Multi-Vehicle Temporal Planning for Underwater Applications

Yaniel Carreno and **Yvan Petillot** and **Ronald P. A. Petrick**
Edinburgh Centre for Robotics
Robotarium West, Heriot-Watt University
Riccarton, Edinburgh EH14 4AS
Scotland, United Kingdom
{yc66, y.r.petillot, r.petrick}@hw.ac.uk

Abstract

This paper investigates the application of temporal planning to multiple robots in marine applications. We analyse plan concurrency, makespan and plan generation time in the multi-robot problem and propose a new schema called GA+TP which combines a Goal Allocation (GA) strategy and Temporal Planning (TP) to improve plan quality in multi-vehicle missions. We implement a new planning domain, motivated by realistic scenarios in maritime environments, considering dynamic reallocation of the refuelling point and the implementation of tasks with multi-domain robots. The ROSPlan framework, the UUV-Simulator and the surface vehicle WAM-V are integrated to implement effective missions and analyse plan execution using ROS and Gazebo. Experiments have shown the dynamic refuelling point method enhances the implementation of long-term missions with small robot fleets. GA+TP demonstrates robustness to domain changes and improved plan quality while significantly reducing planning time for the multi-agent problem.

Introduction

Autonomous Underwater Vehicle (AUV) technology is now routinely used for small-scale surveys of the subsea world. One of the key drivers of current research is therefore the introduction of AUVs into more challenging missions, with much greater complexity, longer execution times and higher risks of failures, especially in unknown dynamic environments. Recent works (Landa-Torres et al. 2017; Zhang et al. 2017) have favoured the approach of using multiple platforms which collaborate to achieve a common set of goals. Multi-vehicle systems are more robust to individual platform failures and have advantages in terms of their overall system capabilities and mission execution performance. The difficulty with such an approach is that it must also be supported by robust planning, coordination, and execution tools in order to increase its optimal performance. As a result, most autonomous missions successfully implemented in the maritime domain rely on a single platform (Cashmore et al. 2014; McGann et al. 2008; Cashmore et al. 2015) or use very simple coordination schemes between platforms (Chrpa et al. 2015; Marques et al. 2017), limiting the effectiveness and benefits of full multi-robot solutions. Planning is particularly promising for maritime robotics since it offers techniques for managing typical problems that arise in the underwater domain, such as limited power and communication bandwidth, navigational uncertainty, and perception noise. However, whilst many planners have been proposed (Coles et al. 2010; Benton, Coles, and Coles 2012; Gerevini and Long 2006; Eyerich, Mattmüller, and Röger 2012), identifying a single "optimal" problem solver is a difficult task. The differences in representational power, plan generation efficiency, highly constrained domain definition, and the use of multiple robots limit the planners' performance, often resulting in a severe lack in plan quality.

In this paper, we explore the application of planning techniques for multi-robot fleets, to enhance long-term autonomy and general fleet robustness. The main contribution of this work is a new strategy, called the GA+TP, that combines a Goal Allocation algorithm (GA) and Temporal Planning (TP) to improve plan quality for temporal multi-robot tasks. We implement this approach in the context of a new realistic planning domain with high levels of expressiveness that supports the execution of AUV missions. We also define a method to reduce the plan generation failures using dynamic reallocation of the refuelling point. Finally, we integrate ROSPlan, a realistic AUV and Autonomous Surface Vehicle (ASV) simulators to execute multi-domain coordinated missions.

Related Work

Automated planning is the process of reasoning about the actions needed to achieve a set of goals. Planning usually involves explicit representations of time to implement complex missions with multiple robots. The agents must work concurrently and execute actions with different time slots which force the use of temporal references for solving the problem. Temporal planning problems can be addressed using PDDL2.1 (Fox and Long 2003), or the latest extensions of the standard Planning Domain Definition Language (PDDL) (McDermott et al. 1998), which adds support for temporal planning through additional language constructs. A number of temporal planners support PDDL2.1, such as Temporal Fast Downward (TFD) (Eyerich, Mattmüller, and Röger 2012), UPMurphi (Della Penna, Magazzeni, and Mercorio 2012), SGPlan (Hsu and Wah 2008), and LPG-TD (Gerevini and Long 2006). However, these planners face problems of scalability, concurrency, or issues with actions' time slot allocation which have limited their introduction in marine applications.

Two temporal planners are particularly promising for underwater applications: the Forward-Chaining Partial-Order Planning (POPF) (Coles et al. 2010) and the Optimizing Preferences and TIme-dependent Costs (OPTIC) (Benton, Coles, and Coles 2012). These planners support numerical fluents, concurrency and exogenous events. POPF is characterised by its ability to find plans with low *makespan*—the time that elapses from the start of plan implementation to the end—while OPTIC's main application is in problems where preferences or time-dependent goal costs define the plan cost. POPF and OPTIC have been successfully tested in real missions (Cashmore et al. 2014; 2015; Bernardini, Fox, and Long 2017). However, little work (Tran et al. 2017; Hertle and Nebel 2018) addresses the multi-agent problem using these planners. In general, these approaches present limited domain complexity which prevent the evaluation of the planners' performance in highly constrained domains.

Multi-Agent Planning (MAP) has also been addressed in (Crosby, Rovatsos, and Petrick 2013; Kvarnström 2011; Muise, Lipovetzky, and Ramirez 2015; Sreedharan, Zhang, and Kambhampati 2015), which apply a distributed problem-solving design to substitute the classical single-agent planning paradigm. However, these solvers do not support tasks with advanced requirements such as temporal constraints (Torreño et al. 2018), which make them less attractive for the implementation of complex missions with concurrent actions. Although several approaches (Largouët, Krichen, and Zhao 2016; Nikou et al. 2018; Schillinger, Bürger, and Dimarogonas 2017) consider mission timing constraints to solve multi-agent problems, they use specific language representations or provide a solution to particular planning problems (e.g., path planning optimisation) in the global plan which limit the approach generalisation.

Other works (Ponda et al. 2010; Nunes and Gini 2015; Nunes, McIntire, and Gini 2017) have explored temporal constraints to task allocation using auction-based approaches. (Ponda et al. 2010) proposes a multi-item auction strategy which replaces the auctioneer for distributed consensus phases and considers tasks with time windows. (Nunes and Gini 2015; Nunes, McIntire, and Gini 2017) present the Temporal Sequential Single-Item auction (TeSSI) algorithm for tasks with temporal constraints. The strategy allocates tasks with time windows to cooperative robots. However, its optimality is subject to the number of robots and regions to explore. In addition, TeSSI assumes the robots can execute all the actions indefinitely and there is not further analysis considering robot resources (e.g., energy level) which can affect the sequence of actions defined.

Recent research (Landa-Torres et al. 2017; Zhang et al. 2017) has considered MAP problems in the context of AUV fleets. In particular, the ScottyActivity planner (Fernandez-Gonzalez, Williams, and Karpas 2018) introduces levels of coordination in maritime applications and provides good results controlling continuous variables on the domain. The strategy accomplishes task planning and trajectory optimisation in long horizons for multiple robots. However, the approach does not guarantee optimal action sequences and the small number of robots in the mission prevents a scalability analysis. Few works directly consider temporal planning for multi-agent applications. In (Crosby and Petrick 2014), planning problems are modelled using standard PDDL and then translated to a temporal planning model for generating and distributing plans to individual robots. (Hertle and Nebel 2018) implement a strategy based on an auction algorithm and temporal planning which reduces complexity during the planning process. The approach presents limitations around the auction time and does not support domains which requires concurrency. (Schneider, Sklar, and Parsons 2017) also considers auction mechanism. However, this work does not analyse the coordination of multiple robots and the way the tasks are clustered. Finally, (Buksz et al. 2018) proposed an strategy based on clustering to allocate the mission goals. Nevertheless, the method does not discuss robot capabilities to implement the allocation which is consider in our work.

While there are several examples of architectures which support plan execution in robotic systems (Ingrand et al. 1996; Chanel, Lesire, and Teichteil-Königsbuch 2014; Muscettola et al. 2002; Hofmann et al. 2016; Dornhege et al. 2012), including planning for AUV missions (McGann et al. 2008; Rajan, Py, and Barreiro 2013; Marques et al. 2017), many approaches employ domain-specific or inflexible language specifications and methodologies, which lead to bespoke implementations not easily ported to new applications. Additionally, the complexity of the planning systems and the robotic architectures entails most of these frameworks to experience difficulties connecting high-level symbolic plan generation with the actuator level.

A notable exception is ROSPlan (Cashmore et al. 2015) which connects the widely used Robot Operating System (ROS) (Quigley et al. 2009) and PDDL2.1. ROSPlan allows different task planners to be embedded in a modular architecture, making it suitable for testing plan feasibility and quality while varying the underlying planning approach. ROSPlan framework solves plans with concurrent actions. Prior implementations using ROSPlan in maritime contexts have only explored the performance of a single agent (Cashmore et al. 2015; 2018), often in simple domains, and have therefore not evaluated ROSPlan's ability to deal with complex temporal and resource constrained missions. Our paper explores the advantages of combining multi-agent temporal planning and goal allocation based on goal positions to solve complex domains using ROSPlan.

Domain Definition

In this work, we use an offshore subsea oil platform scenario which is segmented in several regions for modelling real-world multi-robot problems. This extreme environment reduces the possibility of human intervention; therefore, the platform's supervision and maintenance are autonomously executed by a fleet of surface and underwater vehicles. Figure 1 shows a general map representation of the oil platform, with the **Start & Destination** points for the surface and underwater vehicles, the **operation regions**, which contain local refuelling points (**r-point**) and multiple oil rigs, and the **Transmission Centre (TC)**, which is employed as the point for data transmission. Our planning domain includes a novel combination of properties such as robot capabilities and resources, concurrent actions and temporal constraints which

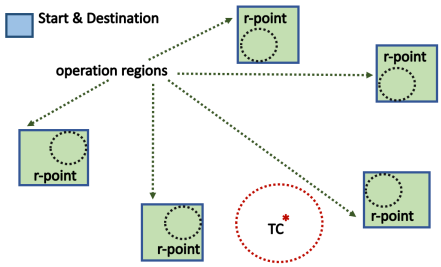

Figure 1: A depiction of the segmented oil platform scenario. This scenario exhibits a fixed transmission centre (TC), multiple deploy regions and refuelling points.

enhance the implementation of long-term missions. This approach introduces a benchmark planning domain representation for multi-robot systems in the marine environment which allows the dynamic allocation of the refuelling point depending on the robot position. Moreover, we consider the optimal implementation of missions in regions separated by long distances, and multi-domain robot coordination based on temporal planning.

The domain contains tasks associated with multiple types of sensors (e.g., camera, sonar, and Doppler Velocity Log (DVL) sensors), enabling important mission scenarios. The mission goals for the AUVs include: exploration of the oil platform, valve state inspection, sensor data acquisition, and target or failure detection. The surface vehicle supports the refuelling strategy, moving to the recharging points every time the robots have low battery level. We here consider a set of homogeneous AUVs and a single surface vehicle. While the domains can be modelled at several levels of abstraction, here we focus on real world implementations and define highly constrained actions to avoid high-level coarse plan generation and handle any discrepancies in the model and the real (or simulated) world during plan execution.

Our domain[1] defines multiple ***types***[2] called `rexrov`, `robot-surface`, `waypoint`, `observation_point`, and `sensors`. Waypoints situate the vehicle in specific sections of the arena, and the robot executes actions related with the sensor system in the observation points. The `?wp` and `?o` instances represent the vehicle's position, part of the robot state supporting exploration. These instances have a fixed location defined by the domain designer. The `?s` describes the robot's sensor system. The domain also includes ***static facts***, ***fluents***, ***functions*** and ***actions*** which defines the system. The capabilities of the robots are captured by the following domain actions:

- `asv_wamv_navigate(?asv_wamv, ?from, ?to)`: a durative action which moves the robot on the surface `?asv_wamv` from waypoint `?from` to `?to`. Action duration depends on the robot's average speed and the distance between the waypoints which is calculated using the Euclidean norm.
- `rexrov_navigate(?rov, ?from, ?to)`: a durative action which is similar to `asv_wamv_navigate` using AUV robot `?rov`.
- `rexrov_explore_ob(?rov, ?wp, ?o)`: a durative action where the robot `?rov` explores around the observation point `?o`, describing a circular trajectory which improves the robot localisation and acquires a global view of the oil rigs.
- `rexrov_refuelling(?rov, ?asv_wamv, ?from, ?to)`: a durative action for robot `?rov` battery recharging. The robot moves from waypoint `?from` to waypoint `?to`, where it stays during recharging time. The action implementation requires as a precondition to deploy the surface vehicle `?asv_wamv` to the point `?to` using the `asv_wamv_navigate` action. We define the `?to` position considering the dynamic refuelling point allocation principle, which will be described in this section. The implementation introduces levels of coordination between the surface and the underwater robots.
- `rexrov_undock(?rov, ?wp)`: durative action to disconnect the robot `?rov` from the refuelling point `?wp`.
- `rexrov_send_data(?rov, ?wp)`: a durative action which moves the robot `?rov` to the waypoint `?wp` (TC) to transmit information considering as a precondition the amount of data stored.
- `rexrov_take -image_data/-sample_data(?rov, ?o, ?wp, ?s)`: durative actions which enable robot `?rov`'s sensors `?s` to capture images or read and store sensor data on the observation point `?o`.
- `rexrov_target_id(?rov, ?o, ?wp, ?s)`: a durative action which enables robot `?rov`'s sensors `?s` to identify pipeline failures, valve states and take pictures of the targets.
- `rexrov_target_approach(?rov, ?o, ?wp)`: a durative action which makes robot `?rov` to implement a maneuver approach to the target. The effect of implementing this action condition the execution of the `rexrov_target_id` action.

Domain Constraints: We claim the domain is directly transferable to the maritime environment and the resulting plans follow representative constraints of real AUV missions. In particular, our domain contains: (i) time constraints, (ii) capability constraints, (iii) resource constraints (energy or storage capacity), (iv) communication constraints, and (v) robot availability constraints; which allow the implementation of long-term missions and prevent mission execution failures. The domain constraints are explicitly represented in the PDDL action definition. For temporal constraints, action durations depend on `distance ?from ?to` and the robot (`?rov` or `?asv_wamv`) average forward speed. The time duration for static actions is decided by the domain designer. In terms of capability constraints, for each action associated with specific sensors,

[1]Domain and problem instances are available at ICAPS-2019 repository on `https://github.com/MA-TemporalP`.

[2]In the following, we will use `?rov` to denote a parameter of type `rexrov`, `?asv_wamv` a parameter of type `robot-surface`, `?wp` a parameter of type `waypoint`, `?o` a parameter of type `observation_point`, and `?s` a parameter of type `sensors`.

a new `rexrov_capable ?rov` predicate is added to the list of predicates. Then, each action that appears gains a precondition `rexrov_capable ?rov`. This ensures the agents performing the action have the appropriate sensors for the implementation. Resource constraints take into account a robot variables state. For instance, the preconditions for the implementation of the action `rexrov_refuelling` considers individually the actual energy level and robot's energy consumption rate. The communication constraints are associated with the action `rexrov_send_data` which is conditioned by the robot `rexrov_ image_capacity/ data_capacity`, and force the robot to move to the surface to transmit information. In terms of availability, the execution of an action prevents the use of the same robot for another action, which reduces action conflicts. Moreover, we define some action dependencies, such as the implementation of `rexrov_refuelling` which is conditioned on the execution of `asv_wamv_navigate`, and `rexrov_target_id` which depends on the implementation of the `rexrov_target_approach` action.

Refuelling Point and Coordinated Actions: Our domain establishes the dynamic allocation of the refuelling point by considering the robot's actual position. We define a set of fixed points or possible refuelling points $DP = \{dp_1, dp_2, ..., dp_n\}$ in critical areas considering the number of regions in the environment (typically one refuelling point per region). Subsequently, we implemented an algorithm which calculates the distance between the waypoints in set $WP = \{wp_1, wp_2, ..., wp_m\}$ and all "possible" docking points. Our approach uses the distance calculation to determine the closest docking point. The method divides the WP set into multiple subsets associated with individual docking points. The results of the algorithm is transformed into new predicates `close_to ?wp ?wp` which are added to the set of predicates. The first `?wp` represents a waypoint from the WP set (actual robot position) and the second `?wp` is the closest docking point. The `rexrov_refuelling` action implementation is constrained by a `close_to ?wp ?wp` precondition which will dynamically define the refuelling point depending on the position of the vehicle. This prunes the search-space, resulting in better planner performance. We also introduce actions with multiple robots (e.g., `rexrov_refuelling`) which increases domain complexity due to the action allocation in the global plan depends on the availability of multiple agents. However, we can evaluate the capacity of temporal planning to deal with coordinated actions. Furthermore, it enhances the realistic representation of the domain and the implementation of long-term missions considering the refuelling method. Finally, the introduction of the robot-surface can support further actions in the domain such as periodic AUVs hardware checks and AUVs recovery after mission completion.

Multi-Vehicle Implementation

Temporal planning is capable of dealing with multi-agent planning problems since time is modelled explicitly: individual actions for different robots can be scheduled and executed in parallel. However, the highly constrained actions

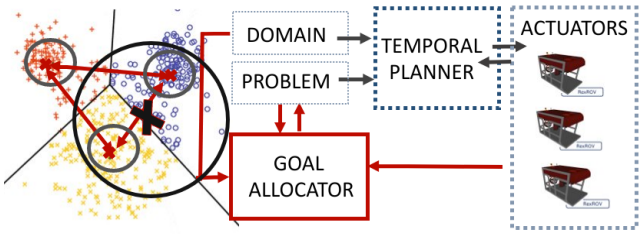

Figure 2: General architecture of the GA+TP strategy which considers robot capabilities and goal positions. GA decomposes the global goals into simpler goal sets for each robot and modify the problem file. The GA strategy uses clustering methods for goal allocation. The temporal planner uses these goal sets to generate plans.

in the domain and the long distances between the oil platform regions limit the planners' performance. In this work, we aim to reduce the complexity of the planning process, assisting temporal planners to achieve a large number of valid plans, improve makespan, planning time, and goal distribution. We define the GA+TP approach based on the combination of a Goal Allocation (GA) strategy and Temporal Planning (TP). The method decomposes a set of mission goals into multiple simpler goal sets for each robot.

Goal Allocation: Figure 2 shows the general architecture of the GA+TP approach, which mainly distributes goals based on goal positions to reduce the distance travelled and the energy consumption. GA takes information from the domain and problem files and modifies the problem by adding new predicates which constraints the goals' allocation. GA takes a set of unallocated goals $G = \{g_1, g_2, ..., g_n\}$, weights the distance between them and decomposes the goals geographically by applying *k-means* clustering method (Hartigan and Wong 1979). This results in goals' subsets $S^G = \{s_1, s_2, ..., s_t\}$ related with individual regions $R = \{r_1, r_2, ..., r_t\}$ with known centres $C = \{c_1, c_2, ..., c_t\}$; where $s_n = \{\}$ contains multiple goals defined in the goal set G. The algorithm considers the robot capabilities to determine the goal's subsets. We define the *capability_condition* which considers the characteristics of the goals and robots.

Definition 1: Given a goal subset s_n, the *capability_condition* is a tuple $cap(s_n, rob) := \langle GC, RC \rangle$, where rob is the set of robots in the mission; GC is the set of capabilities needed to implement the goals in the sub-set and RC the capabilities of the robots in the mission. The condition which must be hold is RC contains all GC.

The strategy (Algorithm 1) takes the set of available robots (rob) and checks the RC and GC. If R is equal rob and RC are equal to GC, the goals of each region are allocated to individual robots. Alternatively, if RC and GC are different, the strategy (i) checks the number of robots able to implement the goals (n_c), (ii) calculates the distance between the centroids (d_{nc}), and (iii) applies clustering again, considering n_c and d_{nc}. As a result, the algorithm will re-

Algorithm 1 Goals Allocator Strategy.

Input: set of robots rob, set of unallocated goals G and goals coordinates C_G.
Output: add a set of predicates to the PDDL problem file.

$R, c = k\text{-}means(C_G)$;
while $G \neq \emptyset$ **do**
  for $r \in rob$ **do**
    $RC, GC = check_{cap}(r, G)$;
  end for
  if $R == rob$ **then**
    if $RC == GC$ **then**
      $G \rightarrow rob$
    else
      $n_c = check_{nc}(GC, RC)$;
      $d_{nc} = distance(c)$;
      $R = k\text{-}means(c, d_{nc}, n_c)$;
    end if
  else
    $d_c = distance(c)$;
    $R = k\text{-}means(c, d_c, rob)$;
  end if
end while

turn R equal to the number of robots with the capabilities, and the allocation is complete. For this case, it is likely that some robots implement actions in multiple segments of the oil platform. Otherwise, if R differs from rob the algorithm calculates the distance between the centroids d_c and recalculates R based on d_c and rob, which solves the goal allocation problem. If we have more robots than regions the algorithm will not consider the extra robots. This technique situates GA as a global decision maker, which relaxes the complexity of the problem and simplifies the temporal planner's work, which now examines a smaller number of constraints for the plan solution. The strategy also reduces the risk of collisions, increases the area covered by the agents, relaxes plan complexity, and helps to guarantee optimality by forcing the robots to reduce distance travelled and mission time.

Temporal Planning: Once the goal allocation is finished, the temporal planner takes the individual goal sets and generates the plan considering temporal constraints. GA populates the list of predicates with new predicates `rexrov_can_act ?rov ?wp`, which modify the original problem allowing the completion of specific goals in the `?wp` for individual AUVs (`?rov`). All the domain actions are conditioned for a precondition `rexrov_can_act ?rov ?wp`, which ensures the vehicle implementing the action has "permission" to execute tasks in the region. The introduction of the action precondition reduces the state-space and relax the plan complexity through the addition of constraints associated with the geographic goals' distribution.

Simulation Environment

The simulation environment is implemented in ROS and Gazebo, and we integrate the ROS packages ROSPlan (Cashmore et al. 2015), the UUV-Simulator (Manhães et al. 2016) and the Virtual RobotX (VRX), which describes the surface vehicle WAM-V (Sarda et al. 2016). The simulation world is customised considering the models from Gazebo and the UUV-Simulator libraries. We implement planning strategies with ROS in the underwater domain to evaluate plan feasibility using maritime platforms. The case study in Figure 3 shows a Gazebo environment (right) and the RViz (left) for a robot fleet (underwater and surface vehicles) in an oil platform with three regions. Each region contains two oil rigs which are the main targets for mission implementation.

ROS nodes were written for actions defined in the domain. This serves as an interface between high-level actions from task planning and low-level robot controls. The vehicles can interact with the environment in multiple ways depending on their capabilities. The robots were instructed to explore certain numbers of points in the environment to collect data, detect valve states, and take pictures, among other actions defined in the domain. The implementation of the navigation and the circular exploration strategies are based on a trajectory-generator function. For navigation, the function calculates a trajectory based on the shortest path between the robot's actual position and the goal's coordinates using point interpolation. For circular exploration, the function takes into account the radius of the circumference and its centre producing a circular path.

Problem	Object Instances	No. Goals
`problem_1`	71	48
`problem_2`	78	60
`problem_3`	99	72
`problem_4`	108	90
`problem_5`	119	96
`problem_6`	127	102

Table 1: Number of object instances and goals distributed in each problem for the oil platform scenario domain. The evaluation considers 6 problem instances with different levels of complexity based on the domain actions.

Evaluation

We now evaluate the performance of our approach, comparing the results with the outcomes of well established benchmark planners: POPF, OPTIC-NP (OPTIC-Non Preferences) with plan utility based on `total-distance` and makespan; and OPTIC-P (OPTIC-Preferences) with plan utility based on `total-distance`, makespan, and preferences that force the robots to recharge at least one time during the mission. Our evaluation considers the oil platform with three regions described in Figure 3. In this work, we evaluate the performance of GA in a homogeneous robot set. However, the strategy can be applied to heterogeneous vehicles considering the system's capability analysis is part of the GA method. All experiments were run on a machine with a 4GHz processor, limiting the planner to 30 minutes of CPU for plan generation and 8GB of memory consumption. The set of test problems considers 6 problem instances of increasing difficulty. Table 1 shows the number of object

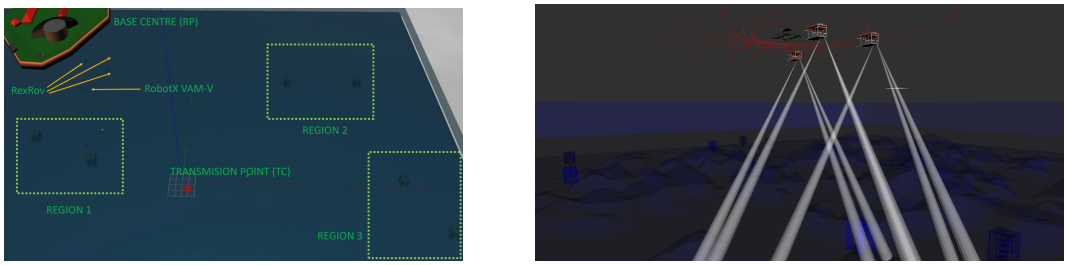

Figure 3: Gazebo simulation environment (left) and RViz (right) for three underwater robots and a surface vehicle in the oil platform scenario with three regions. The robots are in the initial position waiting for the action dispatch to execute a surveillance mission in the domain.

Problems	planning-time (OPTIC-NP)							
	1-robot		2-robots		3-robots		4-robots	
	d	non-d	d	non-d	d	non-d	d	non-d
1	109.5	–	105.3	–	323.2	–	88.4	83.2
2	141.5	–	139.3	–	573.2	–	56.1	47.2
3	358.6	–	920.4	–	633.7	–	138.8	180.5
4	1136.4	–	–	–	–	–	502.4	197.2
5	–	–	–	–	–	–	1315.7	–
6	–	–	–	–	–	–	–	115.8
	planning-time (OPTIC-P)							
1	289.7	–	186.7	–	412.1	567.8	187.3	145.3
2	310.5	–	201.5	–	580.4	582.6	258.3	197.6
3	389.2	–	1301.1	–	503.4	–	358.2	342.7
4	1507.8	–	1420.5	–	606.4	–	401.7	476.7
5	–	–	–	–	–	–	589.1	603.7
6	–	–	–	–	–	–	–	–

Table 2: Planning time (sec) analysis for dynamic and non-dynamic refuelling point allocation using OPTIC-NP and OPTIC-P. Results for the dynamic refuelling point (d) strategy show improvements in plan sovability respect to the non-dynamic refuelling point (non-d) approach.

instances and goals in each problem. Each problem is attempted 30 times, and we measured the average of the simulation results to avoid occasional jitter.

Domain and Strategy Evaluation

We start the evaluation analysing the effect of the dynamic refuelling point allocation on plan solvability and plan generation time. For non-dynamic refuelling points, we situate the static waypoint in the TC's coordinates. Table 2 shows the planning time performance for different robot sets with OPTIC-NP and OPTIC-P. Results reveal the dynamic reallocation of the refuelling point increases the number of problems solved, particularly for robot sets smaller than the number of regions. The introduction of multiple points for recharging relaxes the plan's complexity due to a minimisation of the energy and distance constraints. Therefore, the concept of an "optimal" refuelling point based on the robot position enhances the implementation of long-term missions in complex environments.

We also analyse the approach's scalabilty to larger robot sets (1 to 30 robots) and regions (1 to 30 regions). Figure 4 shows the curves for dynamic and non-dynamic refuelling points based on the number of plans solved, using OPTIC-NP. We assume the same tasks are implemented in all the regions. The dynamic refuelling point allows the actions' execution in a large number of regions using small robot fleets compared to the non-dynamic approach. However, the curves converge to the same values for large robot sets which demonstrates the dynamic strategy is more effective when the number of regions is greater than the robots' figures. Therefore, this domain representation adds a mechanism to solve large plans with a reduced number of resources (robots). The dynamic refuelling point strategy will be used in all the experiments.

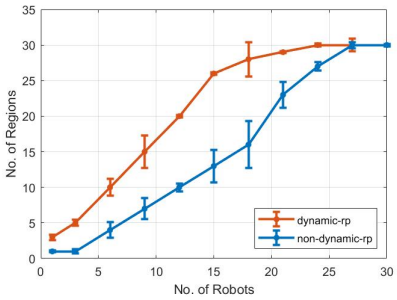

Figure 4: Number of regions explored during a mission for a robot fleet using OPTIC-NP for non-dynamic and dynamic refuelling point allocation. Dynamic refuelling point allocation improves the robot system's efficiency, implementing actions in large numbers of regions with a small group of vehicles.

The plan performance of the GA+TP strategy is also compared with POPF, OPTIC-NP and OPTIC-P in terms of plan solvability, makespan, and plan generation time. We first evaluated the planners' ability to cope with changes in the domain's numeric variables. Table 3 shows the plan makespan (min) and planning time (sec) for different values of the domain variables `consumption_rate`, `data_capacity` and `avg_forward_speed` using a set of two robots. For `consumption_rate` equal to 0.1 and 0.5 the `avg_forward_speed` is 0.33 and the `data_capacity` is 5. For `avg_forward_speed` equal to 0.2 and `data_capacity` of 3, the `consumption_rate` is 0.1.

The results reveal the sensitivity of the benchmark planners towards changes in the numeric constraints. In con-

Problems	consumption_rate = 0.1							
	POPF		OPTIC-NP		OPTIC-P		GA+TP	
	time	span	time	span	time	span	time	span
1	197.1	348	105.3	296	186.7	313	**68.3**	254
2	297.4	272	139.3	180	201.5	222	**87.3**	255
3	–	–	920.4	243	1301.1	317	**216.6**	287
4	–	–	–	–	1420.5	446	**419.1**	456
5	–	–	–	–	–	–	**537.6**	315
6	–	–	–	–	–	–	**547.9**	329
	data_capacity = 3							
1	197.1	348	105.3	296	186.7	313	**68.3**	254
2	297.4	272	139.3	180	201.5	222	**87.3**	255
3	–	–	1892.7	466	2004.4	793	**305.4**	507
4	–	–	–	–	–	–	**529.1**	751
5	–	–	–	–	–	–	**809.6**	762
6	–	–	–	–	–	–	**863.1**	679

Problems	consumption_rate = 0.5							
	POPF		OPTIC-NP		OPTIC-P		GA+TP	
	time	span	time	span	time	span	time	span
1	957.8	769	429.4	594	619.1	852	**138.4**	251
2	1042.3	706	–	–	1154.6	1063	**157.6**	725
3	–	–	984.5	644	1543.2	875	**216.6**	689
4	–	–	–	–	–	–	**759.2**	849
5	–	–	–	–	–	–	**707.1**	789
6	–	–	–	–	–	–	–	–
	avg_forward_speed = 0.20							
1	227.1	468	325.6	497	201.8	543	**138.7**	384
2	426.3	579	368.3	410	371.5	452	**93.4**	392
3	–	–	1130.5	243	1651.2	435	**348.6**	428
4	–	–	–	–	1652.5	646	**484.2**	478
5	–	–	–	–	–	–	**625.5**	495
6	–	–	–	–	–	–	**657.4**	518

Table 3: Plan generation time (sec) and makespan (min) analysis for different values of the `consumption_rate` (top left, top right), `data_capacity` (bottom left) and `avg_forward_speed` (bottom right) in 6 problem instances using temporal planners. Simulations were run for 30 min and show the results of the first solvable plan for 2 robot fleets. GA+TP shows the best performance solving the majority of the problems despite the variable changes.

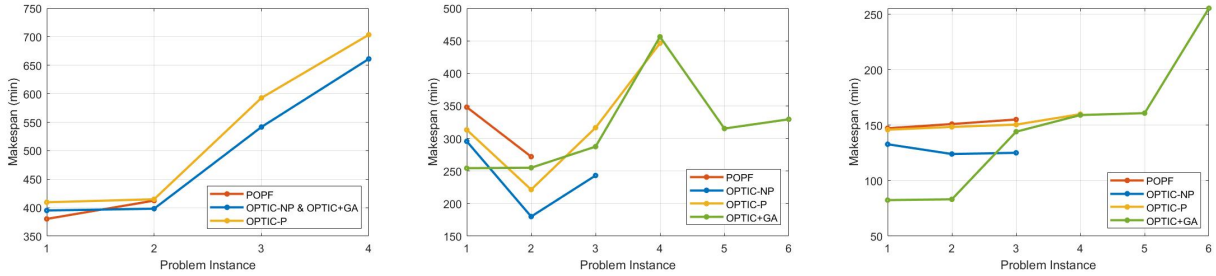

Figure 5: Plan makespan analysis for POPF, OPTIC-NP, OPTIC-P and GA+TP in the oil platform domain using a single robot (left), two robot (middle) and three robot fleet (right). Simulations were run for 30 min and show the first solvable plan makespan for 6 problem instances. For the single robot implementation, OPTIC-NP and GA+TP behave equally. For multi-vehicle implementations, GA+TP shows robustness solving all the problems independently of the number of robots, in shorter time periods.

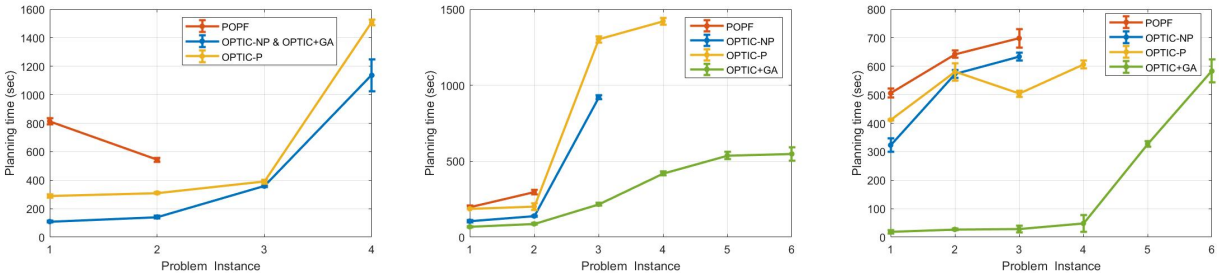

Figure 6: Plan generation time analysis for POPF, OPTIC-NP, OPTIC-P and GA+TP in the oil platform domain using a single robot (left), two robot (middle) and three robot fleet (right). Simulations were run for 30 min and show the first solvable plan makespan for 6 problem instances. OPTIC shows better performance than POPF generating the plan in shorter time periods. For multi-vehicle implementations, GA+TP substantially reduces the planning time.

trast to this, GA+TP generates solvable plans for almost all the problem instances, showing a robust performance despite the domain's changes. Moreover, GA+TP achieves the smallest plan generation times as a result of the plan relaxation obtained with GA. Results indicate the use of GA+TP allows the problem to be scaled to multiple robots with different domain properties. We notice the variations of `avg_forward_speed` change the planning time and makespan but do not affect plan solvability since the variable only constrains action duration. POPF and OPTIC-NP present the poorest performances resulting from the goal complexity, action scheduling and the effect of numeric con-

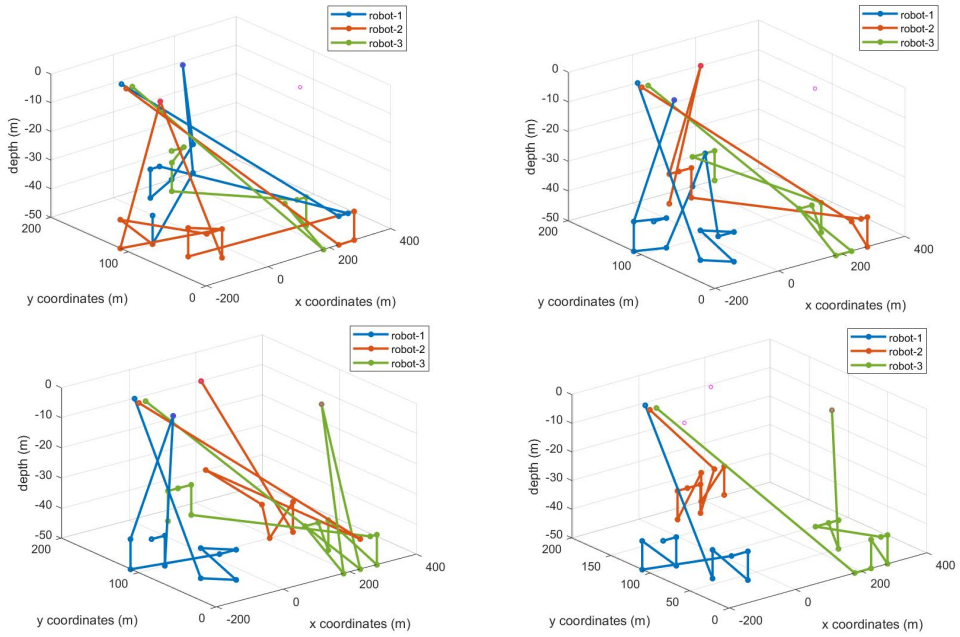

Figure 7: Spatial representation of goal distribution using POPF (top left), OPTIC-NP (top right), OPTIC-P (bottom left) and GA+TP (bottom right). Simulations evaluate `problem_3` results for a set of 3 robots. The Goal Allocator's implementation reduces the complexity of the mission strategy and distributes the robots in an optimal manner considering the position of the waypoints. GA+TP optimises the energy consumption and distance travelled.

straints during plan generation.

In addition, we evaluated the makespan and planning time performance considering different robot sets. We analyse the first solvable plan due to the plan optimisation takes considerably amounts of time (particularly with OPTIC) for multi-agent systems. Figure 5 shows plan makespan performance for a single robot (left), two robot (middle) and three robot (right) fleets. For single agent implementations, the results of OPTIC-NP and GA+TP are equal, since the GA strategy will only define one region. The benchmark planners cannot generate a solvable plan for large problems (`problem_5` and `problem_6`) which supports our claim to introduce multiple robots for the mission implementation. POPF presents the poorest results generating a solvable plan for just two problems. The planner's sensitivity to the ordering of object instances makes the makespan performance highly dependent on the goal position in the problem instance. In contrast, OPTIC is not affected by goal-ordering phenomena which provides freedom for action allocation in the mission schedule. OPTIC's first plan offers better solutions to optimise mission resources and reduce the number of intermediate actions, taking into account the distance travelled for the robots. We use OPTIC for the GA+TP approach since POPF is (i) affected by numeric changes in the domain definition, and (ii) shows errors in the temporal placement of preconditions. For two and three robot sets, GA+TP is the only approach that generates solvable plans for all the problem instances. In addition, we notice a substantial reduction in the mission times, particularly for three vehicle implementations. These results provide evidence that the GA mechanism improves the performance of the temporal planners in these situations.

Plan generation time influences the capacity of the robotic system to react optimally during time sensitive tasks such as underwater missions. Figure 6 shows the plan generation time analysis for the robot fleets mentioned. For a single robot, OPTIC-NP and OPTIC-P produce better results than POPF, generating the first solvable plan in shorter time periods. The preferences considered for OPTIC-P delay the plan generation which explains the time differences with OPTIC-NP. The plots indicate that planning time does not necessarily increase proportionally with problem instance complexity. This phenomena suggests that planning time depends on the actions' interaction with the problem instances and domain constraints rather than the number of goals achieved. For two and three robot sets, GA+TP generates a first solvable plan in the shortest time period, improving the optimality of the mission implementation. The results clearly demonstrate the GA strategy improves plan generation performance by acting as a decision maker which reduces the complexity of the problem and facilitates the temporal planner's work. In addition, the simulations illustrate the advantage of using GA+TP for multi-vehicle strategies, which provide a tool for temporal planning to optimise complex mission planning with large numbers of goals and constraints.

The goal distribution implementation represents one of the key improvements of the GA+TP approach with respect to the benchmark planners. Figure 7 shows the sequence of goals implemented for three robot fleets using `problem_3`.

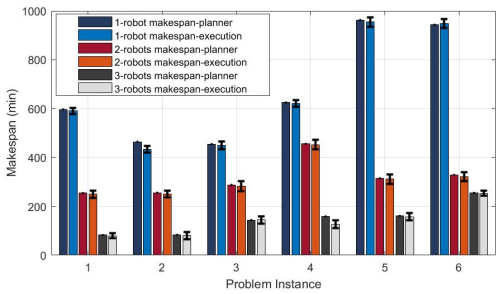

Figure 8: Bar chart comparing the plan execution time and the mission time originally defined by GA+TP. Results for plan makespan and real execution times are significantly similar which demonstrate the domain is able to represent real underwater missions.

The points represent the spatial positions of the goals and the refuelling points (magenta). The results demonstrate GA+TP is able to distribute the robots in different regions which reduces the possibility of robot collisions, the total distance travelled and the energy consumed. POPF generates a plan with all the robots implementing actions in the three regions, which affects the general cost parameters (energy consumption and total-distance). OPTIC-NP and OPTIC-P do not produce good solutions; in both cases the robots implement actions in two different regions. Results show that combining the advantages of temporal planning and GA can substantially improve the general performance of the system and optimise the use of robots and resources.

Plan Execution

This section analyses the performance of the planning strategy based on mission execution. We use the ROSPlan framework to dispatch actions to the underwater and surface vehicles. Each robot run its own action interface/implementation on-board and present individual nodes for path planning and sensor parsing. The vehicles are subscribed to PDDL messages and are able to execute them independently. ROSPlan typically re-plans several times during a mission run, justifying the need for fast planning time. The primary causes for re-planning in our experiments are the errors in robot localisation. When re-planning is triggered the currently executing actions are allowed to be completed. We showed the shortest planning time is achieved with the GA+TP approach. Therefore, we base our analysis of the planning strategy on GA+TP. The experiment establishes the main differences between the makespan obtained for the temporal planner and the real mission execution time.

Figure 8 shows a bar chart which compares mission time results. Simulations demonstrate the plan execution times are similar to those generated for GA+TP, which supports our claim that the domain definition is applicable to real underwater missions. These results are also supported by the fact the robot can recharge in different parts of the environment which reduces the risk of failures. In addition, the time duration for all navigation actions consider an *time-addition-factor* which reduces the system faults. However, we are also assuming a constant sea current and no additional disturbances such as body wrench. In addition, the system is running in a single master which limit the application. Nevertheless, our architecture can be easily segmented in multiple master (one per robot with different ROS core, ROSPlan system, and planner) keeping our approach method. Future work intends to analyse the effect of environmental disturbances on plan execution, providing additional recovery mechanisms to support planning.

Conclusions

In this paper we investigated planning for AUV missions with multiple vehicles, complex models of the environment, and missions in highly constrained domains. While we recognise that expressing domains precisely tends to over-complicate the planning problem and increase planning time, we focus on accuracy. We presented a new approach, GA+TP, which improves the performance of well established temporal planners such as POPF and OPTIC by increasing system robustness despite domain complexity, reducing the planning times and making optimal allocations of the mission goals easier. We also introduced the dynamic reallocation of the refuelling point which reduces the risks of mission failures and enhances the implementation of long-term missions. The dynamic refuelling point method is particularly effective for small robot fleets that need to execute actions in multiple regions. We created a subsea oil platform scenario in ROS and Gazebo to evaluate the performance of this strategy using ROSPlan, a realistic AUV and ASV simulators. Our approach allows the effective execution of concurrent actions with homogeneous robot fleets using ROSPlan framework.

References

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Int. Conference on Automated Planning and Scheduling*.

Bernardini, S.; Fox, M.; and Long, D. 2017. Combining temporal planning with probabilistic reasoning for autonomous surveillance missions. *Autonomous Robots* 41(1):181–203.

Buksz, D.; Cashmore, M.; Krarup, B.; Magazzeni, D.; and Ridder, B. 2018. Strategic-tactical planning for autonomous underwater vehicles over long horizons. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3565–3572. IEEE.

Cashmore, M.; Fox, M.; Larkworthy, T.; Long, D.; and Magazzeni, D. 2014. AUV mission control via temporal planning. In *Proceedings of ICRA*, 6535–6541.

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 *Int. Conf. on Automated Planning and Scheduling*, 333–341.

Cashmore, M.; Coles, A.; Cserna, B.; Karpas, E.; Magazzeni, D.; and Ruml, W. 2018. Situated planning for execution under temporal constraints. *AAAI Spring Symposium on Integrating Representation, Reasoning, Learning, and Execution for Goal Directed Autonomy*.

Chanel, C. P. C.; Lesire, C.; and Teichteil-Königsbuch, F. 2014. A robotic execution framework for online probabilistic (re)planning. In *Proceedings of ICAPS*.

Chrpa, L.; Pinto, J.; Ribeiro, M. A.; Py, F.; Sousa, J.; and Rajan, K. 2015. On mixed-initiative planning and control for autonomous underwater vehicles. In *IROS*, 1685–1690.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-chaining partial-order planning. In *ICAPS*, 42–49.

Crosby, M., and Petrick, R. 2014. Temporal multiagent planning with concurrent action constraints. In *ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*.

Crosby, M.; Rovatsos, M.; and Petrick, R. 2013. Automated agent decomposition for classical planning. In *ICAPS*, 46–54.

Della Penna, G.; Magazzeni, D.; and Mercorio, F. 2012. A universal planning system for hybrid domains. *Applied intelligence* 36(4):932–959.

Dornhege, C.; Eyerich, P.; Keller, T.; Trüg, S.; Brenner, M.; and Nebel, B. 2012. Semantic attachments for domain-independent planning systems. In *Towards service robots for everyday environments*. Springer. 99–115.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments*. 49–64.

Fernandez-Gonzalez, E.; Williams, B.; and Karpas, E. 2018. Scottyactivity: Mixed discrete-continuous planning with convex optimization. *JAIR* 62:579–664.

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

Gerevini, A., and Long, D. 2006. Preferences and soft constraints in PDDL3. In *ICAPS workshop on planning with preferences and soft constraints*, 46–53.

Hartigan, J. A., and Wong, M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1):100–108.

Hertle, A., and Nebel, B. 2018. Efficient auction based coordination for distributed multi-agent planning in temporal domains using resource abstraction. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, 86–98. Springer.

Hofmann, T.; Niemueller, T.; Claßen, J.; and Lakemeyer, G. 2016. Continual planning in golog. In *AAAI*, 3346–3353.

Hsu, C.-W., and Wah, B. W. 2008. The sgplan planning system in ipc-6. In *Proceedings of IPC*.

Ingrand, F. F.; Chatila, R.; Alami, R.; and Robert, F. 1996. Prs: A high level supervision and control language for autonomous mobile robots. In *Robotics and Automation*, volume 1, 43–49. IEEE.

Kvarnström, J. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Twenty-First International Conference on Automated Planning and Scheduling*.

Landa-Torres, I.; Manjarres, D.; Bilbao, S.; and Del Ser, J. 2017. Underwater robot task planning using multi-objective meta-heuristics. *Sensors* 17(4):762.

Largouët, C.; Krichen, O.; and Zhao, Y. 2016. Temporal planning with extended timed automata. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, 522–529. IEEE.

Manhães, M. M. M.; Scherer, S. A.; Voss, M.; Douat, L. R.; and Rauschenbach, T. 2016. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE.

Marques, T.; Pinto, J.; Dias, P.; and de Sousa, J. T. 2017. Mv-planning: A framework for planning and coordination of multiple autonomous vehicles. In *OCEANS–Anchorage, 2017*, 1–6.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 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.

McGann, C.; Py, F.; Rajan, K.; Thomas, H.; Henthorn, R.; and McEwen, R. 2008. A deliberative architecture for AUV control. In *IEEE Int. Conf. on Robotics and Automation*, 1049–1054.

Muise, C.; Lipovetzky, N.; and Ramirez, M. 2015. Maplapkt: Omnipotent multi-agent planning via compilation to classical planning. *Competition of Distributed and Multi-Agent Planners (CoDMAP-15)* 14.

Muscettola, N.; Dorais, G. A.; Fry, C.; Levinson, R.; and Plaunt, C. 2002. IDEA: Planning at the core of autonomous reactive agents. In *NASA Workshop on Planning and Scheduling for Space*.

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

Nunes, E., and Gini, M. L. 2015. Multi-robot auctions for allocation of tasks with temporal constraints. In *AAAI*, 2110–2116.

Nunes, E.; McIntire, M.; and Gini, M. 2017. Decentralized multi-robot allocation of tasks with temporal and precedence constraints. *Advanced Robotics* 31(22):1193–1207.

Ponda, S.; Redding, J.; Choi, H.-L.; How, J. P.; Vavrina, M.; and Vian, J. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*, 3998–4003.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; and Ng, A. Y. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

Rajan, K.; Py, F.; and Barreiro, J. 2013. Towards deliberative control in marine robotics. In *Marine Robot Autonomy*. Springer. 91–175.

Sarda, E. I.; Qu, H.; Bertaska, I. R.; and von Ellenrieder, K. D. 2016. Station-keeping control of an unmanned surface vehicle exposed to current and wind disturbances. *Ocean Engineering* 127:305–324.

Schillinger, P.; Bürger, M.; and Dimarogonas, D. V. 2017. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The International Journal of Robotics Research* 0278364918774135.

Schneider, E.; Sklar, E. I.; and Parsons, S. 2017. Mechanism selection for multi-robot task allocation. In *Annual Conference Towards Autonomous Robotic Systems*, 421–435. Springer.

Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2015. A first multi-agent planner for required cooperation (marc). *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)* 17–20.

Torreño, A.; Onaindia, E.; Komenda, A.; and Štolba, M. 2018. Cooperative multi-agent planning: a survey. *ACM Computing Surveys (CSUR)* 50(6):84.

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

Zhang, Z.; Wang, J.; Xu, D.; and Meng, Y. 2017. Task allocation of multi-auvs based on innovative auction algorithm. In *Proc. of ISCID*, volume 2, 83–88. IEEE.