Planning, Execution, and Adaptation for Multi-Robot Systems using Probabilistic and Temporal Planning

Yaniel Carreno[†], Jun Hao Alvin Ng[†], Yvan Petillot and Ron Petrick
Edinburgh Centre for Robotics
Heriot-Watt University and The University of Edinburgh
Edinburgh, United Kingdom
{y.carreno,alvin.ng,y.r.petillot,r.petrick}@hw.ac.uk

ABSTRACT

Planning for multi-robot coordination during long horizon missions in complex environments need to consider resources, temporal constraints, and uncertainty. This could be computationally expensive and impractical for online planning and execution. We propose a decoupled framework to address this. At the high-level, we plan for multi-robot missions that require coordination amongst robots considering temporal and numeric constraints. The temporal plan is decomposed into low-level plans for individual robots. At the low-level, we perform online learning and adaptation due to unexpected probabilistic outcomes to achieve mission goals. Our framework learns over time to improve the performance by (1) updating the learned domain model to reduce model prediction errors and (2) constraining the robot's capabilities which in turn improves goal allocation. The approach provides a solution to planning problems that require long-term robot operability. We demonstrate the performance of our approach via experiments involving a fleet of heterogeneous robots.

KEYWORDS

Temporal Planning; Multi-Agent Planning; Multi-Robot Systems; Model-Based Reinforcement Learning

ACM Reference Format:
Yaniel Carreno[†], Jun Hao Alvin Ng[†], Yvan Petillot and Ron Petrick. 2022. Planning, Execution, and Adaptation for Multi-Robot Systems using Probabilistic and Temporal Planning. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Automated planning provides the tools to enable intelligent behaviours in robotic systems operating in the real world. *Temporal planning* considers temporal constraints and durative actions to synthesise temporal plans involving concurrent actions and to support multi-agent coordination. However, these planners do not consider probabilistic outcomes of actions as they only deal with deterministic actions. *Probabilistic planning* reasons about the probabilistic outcomes of actions or exogenous events to generate a plan, providing an alternative approach for dealing with the risk of failures. However, probabilistic planners do not reason about temporal constraints and are ill-suited for multi-agent problems as the size of the state space increases exponentially with the number of agents. Existing work combining temporal planning and reasoning under uncertainty [2, 33, 35, 42] does not fully address the requirements of our application of interest: deploying and coordinating a fleet of heterogeneous robots in an environment with temporal uncertainty and probabilistic outcomes, without the requirement of a true (i.e., complete and accurate) domain model. Consider the following example.

[†] Authors with equal contribution.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online. © 2022 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

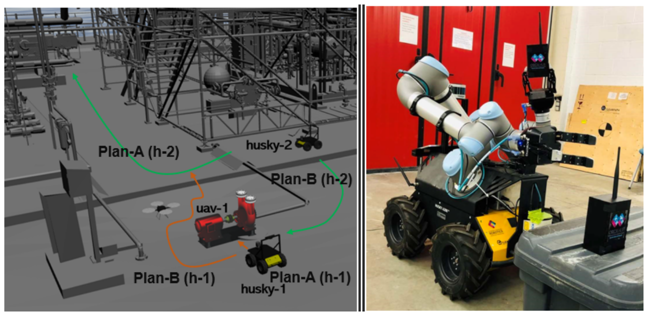


Figure 1: Mission environment in simulation (left) and real (right), where some robots plan, execute, and adapt in the face of uncertainty to achieve some goals.

Example 1. *(Offshore Energy Platform) The maintenance and supervision of an offshore energy platform is performed by a fleet of heterogeneous Unmanned Aerial Vehicles (UAVs) and terrestrial (Husky) robots (see Figure 1) that require coordination. For instance, uav-1 needs to supervise husky-1 while the latter is manipulating the control panel of an industrial motor in an area with high radiation. In addition, robots need to operate over long time horizons to complete goals that depend on the success of previous goals. For instance, husky-2 in Figure 1 (right) needs to pick up and relocate tools that are required to complete other goals. Working on an offshore energy platform requires reasoning about the probability of failure associated with actions. Replanning is not always an option in highly dynamic environments. For instance, husky-1 might fail to manipulate the control panel as a result of a damaged manipulator arm; it needs to navigate to the base for repairs. A planner then needs to reason if it should reallocate the goal to husky-2 or wait for husky-1's repair to complete. Our application involves hard (e.g., temporal and resource) constraints, joint goals (achieved by coordinated actions from multiple robots), and probabilistic dynamics, and cannot easily be solved by solely using either temporal planning or probabilistic planning.*

This paper presents two main contributions. First, we propose the hybrid framework Temporal Planning and Multi-Agent Coordination under Uncertainty (TPMACU). A centralised multi-agent temporal planning (MATP) module, defined as the *deterministic high-level (H-L) planning* component, acts as a strategic long horizon planner coordinating multiple robots and reasons with temporal constraints and robot capabilities. Each robot is a decentralised reinforcement learning (RL) agent, defined as the *probabilistic low-level (L-L) planning* component, which adapts to unexpected events and learns over time to improve reasoning capabilities. The second contribution is the mapping of a temporal H-L problem to a set of single-agent probabilistic L-L planning problems. The search spaces of the single-agent planning problems are reduced by considering only a subset of the state-action space of the MATP problem, which in turn decreases the sample complexity of the RL algorithm. Observations from plan execution are used to inform H-L replanning which manages mission safety and *makespan*—the plan execution time. The two-tier approach significantly reduces the search space on both levels, decreasing the computational cost and enabling online planning and execution.

2 RELATED WORK

Temporal planners such as [16, 21, 23] can solve MATP problems but suffer from scalability issues. More recent work [1, 8–11, 30] improves scalability. Carreno et al. [9] enhance the quality of MATP solutions, but only consider deterministic domains. Prior work dealing with multi-agent planning and uncertainty [4, 26, 39] does not consider multi-agent coordination and temporal constraints.

Next, we consider work which combines probabilistic planning and temporal planning. Our approach extends Schillinger et al. [33] work by temporally constraining actions and goals, and relaxing the requirement of holding a complete and accurate model. Bernardini et al. [2] combine methods for the probabilistic motion of a target with temporal planning. We consider uncertainty in outcomes of task-level actions rather than in motion. In terms of probabilistic dynamics and temporal uncertainty [35, 42], we address the uncertainty by learning action durations from observations. Little et al. [27], Foss and Onder [18] and Bradley et al. [3] present approaches focusing on single-agent problems. Zhang et al. [42] focus on complete decoupled missions while we are interested in coordinated goals. Weld [37] describes a planner that handles concurrent, durative, and probabilistic activities; however, its performance is limited in problems that require coordination.

Multi-agent RL (MARL) algorithms learn in the shared state-action spaces of all agents [41] which are often prohibitively large and do not scale well. An example of MARL complexity is the work of Xinyi et al. [38]. Our approach achieves less expensive data collection by decoupling MATP and goal allocation from single-agent planning and learning. Our work extends the ideas in [7, 13] to solve highly coupled problems with temporal requirements. Our work uses a single-agent model-based RL method which has a lower sample complexity than model-free methods [5, 14, 17].

3 PRELIMINARIES

The H-L component is modelled as a temporal planning problem $\mathcal{P}_t$ using the Planning Domain Definition Language (PDDL) [29] version 2.1 [19]. The solution for the MATP problem decouples the goal allocation from planning.

Definition 1. *The tuple $\mathcal{P}_t := \langle \mathcal{P}, \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{T} \rangle$ represents a temporal planning problem where $\mathcal{P}$ is a set of atomic propositions; $\mathcal{V}$ is a set of numeric variables; $\mathcal{A}$ is a set of instantaneous and durative actions where each action $a \in \mathcal{A}$ is a tuple $\langle a_{pre}, a_{eff}, a_{dur} \rangle$, a_{pre} is a set of conditions that must hold for the action to be applicable, a_{eff} is the set of action effects, and a_{dur} is a set of duration constraints; $\mathcal{I}$ (initial state) and $\mathcal{G}$ (set of goals) are respectively complete/partial value assignment to $\mathcal{P}$ and $\mathcal{V}$; and $\mathcal{T}$ is a set of timed initial literals (TILs) which defines the time t at which a proposition $p \in \mathcal{P}$ becomes true/false and characterises a time window.*

A timed initial literal (TIL) [12] is a pair (t_i, p_i) that defines the time t_i when a Boolean variable $p_i \in \mathcal{P}$ becomes true. Similarly, a TIL $(t_i, \neg p_i)$ describes the time t_i when p_i becomes false $(\neg p_i)$. The solution $\Pi_t = \{a_1 \cdots, a_n\}$ to $\mathcal{P}_t$ is a time-aware plan described by a sequence of durative and instantaneous actions, where each action in the sequence is applicable, and Π_t achieves the goals in $\mathcal{G}$ satisfying the temporal constraints.

Definition 2. *A goal allocation problem is defined by a tuple $\mathcal{G}_{\mathcal{A}} := \langle \mathcal{R}, \mathcal{RC}, \mathcal{G}, \mathcal{GC}, \mathcal{X} \rangle$ where $\mathcal{R}$ is a set of agents; $\mathcal{RC}$ is a set of agents' capabilities; $\mathcal{G}$ is a set of goals; $\mathcal{GC}$ is a set of capabilities required to implement the goals; and $\mathcal{X}$ is a set of coordinates for locations of interest, which represent an abstraction of the world.*

A capability set $C = \{c_1, \cdots, c_z\}$ associated with a robotics problem describes an abstraction of the robot r_i, where $r_i \in \mathcal{R}$, considering the actions $\mathcal{A}$ and goals in $\mathcal{G}$ that r_i can achieve based on its sensors, actuators and intelligent software systems.

The L-L component uses a model-based reinforcement learning (MBRL) method to train a policy for a single-agent problem, modelled as a MDP using the Relational Dynamic Influence Diagram Language (RDDL) [32]. In contrast to PDDL, RDDL allows the modelling of probabilistic action outcomes.

Definition 3. *MDPs model fully-observable problems with uncertainty. A finite-horizon MDP is a tuple $\langle S, A, T, R, s_0, H, \gamma \rangle$ where S is a set of states, A is the set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is the transition function, $R : S \times A \rightarrow \mathbb{R}$ specifies rewards for performing actions, s_0 is the initial state, H is the time horizon, and γ is the discount factor.*

We consider uncertainty in the L-L problem which is modelled as a MDP. In complex domains, the true models (i.e., T and R) are often not known. Thus, probabilistic planners are not feasible. Instead, we use MBRL to learn a Q-function which estimates the expected values, or Q-values, of executing actions in states. This does not require the true model. The agent follows a greedy policy which selects an action with the maximal Q-value.

4 PROBLEM MODEL

In this section, we describe the H-L and L-L domains[1], the mapping from an H-L domain (problem) to an L-L domain (problem), and the decomposition of an H-L plan to a set of L-L plans. Here, we model Example 1 as a planning problem.

[1]Domains and problems at https://github.com/YanielCarreno/bechmark-domains.git

High-Level (H-L) Domain. The H-L domain, modelled in PDDL, incorporates *predicates* that describe the environment and robot properties, including robot capabilities, availability, and *possible* locations each robot can navigate to. For example, the predicate *robot_can_act(?r ?wp)* constrains robot *?r* to a set of locations *?wp* where it can be at. We use *functions* to represent a robot's energy and data resources, and distances between locations. For example, *distance_intime (?wpi ?wpf)* defines the time robot *?r* at *?wpi* takes to reach *?wpf.* We have six durative H-L actions for a robot *?r*: (i) *navigation (?r ?wpi ?wpf), ?r* moves from location *?wpi* to *?wpf*; (ii) *take_image (?r ?s ?wp), ?r* captures an image of the location *?wp* with its camera *?s*; (iii) *check_temperature (?r ?s ?wp), ?r* measures the temperature at location *?wp* with its sensor *?s*; (iv) *valve_inspection (?r ?s ?wp), ?r* inspects the valve at location *?wp* with its camera *?s*; (v) and *manipulate_valve (?r1 ?r2 ?s1 ?s2 ?wp1 ?wp2)*, a husky *?r1*'s manipulator arm *?s1* turns the valve at location *?wp1* while a UAV *?r2* at location *?wp2* records the process with its camera *?s2.*

Our application involves two types of robots, a husky and an UAV, which have different sensors onboard. Thus, some PDDL actions and predicates could apply to one type of robot and not the other. The duration of the H-L actions is determined from data collected in past experiments involving real robots. The H-L domain models multiple robot behaviours and but does not consider probabilistic outcomes due to actions.

Low-Level (L-L) Domain. The L-L domain models probabilistic actions which allows reasoning over every probable outcome during plan execution. Probabilistic outcomes are: (i) the loss of robot's localisation while navigating, (ii) the loss of robot's camera calibration, and (iii) the damage of a husky's manipulator arm while navigating. To model these probabilistic outcomes, we use RDDL which cannot represent temporal aspects. Hence, we map the H-L actions to non-durative L-L actions as shown in Figure 2. The L-L actions which a robot *?r* can execute are: (i) *capture_image(?r ?poi), ?r* takes an image of *?poi*; (ii) *locate_poi(?r ?poi), ?r* search for *?poi* at its current location; (iii) *inspect_poi(?r ?poi)*, a husky *?r* inspects the temperature, pressure, or valve, or a UAV *?r* records a husky manipulating the valve *?poi*; (iv) *localise(?r), ?r* localises itself; (v) *goto_waypoint(?r ?wpi ?wpf), ?r* navigates from location *?wpi* to *?wpf*; (vi) *manipulate_valve(?r ?poi)*, a husky *?r* manipulates the valve *?poi*; (vii) *repair_manipulator(?r)*, a husky *?r* repairs its manipulator arm, possibly with external aid; (viii) and *calibrate_camera(?r), ?r* calibrates its camera.

We introduce the actions *localise*, *repair_manipulator*, and *calibrate_camera* to handle unexpected events. An ill-calibrated camera reduces the probability of success of *inspect_poi* and *manipulate_valve*, while a damaged manipulator arm reduces the probability of success of *manipulate_valve.* A robot can only calibrate its camera or repair its manipulator arm at the base. If it loses localisation, it cannot execute any other actions except *localise.*

We consider and model temporal elements of the H-L domain in the L-L domain. First, the costs of L-L actions are the duration of executing the actions. An RL algorithm which maximises the expected return will therefore learn policies which minimise the makespan. Second, joint goals are represented as predicates which must be made true by the time instances as defined by the end time of the H-L action *manipulate_valve* in the temporal plan. Robots

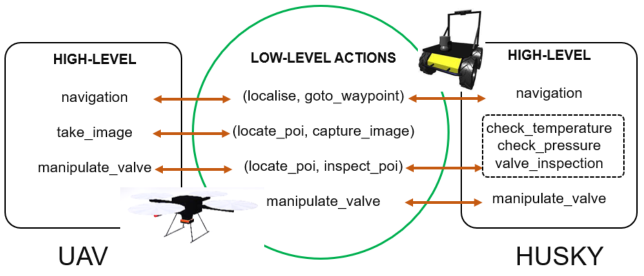


Figure 2: Mapping from an H-L action for a husky and UAV to a sequence of L-L actions. More than one H-L action can be mapped to an L-L action using different grounding.

H-L plan	L-L plan for husky-2	L-L plan for uav-1
Time: Action [Duration]	**Action**	**Action**
0.0: nav(h2 ...) [166.4]	loc(h2)	loc(uav-1)
0.0: nav(h1 ...) [115.2]	goto_wp(h2 wpg1 wpg52)	goto_wp(uav-1 wpa0 wpa35)
0.0: nav(uav-1 ...) [111.5]	loc_poi(h2 pressure_wpg52)	...
...	...	loc_poi(uav-1 valve_wpa35)
166.4: check_press(h2 ...) [20.0]	goto_wp(h2 wpg52 wpg35)	inspect_poi(uav-1 valve_wpa35)
268.0: valve_inspection(h2 ...) [50.0]	man_valve(h2 valve_wpg35)	
318.0: man_valve(h2 uav-1 ...) [30.0]		

Table 1: A fragment of an H-L temporal plan for uav-1, husky-1 (h2), and husky-2 (h2). Fragments of the L-L plans mapped from the H-L temporal plan for husky-2 and uav-1.

fail to achieve a joint goal if the mission time exceeds the start time of the joint goal. A joint goal of manipulating a valve is achieved when a husky executes *manipulate_valve* and a UAV executes *inspect_poi* concurrently. Either robot can execute the joint action before the start time of the joint goal. When this happens, the robot simply idles till the start time. Planning (or replanning) at the H-L is required to coordinate robots to achieve a joint goal.

Mapping between H-L and L-L Representations. The L-L problems and L-L plans are mapped from the H-L problem and temporal plan. By using only objects which are relevant to each robot, the state-action space for an agent is reduced which decreases the sample and computational complexities of MBRL.

Definition 4. *An object is relevant for a L-L problem if it is in the initial state of the robot, is associated with the goals allocated to the robot, or is associated with an action in the L-L plan.*

We illustrate the mapping from a temporal plan for an H-L problem used in our experiments (see Section 6) to L-L plans. To generate the L-L plan for *husky-2*, we extract H-L actions from the temporal plan which involve *husky-2.* Each of these H-L actions is mapped to L-L actions following the schema shown in Figure 2. For example, the *navigation* action is mapped to *localise* and *goto_waypoint* if the robot is not localised in the initial state, and subsequent *navigation* actions are mapped to *goto_waypoint.* This low-level detail is abstracted in the H-L domain but is important in the L-L domain as the actions relate to different actuation commands.

A fragment of an H-L plan and fragments of the associated L-L plans are presented in Table 1 (action names and parameters are abbreviated in the table). The goal of manipulating the valve *valve_wpg35* is a joint goal implemented by *uav-1* and *husky-2* (*h2*).

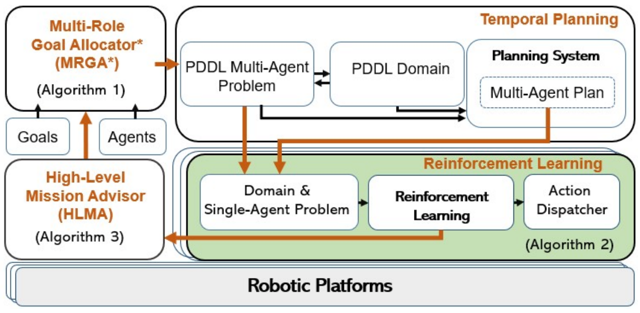


Figure 3: TPMACU framework for planning, acting, monitoring, and replanning. The H-L component deals with temporal and multi-agent planning while the L-L component deals with online learning and single-agent planning.

The actions which achieve this joint goal are highlighted in blue in Table 1. The H-L joint action *manipulate_valve* achieves this joint goal and requires the coordination of *uav-1* and *h2*. This H-L action is mapped to two different L-L actions. Both robots have to execute their L-L actions by the mission time of $t = 318$. Thus, a joint goal has a time constraint which is modelled in the L-L problem representation. If one of the robots attempts to achieve the joint goal before t, it will have to wait for the other robot. If either robot did not attempt the joint goal (by executing their respective L-L actions) by t, the joint goal can no longer be achieved until H-L replanning is done. Both robots will adapt to the situation by abandoning their L-L plans and follow their L-L policies instead. A L-L policy maps a state to an action. We describe the generation of L-L policies in Section 5. Mapping from L-L to H-L is required to update the H-L problem for H-L replanning. The H-L problem is updated to reflect changes in knowledge on the action durations, dynamics of the environment, and robot capabilities. Details are presented in Section 5.

5 FRAMEWORK IMPLEMENTATION

This section describes the algorithms for plan generation, execution, and adaptation at the H-L and L-L. Temporal planners generate plans which distribute the goals over multiple agents and minimise the makespan. However, the goal allocation quality is often relatively poor [9]. We address this issue by decoupling goal allocation from temporal planning. Next, we introduce a model-based reinforcement learning (MBRL) algorithm which deals with probabilistic dynamics. We use a H-L domain for temporal planning and a L-L domain for MBRL.

TPMACU Framework. Figure 3 illustrates our TPMACU framework which integrates the Multi-Role Goal Allocator* (MRGA*) algorithm (see Algorithm 1), temporal planning, and MBRL algorithm (see Algorithm 2). TPMACU performs planning, plan execution, replanning, and learning in an integrated cycle where observations from plan execution are used to improve planning performance. A MATP problem is decomposed into several single-agent probabilistic planning problems, reducing computational costs by avoiding planning in the combinatorial state space representing both temporal constraints and probabilistic dynamics.

We provide an overview of TPMACU here. Firstly, MRGA* augments the MATP problem with information about goal allocation. The augmented problem is given to the temporal planner which synthesises a multi-robot plan. Secondly, the H-L domain, problem, and plan are mapped to L-L domains, problems, and plans. We assume that the true probabilistic dynamics of the environment are unknown and rely on approximate (i.e., partially correct and incomplete) models. The models are generative; given a state-action pair (s, a), it predicts the successor state (s') and immediate reward (r). Thirdly, each robot will execute its L-L plan. Unexpected events could render the L-L plan inapplicable (i.e., the precondition of the action suggested by the L-L plan is not satisfied in the current state). Replanning at the H-L is impractical; if a robot requests replanning at the H-L while another robot is performing an inspection, this forces the latter to abort the inspection. Therefore, the robot acts according to a greedy policy generated from the Q-function, or its L-L policy. The Q-function is trained with imagined observations (s, a, r, s') produced by the generative model. Each robot adapts to unexpected events and attempts to achieve its allocated goals within the makespan of the H-L plan. Lastly, the observations acquired during plan execution are used to improve the L-L domains and update the actions' expected duration. At the end of the mission, robots feed back observations to the High-Level Mission Advisor (HLMA). When feedback from every robot is received, HLMA updates MRGA* on the robots' capabilities, observed action durations, and status of goals (completed or not). If there are unsatisfied goals, MRGA* reallocates these goals to the robots and the cycle repeats.

Goal Allocation and Coordinated Actions. Our goal allocation approach, MRGA*, extends the MRGA [9] strategy to address problems where collaborative and coordinated actions are required to achieve goals by recognising goal dependency. Algorithm 1 describes MRGA* (our extensions are highlighted in blue). The inputs are a goal allocation problem $\mathcal{G}_{\mathcal{A}}$ (see Definition 2), sensor redundancy $\mathcal{S}_e$, and $\mathcal{M}_m^r$, the set of cumulative makespan (initialised to 0) for each robot r which includes the time required $(\mathcal{M}_{Goal})$ to achieve a goal and the time history of the goals achieved by a robot r. A goal could require a robot or multiple robots to achieve. In the first step (line 2), the goals each robot can achieve are identified based on the robot capabilities and goal requirements. Next, the dependency between goals in $\mathcal{G}$ are determined (line 3) and embedded into $\mathcal{W}^*$. Therefore, each dependency $w \in \mathcal{W}^*$ is an ordered subset of $\mathcal{G}$. For example, a robot *husky-1* (see Figure 1, right) needs to navigate to the location *wp2* (g_2), pick up a tool (g_3), then navigate (g_4) to *wp20* to manipulate a valve (g_1). The dependency w is the order $g_2 \prec g_3 \prec g_4 \prec g_1$.

Next, we initialise a set of parameters for each dependency $w \in \mathcal{W}^*$ (line 4), including the goals' order and the first goal in w (line 5), the total makespan of achieving w, $\mathcal{M}_{Goal}$ (line 6), and the set of capabilities required to achieve w (line 7). $\mathcal{G}$ is updated by denoting the goals with dependency by the name of the first goal in w (line 8). The first goal in each w has a makespan that denotes the time required to achieve all goals in w. Next, goals in $\mathcal{G}$ are clustered into regions (line 9); C_{sol} is a set of generated clusters, $\mathcal{GR}_{sol}$ is the set of goals each robot can achieve in each cluster, and $\mathcal{JG} \subset \mathcal{W}^*$ is the set of goals with dependency that cannot be achieved by a single robot. MRGA* assigns robots to regions considering the number of goals the robots can achieve in the region (line 10-11). At most one robot assigned to each region. The goal in the robot's region with the closest proximity to the robot is allocated to it (line 12-13). The initial set of allocated goals $\mathcal{GA}_{INIT}$ is assigned to the final goal allocation set $\mathcal{GA}_{FINAL}$ (line 14). When a goal associated with a dependency w is allocated to the robot, the remaining goals in w are also allocated to it.

MRGA* allocates the remaining goals in $\mathcal{G}$ to robots such that the cost of achieving each goal is minimised (line 15-23). If a goal g is in $\mathcal{JG}$, MRGA* finds the best robots ($\mathcal{R}_{set}$) to achieve g (line 18). The accumulated makespan for each robot is used to find the maximum makespan (line 19). The accumulated makespan $\mathcal{M}^r_m$ considers the time associated with the goals that were allocated to the robots in the set in previous iterations. In this iteration, we add $\mathcal{M}^r_m$ to the cost of implementing g which is $\mathcal{M}_{Goal} = \mathcal{M}^w_{\mathcal{W}^*}$. Having the new $\mathcal{M}^r_m$, we calculate the cost of implementing the goal g which is saved in the b set (line 20). If the goal is not in $\mathcal{JG}$ (line 21), the goal or the set of goals with dependencies that require a robot to solve the problem is evaluated to find b (line 22-23). b is calculated considering $\mathcal{M}^r_m$, $\mathcal{X}$, and the redundancy of the sensory system. The goal is allocated to the robot with the lowest bid (line 24). This process is repeated until all goals are allocated. In every iteration we evaluate all remaining goals and robots to find the lowest bid. If the lowest bid is associated with a goal $g \in \mathcal{JG}$, g is allocated to all robots in $\mathcal{R}_{set}$ and their $\mathcal{M}^r_m$ are updated. The goal allocation is transformed into PDDL instances of the predicate *(robot_can_act ?r - robot ?wp - poi)* (line 25) which are included in the H-L problem (see Section 4).

Temporal Planning. The H-L domain, H-L problem, and the information provided by MRGA* are given inputs to a temporal planner to generate the temporal plan or H-L plan. We use the OPTIC [1] planner for temporal planning. The H-L plan specifies the actions each robot executes at a time instance. The H-L plan is decomposed into individual plans for each robot which are then mapped to L-L plans. A L-L plan is given as input to our MBRL method.

Online Planning, Execution, and Learning. During plan execution of the L-L plan, an unexpected event could invalidate the plan which necessitates replanning. For example, the robot's manipulator arm might be damaged forcing the robot to return to the base for repairs. This leads to two consequences. Firstly, the robot is now at the base and could be farther away from the goals it has been allocated. Secondly, joint goals which require the coordination of the robot cannot be achieved unless the repair is done. We propose to replan at the L-L where every robot is capable of online replanning. The robot deliberates over whether it should return to the base to repair its manipulator arm. This might be undesirable if the mission time exceeds the start time of a joint goal after the repair. However, if the probability of success with a damaged manipulator arm is low, then the robot might focus on completing other goals which does not require manipulation. Alternatively, it could choose to prioritise joint goals. A probabilistic planner is unable to reason about these as it requires the true model; we assume only an incomplete and deterministic model is available initially. Hence, we use a MBRL approach to learn a policy given observations. MBRL methods have a lower sample complexity than model-free methods [15] and are more suited to applications where data collection is expensive.

Algorithm 1: MRGA*

```
   Inputs: $(\mathcal{G}_{\mathcal{A}}, \mathcal{S}_e, \mathcal{M}^r_m)$
1  begin
2      $\mathcal{K} \leftarrow$ANALYSER$(\mathcal{G}, \mathcal{R}, \mathcal{RC}, \mathcal{GC})$
3      $\mathcal{W}^* \leftarrow$ find_dependency$(\mathcal{G})$
4      for each ordered $w \in \mathcal{W}^*$ do
5          $\mathcal{F}^w_{\mathcal{W}^*} \leftarrow$extract_first_goal$(\mathcal{G}, w)$
6          $\mathcal{M}^w_{\mathcal{W}^*} \leftarrow$ calculate_total_makespan$(\mathcal{G}, \mathcal{R}, w, \mathcal{X})$
7          $C_{req}{}^w_{\mathcal{W}^*} \leftarrow$ check_capability_requirement$(\mathcal{G}, \mathcal{R}, w)$
8      $\mathcal{G} \leftarrow$update_goal$(\mathcal{G}, \mathcal{F}^w_{\mathcal{W}^*}, \mathcal{W}^*)$
9      $[C_{sol}, \mathcal{GR}_{sol}, \mathcal{JG}]$ = cluster_cal $\left(\mathcal{G}, \mathcal{R}, \mathcal{X}, \mathcal{K}, C_{req}{}^w_{\mathcal{W}^*}\right)$
10     $k$ = weight_calculation$(C_{sol}, \mathcal{GR}_{sol})$
11     $\mathcal{RA}_{INIT} = max\{k\}$
12     regions_distance_evaluation$(\mathcal{RA}_{INIT}, \mathcal{X}, C_{sol})$
13     allocate_first_goal$(\mathcal{GA}_{INIT} \leftarrow \mathcal{G})$
14     $\mathcal{GA}_{FINAL}\ (\mathcal{G}, \mathcal{R}) \leftarrow \mathcal{GA}_{INIT}$
15     while not $\mathcal{G}$.update$(\mathcal{GA}_{INIT}) \leftarrow \emptyset$ do
16         for $g \in \mathcal{G}$ do
17             if $g \in \mathcal{JG}$ then
18                 $\mathcal{R}_{set} \leftarrow$ find_required_robots$(\mathcal{R}, g, C_{req}{}^w_{\mathcal{W}^*})$
19                 $\mathcal{M}^r_m \leftarrow$ find_maximum_makespan$(\mathcal{R}_{set}, \mathcal{M}^r_m)$
20                 $b$ = allocate_goal$(\mathcal{R}_{set}, g, \mathcal{S}_e, \mathcal{M}^r_m, \mathcal{X}, \mathcal{W}^*)$
21             else
22                 for $r \in \mathcal{R}$ do
23                     $b$ = allocate_goal$(r, g, \mathcal{S}_e, \mathcal{M}^r_m, \mathcal{X}, \mathcal{W}^*)$
24         $\mathcal{GA}_{FINAL}\ (\mathcal{G}, \mathcal{R}) \leftarrow$ findMin$(b)$
25     return transformPDDL$(\mathcal{GA}_{FINAL})$
```

Algorithm 2 describes our MBRL method. It takes as input a generative model ($\mathcal{M}$), a problem (P), goals (G) and joint goals ($\mathcal{JG}$) allocated to the robot, an L-L plan (A) which is mapped from the H-L plan given by the temporal planner, the rollout horizon ($H_{rollout}$), the maximum allowable makespan (H_{mk}), and the number of rollouts ($N_{rollout}$). The problem P is represented as an MDP $\langle S, A, T, R, s_0, H, \gamma, t\rangle$ with a continuous time dimension t for mission time. The transition function T is unknown. We start by training a Q-function Q_θ (line 8) with imagined observations generated using $\mathcal{M}$ which predicts the successor state (s_i), reward (r_t), and execution duration (Δt) (lines 3-8). This process is similar to Dyna [34]. The length of each rollout, or the number of successive simulated steps from the initial state, is at most $H_{rollout}$ (less than $H_{rollout}$ if a terminal state is reached). During simulation training, actions are selected with an epsilon-greedy policy, π_{eps} (line 6). The reward (or cost) of executing an action is $-\Delta t$, and a reward of +100 is given for each goal achieved while no reward is given for achieving a joint goal after its start time (or its time constraint). Q_θ is approximated as a linear function of the weight vector θ and a set of basis functions $\mathcal{F}$. The basis function is represented by features which are conjunctive predicates which maps a state to a lower dimension. The initial set of features comprises of every state predicate and its negation. New features, which are conjunctions of any two existing features, are added incrementally using iFDD+ [20], an online feature discovery algorithm. The weight vector is updated with Double Q-learning [22] (line 8). The simulation training is computationally expensive but can be done offline.

After simulation training, the robot executes the plan A sequentially (line 12). If an unexpected outcome occurs which renders the next action in A to be inapplicable (line 13), the robot then follows

Algorithm 2: MBRL

```
Inputs: $(\mathcal{M}, P, G, \mathcal{JG}, A, H_{rollout}, H_{mk}, N_{rollout})$
begin
    $\theta \leftarrow \mathbf{0}$, $\mathcal{F} \leftarrow$ initialise_features($\mathcal{M}$, $P$)
    for 1 to $N_{rollout}$ do
        $t = 0, \Delta t = 0$
        for $i$ = 1 to $H_{rollout}$ do
            $a_{i-1} \leftarrow \pi_{eps}(s_{i-1})$
            $s_i, r_i, \Delta t \leftarrow imagine(\mathcal{M}, \mathcal{JG}, G, s_{i-1}, a_{i-1}, t + \Delta t)$
            $\mathcal{F}, \theta \leftarrow update(\mathcal{F}, \theta, s_{i-1}, a_{i-1}, r_i, s_i)$
    $t = 0, Follow = \top$
    for $i$ = 1 to $H$ do
        if Follow then
            $a \leftarrow$ follow_plan($A$)
            if $a$ is not applicable in $s_{i-1}$ then $Follow = \bot$
        if $\neg$ Follow then $a \leftarrow \pi_{greedy}(s_{i-1})$
        $s_i, \Delta t \leftarrow execute(s_{i-1}, a)$, $t \leftarrow t + \Delta t$
        $\mathcal{M} \leftarrow update\_model(\mathcal{M}, s_{i-1}, a, s_i, \Delta t)$
        if all goals achieved or $t \geq H_{mk}$ then return $(\mathcal{M}, s_i, t)$
    return $(\mathcal{M}, s_H, t)$
```

a greedy policy π_{greedy} generated by Q_θ (line 14). Exploitation is desired for safety reasons and because Q_θ is updated only during simulation training. We do not perform online learning for Q_θ as the number of real observations is typically much smaller than $N_{rollout} \times H_{rollout}$, the number of imagined observations. A policy maps each state to an action while a plan maps a sequence of states to actions. In addition, our policy is computed offline. Thus, our approach incurs a lower computational cost during plan execution than probabilistic planning and enables fast decision-making [6]. Since the policy is trained with $\mathcal{M}$ which is often an approximation of the true model, the policy could be suboptimal or unsound in regions of the state spaces where $\mathcal{M}$ is a poor predictor. This is an issue known as distribution shift. Recent works have proposed solutions to deal with this [24, 36, 40]. A plausible solution is to terminate the mission when a state that is not seen during simulation training is visited.

The action is executed (line 15) and the resulting observation is used to update $\mathcal{M}$ (line 16) which could then generate more accurate imagined observations. We use the model learner from [28] to learn first-order rules for the transition function and translate the rules to RDDL syntax. The current state and mission time are inputs to the temporal planner for H-L replanning, if necessary. The mission is terminated (line 17) if (1) there are no remaining goals, or (2) the mission time exceeds H_{mk}. H-L replanning is done only after all robots have completed or terminated their missions.

High-Level Mission Advisor (HLMA). The High-Level Mission Advisor (HLMA) acts as a bridge between MBRL and H-L. This is detailed in Algorithm 3 where $\mathcal{R}$ is the set of robots, Π_t is the H-L plan, $\mathcal{RL}$ is a set of advice from MBRL, and $\mathcal{RC}$ is the set of robots' capabilities. We consider the feedback from MBRL to update the H-L problem. An advice $\vartheta \in \mathcal{RL}$ associated with a robot $r \in \mathcal{R}$ is appended to Ξ as a pair $\langle r, \vartheta \rangle$ (lines 2-5). This process ends when all robots are evaluated and all advice are considered (lines 2-5). The H-L problem is updated to consider the robots' current locations (line 6), the expected duration to achieve a goal (line 7), and the remaining unachieved goals (line 8). $\mathcal{M}_{Goal}$ is the duration of actions which

Algorithm 3: HLMA

```
Inputs: $(\mathcal{R}, \Pi_t, \mathcal{RL}, \mathcal{RC})$
begin
    for $r \in \mathcal{R}$ do
        $\vartheta \leftarrow$ acquired_modification($|\mathcal{RL}|, r$)
        $\Xi \leftarrow$ append_modification($\vartheta$,$r$)
        $\mathcal{RL}$.update($|\mathcal{RL}|, \vartheta$)
    $\mathcal{X} \leftarrow$ modify_robot_location($\mathcal{R}, \Xi$)
    $\mathcal{M}_{Goal} \leftarrow$ modify_action_duration($\mathcal{R}, \Xi$)
    $\mathcal{G} \leftarrow$ modify_goal($\mathcal{R}, \Xi$)
    $\mathcal{KK} \leftarrow$ repair_time($\mathcal{R}, \Xi$)
    if contain_information($\mathcal{G}$) then
        $\mathcal{RC} \leftarrow$ modify_capability($\mathcal{RC}, \mathcal{M}_{Goal}, \mathcal{KK}, \Pi_t$)
    return $(\mathcal{X}, \mathcal{M}_{Goal}, \mathcal{G}, \mathcal{KK}, \mathcal{RC})$
```

achieve a goal (line 7). For example, in our domain, the duration of *take_image* is defined by the function *(take_image_dur)*. If an advice in Ξ changes the duration of *take_image*, this updated duration is appended to $\mathcal{M}_{Goal}$. $\mathcal{KK}$ is the additional duration required to calibrate the robot's camera or repair its manipulator arm (line 9). This duration is included in the time required to achieve any remaining goal. In cases where the robots fail to achieve goals, HLMA evaluates the robots' capabilities $\mathcal{RC}$ (lines 10-11) using the equation $\mathcal{E} = arg_{\text{MIN}}\{\sum_{r_i \in \mathcal{R}'} cost(\pi_{t_i}, \mathcal{M}_{Goal}, \mathcal{KK}, \mathcal{G})\}$, where $\mathcal{R}' \subset \mathcal{R}$ are robots with the capabilities required to achieve the goals in $\mathcal{G}$, and π_{t_i} is the subsequence of the H-L plan Π_t that encloses the actions associated with the robot r_i. The approach evaluates the duration required by each robot to achieve the goals. For the robot associated with the failure, HLMA adds to its plan's makespan the repair time in $\mathcal{KK}$. The other robots include the time required to achieve the goal. HLMA finds the best robot(s) to achieve the goals. If the robot associated with the failure does not obtain the plan with the minimum $\mathcal{E}$, the capability associated with the goal is removed from its capability set.

Example 2. *(From L-L to H-L) HLMA improves the plan's quality by applying the information learned by individual robots during plan execution. Considering Example 1 (see Figure 1), the tasks in an initial H-L plan (Plan-A) are distributed amongst a fleet of three robots. A husky robot (h-1) fails the manipulation task which requires coordination with uav-1 in charge of supervising the task execution. Assuming the remaining goals are achieved, replanning is required for the joint goal. HLMA reasons about the additional time required for h-1 to return to the base for repairs, which is predicted by $\mathcal{M}$, to advise MRGA* on the reallocation of goals. In Plan-B, the goal is reallocated to h-2 as a consequence of HLMA's advice (i.e., the time incurred to repair h-1 is larger than that of having h-2 achieve the goal).*

Complexity Analysis. We consider Theorem 12 (polynomial reduction to classical planning) in [31] to claim PSPACE-completeness in MATP problems. MRGA* prunes the search space by constraining the number of agents considered to achieve goals. Next, the sample complexity of many RL algorithms depends polynomially on the size of the state space [25]. MARL algorithms suffers from poor scalability as they learn in the joint state and action spaces which scale exponentially with the number of agents. The joint action space for a multi-agent problem with k agents is $\prod_{i=1}^{k} |A_i|$

where A_i is the action space for the i-th agent. We decompose the MATP problem into single-agent problems and include only relevant objects in the single-agent problems. This reduces the size of the state-action space further. As an illustration, the size of the state space for a single agent is $2^{|\mathcal{P}|}$ where $\mathcal{P}$ is the set of predicates ground over the set of objects O. A lifted predicate with an arity of n can be grounded to $\prod_{i=1}^{n} |O_i|$ ground predicates where $|O_i|$ is the number of objects of the same type as the i-th argument of the predicate. The number of actions relate to the number of predicates in the same manner.

6 EXPERIMENTS AND RESULTS

We evaluate our work with a mission: *husky-1*, *husky-2* and *uav-1* are tasked with checking the temperature at a *poi*, checking the pressure at another *poi*, taking images of two other *poi*, and manipulating two valves. These six H-L goals are mapped to eight L-L goals. The manipulation of a valve is a joint goal which requires the coordination of a husky and a UAV to achieve. This is mapped to two L-L goals, one for the husky and one for the UAV. While our mission involves only three robots, our method does not have significant scalability issues because (1) MRGA* simplifies the MATP problem for a temporal planner by performing goal allocation in its place, and (2) MBRL handles single-agent problems and are not susceptible to the exponential increase in the size of the joint state and action space inherent in MARL algorithms. Problems which are challenging for our method possess a high goals-to-robots ratio (i.e., limited resources available to achieve the goals) rather than a large number of robots. In view of this, we posit a mission with eight goals for three robots is sufficient to provide some interesting empirical insights on our work.

We conducted simulated experiments using RDDLSim [32] as the simulator. We added normally-distributed noise to action durations (20% of its mean duration) which is not made known to our algorithms. At the start, MRGA* allocates the goals to the robots. Each husky is tasked with manipulating a valve which requires the supervision of *uav-1* (i.e., *uav-1* has two temporally-dependent joint goals since it cannot perform both at the same time). In addition, *uav-1* needs to take images of two *poi* and *husky-2* needs to check the temperature and pressure of two *poi*. The state-action spaces of the L-L problems for *uav-1*, *husky-1*, and *husky-2* are $2^{36} \times 45$, $2^{12} \times 12$, and $2^{29} \times 41$, respectively. The joint state space is 2^{167} and the joint action space is 170×173^2. The significant reduction in the state-action space demonstrates the essential step of decomposition that leads to only relevant objects for each robot being considered. Consider a hypothetical mission where the number of objects are doubled (i.e., four huskies, two UAVs, four valves, etc.) and the goals allocated to *uav-1*, *husky-1*, and *husky-2* remain the same as before. The state-action spaces for their L-L problems remain unchanged. Conversely, the joint state space is 2^{454} and the joint action space is $626^2 \times 631^4$, which are exponentially larger.

We evaluate the utility of learning over time using MBRL. An L-L policy is a greedy policy generated from the Q-function which is learned during simulation training with an approximate model (see Algorithm 2). The hyperparameters are $H_{rollout} = 30$, $N_{rollout} = 1000$, and $H = 40$. We compare two possible scenarios: **Scenario 1** where no observations are available and a deterministic model is used, and **Scenario 2** where a learned probabilistic model is learned from observations acquired from previous missions using the model learner from [28]. Both models are approximate (i.e., partially correct and incomplete) though the true preconditions of L-L actions are assumed to be known. As the deterministic model does not predict any probabilistic outcomes, it cannot predict states where the robot loses localisation, its camera loses calibration, or its manipulator arm is damaged. Therefore, these states are never encountered in the simulation training and the L-L policy is suboptimal or even unsound in these regions of the state space. On the other hand, the learned model predicts all probabilistic outcomes, albeit with the wrong probabilities.[2]

Since the environment is probabilistic, we simulated 100 independent runs with different random seeds for each scenario. The performances of both scenarios are shown in the table in Figure 4 and are measured by the number of goals completed/$\mathcal{G}$ (TC) and the number of missions completed by following the L-L plan/total number of missions (MC_{Plan}), or by following the L-L policy/total number of missions (MC_{Policy}). A robot will only follow its L-L policy after its L-L plan is invalidated. Thus, MC_{Policy} is the number of additional goals achieved due to our framework rather than aborting a robot's mission when its L-L plan cannot be executed. A robot's mission is completed if all of its goals are completed. If no unexpected events are encountered during plan execution, a robot can complete all of its goals by executing its L-L plan. Otherwise, it has to adapt by following its L-L policy.

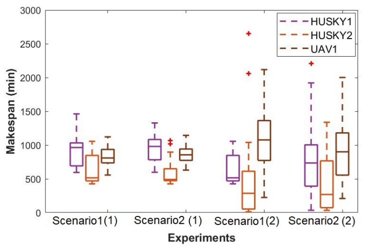
Our empirical results show that the learned model leads to improved performances. In general, the robots completed more missions and goals in Scenario 2 than in Scenario 1. Following the L-L policy trained with the learned model, *uav-1* completed 255 out of 400 goals (*uav-1* is allocated 4 goals, and there are 100 missions), 10 goals more than the deterministic model. The largest performance improvement due to the learned model is demonstrated in the results for *husky-1* where *husky-1* completed 22 missions out of 100 missions by following the L-L policy as compared to 11 missions using the deterministic model. Due to the probabilistic nature of the environment, only a small number of the missions are completed when the robots follow the L-L plan (MC_{Plan}). The utility of adapting to unexpected events by switching to the L-L policy is demonstrated by the relatively significant number of missions completed by following the L-L policy (MC_{Policy}).

At the end of the missions, the robots feed back to HLMA. If at least one goal is not completed, replanning at the H-L is required; a new temporal plan and problem instance are produced which are then mapped to a set of L-L problems. We do not reuse the L-L policies from before as objects and goals have changed. Instead, we train new L-L policies with simulation training using the approximate models again. Observations acquired in the previous mission are insufficient to improve the model, thus we use the same models as before.[3] Scenario 2 generally outperforms Scenario 1 as more goals are achieved in Scenario 2 than in Scenario 1. Unachieved

[2]The true model (resp. learned model) predicts the probability of the loss of camera calibration is 0.08 (resp. 0.42) and the probability of damaging a manipulator arm is 0.05 (resp. 0.12). The probability of success for *locate_poi*, *inspect_poi*, and *capture_image* if the camera is damaged is 0.2 (resp. 0.77, 0.69, 0.24, respectively). If the manipulator arm is damaged, the probability of success for *manipulate_valve* is 0.2 (resp. 0.74).

[3]The number of observations in each mission is at most $H = 40$. While observations in 100 independent runs might be sufficient to improve the model, information is

Scenario	Robot	TC[1]	MC_{Plan}[1]	MC_{Policy}[1]	%TC[2]	%MC_{Plan}[2]	%MC_{Policy}[2]
1	*uav-1*	*245/400*	*13/100*	*14/100*	*64*	*42*	*3*
2	*uav-1*	*255/400*	*25/100*	*12/100*	*81*	*46*	*19*
1	*husky-1*	*55/100*	*44/100*	*11/100*	*89*	*23*	*62*
2	*husky-1*	*66/100*	*44/100*	*22/100*	*88*	*23*	*55*
1	*husky-2*	*234/300*	*27/100*	*27/100*	*93*	*36*	*56*
2	*husky-2*	*252/300*	*34/100*	*28/100*	*97*	*43*	*52*



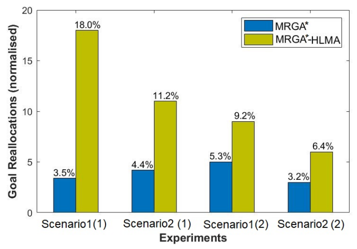


Figure 4: Experiment 1 (left): Comparison in performance using Scenario 1 and Scenario 2 following TC, MC_{Plan} and MC_{Policy} metrics. The superscript 1 (2) denotes the results for missions before (after) the H-L replanning. Experiment 2 (centre): Makespan for (1) planning and (2) replanning in Scenarios 1 and 2, respectively. Experiment 3 (right): Cumulative reallocation of goals using a deterministic and a learned model for planning and replanning.

goals are typically joint goals. In other words, Scenario 2 has a more significant proportion of goals which are joint goals. These can be impossible to achieve if unexpected events occur (e.g., it might not be possible for a husky to repair its arm and manipulate the valve in time). In Scenario 1 (Scenario 2), 21 (26) missions were completed where all goals were achieved. Following H-L replanning, in Scenario 1 (Scenario 2), 34 (46) more missions were completed. This leaves 45 (28) uncompleted missions (at least one goal remains unachieved) with 74 (44) unachieved goals in Scenario 1 (Scenario 2).[4] This shows that our work could improve performances over time even with an approximate model by updating the generative model given new observations. That is, the deterministic model (in Scenario-1) is updated to a probabilistic model (Scenario-2). Furthermore, the absence of probabilistic reasoning, due to the use of a deterministic model, in Scenario 1 causes the performances to deteriorate. Some missions remain uncompleted and require H-L replanning. We limit the duration of a mission to H discrete steps and a makespan of H_{mk}. Our evaluation does not consider the number of H-L replanning required to complete all missions as we intend for H-L replanning (i.e., MRGA* and HLMA) to improve the coordination of robots over time. For example, if the duration of a mission is unbounded, then the mission can be completed without H-L replanning but this does not imply a better performance.

Figure 4 (centre) shows the statistics for the makespan for H-L planning and replanning in both scenarios. In general, the makespans in the replanning outcomes are longer. The advice from HLMA influences the reallocation of goals such that the risk of failure is reduced during plan execution. The learned model experienced the most considerable change as HLMA considers the probabilities of unexpected outcomes. MRGA* is thus forced to reallocate goals less optimally (i.e., trading off makespan for reduced risk). However, this translates into a more significant number of missions completed for the learned model. Figure 4 (right) shows the cumulative reallocation of goals in both scenarios for H-L planning and replanning. Here, the advice from HLMA influences the number of goals that MRGA* reallocates. If HLMA provides advice, the MRGA* executes the strategy considering robot positions and new action durations. HLMA also reasons about the trade-off in allocating a goal to a robot which needs to repair its manipulator arm or calibrate its camera such that it can achieve a goal with a high probability, versus assigning the goal to a fully-functional robot that may be farther away from the goal. Based on this reasoning, HLMA advises MRGA* to retain the capability required to achieve the goal. The number of goals which have to be reallocated is reduced from H-L planning to H-L replanning. Our approach optimises the goal distribution considering learned knowledge from observations. Scenario 2 outperforms Scenario 1 in terms of the number of reallocations required (the fewer the better) for planning and replanning as a result of using a more accurate model. HLMA allows the approach to reach the best possible plan in comparison to the initial deterministic plan. Therefore, the number of reallocations due to risk of failures, which is feed back by MBRL, is gradually reduced over time. This explains the similarity of the results (3.2% ∼ 6.4%) for replanning in Scenario 2.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a framework, TPMACU, which decouples multi-goal allocation, MATP, and MBRL. This reduces the computational cost and the search space considered by each of them which improved scalability to large scale planning problems and is also well-suited for online planning, adaptation, and execution. MATP does not need to reason over all probable outcomes of actions while MBRL only deals with single-agent problems without temporal constraints. To introduce robustness to uncertainty, feedback from MBRL advises the multi-goal allocation and MATP. We demonstrated the applicability of our approach with a fleet of heterogeneous robots operating in an offshore energy platform. For future work, different strategies for goal reallocation can be explored (e.g., when a robot is unlikely to complete a joint goal or a severe unexpected outcome is encountered).

ACKNOWLEDGEMENTS

This work was supported by the ORCA Hub (https://www.orcahub.org/), under EPSRC grant EP/R026173/1.

not exchanged between the runs as they are independent and serve to give statistical confidence in our results.

[4]The metrics are shown in percentages for missions after replanning because the number of remaining goals is different which makes comparison of absolute numbers between the two scenarios difficult.

REFERENCES

[1] J Benton, Amanda Jane Coles, and Andrew Coles. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *ICAPS*. 2–10.

[2] Sara Bernardini, Maria Fox, and Derek Long. 2017. Combining temporal planning with probabilistic reasoning for autonomous surveillance missions. *Autonomous Robots* 41, 1 (2017), 181–203.

[3] Christopher Bradley, Adam Pacheck, Gregory Stein, Sebastian Castro, Hadas Kress-Gazit, and Nicholas Roy. 2021. Learning and Planning for Temporally Extended Tasks in Unknown Environments. *ICRA* (2021).

[4] Michael Brenner and Bernhard Nebel. 2009. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems* 19, 3 (2009), 297–331.

[5] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. 2018. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*. 8224–8234.

[6] Thiago Bueno, Leliane Barros, Denis Mauá, and Scott Sanner. 2019. Deep Reactive Policies for Planning in Stochastic Nonlinear Domains. *AAAI* 33 (07 2019), 7530–7537.

[7] Rafael C Cardoso and Rafael H Bordini. 2019. Decentralised Planning for Multi-Agent Programming Platforms. In *AAMAS*. 799–818.

[8] Yaniel Carreno, Èric Pairet, Yvan Petillot, and Ronald P A Petrick. 2020. A Decentralised Strategy for Heterogeneous AUV Missions via Goal Distribution and Temporal Planning. In *ICAPS*, Vol. 30. 431–439.

[9] Yaniel Carreno, Èric Pairet, Yvan Petillot, and Ronald P A Petrick. 2020. Task Allocation Strategy for Heterogeneous Robot Teams in Offshore Missions. In *AAMAS*. 222–230.

[10] Amanda Jane Coles, Andrew Coles, Maria Fox, and Derek Long. 2010. Forward-Chaining Partial-Order Planning. In *ICAPS*. 42–49.

[11] Amanda Jane Coles, Andrew I Coles, Maria Fox, and Derek Long. 2012. COLIN: Planning with continuous linear numeric change. *JAIR* 44 (2012), 1–96.

[12] Stephen Cresswell and Alexandra Coddington. 2003. Planning with timed literals and deadlines. In *UK PlanSIG*. 23–35.

[13] Matthew Crosby, Michael Rovatsos, and Ronald Petrick. 2013. Automated Agent Decomposition for Classical Planning. In *ICAPS*. 46–54.

[14] Marc Deisenroth and Carl E Rasmussen. 2011. PILCO: A model-based and data-efficient approach to policy search. In *ICML*. 465–472.

[15] Marc Peter Deisenroth, Gerhard Neumann, and Jan Peters. 2013. *A Survey on Policy Search for Robotics*. Vol. 2. Foundations and Trends®in Robotics. 1–142 pages.

[16] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. 2009. Using the context-enhanced additive heuristic for temporal and numeric planning. In *ICAPS*.

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

[18] Janae N Foss and Nilufer Onder. 2006. A Hill-Climbing Approach for Planning with Temporal Uncertainty.. In *FLAIRS*. 868–870.

[19] Maria Fox and Derek Long. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR* 20 (2003), 61–124.

[20] Alborz Geramifard, Finale Doshi, Josh Redding, Nicholas Roy, and Jonathan How. 2011. Online Discovery of Feature Dependencies. In *ICML*. 881–888.

[21] Alfonso Gerevini and Derek Long. 2006. Preferences and soft constraints in PDDL3. In *Proceedings of ICAPS Workshop on Planning with Preferences and Soft Constraints*. 46–53.

[22] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*. 2094–2100.

[23] Chih-Wei Hsu and Benjamin W Wah. 2008. The SGPlan planning system in IPC-6. In *ICAPS*.

[24] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*. 12519–12530.

[25] Sham M. Kakade. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation. University College London.

[26] Thomas Keller and Patrick Eyerich. 2012. PROST: Probabilistic Planning Based on UCT. In *ICAPS*.

[27] Iain Little, Douglas Aberdeen, and Sylvie Thiébaux. 2005. Prottle: A Probabilistic Temporal Planner. In *AAAI*, Vol. 3. 1181–1186.

[28] David Martínez, Tony Ribeiro, Katsumi Inoue, Guillem Alenyà Ribas, and Carme Torras. 2015. Learning probabilistic action models from interpretation transitions. In *ICLP*. 1–14.

[29] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL – The Planning Domain Definition Language (Version 1.2)*. Technical Report CVC TR-98-003/DCS TR-1165. Yale Center for Computational Vision and Control.

[30] Masood Feyzbakhsh Rankooh and Gholamreza Ghassem-Sani. 2015. ITSAT: an efficient sat-based temporal planner. *Journal of Artificial Intelligence Research* 53 (2015), 541–632.

[31] Jussi Rintanen. 2007. Complexity of Concurrent Temporal Planning. In *ICAPS*, Vol. 7. 280–287.

[32] Scott Sanner. 2010. Relational dynamic influence diagram language (RDDL): Language description. *Unpublished manuscript. Australian National University* 32 (2010), 27.

[33] P. Schillinger, M. Bürger, and D. V. Dimarogonas. 2018. Auctioning over Probabilistic Options for Temporal Logic-Based Multi-Robot Cooperation Under Uncertainty. In *IEEE-ICRA*. 7330–7337.

[34] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael H Bowling. 2008. Dyna-style planning with linear function approximation and prioritized sweeping. In *UAI*.

[35] Ioannis Tsamardinos. 2002. A Probabilistic Approach to Robust Execution of Temporal Plans with Uncertainty. In *SETN*.

[36] Yi Wan, Muhammad Zaheer, Adam White, Martha White, and Richard S Sutton. 2019. Planning with expectation models. In *IJCAI*. 3649–3655.

[37] Daniel S Weld. 2005. Concurrent probabilistic temporal planning. In *ICAPS*. 120–129.

[38] Zhao Xinyi, Zong Qun, Tian Bailing, Zhang Boyuan, and You Ming. 2019. Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning. *Aerospace Science and Technology* 92 (2019), 588 – 594.

[39] Sung Wook Yoon, Alan Fern, and Robert Givan. 2007. FF-Replan: A Baseline for Probabilistic Planning.. In *ICAPS*, Vol. 7. 352–359.

[40] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. MOPO: Model-based Offline Policy Optimization. *arXiv preprint arXiv:2005.13239* (2020).

[41] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2019. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635* (2019).

[42] Shiqi Zhang, Yuqian Jiang, Guni Sharon, and Peter Stone. 2017. Multirobot Symbolic Planning under Temporal Uncertainty. In *AAMAS*. 501–510.