An Integrated Framework for Remote Planning

Yaniel Carreno [1,2,3] *, Pierre Le Bras [2] *, Èric Pairet [1,2,3], Paola Ardón [1,2,3], Mike J. Chantler [1,2], Ronald P. A. Petrick [1,2]

[1] Edinburgh Centre for Robotics, Edinburgh, UK
[2] Department of Computer Science, Heriot-Watt University, Edinburgh, UK
[3] School of Informatics, The University of Edinburgh, Edinburgh, UK
{Y.Carreno, P.Le_Bras, Eric.Pairet, Paola.Ardon, M.J.Chantler, R.Petrick}@hw.ac.uk

Abstract

Robotics opens the possibility for safer operations in remote and hazardous environments, with multiple robots deployed to perform tasks that would otherwise present risks for human operators. However, these missions must be carefully planned and monitored to ensure their successful completion while keeping human supervisors in the loop for accountability. While many tools have been developed to tackle individual aspects of such processes, there are few systems combining plan development, review, and supervision in one framework. This paper proposes a mission planning framework designed for remote operations and integrating the following features: a user-friendly problem editor, a task-allocation algorithm, visual plan inspection, digital-twin progression reports, and plan deviation analysis. We show how this system is designed to support non-technical users with planning activities. In particular, the system provides continuous feedback on plan performance, comparing predictions with real implementations, enabling users to improve the requirements for future missions and correct modelling assumptions.

Motivation and Introduction

Robotic platforms provide the potential for safer operating solutions in remote and dangerous environments that would otherwise put human workers at risk (e.g., search and rescue in disaster zones or maintenance of offshore energy platforms). The development of such solutions typically faces two constraints. First, the uncertainty associated with hazardous environments often limits the practicality of deploying fully autonomous systems, e.g., cluttered and unstructured legacy installations, rapid weather changes, etc. As such, for safety and accountability reasons, the oversight of a human supervisor becomes necessary. Second, the remote settings for such missions often necessitate a variety of advanced robotic capabilities that must be highly coordinated to accomplish complex tasks, e.g., inspection and manipulation capabilities in ground, aerial, and subsea domains. As a result, robots are not only expected to coordinate and collaborate with each other, but must also be robust enough to support long-term autonomous operations where direct human interventions are impractical.

This paper proposes a mission planning framework designed for remote operation that integrates the following features: (i) user interfaces to facilitate the development and monitoring of remote missions by human supervisors, and (ii) robust AI planning solutions for heterogeneous multi-robot systems that implement intelligent behaviour in dynamic environments. The benefits of this framework are two-fold: it provides an intuitive end-to-end mission planning and execution system with human end users in mind, and accommodates the capture of offline and online performance data to enable planning experts to enhance model accuracy, system adaptability, and plan optimisation.

Our contribution is a symbiotic framework (see Figure 1) between mission specification, mission planning, and mission monitoring, each aiding the others and leading to a more user-friendly approach for remote planning. None of the individual modules, nor a trivial integration of them all, provides the same functionality. Our work presents newly developed features: (i) the design and development of the Problem Editor interface, fully linked with high-level planning; (ii) the extension of the planner to support offline and online mission planning in favour of enhanced accuracy and on-the-fly plan re-definition; and (iii) the introduction of a Plan Deviation Analysis which supports long-term missions and unexpected changes in the environment. Overall, the proposed framework enables non-expert users to plan missions in complex environments, which, as we believe, constitutes a robust system of interest to the planning community.

The rest of this paper is organised as follows. We first survey previous work related to the development of end-to-end planning frameworks. We then give an overview of the system before expanding on its five major components in detail. We finally conclude with remarks on future development.

Related Work

We begin by reviewing the literature regarding systems that address the problem of planning with human oversight and robust autonomous execution. This survey first considers *human-in-the-loop* systems, then focuses on planning techniques. We then present how these components have been integrated into several ambitious projects over recent years.

Keeping humans involved in automated processes has been recognised as a beneficial approach for optimisation (Scott, Lesh, and Klau 2002), notably for the ability to use up-to-date local knowledge to complement automated models (Fraternali et al. 2012).

*The first two authors have equal contribution.

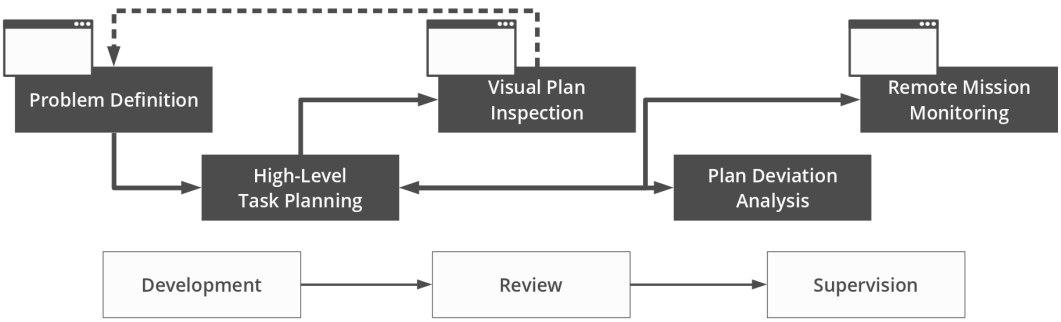

Figure 1: Overview of our planning framework. The approach is designed around 3 user tasks: mission development, plan review, and execution supervision. We integrate 5 components in the framework: problem definition, high-level task planning, visual plan inspection, remote mission monitoring, and plan deviation analysis.

With regard to planning systems, there have been many systems developed to assist in the generation of plans. The most common approach has been the implementation of graphical editing tools for domains and problems using node-link diagrams (Hatzi et al. 2010; Vodrázka and Chrpa 2010). Vrakas and Vlahavas (2003) also include an interface assisting users in defining problems for predefined domains. These techniques have been widely applied within larger planning systems such as EUROPA (Barreiro et al. 2012), GIPO (McCluskey and Simpson 2006), ModPlan (Edelkamp and Mehler 2005), and itSIMPLE (Vaquero et al. 2007). It is also common to provide users with a graphical depiction of generated plans for review. For example, (Kim and Blythe 2003) reuse the node-link diagram representation to describe and provide explanations for plan actions and their components. The PlanCurves technique visualises plans and enables the exploration of interactions between multiple robots (Le Bras et al. 2020).

It is also crucial for users to monitor the progress of plan execution, to check the completion of tasks against the schedule, and provide commands to rectify exceptions. For example, (Bernardini et al. 2020) propose to integrate both planning and monitoring interfaces in their onshore control centre. Relevant work in the area of Explainable AI Planning (XAIP) proposes a framework that utilises the existing planners to assist in answering contrastive questions (Cashmore et al. 2019) showing effectiveness explaining plan solutions for safety-critical domains. To optimise context awareness, such monitoring interfaces must portray agents executing plans within the environment, for example by overlaying their positions on a map (Cummings et al. 2019). The intricacies of offshore installations, however, require the system to render a more accurate depiction of the environment. The ORCA Digital Twin system is an example of a detailed monitoring interface, allowing users to navigate through a 3D simulation while robots are shown to be executing missions (Pairet et al. 2019).

AI temporal planners such as OPTIC (Benton, Coles, and Coles 2012) and POPF (Coles et al. 2010) often lack high-quality task distribution in the generated plans, when planning for multiple robots (Carreno, Petillot, and Petrick 2019). This is the result of their search strategies which focus on satisfying propositional action preconditions first and then action scheduling. Hence, further work has been done to find solutions that improve performance: (Bernardini et al. 2017, 2020), for example, use the POPF-TIF system (Piacentini et al. 2015). This general-purpose planning technology supports required concurrency, metric variables, predictable exogenous events and external advisors. However, this approach does not focus on the optimisation of task allocation for heterogeneous multi-robot systems. The MRGA+TP approach (Carreno et al. 2020) instead favours reasoning about the task allocation problem using the OPTIC planner which enables the introduction of preferences in the planning problem. In this work, we explore the potential of combining task allocation and AI temporal solvers.

In the past decade, AI planning techniques have been combined in *human-in-the-loop* systems to achieve solutions to challenging robotic problems. Examples of such projects include JAMES (Foster et al. 2012), SWARMs (Real-Arce et al. 2016), ORCA (Hastie et al. 2019), and MIMRee, (Bernardini et al. 2020), among others. The JAMES project used AI planning for socially-appropriate interaction using a single robot. Our main target is multi-robot systems. More closely related to our work are the SWARMs and MIMRee projects. The first, focused on coordinating cooperative behaviours in multi-vehicle (underwater) missions. SWARMs explores the areas of task allocation and scheduling, presenting solutions that include genetic algorithms and temporal planning. MIMRee uses AI agent technology to coordinate heterogeneous robotic assets while cooperating with onshore human operators who supervise the mission at a distance, via the use of shared deliberation techniques. While we also consider such goals in our work, our approach differs in a number of significant ways and can be applied to a wider range of applications in extreme environments: our approach is not domain-specific, it incorporates a plan deviation analyser to achieve robustness while executing missions, and we provide a tool to acquire data associated with planning and execution performance to enhance model accuracy, system adaptability, and plan optimisation over time. Our work is being developed in the context of the ORCA project, which considers similar deployment environments to MIMRee, including offshore energy applications.

System Overview

The application domain motivating our work centres around the automation of inspection and basic maintenance tasks on offshore energy installations, including legacy carbon decommissioning and maintenance of renewable resources. In these remote and hazardous environments, the implementation of robotic systems provides safer conditions for the completion of missions.

As such, we designed our framework for remote planning around three major stages of robot deployment:

1. **Development** of the mission specifications, defining goals and available resources to edit problem definitions and generate plans;
2. **Review** of the proposed plan, ensuring its safety and adequacy with the user's up-to-date knowledge and proposing updates to the problem if needed; and
3. **Supervision** of plan execution, assessing plan deviation and guarantees concerning mission success.

Five components[1] are integrated to help users fulfil these tasks (see Figure 1). First, we developed a graphical **problem editor**, allowing users to edit goals and preferences and to select the robots to use for missions based on a model of the domain. We use a **high-level task planning** system to allocate the appropriate robots to goals and generate the mission plan. The plan is then presented using a **visual plan inspector**, enabling users to visually inspect the plan, query details of it if necessary, and approve its execution. Users can then follow the mission progress on a **remote mission monitoring** interface, tracking the robots' movement within the environment in detail. Simultaneously, the **plan deviation analysis** reasons about changes between the scheduled tasks and the robots' progress in real-time, querying the planner for adjustments when needed. We present these components in more detail in the following sections, detailing their inner structure and how they communicate with one another.

Problem Definition

The task of defining the problem is the starting point for any mission. Our framework distinguishes between two problem statements: system predicates, which do not change from one mission to the other (e.g., the distance between waypoints), and user predicates, that users may update for specific missions (e.g., robot starting points). To support both types of statements, our system first uses a domain model that defines the system predicates and the structure of user predicates. A problem editor interface then uses these structures to assist users in defining their statements.

Domain Model: We model three aspects of the domain: the environment (including waypoints, a neighbourhood graph, and objects), the robots (with their capabilities and fluents, e.g., speed or battery level), and finally the structures for goals and preferences. Figure 2 presents examples from our model. While some aspects of the model are domain-specific

[1] In `https://github.com/plebras/PlanVisualisationLive` we present the framework's components linked to the Development and Review stages.

```
Environment Model:
 waypoints:
  {name:wpg0, coordinates:[22,14,0],
   neighbors:[wpg2], type:ground}, ...
 objects:
  {type:valve, position:wpg35}, ...
Robots Model:
 {name:robot0, type:ground, *position:wpg0, *energy:100,
  *available:true, recharge_points:[wgp0], speed:0.5,
  capabilities:[can_turn_valve,...], ...}, ...
Goals and Preferences Model:
 goal types:
  {name:valve_inspected, parameters:[valve]},
  {name:image_captured, parameters:[waypoint]}, ...
 preferences types:
  {name:within, goals:[number,predicate]}, ...
 preferences predicates:
  {name:at, type:state, parameters:[robot,waypoint]},
  {name:energy, type:function, parameters:[robot]}, ...
```

Figure 2: Excerpts from the JSON Domain Model. The environment model defines waypoints and objects. The robot model describes features and capabilities; fields with an * can be edited on the problem editor. The goals and preferences model describes their structures, and allows the editor to assist users when defining such statements (see Figure 3).

(e.g., environment data), the system architecture offers scalability to augment the domain in future development or ensure reusability across different domains. The modularity of this architecture, therefore, allows our work to be deployed onto any platform (real or simulated) and supports different domains to cope with new operational characteristics. In this paper, we present a domain and problems well-aligned with real-world applications, provided by ORCA-Hub's industrial partners. These components have been improved over time considering industrial experiences and necessities. We have used this system architecture to solve problems at different scales, increasing the number of robots and permuting their capabilities.

Problem Editor: We base our problem editor interface on Vrakas and Vlahavas's work (2003), assisting the user in creating three types of statements split in three lists (see Figure 3). The first one determines the set of available robots for the mission (Figure 3B). The user can select which robots to include in the plan and define their starting point and initial charge. The second is the list of goals for the mission (Figure 3C). While initially empty, the editor will use the domain model data to create an interactive form for the user to add goals. Similarly, the user can add constraint preferences to the third list (Figure 3D), providing decisive support for querying a plan with specific characteristics, such as maintaining a minimum charge level or pushing a robot towards a particular path, notably after having reviewed a previous plan that did not meet the user's expectations. Upon sending the user predicates to the back-end for planning, the domain model saves these statements in case the user decides a different plan is needed.

Once instructed to generate a plan via the problem editor in-

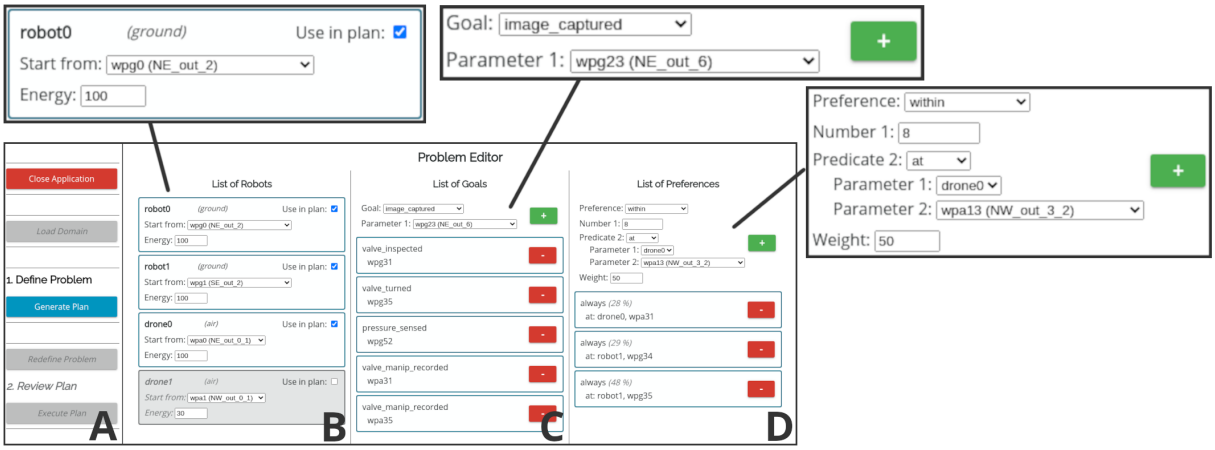

Figure 3: The problem editor interface. The control panel (A) allows users to load a domain and trigger planning and execution. The editor displays 3 lists: robots (B), goals (C), and preferences (D). For the latter two, the interface guides users into adding elements using the domain model data (top of lists); the properties already added are also visible to the user (bottom of the list).

terface, the system performs two actions. First, it saves the user-defined statements for future potential replanning. Second, it compiles the domain model and user-defined statements into a set of problem files, from which the high-level task planning processes can proceed.

High-Level Task Planning

The high-level task planning architecture is responsible for task allocation and task planning, taking into account all available information about system properties (e.g., domain models, physics, etc.) and mission specifications (e.g., goals, constraints, etc.). The system uses a Task Assignment (TA) component to allocate tasks to a set of robots and a Central Temporal Planner (CTP) to generate a plan solution. Here, we describe the key characteristics of these two elements.

Task Assignment: The task assignment component in our strategy is based on the MRGA approach (Carreno et al. 2020). TA is responsible for allocating mission goals to a fleet of multiple heterogeneous robots before planning. This method considers two cost functions to allocate goals: (i) the number of solvable tasks based on the robot capabilities, and (ii) the linear combination of the task makespan, the distance between the points of interest (POIs) and redundancy of the robot's sensory system. TA aims to optimise the distribution of robots in the environment to reduce mission time and avoid worst-case scenarios where all goals are allocated to a single robot. However, robots can implement tasks in different parts of the environment by considering goal capability requirements and robot capabilities (e.g., the ability to inspect a region, manipulate a valve, etc.). We claim this method improves plan solutions presented by benchmark temporal planners. The approach is planner agnostic, with the output of TA described in standard Planning Domain Definition Language with temporal constraints PDDL2.1 (Fox and Long 2003). The high-level task planning approach has been evaluated using a large number of temporal solvers (Carreno et al. 2020) supported by PDDL. Current system evaluations have not considered other planning languages such as RDDL (Sanner 2011). Task allocation distributes tasks by analysing robot and mission characteristics. The approach first evaluates the capabilities of each robot and the capabilities required to implement each goal to allocate the set of solvable tasks to appropriate robots. It then works to define regions where the goals are allocated using clustering methods. The number of designated regions is always equal to the number of robots available. Robots are distributed in the regions by considering the number of tasks they can implement in each cluster and the distance that separates them from the closest goal in each cluster. At the end of this process, each robot will have a goal allocated. The distribution of robots in the environment leads to the remaining tasks being allocated by considering task makespan, the distance between the robots and tasks, and the redundancy of the sensory system to execute critical tasks. Note that the robots are not tied to a single region; they can freely move if required to complete mission tasks.

The final (allocated tasks) set is then transformed into a set of PDDL instances of the fluent `(robot_can_act ?r - robot ?wp - poi)` which is defined in our domain. The PDDL domain constraints the implementation of the different actions in the environment to the appropriate robots that can work at different POIs. The decision of who is capable of executing a particular task depends on the TA reasoning. The set of instances of the fluent is added to the PDDL problem file with all other system specifications and therefore they are considered to generate the plan. Figure 4 (top) shows a set of instances generated by the TA which constraints the execution of tasks in different POIs to the robots that can act in these locations. We use this representation to generate plans using the benchmark planners. In this case, the introduction of the TA means the AI temporal solver does not deal with the task allocation problem directly, which reduces the planning times and improves the final plan solution. However, the flexibility of this system allows the user to decide over the task allocation if that is desirable. In this

```
Problem Instances:
(robot_can_act husky1 wpg52)
(robot_can_act husky0 wpg31)
(robot_can_act uav0 wpa35)
(robot_can_act husky1 wpg35)
(...)
Temporal Plan Solution:
   Time: (Action Name)                              [Duration]
  0.000: (navigation husky1 wpg1 wpg52)              [166.348]
  0.000: (navigation husky0 wpg0 wpg31)              [115.181]
  0.000: (navigation uav0   wpa0 wpa35)              [111.496]
115.182: (valveInsp husky0 camera_h0 wpg31)          [50.000]
166.349: (checkP husky1 p_analyser1 wpg52)           [20.000]
186.350: (navigation husky1 wpg52 wpg35)             [81.687]
268.038: (valveInsp husky1 camera_h1 wpg35)          [50.000]
318.039: (manValve husky1 uav0  wpg35 wpa35)         [30.000]
(...)
```

Figure 4: A fragment of the set of PDDL instances (top) generated by the TA. A temporal plan solution (bottom) for a set of huskies and a UAV in the environment.

case, the near-to-optimal task distribution considering goals and robot fleet characteristics is not a guarantee.

Central Temporal Planner: The planning module is responsible for generating plans that links a robot's actions with the implementation of goals previously assigned to it by the TA component. Missions are created by considering robot capabilities and the characteristics of the environment. This module interacts with other modules (TA and the environment) to obtain a world model that provides information about the robot states, capabilities, and information of the operating environment (e.g., distance between the POIs and map of possible refuelling points, etc.). Such information is used to generate domain and problem descriptions[2] in PDDL. The task planner uses mission knowledge to generate a plan which satisfies the goal allocation restrictions imposed by the TA component. Plans are built using the OPTIC planner which shows good planning performance in a large number of domains, with domain-independent heuristics and fast generation. The quality of the plan is determined by the metrics the user needs to optimise. The most standard is the minimisation of the *makespan*—the time that elapses from the start of plan implementation to the end. However, the OPTIC planner allows considering preferences and time-dependent goal costs.

Figure 4 (bottom) shows a fragment of a plan solution that involves two instances of Husky robot and a UAV (Unmanned Air Vehicle). The user requires the robots to (i) inspect (`valveInsp`), (ii) manipulate a valve (`manValve`) in the environment, and (iii) check the pressure of a boiler (`checkP`) in its digital panel. The UAV implements surveillance tasks to provide visual information to the user. The CTP takes as inputs the PDDL and problem files to generate a solvable plan. The plan's solution takes into account the TA output. For instance, TA evaluates `husky1` is the best robot to implement tasks in `wpg52`. Therefore the planner is restricted to find a solution where the action associated with checking the pressure of the boiler (`checkP`) is executed by `husky1`. The introduction of the High-Level Task Planning is fundamental to achieve a sequence of actions that leads a set of heterogeneous robots from an initial state to a goal state. The planning solution responds to a set of requirements the user presents to the system including goals and constraints (e.g., temporal, resources, etc.). As a result, the High-Level Task Planning provides an executable solution that can be analysed and visualised by the user in order to decide its implementation. In this work, we assume each goal is a single task. However, it does not impede the system to implement coordinated actions. For instance, action `manValve` requires a Husky and a UAV. The TA finds the best robots to execute the goals `valve_manipulated wpg35` (that require a Husky) and `valve_inspected wpa35` (that requires a UAV), which are effects of executing the same action. Using our centralised planning approach we deal with these types of dependencies allowing the robots to coordinate their efforts. In addition, the CPT supports reasoning regarding mission survivability dealing with mission numeric constraints. For instance, the actions associated with the battery recharge are introduced by the planner that keeps robot operation requirements in consideration when planning.

[2] In `https://github.com/YanielCarreno/MRGA` we present Task Allocation algorithm and the domain and problems used in this work (folder ICAPS-IntEx 2021).

Visual Plan Inspection

While the task planning processes will optimise the allocations of robots and ensure the plan safety, human supervisors will still be held accountable for the safe and efficient progress of the mission. Thus, it is necessary to provide them with means to assess the plan generated for them. We, therefore, integrated the visualisation system introduced by (Le Bras et al. 2020). This approach displays plans in three coordinated views (see Figure 5).

Activity Chart: The activity chart follows a common representation for planned tasks: Gantt charts. It displays scheduled tasks as horizontal bars, positioning them to reflect their timing and duration (Figure 5C). In this interface, activities are grouped by robots to highlight their individual roles in the plan. It also connects tasks that are meant to be performed in coordination (e.g. one robot manipulating an object and another robot recording the action). This visualisation is built directly from the plan data (i.e. the list of actions).

Scene Map: While the activity chart displays the planned actions in detail, our domain of application (offshore energy) often includes unstructured legacy installations and involves unpredictable environment factors (e.g. rapid weather changes). It becomes, therefore, necessary for the user to also inspect the planned robots' movements and possible interactions, and assess their safety. To address this issue, the scene map visualisation shows a top-down view of the environment and simulates the position of robots within it, with the robots' elevation shown on the left (Figure 5D). Panning the activity chart or selecting states on the time curve update

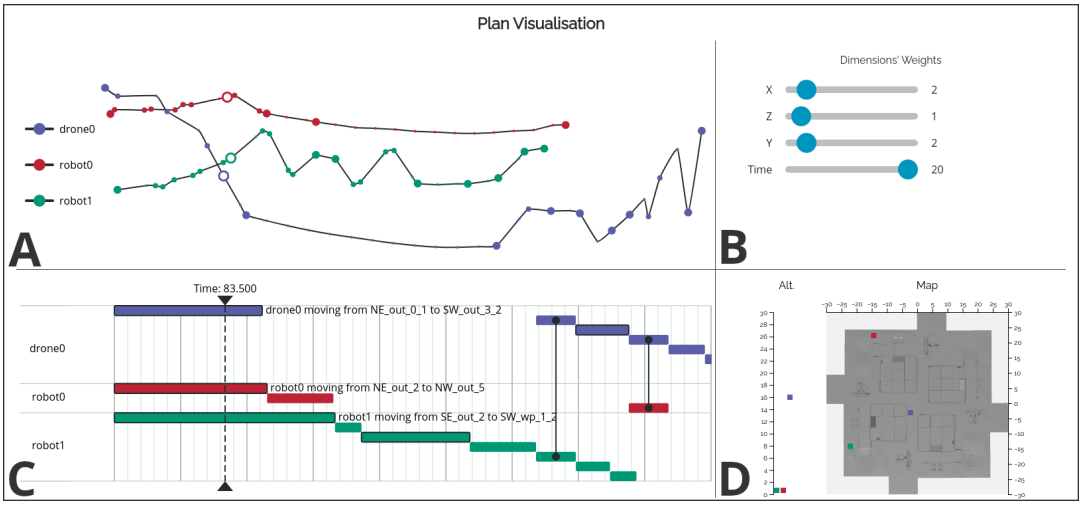

Figure 5: Screenshot of the plan visualisation interface. The plan is shown as: time curves (A) to give an overview of the distances between robots across the plan, sliders (B) that allow the tuning of dimensions' weights to view multi-dimensional interactions from different perspectives, an activity chart (C) for detailed task schedule and robot coordination, and a map view (D) for details on the robots' positions.

the timeframe shown on the scene map, effectively animating it.

Time Curves: The two previous visualisations address the two main requirements for plan inspection: assessing the robots' tasks and movements. On offshore installations, however, there are situations when missions are made in response to emergencies. As such, the plan assessment, notably for movements, needs to be quick. From the list of actions described in the plan, the system will automatically infer the set of robot states throughout the mission, each with spatial and temporal coordinates. To enhance accuracy, the system will also estimate states every 20-time units if such interval is originally missing. The time curve visualisation technique (Bach et al. 2015) proposes to marginalise the multi-dimensional distances between states (three spatial and one temporal) down to two dimensions (Figure 5A). As a result, it creates a set of timelines (one per robot) distorted to reflect the closeness of robots both spatially and temporally. Note that the resulting chart expresses the robots' states in two abstract dimensions: the composites best preserving the original multi-dimensional distances. It is, therefore, impossible to label or interpret the axes of time curves, however, we introduce a posterior manipulation to "correct" the general orientation of curves, from left (start) to right (end).

This representation allows users to get a quick overview of the planned movements for robots and make rapid sense of their potential interactions. The weights of dimensions towards the time curves projection can be controlled using sliders, allowing the user to query details (Figure 5B). If the user decides the plan is not suited for the mission, the system allows them to return to the problem editor interface, where robots, goals, and preferences can be adjusted. Once content with the plan generated for the mission, the user may trigger its execution and monitor mission progress remotely.

Remote Mission Monitoring

We contextualise our work in the implementation of missions in remote and hazardous zones. Plan execution has been mainly evaluated in simulation scenarios using our ORCA-Hub simulator (Pairet et al. 2019). Our simulator is a ROS-enabled offshore energy platform environment composed of four gas and oil tower sites. Figure 6 shows a top (left) and field (right) view of the environment. The simulator supports the simultaneous deployment of multiple instances of robotic platforms, thus enabling a wide range of capabilities for cooperative inspection of large areas and emergency response. In this work, we employ UAV (ideal for aerial inspections) and Husky robots (medium-size robot with large payload capabilities, capable for example of extinguishing a fire) to implement missions that require totally decoupled as well as coordinated tasks. Figure 6 (right) shows a set of robots executing the plan presented in Figure 4 which involves tasks that require different sets of capabilities.

The simulator allows the user to evaluate the performance of the multi-robot system when executing the mission plan. This system provides a semantic description of the offshore energy structure, i.e., a map from 3D coordinates to high-level labels, which bridges the human-robot communication gap. Moreover, to ease some of the inherent robotic challenges, the simulator provides a semantic road map for autonomous point-to-point navigation and collision-free planning. Figure 6 (left) shows a representation of these points in the ground floor and the possible direction of navigation that robots can take. The POIs defined in our PDDL problem are directly aligned with the set of points in the road map. For instance, the high-level POI `wpg1` is defined by:

$$[\mathtt{x},\ \mathtt{y},\ \phi,\ \mathtt{roadmap\_poi\_name},\ \mathtt{poi\_property}],$$

where `x`, `y` and ϕ is the robot position and heading,

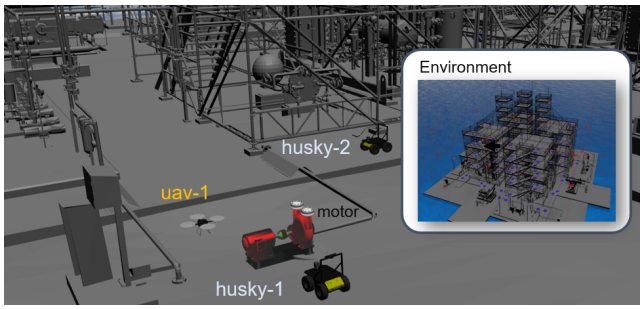

Figure 6: Overview of the simulator scenario with multiple aerial and ground robots executing a mission plan. The view of the platform shows the possible navigation paths.

`poi_property` provides relevant location information (e.g., goal location, recharge point, etc.), and `roadmap_poi_name` is the semantic tag of `wpg1` in the roadmap (e.g., SE_out_2). This information enables high-level task planning to be connected to the simulation execution tools. Therefore, if a plan action asks a robot to navigate from `wpg1` to `wpg52` the simulator makes use of the `roadmap_poi_name` variable to query the roadmap for a path that connects both points.

The simulator supports Human-Robot-Interaction (HRI), including remote interaction with the robotic platforms through natural language commands and can receive vehicle and mission status through natural language such as *inspect the valve1*. However, this work does not directly use this type of single-action command implementation. Instead, we are interested in leveraging the strength of AI planning solutions to interact with multiple robots in the environment to solve long-term missions with a large number of goals. Moreover, the robots can reason about additional actions in the plan which are not directly related to achieving a goal. For instance, the plan solution can contain actions related to recharging the robot's battery, which is not a mission goal, but still fundamental for successful mission execution.

Overall, our simulator provides a comprehensive platform for developing and supporting HRI techniques, to aid in building human-robot trust in high stakes scenarios such as emergency response. It also enables testing of task planning algorithms for cooperative inspection and long-term autonomy, and human-guided supervision and control of the robotic assets from remotely located control stations. Being able to exhaustively test these applications ensures the coherence and efficiency of the execution plans, thus increasing the likelihood of adoption of robotics and autonomous systems for high-risk environments.

Planning and Execution

Robustness is a fundamental requirement for the success of complex robotic missions in extreme environments, requiring the consideration of mission failures, adaptability and survivability. Figure 7 shows a general description of the three elements involved in the mission implementation: Planning Framework (PF), Planning Execution Framework (PEF), and Plan Deviation Analyser (PDA). This architecture is an extension (grey components represent the additions) of ROSPlan (Cashmore et al. 2015). Here, we did not mention the users, considering this process occurs when they decide the plan obtained in the Development and Review phases (see Figure 1) can be executed. However, in the Supervision phase, the user can decide to stop the mission and replan it. In this section, we aim to describe the processes of plan dispatch, execution, and replanning in case of failures. We consider the system can adapt to a set of mission failures. Therefore, we evaluate the system's ability to overcome unexpected situations as a consequence of using an incomplete domain model.

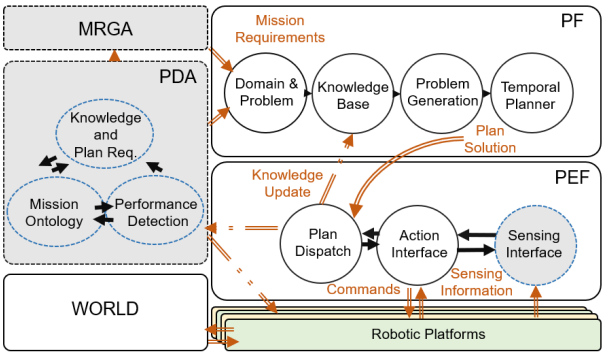

Figure 7: Framework for offline-online plan generation and execution. The system presents three main parts: Planning Framework (PF), Plan Execution Framework (PEF) and Plan Deviation Analyser (PDA).

The PF encloses the process of generating a plan solution. The planner produces plans using a domain model and a problem which are inputs to the Knowledge-Base (KB). As a result of using the PF we have a parsed plan available to be executed. The PEF dispatches the plan to a set of robots using the Plan Dispatch and Action Dispatch components from ROSPlan. The robotic platforms receive the actions through the action interface and provide feedback on the execution. If the action is successfully completed the next action in the plan is dispatched. If an action fails, however, further action dispatches are cancelled.

Plan Deviation Analysis: Our system introduces the PDA framework which can be used in two different cases:

- **Case 1:** There is a failure associate with the implementation of the action that makes the PEF to cancel the Plan Dispatch and claim replanning.
- **Case 2:** The Performance Detection component of the PDA detects possible problems during the execution of the action and interacts with the system to deal with them at the execution time.

In Case 1, the PDA takes the information the plan fails without completing all mission goals. The system checks the action that fails and queries the KB to evaluate possible reasons that guarantee the new plan solution is solvable. For instance, a Husky lost track of the image reconstruction of a structure while it is mapping an area. The PDA will consider the last location the quality of the map was acceptable and it

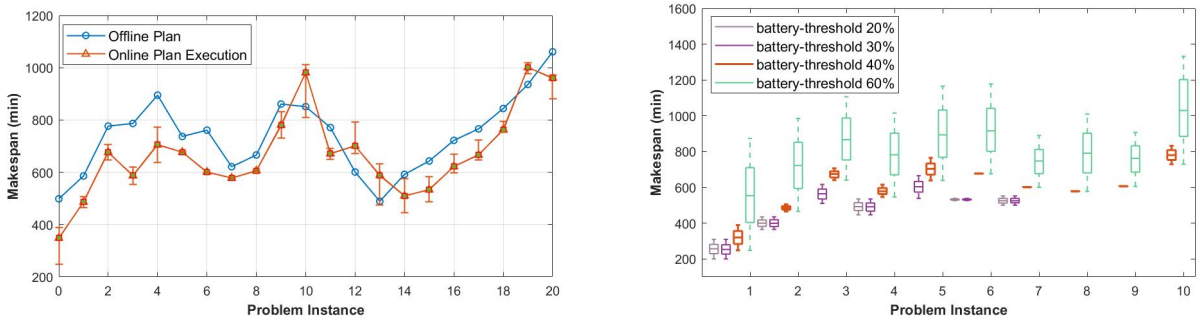

Figure 8: Makespan performance (offline) and real mission execution times (online) for 20 problem instances over 100 experiments (left). Makespan performance for different battery thresholds (right) for 10 problem instances over 50 experiments.

will add this information to the KB; based on the feedback from this evaluation, the system decides the next step. In addition, the PDA will evaluate other required preconditions to execute actions that were removed from the KB during execution and add them. For instance, if action `navigate` requires knowing the robot location (precondition `robot_at ?r - robot`) the PDA adds the robot's actual location to the KB before calling the planner to replan.

In Case 2, replanning is not needed. The PDA aims to observe the execution of particular actions and identify possible problems that might force the system to replan in the future. For instance, a Husky navigates from `wpg1` to `wpg52` and the navigation is delayed, as a consequence, the robot took longer to reach the second floor (`wpg52` location). The PDA can suggest the Husky to increase the speed to complete the action in the required time. These suggestions will depend on the action and robot characteristics that are defined in the Mission Ontology (MO). This type of online solutions makes robots adapt their behaviour while executing the plan avoiding the need for replanning. The MO is populated with multiple properties associated with all actions. Therefore, we can have a set of propositions to deal with failures associated with domain actions.

Performance Evaluation: Our framework can store data related to (i) offline planning, (ii) online planning, (iii) execution, and (iv) sensing information. This data includes mission goals; actions failures; planning time; makespan (at planning time and during the execution); and data associated with the sensors involved in the mission. Figure 8 shows examples of the results the framework can present. The first, shows the makespan performance and mission execution times for a set of 20 missions (some of them can take a day approximately). Results exhibit the execution times differ from the plan makespan originally obtained in a set of missions. The acquisition of this information over time allows planning experts to enhance model accuracy, system adaptability, and plan optimisation. Figure 8 (right) shows execution performance for the first 10 missions considering the Husky needs to recharge for different battery levels (battery life $\sim$ 5 hours and recharge time $\sim$ 2 hours). These results indicate best performance is obtained when the threshold for recharging is 40%. For 60% makespan increases substantially and for 20% and 30% the planner cannot solve all missions. However, we can use these small values to execute certain problems (e.g., 1 and 2) as the execution times are reduced as a result the Husky does not need to recharge. This information can be useful to the user to define which is the best time to recharge the battery of the robots to avoid unexpected situations where robots fail to maintain long-term operations. This project leads to a set of new autonomous solutions for extreme environments. Our framework supports the plan's evaluation considering industry-standard metrics such as implementation times and mission survivability.

Conclusion and Future Work

In this paper, we presented an end-to-end framework for planning and executing remote missions for multiple coordinated heterogeneous robots. Our system targets two main goals: providing oversight capabilities to human operators for safety and accountability while ensuring a robust implementation of long-term autonomous activities. In particular, the system is designed around three mission stages: plan generation, plan inspection and plan monitoring during mission execution. Overall, the proposed framework enables non-expert users to plan missions in complex environments which, we believe, constitutes a robust system for integrating AI planning solutions in industrial applications.

As future work, we are considering some additions to this framework. While our plan deviation analysis provides answers to exceptions occurring during execution, integrating contingencies into the plan generation process would also provide additional benefits reducing the amount of replanning needed during mission execution, and redefining mission goals or preferences by considering contingent plans. A second addition to our framework would be to implement an interface for users to visualise the performance of previous missions. Such a tool would allow planned activities to be analysed and compared against the reality of execution, understand the pitfalls or strengths of past missions, and make better judgements when developing new plans.

Acknowledgements

This work was funded and supported by the ORCA Hub (`orcahub.org`), under EPSRC grant EP/R026173/1.

References

Bach, B.; Shi, C.; Heulot, N.; Madhyastha, T.; Grabowski, T.; and Dragicevic, P. 2015. Time curves: Folding time to visualize patterns of temporal evolution in data. *IEEE Transactions on Visualization and Computer Graphics* 22(1): 559–568.

Barreiro, J.; Boyce, M.; Do, M.; Frank, J.; Iatauro, M.; Kichkaylo, T.; Morris, P.; Ong, J.; Remolina, E.; Smith, T.; et al. 2012. EUROPA: A platform for AI planning, scheduling, constraint programming, and optimization. *4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS)* .

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of ICAPS*.

Bernardini, S.; Fox, M.; Long, D.; and Piacentini, C. 2017. Boosting Search Guidance in Problems with Semantic Attachments. In *Proceedings of ICAPS*, 29–37.

Bernardini, S.; Jovan, F.; Jiang, Z.; Watson, S.; Weightman, A.; Moradi, P.; Richardson, T.; Sadeghian, R.; and Sareh, S. 2020. A Multi-Robot Platform for the Autonomous Operation and Maintenance of Offshore Wind Farms. In *Proceedings of AAMAS*, 1696–1700.

Carreno, Y.; Pairet, È.; Petillot, Y.; and Petrick, R. P. 2020. Task Allocation Strategy for Heterogeneous Robot Teams in Offshore Missions. In *Proceedings of AAMAS*, 222–230.

Carreno, Y.; Petillot, Y.; and Petrick, R. P. 2019. Multi-Vehicle Temporal Planning for Underwater Applications. In *Proceedings of ICAPS Workshop on Planning and Robotics (PlanRob)*.

Cashmore, M.; Collins, A.; Krarup, B.; Krivic, S.; Magazzeni, D.; and Smith, D. 2019. Towards explainable AI planning as a service. In *Proceedings of the ICAPS Workshop on Explainable Planning (XAIP)*, 104–112.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; and Carreras, M. 2015. ROSPlan: Planning in the Robot Operating System. In *Proceedings of ICAPS*, 333–341.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings of ICAPS*, 42–49.

Cummings, M.; Huang, L.; Zhu, H.; Finkelstein, D.; and Wei, R. 2019. The impact of increasing autonomy on training requirements in a UAV supervisory control task. *Journal of Cognitive Engineering and Decision Making* 13(4): 295–309.

Edelkamp, S.; and Mehler, T. 2005. Knowledge acquisition and knowledge engineering in the modplan workbench. *Proceedings of the First International Competition on Knowledge Engineering for AI Planning* 26–33.

Foster, M. E.; Gaschler, A.; Giuliani, M.; Isard, A.; Pateraki, M.; and Petrick, R. P. 2012. Two people walk into a bar: Dynamic multi-party social interaction with a robot agent. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI)*, 3–10.

Fox, M.; and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR* 20: 61–124.

Fraternali, P.; Castelletti, A.; Soncini-Sessa, R.; Ruiz, C. V.; and Rizzoli, A. E. 2012. Putting humans in the loop: Social computing for Water Resources Management. *Environmental Modelling & Software* 37: 68–77.

Hastie, H.; Robb, D. A.; Lopes, J.; Ahmad, M.; Le Bras, P.; Liu, X.; Petrick, R.; Lohan, K.; and Chantler, M. J. 2019. Challenges in Collaborative HRI for Remote Robot Teams. *arXiv preprint arXiv:1905.07379* .

Hatzi, O.; Vrakas, D.; Bassiliades, N.; Anagnostopoulos, D.; and Vlahavas, I. 2010. A visual programming system for automated problem solving. *Expert Systems with Applications* 37(6): 4611–4625.

Kim, J.; and Blythe, J. 2003. Supporting plan authoring and analysis. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, 109–116.

Le Bras, P.; Carreno, Y.; Lindsay, A.; Petrick, R. P.; and Chantler, M. J. 2020. PlanCurves: An Interface for End-Users to Visualise Multi-Agent Temporal Plans. In *Proceedings of ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*.

McCluskey, T.; and Simpson, R. 2006. Tool support for planning and plan analysis within domains enbodying continuous change. In *Proceedings of the ICAPS Workshop on Plan Analysis and Management*.

Pairet, È.; Ardón, P.; Liu, X.; Lopes, J.; Hastie, H.; and Lohan, K. S. 2019. A digital twin for human-robot interaction. In *Proceedings ACM/IEEE HRI*, 372–372.

Piacentini, C.; Alimisis, V.; Fox, M.; and Long, D. 2015. An extension of metric temporal planning with application to AC voltage control. *Artificial Intelligence* 229: 210–245.

Real-Arce, D. A.; Morales, T.; Barrera, C.; Hernández, J.; and Llinás, O. 2016. Smart and networking underwater robots in cooperation meshes: the swarms ECSEL: H2020 project. In *Instrumentation viewpoint*, 19, 19–19. SARTI.

Sanner, S. 2011. Relational dynamic influence diagram language (RDDL): Language description (2010). *URL http: // users. cecs. anu. edu. to / ssanner / IPPC* 59–79.

Scott, S. D.; Lesh, N.; and Klau, G. W. 2002. Investigating human-computer optimization. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 155–162.

Vaquero, T. S.; Romero, V.; Tonidandel, F.; and Silva, J. R. 2007. itSIMPLE 2.0: An Integrated Tool for Designing Planning Domains. In *Proceedings of ICAPS*, 336–343.

Vodrázka, J.; and Chrpa, L. 2010. Visual design of planning domains. In *Proceedings of ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 68–69.

Vrakas, D.; and Vlahavas, I. 2003. A Graphical Interface for Adaptive Planning. In *Proceedings of the ICAPS Doctoral Consortium*, 137–141.