Integral AI-based planning for management of WSNs in military operations

J. Caballero
Universidad de Alcalá, ISG
Alcalá de Henares, Spain
ORCID: 0000-0003-3099-3640

Olaya Perez-Mon
Universidad de Alcalá, ISG
Alcalá de Henares, Spain
ORCID: 0000-0002-4527-6698

Maria D. R-Moreno
TNO & Universidad de Alcalá
The Hague, The Netherlands
ORCID: 0000-0002-7024-0427

Julio de Oliveira Filho
TNO, IAS
The Hague, The Netherlands
ORCID: 0000-0001-5152-4902

***Abstract*—Military tactical scenarios have been shifting to more often consider combat situations in urban environments. Threats in these environments are generally more dynamic in nature, imposing new requirements on sensors and communications systems that support military operations. Wireless sensor networks (WSNs) with a large number of small and mobile computing nodes became the typical solution. However, WSNs demand additional complexity to dynamically manage their tasks, resource allocation, mobility, power consumption, and communication.**

This paper illustrates the integration of AI techniques into a Battle Management System (BMS) to support military operations in urban environments. The BMS is enhanced with an AI-based planner able to plan tasks, allocate resources, and monitor the WSN operation. The planner takes into consideration energy harvesting capabilities, secure data transfer, and authorization procedures. It generates plans using the information received from the sensors. In case new situations emerge, based on data fusion information, it automatically replans to adapt to the uncertainty in the environment. Finally, it takes into account the coverage between the different components to optimize the communications and better support WSN's operator(s) and their activities.

***Index Terms*—AI planning, Replanning, Goal Reasoning, Data Fusion, WSN, Sensor Data.**

I. Introduction

In recent years, there has been a notable shift in military tactical scenarios with more emphasis on asymmetric urban combat situations than on traditional open-field warfare. That shift creates new requirements for sensing and communication military capabilities to address challenges related to the urban environment. Key challenges include the more unpredictable nature of threats, a larger number of locations where a threat can arise, and the need to continuously monitor extensive areas. The use of large-scale networks of wireless and unattended sensors became the typical solution proposed to address these challenges [6] [5]. The development of such networks has been supported in the Network Enabled Capabilities (NEC) [8] paradigm, introduced by the British military and adopted by NATO members [7]. NEC lays the base for a dynamic, network-centered combat infrastructure.

Numerous studies on Wireless Sensor Networks (WSN) advanced specific design aspects, such as selecting sensor technologies [4] or the semantic interoperability of sensor services [3]. These aspects are predominantly determined during the network's design phase, resulting in a static deployment that lacks adaptability to changes on the battlefield. During contemporary conflict situations, sensors can be highly mobile (e.g. deployed as UxVs' payloads), remotely exchange processing and communication tasks, and harvest energy when possible or necessary. Ongoing research, therefore, aims to develop dynamic and adaptive WSNs that can reconfigure themselves in response to changing conditions [2] [1].

New advances in fields such as Artificial Intelligence (AI), Machine Learning (ML), and Robotics, offer adequate mechanisms to exploit the dynamic aspect of WSNs to their full potential. AI and ML techniques benefit from the large amount of data produced by numerous sensors and are prominently efficient for tasks such as detection and recognition. Besides that, AI-based planners can exploit the dynamic network adaptation to optimize the system's performance in real-time, effectively addressing the unique demands of urban combat scenarios.

This paper illustrates the research and the integration of AI techniques to support the Battlefield Management System (BMS) in urban environments. The work is funded by the European Defense Agency, under the project ***WI**reless sensor **N**etworks for urban **L**ocal **A**reas **S**urveillance* (WINLAS). We introduce an AI-based planner and executor for mission tasks and WSN resource allocation that considers many factors including the dynamic behavior of urban environments, the node's energy harvesting capabilities, the dynamic exchange of sensor and computing capabilities, and communication protocols. In particular, our BMS integrates AI techniques that allow:

- Automated planning: It generates a sequence of actions to achieve specific goals in a dynamic and uncertain environment. It considers various factors such as mission objectives, available resources, sensor coverage requirements, authentication, and potential threats.
- Intelligent Resource Allocation: It assists in dynamically allocating resources in WSNs, including energy, bandwidth, and computational capabilities.
- Monitoring and Failure Recovery: It monitors the plans and goals based on the information received from the different components through data fusion techniques. It ensures the feasibility and assesses the relevance of the

European Defence Agency (EDA)

planned tasks for the mission.

The paper is structured as follows. Section II introduces the main WINLAS components.Then, section III describes MA-LAMA, an AI planner developed to carry out the automatic generation of plans to support military operations. Section IV outlines the techniques for monitoring and failure recovery of the plans. Section V shows some of the scenarios that we are evaluating in WINLAS. Finally, conclusions and future work are outlined in section VI.

II. WINLAS Overview

At the beginning of a military campaign, the operator initiates a mission through the BMS (e.g., red force tracking). Using the Graphical User Interface (GUI), the operator has the ability to select different areas on a map, defining the target regions of interest. This selection triggers a call to INFOSTAR [26], a data fusion and authentication tool, which in turn provides the operator with a comprehensive list of available resources within the WINLAS network. In particular:

Platforms
: We consider squads, drones, and vehicles as platforms. Vehicles and drones can be electrical, fuel-based, or hybrid. Drones can also be a "smart drone" type, which plans tasks to perform an assigned goal without the help of our BMS planner.

Sensors
: In WINLAS there are different types of sensors available such as infrared (IR), cameras, radio monitor, optical, acoustic, seismic, life signs (e.g. Body Network), and Chemical, Biological, Radiological and Nuclear (CBRN).

Functions
: Depending on the mission, there are several functionalities. In our red force tracking mission, there are defined functions such as threat alerting, threat monitoring, etc.

With the list of available components, the operator can now make informed decisions regarding the specific resources to be utilized for the operation. Through the GUI, the operator selects them, indicating the platforms, sensors, and functions that will contribute to the mission. This selection is then transmitted to an AI planner, MA-LAMA (see next section for further details), a powerful computational tool designed to optimize operations.

The AI planner takes into account the selected components, along with the mission objectives and constraints, and generates a detailed plan of action. These actions outline the specific commands that will be executed by the platforms, sensors, and functions involved in the operation.

With the action plan in hand, the operation enters the execution phase if the operator agrees with the resulting plan. At any moment, the operator can accept or reject the plans that are visually presented to the GUI; add or remove any of the platforms, functions, and sensors available in the area of operation; and set new goals or discard previous ones.

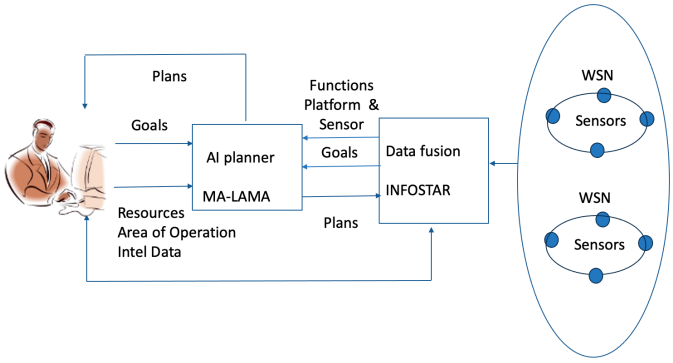

Fig. 1. WINLAS Components

If the plan is accepted, the platforms and sensors start carrying out the assigned tasks. As the operation progresses, the parameters of the platforms and sensors may dynamically change due to environmental factors or unexpected events. These changes serve as crucial feedback for the system. The data fusion and authentication tool (i.e. INFOSTAR) continuously monitors and analyses these changes. If the observed changes significantly deviate from the expected or planned parameters, the tool triggers a need for replanning. The process of replanning involves reassessing the mission goals, incorporating new information, and adapting the action plan to ensure optimal performance in light of the evolving circumstances. This iterative cycle of planning, execution, and potential replanning ensures that the operation remains flexible, adaptive, and responsive to the dynamic nature of the mission.

In Figure 1, we can see the WINLAS components and the information flow among them.

III. Planning Military Operations

Automated planning (AP), or simply planning, is the area of AI that computationally studies the deliberation process of finding a set of actions that achieve a set of goals. This set of actions, also called a plan, is determined by a search process, which is based on constraints, rules, and characteristics defined in a domain, and an initial and final state defined in a problem.

Depending on these characteristics, the different AP branches study different types of problems, such as classical, numerical, temporal, or probabilistic planning [14]. In WINLAS, we study a temporal problem, which considers that actions that are set in the domain have a duration and can define conditions and effects at their start and end.

The deliberative process that is performed to solve the problem and produce a plan carries out a search, following the rules set by the domain. This search is done by a planner, which will make use of different search algorithms, such as greedy and A*, and heuristics, mathematical formulas that will serve to guide and optimize the search through estimations on how close the intermediate steps of the search are to the solution of the problem. Additionally, the quality of a plan can be measured by other numerical variables defined in the

domain, such as the duration of all present actions, or other domain-dependent variables such as the risk, which shall be defined in the problem metric.

In the following subsections, we will review our proposition for the domain and problem that will be used in the WINLAS system, as well as the main characteristics of the utilized planner, MA-LAMA, a temporal planner that makes use of state-of-the-art multi-agent temporal planning techniques to tackle and solve temporal problems, optimizing any set of numeric variables defined in the metric.

A. Domain description

In the frame of AP, the domain definition is represented in the Planning Domain Definition Language (PDDL) [16], a standard planning language that allows to declare domains and problems.

- The domain is described by sets of variable types, logical predicates, numerical functions, and actions. Thus, a domain defines the logic proposition of the world: present objects and agents, the environment where they exist and operate, and the rules under which they can interact. These rules, also called actions, are transformations that can be applied to the current state of the world. They are formed by parameters, objects, and agents in the world that are involved in the world transformation. They also consist of preconditions – logic prepositions that need to be true in the current state in order for the action to be eligible to be applied – and effects, which are the logic prepositions that will change in the world state once the action is applied.
- For the problem, we need to define the instances of the typed variables, the initial state of the world (i.e. the instances of the predicates defined in the domain and the numerical functions of the initial values) as well as the goal state that needs to be achieved following the established rules, and the metrics.

Additionally, since PDDL 2.1 [15], the temporal framework is introduced and defined, including in the actions the concept of duration, which is the time that must pass for the next action to take place. Including this concept in the formulation of an action means that the conditions and effects are affected. The conditions are divided into three categories: (1) preconditions that need to be true before the action starts, (2) preconditions that need to be true before the action ends, and (3) preconditions that need to be true at all times for the entirety of the duration of the action. Effects on the other hand are divided into two types: (1) effects that are applied to the world when the action starts, and (2) effects that are applied to the world when the action ends.

An example of a PDDL2.1 action of this domain is shown in Figure 2. We can see that the action MoveSquad involves a squad and two waypoints, one for the origin and one for the destiny; and a duration that depends on the distance between the two points and the speed at which the squad moves. Then, the pre-conditions are declared (condition field in Figure 2). The first one, that needs to be true at the start of the action, indicates that the squad must be at the origin position. The second one, which needs to be true for the entirety of the action duration, indicates that the path between the two points must be traversable. For the effects (effect field in Figure 2), the first one will be applied at the start of the action, and it defines that the squad is no longer in the origin position. The next two, which will be applied at the end, set that the squad is at the end of the action at its destiny and that the total distance traveled by the squad increases by the distance between the two points.

In order to work inside the WINLAS architecture, our PDDL2.1 domain contains the following definitions. We instantiate several types of acting entities, also called agents: squads, vehicles, and drones, which all can move around a set of positions in an area of a map decided by the operator. We also declare several types of sensors (see previous section for the list of sensors considered). As the agents move through the different map positions, they all can carry, place, activate, and deactivate the sensors, which can take measures once they are placed and are active in a given position, or while they are carried by an agent. The numerical values will define the speed at which the agents move, the distances between positions, and the duration of all actions, using a combination of the distance traveled by the agents and the total duration of the plan as the metric to measure quality. Energy constraints are also taken into consideration, so recharging actions are also planned. Finally, the goals will be a set of measurements being done in defined map positions to provide better coverage.

The main planning challenge in this domain is the optimization of the available resources, i.e. the sensors between all the agents, so that the final solution makes the best possible use of the available resources through the cooperation of the present agents, striving to optimize the defined metric. In the following section, we will describe how this planner works and its main features.

B. LAMA

The planner developed to solve the WINLAS domain is a multi-agent temporal planner built upon the LAMA planner [17]. LAMA was the winner in the sequential satisficing track of the International Planning Competition (IPC) in 2008. It utilizes forward multi-heuristic search to solve classical planning problems, using both a landmark heuristic (i.e. based on propositional formula sets that need to be true in every possible solution that the planner can achieve) and the FF heuristic (i.e. ignore the delete effects in a relaxed planning graph) [27]. Prior to the search phase, LAMA conducts a translation and preprocess phase over the PDDL read domain and problem, as it operates internally under the finite-domain variables paradigm [25] [24], rather than the PDDL defined binary variables one. By searching for two cost-sensitive heuristics, LAMA is able to produce high-quality classical plans through iterative weighted A* searches. However, it does not support metrics and the temporal framework of PDDL 2.1.

```
(:durative-action MoveSquad
 :parameters (?a - squad ?c1 ?c2 - waypoint)
 :duration (= ?duration (/ (distance ?c1 ?c2) (speed ?a)))
 :condition (and (at start (position ?a ?c1))
                 (over all (traversable ?c1 ?c2)))
 :effect (and (at start (not (position ?a ?c1)))
              (at end (position ?a ?c2))
              (at end (increase (agent_dist ?a) (distance ?c1 ?c2))) ))
```

Fig. 2. Move Squad action in PDDL2.1 syntax

C. MA-LAMA

In order to make use of the highly optimized search process of LAMA, we have introduced several changes that allow us to solve temporal multi-agent complex-metric problems. This version is named MA-LAMA.

First, a temporal framework is introduced by modifying the translate, preprocess, and search LAMA phases. We included the 'snap-actions' [19] solution in the translation phase, used in other modern temporal planners such as OPTIC [18]. That permits to create a new init-end pair of actions for each durative one, inheriting the appropriate preconditions and effects and dealing with the duration as a numerical function. The benefit of including the 'snap-actions' solution is that it allows us to later translate the full planning task to the limited-range variables paradigm without additional effort. The only remaining point to make this translation possible is the functional variables encoding, which we will address later in this section. Then, during the search phase, the duration numerical function is used as a precondition for the end-actions to be included in the plan, and as an effect to correctly monitor the global duration of the plan.

Multi-agent techniques have already proven that they are capable of reducing the complexity of problems that involve more than one acting entity (agents) [20]. During the LAMA translate phase, MA-LAMA performs a problem automatic agent decomposition, creating a set of individual domains and problems that can be solved individually. This decomposition is based on the work of Crosby et. al [21], where the origin nodes of the Casual Graph after a level-two cycle removal are used as the root to define the agents, but some changes were introduced in order to make this decomposition reliable in a diverse set of domains.

Once the number of agents has been detected, a goal assignment process is carried out by launching relaxed non-delete searches, one for each detected agent. This process is similar to the goal assignments that are present in planners such as CMAP [22] in the multi-agent planning competition CoDMAP [23]. This allows us to estimate the metric cost of reaching each goal for each agent, and the final goal assignment is the one that minimizes the global metric estimation. The other elements of the domain and problem, such as the metric components, predicates, and actions, are only added to each agent problem if they are needed to achieve the goals they were assigned to. Later, each domain and problem pair will be solved individually, so MA-LAMA will launch as many search phases as the number of agents that were detected in the translation phase.

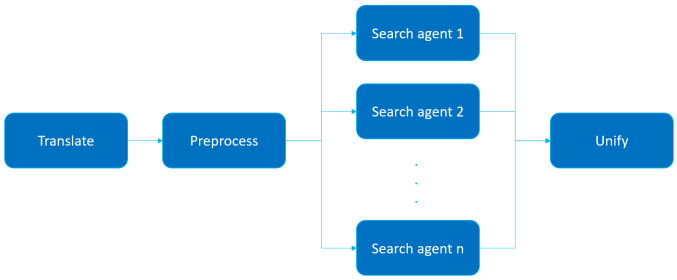

Fig. 3. MA-LAMA architecture overview

In order to deal with more complex metrics, a new numerical framework has been implemented on top of the finite-domain representation. This means that we create finite-domain numerical variables by expanding all the possible values the numerical functions can take, dealing with them as logical variables until the search phase. This allows MA-LAMA to correctly deal with complex metrics, as any weighted combination of domain functions, and numerical conditions for actions.

As MA-LAMA can launch several search phases if more than one agent is detected, a plan repair and unification phase is needed. This phase will analyze each agent's actions and will try to build the final cooperative plan, taking into account the temporal and logical constraints that emerge when dealing with shared domain variables, such as shared resources. Once this phase has finished, the final plan will involve all agent's actions and will accomplish all the goals that were set in the original problem file.

Thus, the final structure of the MA-LAMA planner is the one that can be seen in Figure 3. We can see the Translate and Preprocess phases are launched one time, extracting the necessary information to perform the automatic agent decomposition. Later, once this decomposition has been performed, the Search phase will be launched for each agent, and the Unify phase will recollect each agent's plan, building the final cooperative multi-agent plan and repairing it if it was necessary.

D. Domain and problem generation

Most of the work in the generation of PDDL domains is done by planning experts, mostly manually. This process requires a deep understanding not only of the language itself but also of the specificity of the domain to correctly define actions, the relationships between agents, and the constraints of the problem. Due to the time-consuming and tedious nature of this task, an automated procedure has been developed to generate PDDL domains and problems. It uses the external source of information coming from INFOSTAR from which the necessary data is extracted and transformed into the PDDL format using Python and Jinja2. It extends the work of Gregory [28] that focuses only on problem generation and custom reporting. We also automatically generate the domain description. This allows a quick generation of problems and domain descriptions without requiring expertise in planning. It also facilitates working with large amounts of data when compared to traditional approaches. All this translation process is seamless for the user.

IV. Intelligent Execution of the plans

Planning is the reasoning side of acting. When acting (or executing a process), we need to decide how to perform the chosen activities while reacting to the environment where they are taking place. Each action in the plan can be seen as an abstract task that needs to be refined into subactions or commands that are more concrete.

In real environments, as is the case of WINLAS, we need to examine the situation where an initial plan has been created, but the context in which it is being executed deviates from the expected conditions. This deviation could be caused by a mismatch between the anticipated and the observed values due to unmanageable factors in the environment. Alternatively, it could be a result of changes in the goals of our original plan, requiring us to address new goals while potentially disregarding others.

In the following subsections, we address the techniques used to face these two problems.

A. Failure Detection

We have studied two main approaches for solving the problem of failure detection: *plan repair* and *replanning*.

Plan repair means adapting an existing plan to a new context while perturbing the original plan as little as possible. One common technique is *plan repair* or *plan stability* [12]. It refers to an adaptation strategy that takes into account the differences between 2 plans, based on the number of actions in common. By contrast, *replanning* is the work of generating a new plan from scratch without considering stability [13].

In some of the scenarios analyzed, the new plans had in general very few (or none) goals in common, then, the strategy presented by Fox et al. [12] was less suitable. However, in the future, we want to study more in detail this functionality and add it to our planning system MA-LAMA. Instead, we have followed a similar approach as Cashmore et al. [13], framing the replanning problem as a temporal planning problem with a dynamic initial state. We follow the same classification for the type of actions: non-interruptible and interruptible. The first type represents actions that once they have started, cannot be modified during the execution time. It will either continue executing until it has accomplished the goal or it will fail and cause replanning. An example of a non-interruptible action is the deployment of a sensor, the duration of the action during the execution phase cannot be altered. Instead, in an interruptible action, the duration can be adjusted. An example of an interruptible action is the flying action of a drone. Its duration can be reduced or the end position where the drone has to move can be changed.

In the same way as Cashmore et al. [13] state in the paper, we never start executing a new action from the original plan when replanning has started, we just finish executing the already started executing actions. Then, we take into consideration their invariant conditions and end effects. In this way, we ensure there are no time inconsistencies.

B. Goal reasoning

Goal reasoning, also known as Goal Driven Autonomy (GDA), operates as a supervisory function within mission management. Its primary objective involves monitoring the current goals of a system, ensuring their ongoing feasibility and relevance to the mission at hand, and initiating the establishment of new goals which are passed to the planner. [9].

As a monitoring function [11], it continuously checks for unexpected events or situations. It does not rely on the explicit prediction of the current plan and it may react to unexpected events by establishing new goals.

In WINLAS this role has been delegated to INFOSTAR. Thanks to the data fusion algorithms over the data received from the sensors and platforms, it is able to determine which new goals should be established and which ones should be discarded. Then, the planner generates a new plan having in mind the conditions established for the failures detection module.

V. Experiments

In this section, we present two scenarios to show the capabilities of MA-LAMA and the overall functionality of the WINLAS system.

A. Scenario example

The Operations Area used to carry out the experiments is located within the Campus of the Universidad de Alcala. This scenario has been chosen as an example to preserve the confidentiality of the actual area where the demonstration of WINLAS will be conducted, and because it represents an urban area with similar characteristics to the real demo. Furthermore, for the sake of easier visualization of the plans, a low number of sensors has been used, whereas, in reality, these plans should consist of a large number of sensors, making this capability one of the main demonstration objectives of WINLAS.

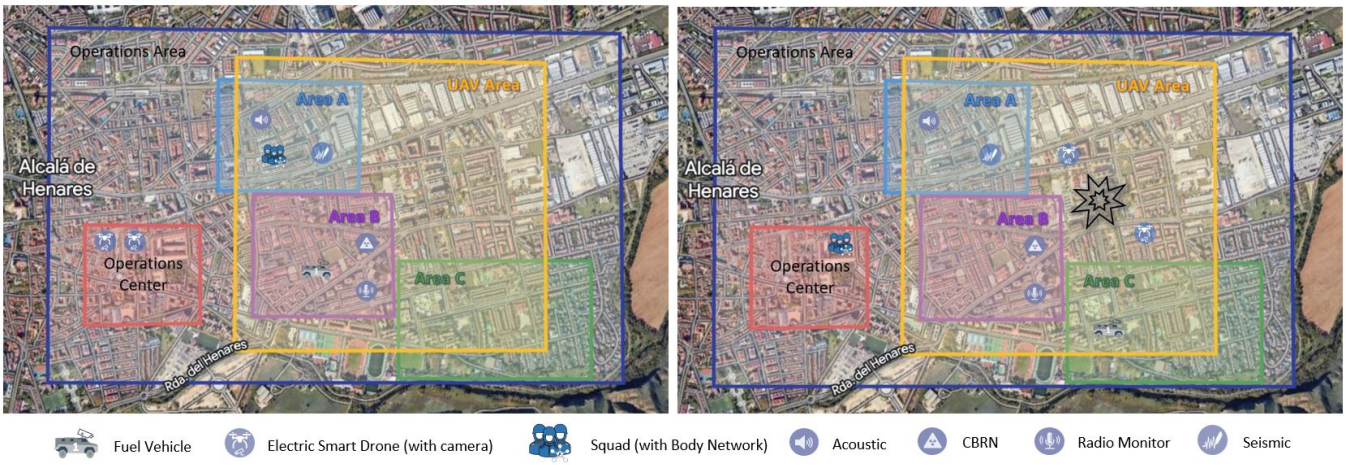

Fig. 4. Military operation plans. a) Deployment of sensors in a nominal situation. b) Detection of an explosion with new goals setting.

Initially, the operator selects the location area to operate which triggers INFOSTAR to return the platforms, sensors, and/or functions that can be used in those areas. At the beginning, all the platforms and sensors available are at the Operations Center (red area of Figures 4a and b). In the first example (see Figure 4a), she establishes as the mission goal the deployment of a network of sensors with the following functionalities: Chemical Threat Detection, Vibration Detection, Radio Monitoring, and Body Monitoring. With this information, the planner is launched, which provides us with the solution shown in Figure 5a). The first step involves switching them on (if they were off), deploying sensors at points where the communication between the different components is optimal, and gathering information about the situation in those areas. Due to the difficult accessibility of Area A (blue area), a squad equipped with a Body Network is decided to be used to place the sensors. This squad will place two sets of sensors: a seismic detector to monitor vibrations in the area and an acoustic sensor to detect shots. Simultaneously, a vehicle is preferable to be used to dispatch the sensors in Area B (purple area) for the CBRN sensor. Additionally, a Radio Monitoring sensor is also deployed to intercept radio communications in a position where all the elements have the best communication possible.

Once all the sensors were deployed, they started acquiring data. According to the information captured, an explosion has occurred (see Figure 4b). Consequently, the INFOSTAR data fusion tool, based on the obtained information, inferences that there will be red forces in op top of area C, and it should be checked. It automatically sets new goals to be achieved: detect people and record the area of the explosion. The area and the resources available (2 Electric Smart Drones equipped with cameras, and the vehicle) are shown to the operator. Once she approves it, these new goals are sent to the planner, generating a new plan (see Figure 5b). The planner decides to send the 2 Electric Smart Drones to the new area to detect any potential individuals present in the explosion area. And, simultaneously, on the ground, the vehicle previously located in Area B moves towards Area C. Its objective is to place a camera in a suitable position to record the area.

The duration of the Move/Fly actions varies depending on the distance between the platform and the point of interest, as well as the speed at which the platform moves. On the other hand, the duration of the different actions associated with the sensors depends on the specification of the manufacturer. This information is registered from the GUI.

B. Comparison to other planners

In order to prove the suitability of MA-LAMA for WINLAS, we have launched several tests and compared the performance of our new planner against other temporal planners. Before explaining the planners we have compared it to, it is important to clarify the reason why we have not compared MA-LAMA against the most obvious choices, being these the participants of the Competition of Distributed and Multi-agent Planners (CoDMAP) [23], and other temporal multi-agent planners. First, although all CodMAP participants are able to deal with multi-agent domains, the competition was performed under a classical planning paradigm, not temporal, which means that they are not able to correctly deal with durative actions and, therefore, with domains such as the ones we are considering here. Additionally, other multi-agent planners that are able to deal with temporal problems, such as TFPOP [29] are not available for free use. This means that our main goal in this test will be to prove that MA-LAMA is necessary so that we obtain the best possible solution measured by two key factors: (1) plan metric value, and (2) cooperativeness degree, which means to make the best possible temporal use of each agent, which can be measured as the total number of actions per agent and in the total plan.

Thus, as MA-LAMA do not need any multi-agent dependent definition in the domain, as would be necessary using MA-PDDL [30], a multi-agent variation of PDDL 3.1, MA-LAMA can be compared against temporal planners that are able to consider plan metrics, such as OPTIC, SGPlan and

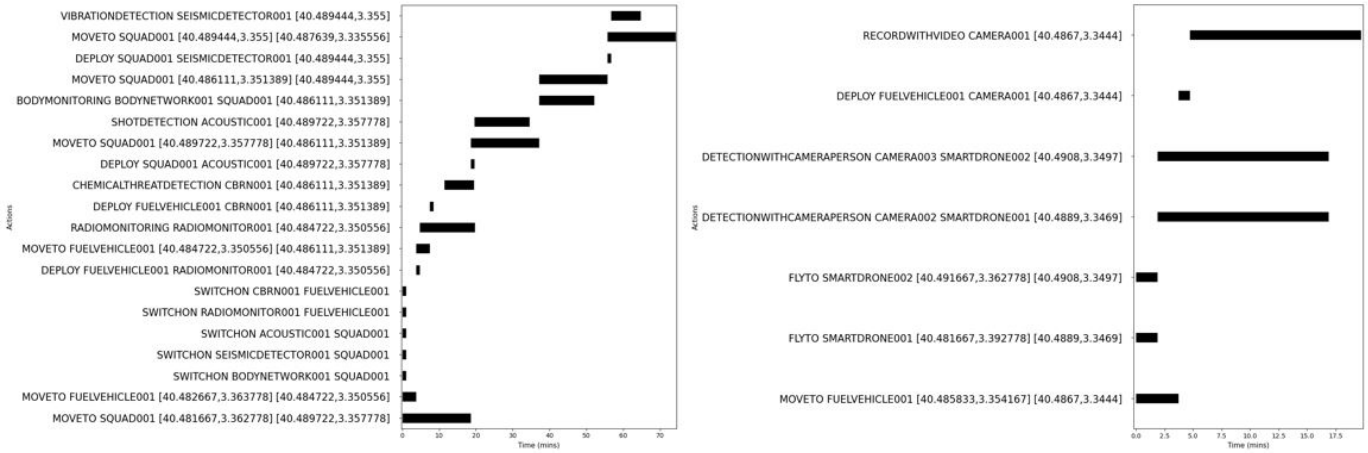

Fig. 5. Plans of the two scenarios considered in Figure 4 generated by MA-LAMA. The duration of the actions is in minutes.

LPG. OPTIC [18] is a temporal planner that is able to deal with plan quality metrics that are not related to the duration of the plan, supporting the use of hard and soft temporal constraints and continuous cost functions. SGPlan [31], whose approach is dividing the full problem into subproblems by finding subgoals, uses multi-valued world representation to produce more efficient heuristics in order to deal with temporal constraints. And, finally, LPG [32], a multi-heuristic planner that produces high-quality plans using partial planning, and that is able to keep track of several parallel steps in the cost function and plan duration. The results for the first problem can be seen in Table I.

TABLE I
Results for the problem of Figure 4a) by each planner. The values taken into account are the number of actions in the final plan, the metric value (i.e. duration) and the search time in seconds.

	MA-LAMA	SGPlan	Optic	LPG
Number of actions	20	21	21	20
Metric	74.269	109.881	92.847	74.269
Search Time (sc)	0.02	0.01	0.06	0.01

What can be seen in the first experiment is that the main complexity of the plan comes from the ability of the planner to deal with simultaneous actions in each and between agents. MA-LAMA demonstrates here that, by reducing the complexity and the amplitude of the search space, it becomes easier to find the best possible action sequence to minimize the metric, producing better solutions. After the agent decomposition is done, the set of actions that the planner needs to consider the possibility to execute each time unit becomes much smaller, and key interactions between simultaneous actions are found faster. LPG and MA-LAMA find the best possible solution. Then, OPTIC and SGPlan find worse solutions in the number of actions and the duration of the plan.

For the second problem, the results can be seen in Table II.

In this case, we can see that the decrease in complexity of the problem helps all planners to find the best possible solution in terms of metric, as the search space is reduced and simultaneity can only happen between agents, not inside each agent plan. Nevertheless, it is important to note that LPG introduces an extra action that does not affect the metric. This would normally not be relevant to mention since it can be still considered that it is the best possible solution, but if we are evaluating the capacity of a planner to deal with multi-agent problems, not introducing extra actions that delay each agent plan, even if it does not affect the overall metric, is not desirable, as in real-life scenarios would mean that the Electric Smart Drone is available for other uses later in time.

TABLE II
Results for the problem of Figure 4b) by each planner. The values taken into account are the number of actions in the final plan, the metric value and the search time in seconds.

	MA-LAMA	SGPlan	Optic	LPG
Number of actions	7	7	7	8
Metric	19.713	19.721	19.715	19.714
Search Time (sc)	0.01	0.00	0.02	0.00

Thus, we can conclude that in these two scenarios, MA-LAMA performs better than the other planners in this multi-agent domain, as it produces the best possible solution in the two cases.

VI. Conclusions and Future Work

In this paper, we have presented an AI-based BMS to support military operations in urban environments. The tool provides several important functionalities in the context of sensor networks and operational activities. It enables the scaling of sensor networks with the capability of self-energy harvesting, ensuring a sustainable and independent power source for the sensors. Additionally, it facilitates secure data transfer and authorization, ensuring the integrity and confidentiality of the transmitted information.

One of the main features of the tool is its ability to generate plans based on the information received from the sensors. By

analyzing the sensor data, it formulates actionable plans and strategies for various operational activities. We have described the AI-planner MA-LAMA, a multi-agent temporal planner that provides optimized solutions for the scenarios. Since MA-LAMA supports PDDL 2.1 language, we can model a wide range of problems regardless of the scenario. In situations where new and unexpected events or circumstances arise, the tool employs data fusion techniques to integrate and analyze information from multiple sensors. This enables it to gain a comprehensive understanding of the environment and generate new goals or discard old ones. By automatically replanning, it ensures that the operations remain effective and responsive to the changing dynamics and uncertainties in the environment.

In the future, we want to test MA-LAMA in more complicated scenarios where charging stations are placed in different positions on the map and vehicles are considered charging stations for the drones. In this way, we will be able to consider other metrics such as energy consumption or risk as the metrics for the plan optimization.

ACKNOWLEDGMENT

This project is funded by the EDA project number B1486IAP4GP.

REFERENCES

[1] L. B. Ruiz, J. M. Nogueira, and A. A. F. Loureiro. Manna: A management architecture for wireless sensor networks. IEEE Communications Magazine, 41(2):116–125, 2003.

[2] C. De Marziani, R. Alcoleas, F. Colombo, N. Costa, F. Pujana, A. Colombo, J. Aparicio, F. J. Alvarez, A. Jimenez, J. Urena, et al. A low cost reconfigurable sensor network for coastal monitoring. In Procs. of the IEEE OCEANS 2011, pp: 1–6, 2011.

[3] B. J. Powers. A multi-agent architecture for nato network enabled capabilities: enabling semantic interoperability in dynamic environments (nc3a rd-2376). In Service-Oriented Computing: Agents, Semantics, and Engineering: International Workshop, SOCASE 2008, Estoril, Portugal, May 12, 2008 Procs., pp: 93–103. Springer, 2008.

[4] M. Winkler, K.-D. Tuchs, K. Hughes, and G. Barclay. Theoretical and practical aspects of military wireless sensor networks. Journal of Telecommunications and Information Technology, pp: 37–45, 2008.

[5] A. Ali, Y. K. Jadoon, S. A. Changazi, and M. Qasim. Military operations: Wireless sensor networks based applications to reinforce future battlefield command system. In 2020 IEEE 23rd International Multitopic Conference (INMIC), pp: 1–6. IEEE, 2020.

[6] M. P. Urisic, Z. Tafa, G. Dimic, and V. Milutinovic. A survey of military applications of Wireless Sensor Networks. In 2012 Mediterranean conference on embedded computing (MECO), pp: 196–199. IEEE, 2012.

[7] C. R. F. Palaganas. Implementing NATO network enabled capability: Implications for nato response force's envisioned roles. Information as Power, 1:175–197, 2007.

[8] D. Ferbrache. Network enabled capability: concepts and delivery. Journal of Defence Science, 8(3):104–107, 2003.

[9] M. Molineaux, M. Klenk, D. W. Aha. Goal-driven autonomy in a navy strategy simulation. In Procs. of the 24th AAAI Conference on Artificial Intelligence, pp: 1548–1554, 2010.

[10] M. Iovino, E. Scukins, J. Styrud, P. Ögren, C. Smith. A survey of Behavior Trees in robotics and AI. Robotics and Autonomous Systems, 154:104096, 2022.

[11] F. Ingrand, M. Ghallab. Deliberation for autonomous robots: A survey. Artificial Intelligence, 247:10-44, 2017. ISSN 0004-3702,

[12] M. Fox, A. Gerevini, D. Long, I. Serina. Plan stability: replanning versus plan repair. Procs. of the 16th International Conference on Automated Planning and Scheduling (ICAPS), 2006.

[13] M. Cashmore, A. Coles, B. Cserna, E. Karpas, D. Magazzeni, W. Ruml. Replanning for Situated Robots. Procs. of the 29th International Conference on Automated Planning and Scheduling (ICAPS), 2019.

[14] A. Cales, A. Coles, M. Martinez, P. Sidiropoulos. International Planning Competition in ICAPS, 2018.

[15] M. Fox, D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Journal of Artificial Intelligence Research, 20:61-124, 2006.

[16] M. Ghallab, C. Knoblock, D. Wilikins, A. Barrett, D. Christianso, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, D. Weld. PDDL - The Planning Domain Definition Language. The 4th International Conference on Artificial Intelligence Planning Systems 1998 (AIPS 98), 1998.

[17] S. Richter, M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research, 39:127-177, 2010.

[18] J. Benton, A. Coles, A. Coles. Temporal planning with preferences and time-dependent continuous costs. Procs. of the 22nd International Conference on Automated Planning and Scheduling (ICAPS), pp: 2-10, 2012.

[19] D. Long, M. Fox. Exploiting a Graphplan Framework in Temporal Planning. Procs. of the 13th International Conference on Automated Planning and Scheduling(ICAPS), pp: 52–61, 2003.

[20] A. Torreño, E. Onaindia, A. Komenda, M. Štolba. Cooperative Multi-Agent Planning: A Survey. ACM Comput. Surv. 50, 6, Article 84:32, 2017.

[21] M. Crosby, M. Rovatsos, R. Petrick. Automated Agent Decomposition for Classical Planning. Procs. of the 23rd International Conference on Automated Planning and Scheduling, pp: 46-54, 2013.

[22] D. Borrajo, S. Fernández. MAPR and CMAP. Procs. of the Competition of Distributed and Multi-Agent Planners (CoDMAP), pp: 46-54, 2015.

[23] M. Štolba, A. Komenda, D. L. Kovaks. Procs. of the Competition of Distributed and Multi-Agent Planners (CoDMAP). International Conference on Automated Planning and Scheduling (ICAPS), 2015.

[24] C .Backstrom, B. Nebel, Complexity results for SAS+ planning. Computational Intelligence, 11(4):625-655, 1995.

[25] M. Helmert. The Fast Downward planning system. Journal of Artificial Intelligence Research, 26:191-246, 2006.

[26] P. de Oude, G. Pavlin and J. P. de Villiers. High-Level Tracking Using Bayesian Context Fusion. 21st International Conference on Information Fusion (FUSION), Cambridge, UK, pp: 1415-1422, 2018.

[27] J. Hoffmann and B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research, 14:253 - 302, 2001.

[28] P. Gregory. PDDL Templating and Custom Reporting: Generating Problems and Processing Plans. Procs. of the 30th ICAPS, Nancy, France, pp: 14-19, 2000.

[29] J. Kvarnström. Planning for loosely coupled agents using partial order forward-chaining. Procs. of the 21st International Conference on Automated Planning and Scheduling (ICAPS), pp: 138–145, 2011.

[30] D. L. Kovacs. Kvarnström. A Multi-Agent Extension of PDDL3.1. 22nd International Conference on Automated Planning and Scheduling (ICAPS), pp: 19-27, 2012.

[31] C. W. Hsu, B. W. Wah, R. Huang, Y. X. Chen. Handling Soft Constraints and Goals Preferences in SGPlan*. Proc. ICAPS Workshop on Preferences and Soft Constraints in Planning, 2006.

[32] A. Gerevini, I. Serina:. LPG: a planner based on local search for planning graphs with action costs. Procs. of the 6th International Conference on Artificial Intelligence Planning Systems (AIPS), 2002.