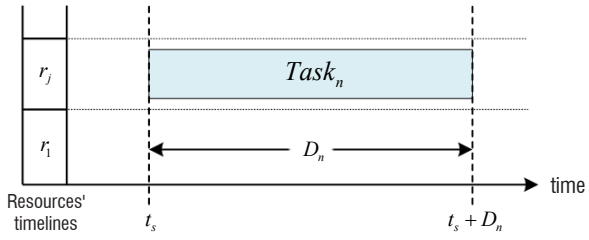


Figure 1 – A task and its parameters

A task plan is an ordered set of tasks to achieve a sensor function (e.g., *Take a picture* requires the use of two resources: An *optical camera* and an *optical image-processing unit*).

Figure 2 represents a plan composed of three tasks, each needing a distinct resource. This figure shows the asynchronous and indivisible features of resource occupations. In fact, Tasks 1 and 2 start and end at the same time, while Task 3 starts before the previous tasks end. The resources are fully allocated during the tasks. The plan weight is determined by agents and reflects the importance of executing the plan at an operational level.

Plan tasks and tasks directly inherit the agent's priority.

Plan P_k is defined such that $P_k = \{\alpha, T_r, T_d, C, T\}$, where α is the plan's priority, T_r is the release time of the plan, T_d is the plan's execution deadline and C is the set of constraints that specifies the order of the set of tasks $T_k = \{T_1, T_2, T_3\}$.

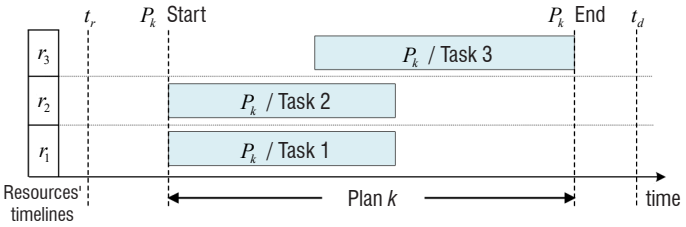


Figure 2 – A task plan involving three resources

Scheduler

The scheduler takes as input the plans issued by the agents and the plans already scheduled on the timelines, as well as their priorities, and defines a global schedule. After sorting all of the plans by priority, the scheduler's algorithm calculates the start time for each task contained in the plans.

The result is a global schedule constituted of interleaved tasks. This scheduling is achieved for a temporal horizon T_H.

The plans that were not accepted within the temporal horizon are not scheduled and will be processed later when the average priority of all of 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.

Given that our algorithm schedules plans one by one, and the industrial context requires the plans with the highest priority to be scheduled despite the low-priority plans, the plans of highest priority are scheduled first.

The scheduling can be based on highest priority first plan insertion through starting date computation, with the possibility of ordering plans if needed.

Experimental Results

Scenario

Since testing 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 that an operator takes during a mission. This scenario gathers up to 10 steps where the platform is deployed in various 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 realistic situations.

Figure 4 shows the visualization of the main window of the simulation engine.

The bottom frame represents functions and resources available in the MSS. Framed resources and functions are currently working and

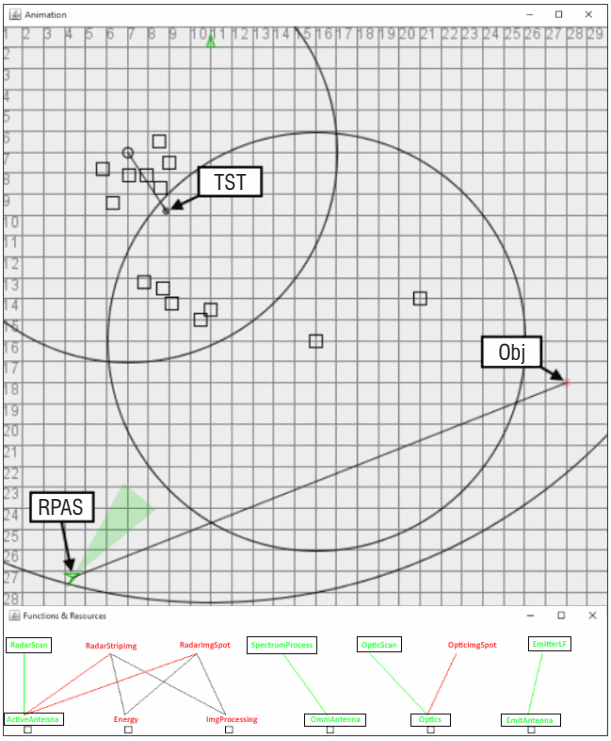


Figure 4 – Visualization of the simulator's main frames

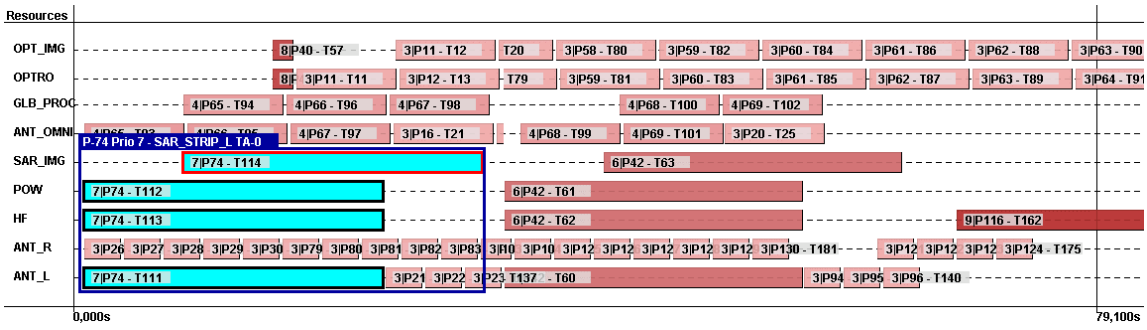


Figure 3 – Resulting global scheduling

unframed ones are not. The links between functions depict the functions' dependencies on the sensors.

At this step of the scenario, the *watch* mode of the RPAS, which was enabled at the power-on of the MSS, has planned and executed the use of an electromagnetic detector. It detected the presence of a radar ("Obj" in the figure), and is heading towards the emitting object to obtain more data about it. As in reality, the MSS is not managing the platform attitude (nor deciding the platform maneuvers or controlling RPAS surfaces), but the Mission Manager is deciding to go toward the object after the MSS shared data received and proposed an identification procedure (proposition emitted by the corresponding agent) at that point. The detection of this object led to the activation of various other sensors. The cone around the RPAS is the visualization of an optical sensor (*i.e.,* camera) turning around the platform. This sensor was also activated by the *watch* function.

After many tasks, an objective is given to the platform: "seek the object TST" (*i.e.,* Time Sensitive Target) in a particular area. After 2 minutes and many achieved tasks, the TST was found as expected without human control over the MSS' sensors.

Some functions were implemented to enhance the robustness of the MSS, including agent death and replication to avoid blocked agent issues by detecting and killing blocked agents and creating a new agent with the data backup from the previous one.

The MSS' global behavior matched our expectations during simulations and sensor tasks were scheduled in time and consistently with regard to the simulated field. The modularity of the MSS is improved by this architecture and the agent nature allows the architecture's characteristics to be specified block by block.

With regard to the system autonomy, the simulation showed the ability that the MSS has for managing high-level objectives depending on its own observations, without any intervention from the operator.

The previous states of the visible tactical situation were accomplished in a fully automated way with no human manipulation. All of the necessary data were embedded in the algorithms, plan descriptions and tactical rules, like for a real platform during flight preparation. The simulation started with the creation of the RPAS agent, which is a particular derivation of the agent class. The RPAS goal is to build plans able to collect data from the field in order to detect objects. Once the RPAS was 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, responsible for agent creation.

After many tasks, an objective was given to the platform: "seek 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 over the MSS' sensors.

Some functions were implemented to enhance the robustness of the MSS, including agent death and replication to avoid agent-blocking situations.

The MSS' global behavior matched our expectations during simulations and sensor tasks were correctly scheduled. Work should be done to refine choice models concerning agent plans, sensor behaviors, and object behaviors. However, the modularity of the MSS is improved by this architecture and the agent nature allows architecture characteristics to be specified block by block.

With regard to the system autonomy, the simulation showed the ability that the MSS has for managing high-level objectives depending on its own observations, without any interventions from the operator other than specifying policies.

Experimentation

Agents 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 Figure 5 and Figure 6.

Figure 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. If the plans are quite temporally constrained, then giving the scheduler more time is not useful to increase the number of scheduled plans. It also shows that the scheduling time depends on the temporal horizon. The larger the horizon is, the less reactive the scheduler is. From an operational point of view, a scheduling time above 660ms is not acceptable: the temporal horizon should be under 120s to keep the scheduling time under 100ms, depending on the mission.

Figure 6 shows that the plans with the highest priority are scheduled as soon as possible, even in a narrow window (temporal horizon).

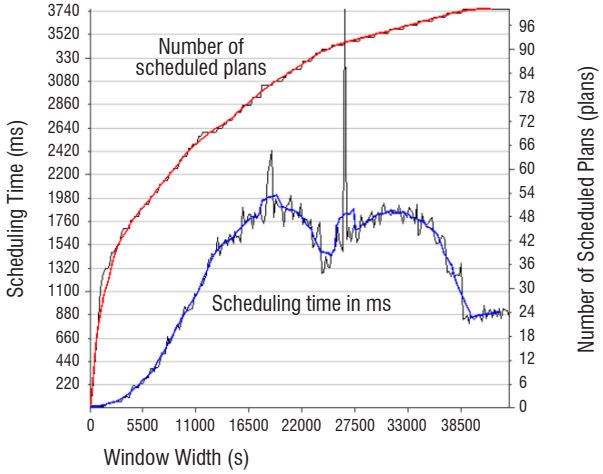


Figure 5 – Number of scheduled plans and scheduling time depending on the temporal horizon.

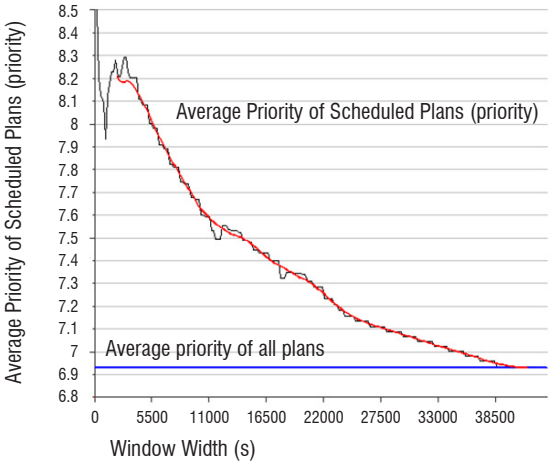


Figure 6 – Average priority of scheduled plans depending on temporal horizon.

In addition, the average priority of the scheduled plans converges to the average priority of all of the plans.

The operational requirements are met by the scheduling that we propose, since most of the time the MSS faces situations that require short-time scheduling with few plans having high priorities.

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 being stopped before they are finished, they are isolated from the schedule 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 is aimed at dealing with new scheduling problems in the context of RPAS. We are interested in the scheduling of task plans instead of the classical scheduling of tasks. This implies several differences with existing algorithms. For instance, removing an unfeasible task plan releases a set of resources, which strongly affects the ongoing scheduling.

We also have to deal with a flow of requests from the agents. This can be roughly viewed as online scheduling, but at this stage we have no information about the probabilities of agent requests.

From the architecture point of view, our design of a multi-agent system allows dynamic and open theaters to be considered. The dynamics of the architecture, its flexibility and the first results of our scheduling mechanism provide a 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.

The closed loop sensor functions and resource allocation control by agents is a first breakthrough concerning agent-based online architectures for strongly constrained systems like MSS.

The results provided by the simulation gave us a first proof of concept concerning the architecture. The general behavior of the simulated MSS, the agents' planning abilities, and the general flexibility of implementation met our expectations.

The various different software blocks are henceforth welcoming innovative algorithms concerning tactical situation forecasting, improved track merging, collaborative agents, refined scheduling, and enhanced communication protocols.

These future algorithm developments will have an important role in the architecture efficiency and sustainability. Scheduling is one of the main algorithms affecting the MSS overall consistency and will be the topic of upcoming research.

A modular software architecture for autonomous and optimized sensor driving has been presented. However, this approach does not propose any hardware architecture. Since the final MSS capabilities and modularity also depend on the supporting hardware and its distribution within the platform, special attention should be paid during its designing task to appreciate all of the features provided by the architecture.

Finally, the architecture may be potentially adapted to less constraining platforms like underwater vehicles, piloted aircraft, or land vehicles.

Perspectives are various, ranging from the improvement of the scheduling by using a 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 ■

References

[1] T. J. CALLANTINE - *CATS-based Air Traffic Controller Agents.* San Jose State University, 2002.

[2] L. CHABOD, P. GALAUP - *Shared Resources for Airborne Multifunction Sensor Systems*. IET International Conference on Radar Systems, 2014.

[3] Y. IBRAHIM, P. HIGGINS, 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.

[4] S. KEMKEMIAN, M. NOUVEL, P. CORNIC, P. LE BIHAN, P. GARREC - *Radar Systems for Sense and Avoid on UAV.* International Radar Conference, October 2009.

[5] S. KEMKEMIAN, M. NOUVEL-FIANI - *Toward Common Radar & EW Multifunction Active Arrays.* IEEE International Symposium on Phased Array Systems and Technology. 77784, 2010.

[6] T. LE, T. J. NORMAN, W. VASCONCELOS - *Agent-Based Sensor-Mission Assignment for Tasks Sharing Assets.* IFAAMA, 2009.

[7] M. NGUYEN-DUC, Z. GUESSOUM, O. MARIN, J.-F. PERROT, J.-P. BRIOT - *A Multi-Agent Approach to Reliable Air Traffic Control*. International Symposium on Agent Based Modeling and Simulation, Vienna, Austria, 2008.

[8] A. OMICINI, A. RICCI, M. VIROLI - *Agens Faber: Toward a Theory of Artefacts for MAS.* Electronic Notes in Theoretical Computer Science 150 (3): 21–36, May 2006.

[9] A. SCHULTE, D. DONATH, F. HONECKER - *Human-System Interaction Analysis for Military Pilot Activity and Mental Workload Determination.* IEEE International Conference on Systems, Man, and Cybernetics, 2015.

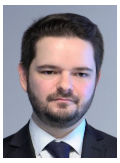
[10] K. TUMER, A. AGOGINO - *Distributed Agent-Based Air Traffic Flow Management.* The Sixth Intl. Joint Conf. on Autonomous Agents and Multi-Agent Systems, AAMAS, 2007.

[11] R. H. BORDINI, A. EL FALLAH SEGHROUCHNI, K. HINDRIKS, B. LOGAN, A. RICCI - *Agent Programming in the Cognitive Era.* JAAMAS (Autonomous Agents and Multi-Agent Systems) volume 34, Article number: 37 (October 2020). https://doi.org/10.1007/s10458-020-09453-y.

Amal El Fallah Seghrouchni (is a Full Professor at Sorbonne University – Faculty of Science and Engineering, and is currently on leave from the CNRS. She is a researcher assigned to the LIP6, and she leads the Multi-Agent Systems Group and co-leads the Artificial Intelligence and Data Science line of research (covering human and machine learning, deep learning, automatic decision-making, intelligent agents and multi-agent systems, etc.). Amal El Fallah Seghrouchni is also a member of the COMEST (World Commission on the Ethics of Scientific Knowledge and Technology) of UNESCO (2020 -2023) and Holder of the Chair of Excellence "Design of hybrid, cognitive and collaborative AI systems" – Sorbonne University / Thales (2020 – 2024).

Her research focuses on Artificial Intelligence, including the Design of Autonomous Systems and Ambient Intelligent Applications. She is developing an approach based on cognitive agents and multi-agent systems. This includes coordination models (negotiation, distributed planning, interaction protocols), and the spatial-temporal design of contextual systems based on adaptation, individual and collective learning, and human-artefact interaction. Her research themes find a large number of applications in the fields of complex system design (e.g., drones of the future), simulation (e.g., urban), contextual applications and smart cities (e.g., intelligent assistants). These applications and topics are supported by a large number of industrial partnerships and international collaborations.

Amal El Fallah Seghrouchni has also held a number of scientific posts in leading international organizations (e.g., General Chair at AAMAS 2020, Track Chair at AAMAS 2019, Track Chair at IJCAI 2019, Chair of the APIA 2019 Program Committee and of the APIA 2020 Program Committee, member elected to the IFAAMAS and EURAMAS boards, etc.). She has also led the research work of more than 30 doctoral students and has published numerous books and articles in the best conferences on Artificial Intelligence and Multi-Agent Systems (for more details, see http: //www-poleia.lip6.fr/elfallah∼/).



Ludovic Grivault is a Doctor Engineer working for Thales Defense Mission Systems – He obtained his PhD in December 2018 under the supervision of Professor Amal El Fallah Seghrouchini at Sorbonne University – Faculty of Science and Engineering. His thesis topic was "Architecture multi-agent pour la conception et l'ordonnancement de systèmes multi-senseur embarqués sur plateformes aéroportées". His research focuses on embedded systems and artificial intelligence, more specifically on complex system architectures, multi-agent technologies, scheduling and airborne sensors.