Adaptive Temporal Planning for Multi-Robot Systems in Operations and Maintenance of Offshore Wind Farms

Ferdian Jovan[1], **Sara Bernardini**[2]

[1] University of Bristol, Bristol, BS1 5DL, UK
[2] Royal Holloway University of London, Egham, TW20 0EX, UK
Ferdian.Jovan@bristol.ac.uk, Sara.Bernardini@rhul.ac.uk

Abstract

With the fast development of offshore wind farms as renewable energy sources, maintaining them efficiently and safely becomes necessary. The high costs of operation and maintenance (O&M) are due to the length of turbine downtime and the logistics for human technician transfer. To reduce such costs, we propose a comprehensive multi-robot system that includes unmanned aerial vehicles (UAV), autonomous surface vessels (ASV), and inspection-and-repair robots (IRR). Our system, which is capable of co-managing the farms with human operators located onshore, brings down costs and significantly reduces the Health and Safety (H&S) risks of O&M by assisting human operators in performing dangerous tasks. In this paper, we focus on using AI temporal planning to coordinate the actions of the different autonomous robots that form the multi-robot system. We devise a new, adaptive planning approach that reduces failures and replanning by performing data-driven goal and domain refinement. Our experiments in both simulated and real-world scenarios prove the effectiveness and robustness of our technique. The success of our system marks the first-step towards a large-scale, multi-robot solution for wind farm O&M.

Introduction

The use of turbines for capturing wind power is considered one of the most promising ways to produce green and clean energy. Combined with the awareness of climate change, the installation of wind energy has been rapidly growing, with the current cumulative wind power capacity reaching 743 GW (GWEC 2021). A large portion of this fast expansion relates to wind turbines located offshore, as they can exploit stronger winds, enjoy larger areas for deployment, and create a minimum conflict of interests with other aspects of society (Bergström et al. 2014). However, because of their remote location, wind turbines are exposed to unpredictable and harsh weather, which results in highly variable operational conditions and, in turn, to intense and costly maintenance operations that can be up to 23% of their total investment cost (Zion Market Research 2019). Such a high cost comes from two factors: i) the length of turbine downtime during maintenance; and ii) the daily use of crew transfer vessels for round trips to and from the farms.

Recent advances in the development of multi-robotic platforms have opened up new opportunities in the O&M automation of offshore assets. Autonomous systems have the potential to both save costs over 20% of the total investment of a windfarm and reduce H&S risks of O&M, especially relating to rope access technicians (Welburn et al. 2019). Some studies show that multi-agent systems tend to achieve better overall performance in their execution (Fernandez-Gonzalez, Karpas, and Williams 2017; Piacentini, Bernardini, and Beck 2019). However, this kind of approach requires solid planning, coordination and execution to perform effectively. A robust implementation of those tools becomes even more challenging in harsh conditions and when the system involves heterogeneous platforms, with each platform having its own role and different sets of functionalities.

AI Planning tools are a promising option for multi-robot systems that rely on coordination and cooperation for achieving shared goals since they allow experts to represent actions and goals at multiple levels of abstraction and enable modularity and hierarchical control across different agents. Yet, current AI Planning tools require domain experts formulating a correct and precise representation of the problem. Since the cost advantage offered by robots over conventional methods lies in performing more tasks in less time, an inaccurate domain representation (e.g. incorrect vehicle's energy consumption or inaccurate task duration), leading to frequent mission failures and replanning, would temper or even nullify the benefit of using autonomous systems.

In this paper, we present the use of AI planning technologies to underpin the autonomous operations of multi-robot systems in offshore wind farms while achieving general robustness and adaptability to domain changes. Our contributions are (1) We offer the first planning formulation of a domain with multiple robots tasked with the Inspection, Maintenance, and Repair (IMR) of offshore wind turbines. (2) We present data-driven adaptive strategies based on statistical models for (i) monitoring energy consumption and action duration, and (ii) correcting the domain specification to avoid divergences between planned and executed behaviour. (3) The proposed data-driven strategies together with the planning formulation have been tested both in realistic simulations and real-world deployments to execute multi-robot coordinated missions.

Related Work

Multi-agent planning is a very active area of research, with several planners having been developed recently (Crosby, Rovatsos, and Petrick 2013; Sreedharan, Zhang, and Kambhampati 2015). These planners apply a distributed problem-solving design to replace the classical single-agent planning paradigm. However, many of them do not support tasks with concurrent and synchronous actions (Torreño et al. 2017), while the others tend to have their own languages and representations to solve specific planning problems (Largouët, Krichen, and Zhao 2016; Nikou et al. 2018).

General-purpose planners that use the de-facto standard planning language PDDL are easier to integrate in complex architectures than specific multi-agent planners with their own languages and representations. We use PDDL2.1 (Fox and Long 2003) to represent planning domains since, in missions involving multiple robots as well as concurrent and coordinated actions, time is crucial. PDDL2.1 includes numerical fluents, concurrency, and exogenous events. Two of the temporal planners for PDDL 2.1, POPF (Coles et al. 2010) and OPTIC (Benton, Coles, and Coles 2012), have been proven to work in multi-agent domains in several real-world scenarios (Tran et al. 2017; Piacentini, Bernardini, and Beck 2019). Piacentini et al. (Piacentini, Bernardini, and Beck 2019) consider multi-agent problems in the context of UAV fleets for search-and-tracking applications. Unfortunately, their work considers only homogeneous robots with similar capabilities and actions, which limits their applicability in contexts with heterogeneous assets.

Unlike the works above, in our application, we deal with different types of robots, each type having a different set of actions and functionalities. Due to the complexity of coordinated actions in heterogeneous robots, there are fewer works in this field. Carreno et al. (Carreno, Petrick, and Petillot 2019; Carreno et al. 2020) adopted a combination of task allocations and temporal planning to tackle multi-vehicle systems involving underwater vehicles and ASVs for subsea oil rigs investigation. They demonstrated that a temporal planner (POPF or OPTIC) is capable of supporting complex actions with multiple robots, especially when the agents must work concurrently and execute actions as part of joint plans. Concurrency is managed by distributing goals over multiple underwater vehicles based on their distance to the actions' locations and robot's capability. Our work is inspired by Carreno et al. (Carreno et al. 2020) with two main differences. While Carreno et al. formulate planning for multi-homogeneous robots with heterogeneous capabilities, we formulate planning for several heterogeneous robots (UAVs, ASVs, and IRRs) with a set of more complex coordinated and concurrent actions. In addition, unlike Carreno et al. (Carreno et al. 2020)'s technique, which centers on optimizing the task allocation, our work focuses on resource and time limitations (e.g., fuel level of the surface vehicle or daily time limit trip), which can restrict the sequences of actions being generated by the planner and increase the total makespan. Our adaptive planning approach not only considers resource and time limitations to create high-quality plans but also improves the makespan's accuracy to better support the actual execution of the plan.

Planning Domain Formulation for O&M Tasks

We now present the PDDL2.1 planning domain, which we formulated based on input from experts and technicians in the offshore energy field. We model the robots' capabilities (e.g. sensors, actuators) as *actions*, the robots' limitations (e.g. battery, fuel level) as *domain constraints* and concurrency as *coordinated actions*. We include 13 different actions in the domain, spread across multiple robot types.

Domain Objects and Instances. Our planning domain[1] contains several object types for the different types of vehicles (`uav, asv, irr`) and `waypoint` for the waypoints. Some waypoints, such as those for the ASV to move between wind turbines, are fixed and reusable for different tasks. Others have properties specific to tasks that can be performed in them. The mission operator specifies the waypoints of interest via a web-based tool, where (s)he can set the exact positions (in GPS coordinates) and the tasks corresponding to each waypoint (see Figure 1a).[2]

Actions. Our planning domain contains durative actions associated with the sensors and actuators available, which enable different O&M tasks on the wind farms. They are:

- `asv_inspect_wt(?asv, ?wp)`: the ASV `?asv` performs a long-ranged visual inspection of a wind turbine at waypoint `?wp`. The action requires that the ASV has a long-ranged camera on it.
- `uav_inspect_blade(?uav, ?wp)`: the UAV `?uav` closely inspects one of the blades of a wind turbine around waypoint `?wp`. A close-up camera is required for the UAV to perform this action. Other preconditions include battery availability and UAV `?uav` being airborne.
- `irr_ndt_inspect(?irr, ?wp)`: the IRR `?irr` performs an NDT inspection on a blade of a wind turbine near waypoint `?wp`. An NDT testing-kit and high battery level are required for the IRR to perform this action.
- `irr_repair_wt(?irr, ?wp)`: the IRR `?irr` performs a crack repair on a blade of a wind turbine near waypoint `?wp`. A repair arm and high battery level are required for the IRR to perform this action.

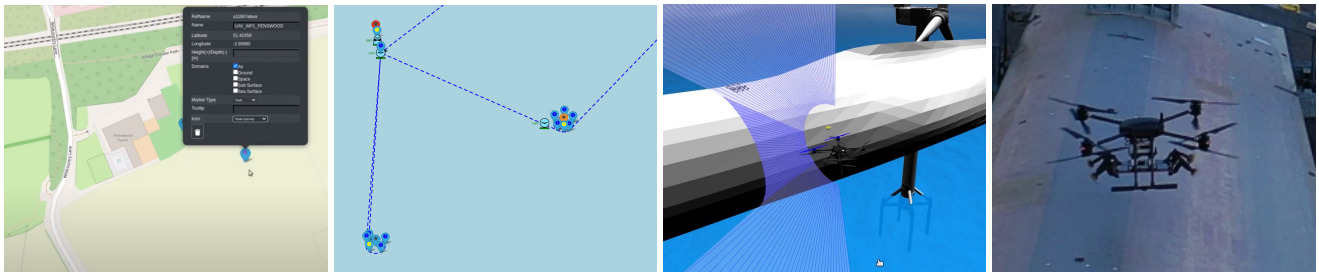
Mission Objectives and Failures. The mission objectives are represented by the following predicates:

- `turbine_inspected_at(?wp)`
- `turbine_nd_tested_at(?wp)`
- `turbine_repaired_at(?wp)`

These are the effects of the actions listed above. These mission objectives (or missions) are pursued via daily trips. Missions correspond to tasks that must be done at different wind turbines within a wind farm (e.g. repair a crack in a blade) specified by the waypoint parameter `?wp`. In one single trip, the multi-robot system starts from a port, visits a wind farm, and goes back to the port at the end of a day. Based on our discussions with wind turbine operators, we define a trip as failed if either (i) one of the robots has insufficient fuel to

[1] https://github.com/ferdianjovan/mimree_executive

[2] The web-based GUI has been developed by our industrial partner (Thales UK); hence, it is outside the scope of this work.



(a) Waypoint and task creation (b) Waypoint configurations (c) Plan execution in simulation (d) Plan execution in real world

Figure 1: A full cycle of the proposed multi-robot system for wind farm O&M. (a) A human operator remotely designs and configures waypoints and tasks for each robot via a user interface. (b) A complete configuration (waypoints and tasks) in an offshore wind farm including all involved robotic assets; a trip is ready to be planned. (c) Plan execution in a Gazebo simulated environment. (d) Plan execution in a real-world setting.

carry out its tasks, or (ii) the system underestimates the time to perform the tasks, which potentially could run over the daily time limit. If a trip fails, replanning occurs, which prioritizes the return of all robotic assets to shore.

Domain Constraints. We model several constraints including time, capability, and resource constraints. Temporal constraints of the type `min_dur(?wp1, ?wp2)` and `max_dur(?wp1, ?wp2)` are used to represent the duration range for each action, with `?wp1` and `?wp2` showing the waypoints where the action is performed. Some actions are associated with specific requirements on sensors or actuators. For each sensor `X` available, a predicate of the form `has_X(?vehicle)` is added to the list of predicates, e.g. `has_deploy_system(?uav)`. Resource constraints consider fuel or battery depletion of each robot over time due to the consumption of these resources for performing activities. This is represented by the function `consumption_rate(?vehicle)`. For UAVs, there is an action `uav_refuelling(?uav, ?asv, ?wp)`, which allows the UAV to recharge its battery at an ASV, assuming the ASV has a charging station available, which is represented by predicate `has_charging_dock(?asv)`. We assume that ASVs and IRRs are manually refueled when they return to shore at the end of their trips.

Coordinated Actions. All coordinated actions require exclusive use of the robot without any interference, e.g. when the UAV deploys or retrieves an IRR, both the UAV and the IRR are not allowed to do anything else. This is obtained by the extra predicate `idle(?vehicle)`, which is added to the precondition of the action definition. Conflicts between actions of different robots can be avoided thanks to the introduction of this predicate. Other requirements are specific to the coordinated actions as follows:

- `uav_deploy_irr(?uav, ?irr, ?uav_wp, ?irr_wp)`: the UAV `?uav` deploys the IRR `?irr` onto a wind turbine blade. The preconditions of this action require that the UAV has a deployment system installed and the IRR is attached to it. The UAV must deploy the IRR on a designated waypoint `?irr_wp`. A successful deployment is conditioned on whether the IRR is fully detached at the end of this action.

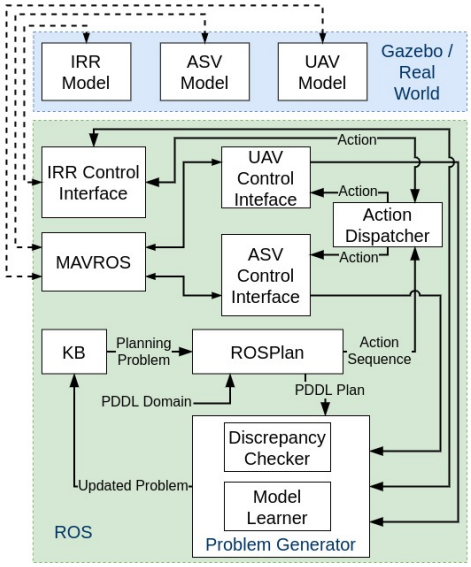


Figure 2: Planning and execution architecture.

- `uav_retrieve_irr(?uav, ?irr, ?uav_wp, ?irr_wp)`: the UAV `?uav` retrieves the IRR `?irr` from a designated waypoint `?uav_wp` on the blade. The preconditions of this action require that the UAV has a retrieval system installed and the IRR is back to the waypoint `?irr_wp` where it was deployed. This requires an extra coordination step from the IRR, which has to move to the waypoint `?irr_wp` via the IRR action `irr_navigate(?irr, ?from, ?to)`.

Adaptive Monitoring Strategy (AMS)

Although automating O&M processes using planning-based multi-agent systems can reduce the costs spent annually, in practice, the efficacy of a planning-based system depends on how accurately the domain experts design the planning domain and problem representation and how tight and precise the time constraints on the actions are.

We keep the planning domain and problem representation

as accurate as possible by implementing an Adaptive Monitoring Strategy (AMS) that reduces the discrepancy between the states expected by the system based on the synthesized plans and the actual states of the system during execution. The AMS does this by continuously updating the *Knowledge Base* (KB), provided by *ROSPlan* (Cashmore et al. 2015), through a *Problem Generator*. In general, a Problem Generator translates tasks and state information into a PDDL problem. Our work enhances the Problem Generator by making it more adaptive to changes. This is obtained by coupling the Problem Generator with a monitoring system. The Problem Generator reasons about: (1) the discrepancy between the actual sensors output and the expected one (*Discrepancy Checker*); and (2) the dynamics of an action duration and fuel consumption rate (*Model Learner*). Figure 2 shows how the modules in our planning system interact. The KB and *Action Dispatcher* are part of ROSPlan, which stores information on the planning problem, solves it by invoking a planner, and distributes the plan to the robots' control systems. The *MAVROS* package is used to control the UAV and ASV and to provide position and velocity controllers. The control interface translates the high-level actions into low-level controls to move the robots.

Discrepancy Checker Through access to sensors and actuators, our approach dynamically monitors any discrepancy between the actual state and the expected state of the system and the environment during plan execution. These discrepancies usually come from sensor / actuator malfunctioning, resulting in failures in the execution of the actions. When a failure occurs, a correction to the discrepancies is made by changing the corresponding predicate values associated to the actions stored in the KB module. Our planning system will then trigger replanning to redeem the action failures that have happened. In this case, however, replanning will not prioritize the robots' return to shore as in the default case. For example, the expected sensor output of the action `asv_inspect_wt(?asv, ?wp)` is that a wind turbine is found and detected, which translates into the goal `turbine_inspected_at(?wp)` being true. However, if the sensor does not detect a wind turbine, the predicate `turbine_inspected_at(?wp)` must be corrected to false at the end of the execution of the action `asv_inspect_wt(?asv, ?wp)`.

A function that refines an action operator, e.g. `asv_inspect_wt(?asv, ?wp)`, into low-level commands resides on each vehicle control interface (shown in Figure 2). Each of these functions communicates its completion of the action to the Problem Generator. A failed output from a function triggers the Problem Generator to remove the corresponding predicates and request a replanning to ROSPlan. A successful output, instead, triggers the Problem Generator to add the predicates of the action's effect to the KB. Other action operators follow similar procedures, the difference resting on the actual low-level control actions[3] needed to execute the high-level action.

[3]The implementation of each low-level control function (e.g. path-planning navigation, turbine visual detection, and arm manipulation) is outside the scope of this paper.

Model Learner In our domain, time and costs are essential, and executing actions within expected duration is necessary. The more accurate the expected duration of each action is, the more missions can be completed in one single trip to the wind farms. Similarly, the fuel rate consumption of each robot, especially the IRR and the ASV, which do not have a refueling option, becomes an important feature in deciding the number of missions that can be performed in one single trip. We estimate and refine the dynamics of the action duration and fuel rate consumption by introducing Bayesian filters that keep track of the expected action duration and the expected fuel rate consumption.

Our Bayesian filters transform an action duration and fuel rate consumption from a single point estimate value into a likelihood distribution. Since both action durations and fuel rate consumptions are continuous values, a Gaussian distribution is appropriate to represent them. A Gaussian distribution, parameterized by a mean μ and a variance σ, is used both for the likelihood distribution and the conjugate prior distribution, as shown in Eq. 1:

$$\mathcal{N}(x \mid \mu, \sigma) \quad = \quad \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{1}$$

A conjugate prior is an algebraic convenience, giving a close-form expression for the posterior distribution. If the likelihood and the prior are normal distributions, the posterior is also a normal distribution. As we are interested in the probability of the action duration (or a fuel rate consumption) x given that the sequence of action durations (or fuel rate consumptions) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ has been observed, we want to obtain a posterior predictive distribution $P(x \mid \mathbf{x})$. This distribution is in the form of a normal distribution $\mathcal{N}(x \mid \mu_0', \sigma_0'^2 + \sigma^2)$ with μ_0' and $\sigma_0'^2$ as indicated in Eq. 2 below:

$$\begin{aligned} \mu_0' &= \frac{1}{\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2}} \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{i=1}^{n} x_i}{\sigma^2} \right) \\ \sigma_0'^2 &= \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2} \right) \end{aligned} \tag{2}$$

with μ_0, σ_0^2 being hyper-parameters for the prior distribution $\mathcal{N}(\mu \mid \mu_0, \sigma_0^2)$ and σ the known variance for the likelihood distribution $\mathcal{N}(x \mid \mu, \sigma^2)$.

Given this learning model, each time a particular action is performed, the duration of the action and the fuel consumption rate are recorded. Then, the predictive posterior distributions are updated by calculating the hyper-parameters as shown in Eq. 2 for each corresponding duration and fuel consumption. The duration is represented by the function `min_dur(?wp1, ?wp2)`, `max_dur(?wp1, ?wp2)` and the fuel consumption by the function `consumption_rate(?vehicle)`. It follows that each pair of waypoints has its own action duration distribution and each vehicle its fuel consumption distribution.

A Bayesian filter can be thought of as a variation of a Kalman filter where the updating step is encapsulated by Bayesian inference, and the prediction step is not treated as a point estimate but rather a distribution.

The advantage is that we can estimate the lower / upper bound of the credible interval that we use to represent the `min_dur(?wp1, ?wp2)`,`max_dur(?wp1, ?wp2)` of actions and `consumption_rate(?vehicle)` of a vehicle. A lower bound credible interval of the posterior predictive distribution $\mathcal{N}(x \mid \mu_0', \sigma_0'^2 + \sigma^2)$ with CDF^{-1} as the inverse of the cumulative density function of a Normal distribution is calculated as:

$$x_{LB} = \int CDF^{-1}(\% = 0.05 \mid \mu_0', \sigma_0'^2 + \sigma^2)\, dx \qquad (3)$$

An upper bound credible interval of the posterior predictive distribution $\mathcal{N}(x \mid \mu_0', \sigma_0'^2 + \sigma^2)$ for action duration and fuel consumption rate respectively are calculated as:

$$x_{UB} = \int CDF^{-1}(\% = 0.95 \mid \mu_0', \sigma_0'^2 + \sigma^2)\, dx \qquad (4)$$

Experimental Results

We now present the metrics that we use to assess the performance of our AMS in a realistic simulation that we developed in Gazebo. Our case study is shown in Figure 1c, which depicts a wind farm consisting of four turbines[4]. We also present tests in real-world scenarios.

AMS Evaluation in Simulation

We test and validate our approach by running the Gazebo simulation with scenarios that might trigger replanning due to insufficient fuel or time. Experimental scenarios simulate daily trips visiting one or more wind turbines to perform up to ten types of inspection or repair missions with a maximum trip time of 150 minutes. A USV, a UAV, and an IRR are used for consistency of the results on each trip scenario. We also set the probability of action failures to 0.2 for those actions that might triggers replanning due to sensor / actuator malfunctions (e.g. actions defined in Section "Domain Formulation for O&M Tasks").

We test four different strategies: (i) POPF; (ii) POPF with Discrepancy Monitoring (DM); (iii) POPF with AMS, which include DM and our Model Learner; (iv) POPF with Decentralized Heterogeneous Robot Task Allocator (DHRTA). DHRTA (Carreno et al. 2020) utilizes a temporal planner in combination with an efficient task allocator to tackle complex actions in multiple heterogeneous robots. For POPF, POPF with DM, and DHRTA, we formulate the planning domain by using the estimated average speed of the vehicles as indicated by the manufacturers as well as inputs elicited from experienced UAV / boat human operators to calculate the estimated duration for each action (`min_dur(?wp1, ?wp2)` and `max_dur(?wp1, ?wp2)`) and the estimated fuel consumption for each vehicle (`consumption_rate(?vehicle)`). On the other hand, for POPF with AMS, the action durations and fuel consumption rates are based on the learned model trained using 30 different trip scenarios with two missions[5]. The similarity between the missions involves the actions that are performed at the different waypoints, which are used to learn the action duration, and the vehicles that are employed in the trips, which are used to learn the fuel rate consumption.

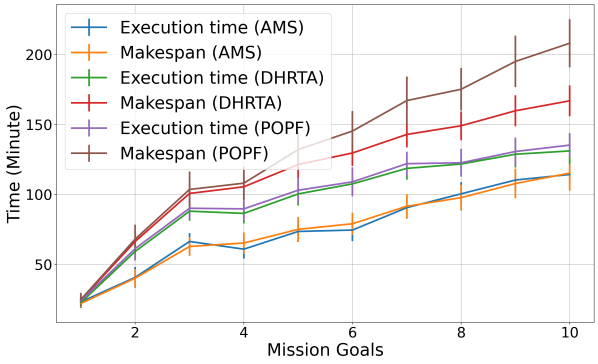


Figure 3: Execution time using plans generated by POPF, POPF with DHRTA and POPF with AMS. The results show that the expected execution time (makespan) for plans generated by POPF with AMS is closer to the actual execution time in simulation.

The performance is evaluated in terms of: (i) total plan generation time, i.e. the sum of plan generation time including replanning; (ii) adjusted makespan, i.e. the total makespan accounting for the makespan created after replanning; and (iii) the number of missions properly completed. Table 1 shows the performance of each strategy averaged over 12 different scenario runs for each number of missions. The table shows that POPF and POPF with DM could not produce plans for trip scenarios with more than six missions. With its efficient task allocator, POPF with DHRTA managed to increase the maximum missions up to eight missions. The main problem comes from overestimating both the action duration at each waypoint and the fuel rate consumption of each vehicle. This causes the expectation that actions take longer to execute and deplete fuel faster than it actually is. On the other hand, POPF with AMS produces plans for all trip scenarios due to its accuracy in modeling the action duration and the fuel rate consumption.

We hypothesized that the trip execution time for plans generated by POPF with AMS plan would be close to the expected execution time (makespan) in simulation. To validate the hypothesis, we ran experiments for the original POPF, POPF with DHRTA, and POPF with AMS by removing the 150-minute time limit per trip so that the original POPF and POPF with DHRTA can generate plans for more missions. Figure 3 shows the results of the experiments, which indicate that our approach closely approximates the execution time in simulation, while the execution time for plans without AMS deviates from the actual execution time more significantly as the number of missions increases.

We also ran an experiment to establish if POPF with AMS is able to reduce the number of replanning due to insufficient fuel. As discussed earlier, one of the causes for major replanning, which forces all robotic assets to go back to shore and abandon all current missions, is an incorrect estimate of

[4]Simulation videos are available at https://www.youtube.com/watch?v=AHdnC0WLK5c&t=11s&ab_channel=FF.

[5]Our experiments suggest that 5 different trips are enough to outperform POPF without AMS.

Missions	POPF			POPF + DM			POPF + AMS			DHRTA		
	time	span	complete	time	span	complete	time	span	complete	time	span	complete
#1	0.21	24.92	0.97	0.29	26.02	1	0.12	22.06	1	0.22	24.45	1
#2	17.95	68.09	1.94	18.50	68.96	2	0.31	40.09	2	9.21	66.38	2
#3	31.36	103.52	2.90	34.75	106.80	3	0.33	62.76	3	18.76	100.67	3
#4	120.21	108.02	3.86	128.34	110.17	4	1.42	65.26	4	32.20	105.43	4
#5	164.14	132.01	4.84	172.78	135.54	5	4.95	75.01	5	45.38	121.33	5
#6	470.76	145.27	5.75	505.11	147.09	6	7.65	79.02	6	76.41	129.60	6
#7	-	-	-	-	-	-	11.86	91.51	7	120.54	142.83	7
#8	-	-	-	-	-	-	33.73	97.66	8	115.72	149.04	8
#9	-	-	-	-	-	-	46.04	107.76	9	-	-	-
#10	-	-	-	-	-	-	61.70	115.22	10	-	-	-

Table 1: Plan generation time (sec), makespan (min), and completed missions (12 different trips per row).

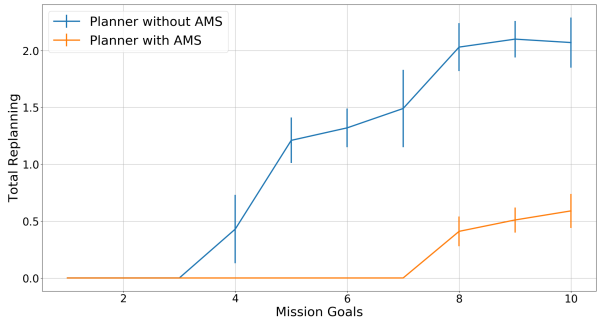


Figure 4: A comparison of the total number of replanning calls between POPF with and without AMS.

fuel consumption rates. Similarly to the previous set of experiments, we removed the 150-minute time limit per trip to allow the original POPF to solve more problems. Figure 4 shows a significant decrease in the number of replanning calls when using the POPF with AMS instead of the original POPF.

Real-World Experiments

Due to the hardware limitation of our IRR prototype and the absence of a real ASV (for budget restrictions), our real-world test consists in an onshore trip (Figure 1d) combining an inspection mission and a deployment and retrieval mission of an IRR to / from a wind blade, both carried out by a UAV. For POPF, we calculate action duration and fuel consumption as in the simulated experiments. For POPF with AMS, the action durations and fuel consumption rates are based on the learned model trained using 15 trips (five trips with a visual inspection mission, five trips with a deployment mission, and five trips with retrieval missions).

Table 2 shows the performance of POPF with AMS compared to the original POPF where the same trip is run twice for each planner. The POPF planning domain overestimates both the duration of each mission and the UAV's fuel consumption rate. As a result, the planner assesses that the trips could not be completed without the UAV returning to the ground to refuel. This decision costs extra actions and time.

POPF				POPF + AMS			
time	span	real	act	time	span	real	act
19.34	26.67	15.73	15	0.2	7.55	8.18	10
19.34	26.67	17.22	15	0.2	8.09	7.96	10

Table 2: Plan generation time (sec), makespan (min), real execution time (min), and total actions to complete three missions on an onshore wind turbine.

Our adaptive planning domain formulation avoids this problem; POPF with AMS accurately estimates the duration of each mission and the fuel consumption rate.

Conclusions

This work looks into the benefits of combining heterogeneous multi-agent systems, temporal planning, and domain refinement to perform IMR tasks on offshore wind turbines. We develop a sophisticated PDDL model and a robust autonomy architecture to underpin the execution of these missions. We also devise a strategy that corrects the domain and problem specifications by monitoring the discrepancies between the expected and the actual states of the system resulting from the execution of the actions over time. A refinement process takes place after every action execution to make the system incrementally more accurate.

We tested our approach in realistic scenarios for the autonomous IMR of offshore wind turbine blades by using a simulated wind farm environment that we developed in Gazebo. The environment includes UAVs, USVs, and IRRs, which are integrated via ROSPlan. Our tests show that the proposed approach is effective in completing more missions for each trip to the wind farms when compared with non-adaptive solutions. We also tested subsets of our multi-robot system in real-world scenarios involving UAVs and IRRs. The results further confirm that our approach is effective in creating valid plans with shorter total makespans.

Acknowledgments

This study has received funding from UKRI through the Innovate UK Grant Agreement No. 104821 and EPSRC Grant EP/R026084/1.

References

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS-2012*.

Bergström, L.; Kautsky, L.; Malm, T.; Rosenberg, R.; Wahlberg, M.; Capetillo, N. Å.; and Wilhelmsson, D. 2014. Effects of offshore wind farms on marine wildlife—a generalized impact assessment. *Environmental Research Letters*, 9(3): 034012.

Carreno, Y.; Pairet, È.; Petillot, Y.; and Petrick, R. P. 2020. A Decentralised Strategy for Heterogeneous AUV Missions via Goal Distribution and Temporal Planning. In *Thirteenth International Conference on Automated Planning and Scheduling*.

Carreno, Y.; Petrick, R. P. A.; and Petillot, Y. 2019. Multi-agent Strategy for Marine Applications via Temporal Planning. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 243–250.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtós, N.; and Carreras, M. 2015. ROSPlan: Planning in the robot operating system. In *Proc. Int. Conf. Automated Planning Scheduling*, 333–341.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *Proc. Int. Conf. Automated Planning Scheduling*. CAN.

Crosby, M.; Rovatsos, M.; and Petrick, R. P. 2013. Automated Agent Decomposition for Classical Planning. In *ICAPS-2013*, 46–54.

Fernandez-Gonzalez, E.; Karpas, E.; and Williams, B. C. 2017. Mixed Discrete-Continuous Planning with Convex Optimization. In *AAAI-2017*, 4574–4580.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. of Artif. Intell. Res.*, 20.

GWEC. 2021. Global Wind Report 2021.

Largouët, C.; Krichen, O.; and Zhao, Y. 2016. Temporal Planning with Extended Timed Automata. In *ICTAI-2016*.

Nikou, A.; Boskos, D.; Tumova, J.; and Dimarogonas, D. V. 2018. On the timed temporal logic planning of coupled multi-agent systems. *Automatica*, 97: 339 – 345.

Piacentini, C.; Bernardini, S.; and Beck, J. C. 2019. Autonomous target search with multiple coordinated UAVs. *Journal of Artificial Intelligence Research*, 65: 519–568.

Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2015. A first multi-agent planner for required cooperation (MARC). *CoDMAP-2015*, 17–20.

Torreño, A.; Onaindia, E.; Komenda, A.; and Štolba, M. 2017. Cooperative Multi-Agent Planning: A Survey. *ACM Comput. Surv.*, 50(6).

Tran, T. T.; Vaquero, T.; Nejat, G.; and Beck, J. C. 2017. Robots in retirement homes: Applying off-the-shelf planning and scheduling to a team of assistive robots. *Journal of Artificial Intelligence Research*, 58: 523–590.

Welburn, E.; Wright, T.; Marsh, C.; Lim, S.; Gupta, A.; Crowther, B.; and Watson, S. 2019. A Mixed Reality Approach to Robotic Inspection of Remote Environments. *Proceedings of the second UK-RAS Conference*, 72–74.

Zion Market Research. 2019. Wind Turbine Operations and Maintenance Market by Application (Offshore and Onshore): Global Industry Perspective, Comprehensive Analysis, and Forecast, 2018–2025. *Zion Market Research Reports*.