Task Allocation Strategy for Heterogeneous Robot Teams in Offshore Missions

Yaniel Carreno[†*] and Èric Pairet[†*] and Yvan Petillot[†] and Ronald P. A. Petrick[†]
Edinburgh Centre for Robotics
[†]Heriot-Watt University, Edinburgh, EH14 4AS, United Kingdom
[*]University of Edinburgh, Edinburgh, EH8 9AB, United Kingdom
{y.carreno,eric.pairet,y.r.petillot,r.petrick}@hw.ac.uk

ABSTRACT

Heterogeneous robot fleets are capable of supporting dynamic and resource-constrained missions. While current temporal AI planners are able to deal with multi-robot planning problems by producing plans that take into account the individual robot capabilities and task requirements, these approaches deal with the high-dimensionality of the state space inefficiently, leading to multi-robot plans with poor plan quality. This paper proposes a novel task allocation strategy called Multi-Role Goal Assignment (MRGA) which enables for more efficient computation of plans using temporal planners. The approach allocates a mission's goals based on robot capabilities, the redundancy of the sensor system, the spatial distribution of the goals and task implementation time, avoiding the need to compute a large number of possible assignments. We demonstrate the applicability of the strategy with multiple robots operating jointly in an offshore platform. Experiments demonstrate that our approach allows for more robust solutions and improved plan quality while significantly reducing planning time.

KEYWORDS

task allocation strategy, temporal-planning, long-term autonomy

ACM Reference Format:
Yaniel Carreno[†*] and Èric Pairet[†*] and Yvan Petillot[†] and Ronald P. A. Petrick[†]. 2020. Task Allocation Strategy for Heterogeneous Robot Teams in Offshore Missions. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Offshore oil and gas structures are usually located in remote and isolated places, which present a challenge for human operators due to the unsheltered maritime environment, extreme weather conditions and unfriendly atmosphere for the personnel working on those platforms. One alternative for addressing these problems is the introduction of fully autonomous robots for the commissioning and operational phases of these facilities, resulting in fewer offshore staff, reduced cost and increased personnel safety. Recent work [22, 35] has proposed a benchmark for improving the role of autonomous vehicles participating in intervention missions in offshore scenarios. However, problems such as long-term inspection in the oil rig and systematic maintenance of structures such as pipes require a level of autonomy not available in currently deployed systems [3, 42]. One approach for managing mission complexity is to equip a single vehicle with all of the necessary hardware and software components needed for complex tasks. However, the resulting robots are typically very expensive, and their designs often make trade-offs to accommodate all possible missions, leading to suboptimal systems. In addition, some sets of tasks are only appropriate for specific robots. For instance, the vertical inspection of a pipe can only be implemented using aerial vehicles.

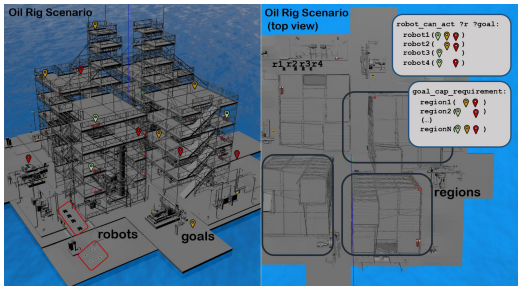

Figure 1: Overview of the offshore mission scenario.

An alternative solution is to use multiple platforms optimised for specific tasks, which collaborate to achieve a common set of goals [1, 31, 36, 40, 47]. Such *heterogeneous* multi-vehicle approaches provide robustness to the overall system and can increase a mission's scope. However, current multi-agent planning solutions struggle to generate plans that produce optimal task allocations, and often suffer from a lack of plan quality. We address this challenge in the context of a fleet of several ground and aerial robots with different sensory systems. Robots must perform regular supervision and control tasks in an oil rig environment which requires the vehicles to move between different towers and levels. For instance, Figure 1 shows the oil rig scenario [35] supervised by a robot team. Mission implementation in this context must consider robot availability for achieving goals in different regions of the oil rig, individual robot capabilities (highlighted in different colours), redundancy of the sensory system, and the resources required (e.g., battery level) to successfully complete tasks.

In this paper, we propose a new strategy for improving the quality of multi-robot plans called Multi-Role Goal Assignment (MRGA). This method focuses on optimising the goal distribution resulting from the use of AI temporal planners. The algorithm considers: (i) the robot sensory and motion capabilities, which constrains the goal allocation to robots that can satisfy the goal capability requirement,

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

and (ii) the linear combination of distance between the points of interest (POI), the level of redundancy of the robot sensory system to implement critical tasks, and the goals makespan—the time that elapses from the start of task execution to the end. We integrate the MRGA algorithm with a set of AI temporal planners, reducing the size of the search space and consequently the planning time. We evaluate the approach by considering relevant aspects of the planning performance, such as spatial goal distribution, makespan, planning time, and the goal allocation rate for different robot sets, to demonstrate the applicability of the strategy. We deploy our multi-robot task allocation and planning strategy to a robot fleet of several Husky and Quadcopter robots operating in an offshore oil platform simulator, providing evidence of the efficacy of the method to deal with highly constrained domains.

2 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 modelled using PDDL2.1 [19], or other extensions of the standard Planning Domain Definition Language (PDDL) [29], which adds support for temporal planning through additional language constructs. A number of temporal planners support PDDL2.1, such as TFD [17], SGPlan [24], and LPG-TD [20]. In addition, the 2018 International Planning Competition (IPC-2018) showed the potential of portfolio-based approaches [12] to deal with larger numbers of problems using a set of individual planners, and the TFLAP planner, which represents an extension of [39]. However, these planners face problems related to scalability, concurrency, and action timeslot allocation, which have limited their application in real missions with multiple robots.

There are two temporal planners which are particularly promising for the implementation of missions in challenging environments: the Forward-Chaining Partial-Order Planning (POPF) [13] and Optimizing Preferences and TIme-dependent Costs (OPTIC) [2], which are later extensions of COLIN [14]. These planners support numerical fluents, concurrency and exogenous events. POPF is characterised by its ability to find plans with low makespan 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 [10, 11]. However, there is little work addressing the multi-agent problem using these planners [23, 46]. In general, these approaches present limited task complexity and diversity which prevents the evaluation of the planners' performance in highly constrained domains.

Multi-Agent Planning (MAP) has also been addressed in [16, 26, 30, 44], 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 [45], which make them less attractive for the implementation of complex missions with concurrent actions. Although several approaches [1, 28, 32] consider mission timing constraints to solve multi-agent problems, they use specific language representations or provide solutions to particular planning problems (e.g., path planning optimisation) in the global plan which limits the generalisation of the approach. Other approaches consider separable tasks [40] which limits robot collaboration. In our work, task allocation conditions the planning process but does not restrict the implementation of collaborative actions. Other work [33, 34, 37] has explored task allocation with temporal constraints using auction-based approaches. [37] proposes a multi-item auction strategy which replaces the auctioneer for distributed consensus phases and considers tasks with time windows. [33, 34] 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. TeSSI also assumes the robots can execute all the actions allocated and there is no further analysis considering robot resources (e.g., energy level) which can affect the sequence of defined actions. In addition, auction-based approaches can lead to solutions where all tasks are allocated to a single robot reducing the solution's optimality. This is mitigated in our strategy by distributing the robots around the arena at the start of the mission.

Recent research [27, 48] has considered MAP problems in the context of more complex domains such as underwater approaches. In particular, the ScottyActivity planner [18] introduces levels of coordination in maritime applications and provides good results for controlling continuous variables in the domain. The strategy addresses 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 used in the missions prevent a scalability analysis. Few works directly consider temporal planning for multi-agent applications. In [15], planning problems are modelled using standard PDDL and then translated to a temporal planning model for generating and distributing plans to individual robots. [23] 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 require concurrency. [41] also considers an auction mechanism. However, this work does not analyse the coordination of multiple robots and the way the tasks are clustered. Finally, [4] propose a strategy based on clustering to allocate mission goals. Nevertheless, the method does not consider the robot capabilities required to implement the allocation which is considered in our work.

3 PROBLEM DEFINITION

We investigate how to allocate tasks with particular capability requirements (sensory and motion) to a set of heterogeneous robots deployed in an offshore oil platform, such that the generated allocation is feasible according to the robot's capabilities. In addition, the task allocation approach considers the minimisation of a cost function based on the combination of three parameters: makespan, the distance between the POIs, and sensor-based optimisation considering redundancy. We assume the sets of robots R and goals G are nonempty and the mission's goals are known upfront. The robots represented by R are heterogeneous. The solution to the task

allocation problem for a heterogeneous fleet intrinsically solves the homogeneous fleet task allocation problem.

Task allocation methods support the implementation of missions with different goal capability requirements distributed over large areas such as search and rescue [23], underwater [4], and surveillance [1]. Our problem considers a set of robots that must complete tasks located at an offshore oil rig with multiple towers and several floors (see Figure 1). The autonomous supervision of an offshore platform requires a considerable number of sensors and actuators. In this work, we consider six types of tasks which require different capabilities: `check_temperature`, `check_pressure`, `observation`, `take_image`, `valve_inspection` and `manipulate_valve`.

Observation tasks involve the exploration of particular POIs and can be executed using drones or husky robots. The `valve_inspection` task requires the husky to identify the state of the valve (on/off), for which the robot needs to be located in an optimal position. The `take_image` task captures images of the structure from different points of view using drones. The tasks `check_pressure` and `check_temperature` are associated with the collection of sensor data by the huskies. The `manipulate_valve` task changes the open/close state of a valve and requires the coordination between drones and huskies (the drone records the execution and the husky turns the valve). Essential actions in the domain include data communication, navigation, and refuelling (see Section 4.2 for a full list) which support robot mobility and long-term autonomy. Mission goals are strongly related to the tasks the robotic system can implement. The AI planner is responsible for generating a sequence of actions that achieves a given set of tasks.

PDDL-based temporal planners (such as those that use PDDL2.1 [19]) deal with multi-agent capability constraints [15, 46] and intrinsically address the task allocation problem. However, in many cases the plans generated by these planners show poor task allocation quality in complex missions [7, 8]. As a result, our approach solves the goal assignment problem before initialising the main planning process. Planning then jointly considers the task allocation (MRGA's output), temporal constraints, and the robots' resources and actions to generate plans. Formally, we define the task allocation problem in multi-robot systems as follows.

Definition 3.1. A *Task Allocation Problem* is defined by a tuple $t := \langle R, RC, G, GC, P \rangle$, where $R = \{r_1, r_2, ..., r_n\}$ is a set of robots, $RC = \{rc_1, rc_2, ..., rc_m\}$ is a set of capabilities, $G = \{g_1, g_2, ..., g_s\}$ is a set of goals, $GC = \{gc_1, gc_2, ..., gc_w\}$ is a set of capabilities required to implement the goals, and $P = \{p_1, p_2, ..., p_s\}$ is a set of goal coordinates. Each $r \in R$ can be related to multiple elements of RC. Each $g \in G$ is related to a single element of GC.

Our strategy first solves the task allocation problem and then introduces a new set of constraints into the planning phase to directly allocate goals to individual robots. The allocation is based on two cost functions: maximising the number of tasks the robots can achieve and minimising the combination of distance between goals, task makespan, and aborted tasks due to problems in a robot's sensory system. The result is a reduction in the complexity of the search tree the AI planners need to explore to generate plans.

Algorithm 1: ANALYSER (R, R_{r_n-cap}, G, GC)

Input:
R: Robot Set
G: Goal Set
GC: Goal Capability Requirements Set
R_{r_n-cap}: Robot Capabilities Set

```
1 begin
2     cnt = 0 /* counter over the set of robots */
3     while not R ← ∅ do
          /* solvable goal set for robot r_cnt */
4         rgoal_cnt (G) ← ∅
5         for g_cnt to length (G) do
6             if GC (g_cnt) → R_cnt−cap then
7                 rgoal_cnt (g_cnt) ← G (g_cnt)
8         cnt = cnt +1
9         R = R.remove.r_cnt
```

4 SYSTEM FORMULATION & DESCRIPTION

We address the goal assignment problem in the context of a heterogeneous robot fleet. The role of the robots during goal execution can be *active* (agents can execute goals) or *passive* (agents cannot execute goals), which is determined by their capabilities. 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. Heuristic forward search planners have shown the best performance generating plans in many domains. However, plan search is not primarily guided to guarantee optimal task allocation. In this section, we present the MRGA algorithm which improves the quality of the plans generated by temporal planners (TP) dealing with the task allocation problem.

4.1 Multi-Role Goal Assignment (MRGA)

The MRGA algorithm is based on two cost functions: (i) the tasks solvable for a robot by considering its capabilities, and (ii) the linear combination of the makespan, distance between the POIs and redundancy of the sensory system. We divide the analysis into two parts: Capabilities Analyser and Regions Delimiter.

4.1.1 ***Capabilities Analyser:*** The capabilities analysis determines the set of mission goals each robot can implement. Algorithm 1 describes the strategy which considers four inputs: the sets of available robots $R = \{r_1, r_2, ..., r_n\}$, goals $G = \{g_1, g_2, ..., g_s\}$, capabilities required to implement the goals GC, and the set of robot capabilities R_{r_n-cap} which includes n subsets (one subset per robot). The method identifies the robots that possess the capability required to implement each goal (line 4), and allocates the tasks (line 5) to the list of solvable goals $\mathbf{r}goal_{\mathsf{cnt}}$ for robot r_{cnt} when the required capability is found. The role the robots (active or passive) regarding the execution of each goal is defined based on the capability analysis. As an example of how the algorithm operates, consider two robots r_1 and r_2 and a goal g_1. We compare the capability required to implement g_1 with the capability set of r_1 and r_2. Each robot is considered and if the capability required to execute g_1 is found, the goal is allocated to the solvable goal set for that robot.

This process is repeated (line 1) until all the robots in the mission know the goals they can achieve based on their capabilities.

4.1.2 ***Regions Delimiter:*** The Regions Delimiter is divided in two parts: Robot Distribution (RD) and Goal Allocation (GA). RD identifies regions in the environment to deploy the robots following the pipeline presented in Algorithm 2, which considers the number of available robots and the goal coordinates. We use the RD strategy to distribute robots around the environment to avoid worst-case complexity [33], which arises when all the tasks are assigned to a single robot. Regions are obtained using an updated version of the k-means algorithm [21], which decomposes the goals geographically (line 3) by partitioning the observations according to the Voronoi diagram generated by their means. Given an initial set of k means $m_i := \{m_1^{(1)} ... m_k^{(1)}\}$, the formal distribution is described as:

$$S_i^t = \left\{ x_p : \| x_p - m_i^t \|^2 \leq \| x_p - m_j^t \|^2 \ \ \forall j,\ 1 \leq j \leq k \right\}, \quad (1)$$

where $S_i^{(t)}$ is a cluster, x_p are goal coordinates, and $m := \{m_1, ..., m_t\}$ is a set of means updated using:

$$m_i^{(t+1)} = \frac{1}{| S_i^{(t)} |} \sum_{x \in S_i^{(t)}} x_j. \quad (2)$$

The algorithm can use all the robots available for the mission. However, the strategy optimises the resources based on the mission requirements and goal positions. The method takes the results from the clustering (line 3) and the Capabilities Analyser (Algorithm 1, line 7) and calculates the number of goals the robots can execute in each region (lines 3-4). These results are used to determine the best robot option for implementing actions in each cluster. We perform this evaluation using linear programming methods [5] which determines the optimal robot distribution over the clusters by addressing the problem as a maximisation problem (line 4):

$$\textbf{max}\, z = \sum_{r=1}^{m} \sum_{c=1}^{n} w_{rc} x_{rc}, \quad (3)$$

subject to $\sum_{c=1}^{n} x_{rc} = a_r$, $\sum_{r=1}^{m} x_{rc} = b_c$, with $r = 1, ..., m$ and $c = 1, ..., n$, $x_{rc} \geq 0$. Where robot $r \in R$ robot set, cluster $c \in C$ cluster set, (r, c) is an edge, w_{rc} is the number of goals r can execute in c, and x_{rc} is the variable in the edge.

The solution to the maximisation problem provides the optimal robot distribution over the clusters by considering the robot that is most appropriate for a region based on the number of tasks it can solve there. Our approach calculates the distance between robot allocations and all the solvable goals in the area (line 8). The closest goal is the first assigned to the robot (lines 11-12). This process is repeated for all the robots until the vehicles are distributed over the environment. The strategy removes the goals allocated from the unallocated goal set (line 13). This method can solve a large number of goals; however, the approach can also lead to suboptimal solutions when the robots cannot execute all the goals in a region. This is due to RD which restricts the operation of robots to particular clusters based on the locations of their closest goals. The approach can be generalised to instead cluster by similar goals, by abstracting the notion of spatial distance to distance between

Algorithm 2: ROBOT-D $\left(R, R_{coord}, G, G_{coord}, R_{\mathbf{r}goal_n}\right)$

Input:
R: Robot Set
G: Goal Set
R_{coord}: Initial Coordinates of Robot Set
G_{coord}: Coordinates of Goal Set
$R_{\mathbf{r}goal_n}$: Set of solvable goals for the n robots in R

1 **begin**
2 $\{ GA_{INIT}, RA_{INIT}, C_{sol}, R_{sol} \} \leftarrow \emptyset$
/* GA_{INIT} is set with the first goal allocated to each robot, RA_{INIT} is set with robot allocated to each cluster, C_{sol} is set with clusters id and POIs, and R_{sol} is the number of robots required */
3 $[C_{sol}, R_{sol}]$ = `cluster_cal` $(G, R, R_{coord}, G_{coord})$
4 RA_{INIT} = max$\{$`weight_cal`$(R_{\mathbf{r}goal_n}, C_{sol}, R_{sol})\}$
5 `cnt` = 0 /* counter over the set of robots */
6 **while** *not* $R \leftarrow \emptyset$ **do**
7 **for** *0* ***to*** `length` $(C_{sol}$ (`cnt`)) **do**
8 $dist$ = `dist_cal` $(RA_{INIT}, R_{coord}, C_{sol}, C_{coord})$
9 **for** *0* ***to*** $\mathbf{r}goal_{\text{cnt}}$ **do**
10 **if** `min` $\sum dist$ **then**
11 R_{coord} (`cnt`) $\leftarrow C_{coord}$ (`cnt`)
12 GA_{INIT} (`cnt`) $\leftarrow G$ (`cnt`)
13 G.remove.GA_{INIT}
14 `cnt` = `cnt` +1
15 R = R.remove.r_{cnt}

goal characteristics when each robot presents a unique capability required for planning.

The GA step distributes the set of unallocated goals and eliminates the possibility of suboptimal goal distribution. We initiate the GA after assigning the first goal to each robot (using the RD step). GA does not consider cluster restrictions. Therefore, the robots can move freely in the environment implementing goals based on the GA cost function. The strategy described in Algorithm 3 combines the system reliability analysis based on the redundancy of the sensory system [25] with vehicle routing and scheduling problems [43], which allows us to consider past distance travelled and makespan to allocate the goals. The method evaluates the cost of implementing each goal for each agent over the set of unallocated goals and robots (lines 6-8). We find the minimum cost associated with the implementation of a goal (lines 9-10) using Equation 4:

$$\textbf{min}\, c = \sum_{g=1}^{n} \sum_{r=1}^{m} \gamma M_{max}^r + \alpha \left[T_{(g_i, g_f)}^r - T_{(g_i, 0)}^r \right] + \beta \left(\frac{T_{g-aborted}^r}{No\_s} \right), \quad (4)$$

where γ, α and $\beta \in [0, 1]$ are weighting factors, $\gamma + \alpha + \beta = 1$, and M_{max}^r is a maximum makespan for robot r, $T_{g-aborted}^r$ is the total wasted time resulting from aborted tasks, No_s is the number of

Algorithm 3: GOAL-A (M^r_{max}, NS,CT, ROBOT-D)

Input:
M^r_{max} : Task Makespan Set
CT : Number of Critical Tasks in the Mission
ROBOT-D: Inputs and Outputs of Algorithm 2
NS : Number of Sensor Sets

```
/* NS defines the redundancy associated with each goal. Some
   inputs and outputs of ROBOT-D are also used here.        */
1  begin
2      GA_FINAL (G) ← GA_INIT
       /* GA_FINAL (G) will contain all the goal allocations,
          initially it has the goals allocated for ROBOT-D   */
3      cnt = 0 /* counter over the set of robots             */
4      while not G ← ∅ do
5          g = 0 /* initialisation of the number of goals    */
6          for g to length (G − GA_INIT) do
7              if GC (cnt) ∈ rgoal_cnt then
8                  for 0 to length(R) do
9                      dist(r) = dist_cal (R_coord,G_coord)
10                     cost(r) = f (γ, β, α, NS, CT, M^r_max, dist)
11                     M^r_max = update.makespan_cal(g)
12             cost_r(g) = min (cost(r))
13             cost_r(G) ← cost_r(g)
14         GA_FINAL ← min {cost_r(G)}
15         G = G.remove.GA_FINAL
```

robot sensors that provide redundancy to implement a goal, and $\left[T^r_{(g_i,g_f)} - T^r_{(g_i,0)}\right]$ represents the distance between the the initial goal g_i and the final goal g_f. The distance travelled is transformed to travel time by assuming all robots are moving at the same speed. We calculate M^r_{max} by considering the accumulated makespan and the makespan associated with the implementation of the actual action. The $T^r_{g-aborted}$ parameter is calculated by defining the critical tasks in the mission which require redundancy and can be implemented by the robot.

The robots update their allocation cost function for each unallocated goal in the environment (independent of the clusters) (lines 11-13) by considering the distance to each goal and its makespan. The robot with the minimum bid will take the goal associated with the smallest cost value (line 14). The robot will then update its position and calculate the cost function for all unallocated goals again. This process is repeated until all the goals are allocated to the robots. The accumulated makespan considers the distance travelled between the POIs. The weight parameters condition the allocation of a goal to a particular robot giving more importance to the distance between the goals, the makespan, or the redundancy analysis. We choose $\alpha + \beta = 0.45$ in an attempt to find a balance between makespan and redundancy while still considering distance. This method optimises the goal distribution allowing the robots to move to other regions when they are required to solve a goal that depends on a capability the robot in that area does not possess (line 7). MRGA's output is defined in PDDL. The results of the allocation (line 14) are transformed to instances of the PDDL predicate (`robot_can_act ?r ?to`), where `?r` is the robot and `?to` represents the POI of the goal. We modify the PDDL problem file to add these instances to the initial state. The PDDL domain file is also modified to add an instance of this predicate to each action as a precondition in order to constrain the selection and execution of the action to the appropriate robots. We introduce capability constraints into our domain actions using two different definitions. We can define a predicate for each action associated with specific sensors and an action precondition associated with each defined predicate. For instance, we define the predicate (`can_inspect_valve ?r`) which ensures the agents performing the inspection action have the appropriate sensor. We use this representation to generate plans using the benchmark planners. In this case, PDDL solvers deal with the task allocation problem directly. Alternatively, we can add `robot_can_act` preconditions to the domain's actions. The MRGA strategy generates a set of instances of the (`robot_can_act ?r ?to`) predicate covering all the mission goals at positions `?to`.

Table 1: Robot capabilities and estimated task duration.

Capabilities	Robots	Time
`navigation`	$R_1, R_2, R_3, R_4, R_5, R_6$	$t = f(d)$
`communicate-data`	$R_1, R_2, R_3, R_4, R_5, R_6$	7
`refuel`	$R_1, R_2, R_3, R_4, R_5, R_6$	$t = f(chr, e)$
`observation`	$R_1, R_2, R_3, R_4, R_5, R_6$	15
`take-image`	R_5, R_6	5
`valve-inspection`	R_1, R_3, R_4	20
`check-temperature`	R_1, R_2	10
`check-pressure`	R_1, R_3, R_4	10
`turn-valve`	R_1, R_3, R_4, R_5, R_6	30

4.2 MRGA+TP Definition

The goal assignment algorithm adds true instances of the predicate to the initial state of the PDDL problem definition which directly allocates goals before using the benchmark solvers to generate the plan. We introduce the results of MRGA into the planning process, referred to as the MRGA+TP strategy, resulting in solutions that improve the performance of the TP.

4.2.1 ***PDDL Domain Definition:*** The planning process requires a realistic representation of the domain. Our domain definition captures the dynamics of the world using standard PDDL features like types, static facts, and functions.[1] The domain file contains the following types: `robot`, `robot_sensor`, `observation_point` and `actuator`.[2] The observation points specify the coordinates of the POIs, which have a fixed location defined by the domain designer. The robots' sensory system is described by the `robot_sensor`. The `actuator` defines the husky's arms. Our work considers a set of heterogeneous vehicles and the robots present differences in their

[1]Domain and problem instances are available in the MRGA repository at https://github.com/YanielCarreno/MRGA.
[2]We will use `?r` to denote a parameter of type `robot`, `?poi` a parameter of type `observation_point`, `?t_s` (temperature sensor), `?p_s` (pressure sensor) and `?camera` are parameters of type `robot_sensor`, and `?arm` is a parameter of type `actuator`.

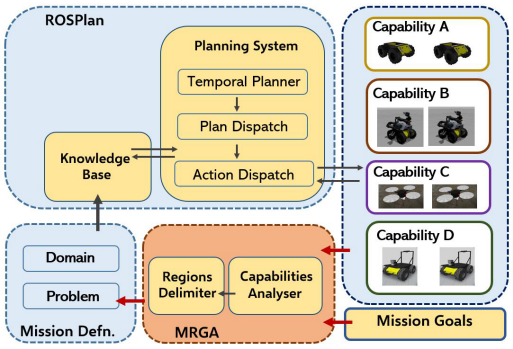

Figure 2: Task Allocation and Planning framework.

sensory systems. PDDL actions and properties will typically be related to the capabilities of the specific robot platforms, captured by the following domain actions:

- **`navigation(?r, ?from, ?to):`** a durative action which moves the robot `?r` from observation point `?from` to `?to`. We employ the semantic roadmap for autonomous point-to-point navigation and collision-free planning defined in [35] to compute navigation actions which meet the `?r` kinematic capabilities, and retrieve the corresponding action duration.
- **`data_communication(?r, ?poi):`** a durative action which allows the robot `?r` to communicate the data recorded at observation point `?poi`.
- **`refuel(?r, ?poi):`** a durative action for robot `?r` to recharge its battery at the closest recharging point `?poi`, where it stays during recharging time. We consider multiple recharging points based on the scenario described in [7]. The recharging point changes according to the robot's position.
- **`observation(?r, ?poi, ?camera):`** a durative action where the robot `?r` explores the structure located at `?poi` from different view points (multiple angles) using `?camera`. The drone records data to analyse the state of the structure.
- **`take_image(?r, ?poi, ?camera):`** a durative action which enables the robot `?r`'s camera `?camera` to capture images on the observation point `?poi`.
- **`valve_inspection(?r, ?poi, ?camera):`** a durative action which enables the robot `?r`'s camera `?camera` to identify the state of the valve located at point `?poi`.
- **`check_temperature(?r, ?poi, ?t_s):`** a durative action which enables the robot `?r`'s temperature sensor `?t_s` to collect and store data at point `?poi`.
- **`check_pressure(?r, ?poi, ?p_s):`** a durative action which enables the robot `?r`'s pressure sensor `?p_s` to collect and store pressure data at point `?poi`.
- **`manipulate_valve(?rh, ?rd, ?poi, ?camera ?arm):`** a durative action which enables the husky robot `?rh`'s actuator `?arm` to turn the valve, and drone `?rd`'s camera `?camera` to record the action at point `?poi`.

The properties of our offshore platform domain are encoded in PDDL. Our encoding is based on adding capability and temporal constraints to the domain. The domain actions consider robot capabilities to distribute the actions among the fleet. Table 1 shows a list of the domain capabilities that each robot holds and the estimated time that it takes to perform the action associated with the capability. Action duration for essential capabilities is based on the distance d between the POIs, the energy e and the recharging rate chr. We assume the domain actions are deterministic and there is complete domain knowledge. The planner generates a plan for the robot fleet distributing the goals based on the domain constraints. The robots acting in the domain can execute actions concurrently.

4.3 System Architecture

There are multiple architectures that support the execution of multi-robot missions. We use ROSPlan [11] which connects the widely used Robot Operating System (ROS) [38] and PDDL2.1. ROSPlan architecture allows the integration of high-level planning with robot-level action implementation methods. Prior implementations using ROSPlan have demonstrated good performance when dealing with single-robot missions [9] and multi-robot approaches [6].

Figure 2 shows the MRGA framework which consists of the Capabilities Analyser and Regions Delimiter ROS nodes. MRGA receives data from the robots (the number of robots and capabilities), mission goals and the capabilities required to execute these tasks, and generates a set of PDDL instances which populate the problem file. The system evaluates the capability constraints using the Capabilities Analyser and defines the regions using the Regions Delimiter. The problem file (see Mission Defn.) updates the Knowledge Base node from ROSPlan. We use a centralised planning schema to generate plan solutions. The planner attempts to optimise the complete decision space and reduce the number of planning failures. We assume a reliable communication between the acting agents and the planning agent which allows plan repairs during mission execution keeping total mission failure at low levels. However, the approach can also be used with a decentralised architecture [6]. This work provides a generic solution using standard languages and methods (PDDL and ROS), but is easily adaptable to a different output structure or language.

We demonstrate the approach in simulation with multiple heterogeneous robots equipped with temperature and pressure sensors, cameras, and arms. We implement our experiments using the ORCA Hub[3] simulator [35] which presents a ROS-enabled oil rig environment to execute tasks with multiple robots. The simulator environment (see Figure 1) allows multiple instances of different robotic platforms to coexist and implement complex missions with large goal sets. The ORCA Hub simulator offers access to a semantic description of the oil rig structure which provides knowledge of the environment such as the labels that describe the 3D coordinates of the POIs (e.g., valve positions).

5 EXPERIMENTS AND RESULTS

We evaluate the MRGA algorithm in simulation in three experiments. In the first experiment, we analyse the performance of the goal distribution in a particular problem setting. The second experiment examines the efficiency of the approach through the analysis of the plan's quality. We compare the performance of MRGA+TP with the results of well-established benchmark planners and a portfolio approach: LPG-TD, TFLAP, TFD, POPF, OPTIC, and TemPoRal

[3] https://orcahub.org/

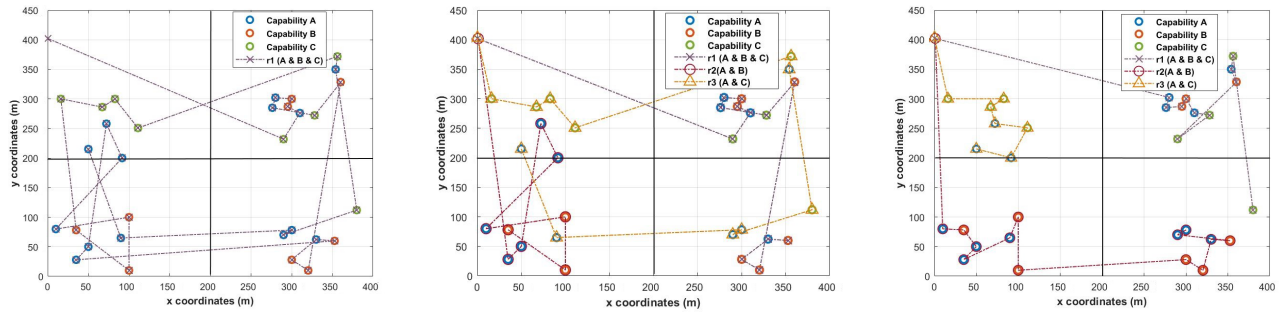

Figure 3: Goal allocation using TFLAP (left), OPTIC (middle) and MRGA+TP (right). Simulations evaluate the sequence of goals implemented for a set of three heterogeneous ground robots using as a reference a top view of the oil platform scenario.

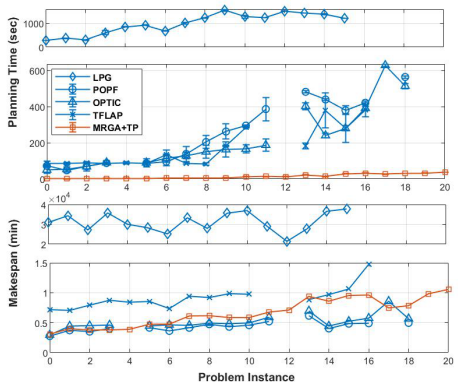

Figure 4: Planning time (top) and makespan performance (bottom) using MRGA+TP and the benchmark planners.

which have been tested in multiple experiments in the past. We evaluate the approach by considering fleets of three, four and six heterogeneous robots supervising up to 8 regions. We use OPTIC as our temporal planner, which we've found provides good plan results in a large number of domains using domain-independent heuristics and fast generation. We consider a set of 20 problems of increasing complexity. Experiments were attempted 150 times and the results show the mean and standard deviation of the data. Finally, we evaluated the scalability performance of the strategy using larger numbers of goals and robots. All experiments were performed on a machine with a 4GHz processor, with the planner limited to 30 minutes of CPU time and 8GB of memory.

Experiment 1: We evaluate the performance of TFLAP, OPTIC and MRGA+TP using a fleet of three robots (R_1, R_2, and R_3). The problem includes 31 goals (including three different types of goals with different capabilities) distributed in four regions. The offshore platform is concentrated in an area of almost a half square kilometre. The robots must execute goals related to the capabilities in Table 1. The initial position of the robots is at a corner of the environment (see Figure 1). We only use husky robots in this experiment to demonstrate the advantages of the allocation approach using the Robot Distribution and Goal Allocation strategies.

Experiment 2: We analyse the performance of our approach comparing the quality of the plans generated by the benchmark planners and the plan results of combining the MRGA algorithm with OPTIC (MRGA+TP). We evaluate the results of executing plans for the 20 problems. The makespan and planning time metrics are assessed for fleets of 3-6 robots.

Experiment 3: We evaluate the scalability performance of the strategy for a large number of robots and goals. We generate missions with large numbers of goals (up to 150) and robot sets, increasing the number of husky robots and drones in the same proportion up to 30. We analyse the number of robots required to execute a large number of goals. The experiment attempts to determine the maximum number of goals a robot team can achieve without intermediate battery recharging, to demonstrate the advantage of using MRGA to optimise resource consumption.

5.1 Results and Analysis

In comparing MRGA+TP to TFLAP and OPTIC, we found that MRGA+TP outperforms the other approaches in distributing goals for the same number of robots. The results of the goal allocation presented in Figure 3 improve the overall plans in Experiment 1.

MRGA+TP effectively distributes the goals among the robot set while trying to maintain the agents at different regions to reduce the possibility of robot collisions, the total distance travelled and the energy consumed. However, robot capabilities influence the allocation and can force the assignment of goals from different regions to the same robot. For instance, r1 moves to the bottom-right region to implement a goal with capability C as a consequence of r2 not having the required capability. Experiments show that the strategy takes advantage of the distance between the POIs and makespan. We found the algorithm effectively considers the makespan parameter to allocate a goal when the number of robots is smaller than the number of regions. Robot r2 implements most of the goals in two regions (bottom-right/left) since the time to implement capability B is the smallest, making r2 the best option to implement the goals. On the other hand, goal distribution in TFLAP and OPTIC provide suboptimal plans with poor task allocation. TFLAP allocates all tasks to a single robot (an agent with all the capabilities) which affects the goal allocation and action schedule in the plan, while OPTIC generates the best performance among the

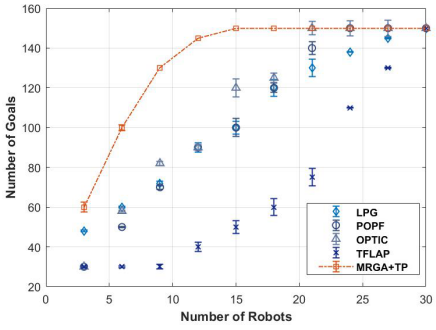

Figure 5: Task allocation results for multiple robot sets.

benchmark solvers. Results show that combining temporal planning and MRGA can substantially improve the general performance of the system and optimise the use of the robots and resources.

In Experiment 2, we evaluate plan performance by considering planning time and makespan. Figure 4 shows planning time (top) and plan makespan (bottom) for 20 problems. We found MRGA+TP generates solutions for all problems in less than 5 minutes of planning time. Since planning time influences the capacity of the robotic system to react optimally during time-sensitive tasks, minimising planning time is essential for optimising the performance of the system. The results show that MRGA+TP substantially reduces the planning time over all problems compared with the benchmark planners. Thus, task allocation in multi-agent domains can significantly speed up the planning time. Solvers such as POPF and OPTIC (the second and third best results, respectively) have longer planning times which are not desirable in many real-world applications. The performance of MRGA also provides evidence that the strategy improves plan generation for a heterogeneous fleet by acting as a decision maker which reduces the complexity of the resulting planning problem, thereby facilitating the temporal planner's work. In addition, the simulations illustrate the advantage of using MRGA+TP for multi-vehicle strategies, and provide a tool for temporal planning to optimise complex mission planning with large numbers of goals and constraints. MRGA also improves the capacity of the robotic fleet to react to replanning situations. The performance of benchmark planners is particularly poor for problems where the planning time exceeds the time limit of 30 minutes. We also ran simulations using TDF and TemPorAl, however, these solvers did not generate solvable plans in the required time.

We also evaluated plan makespan performance as part of Experiment 2. The makespan results for MRGA+TP are similar to the values obtained by the other planners in most of the problem instances. However, the strategy optimises the number of robots used during the mission leading to implementations where MRGA+TP uses a smaller number of vehicles compared with the benchmark solvers to achieve the same goals. The results demonstrate MRGA supports the generation of plans with higher quality and similar mission times but using fewer resources.

In Experiment 3, we analyse the scalability performance of the MRGA+TP approach. Figure 5 shows the results of generating plans with different robot sets. The results indicate that MRGA+TP is capable of generating plans with a large number of goals using smaller robot sets compared with the planners evaluated. In some cases, the approach reaches the maximum number of goals using half the number of robots that most benchmark planners need to execute the missions without refuelling.

In general terms, our method distributes goals while trying to optimise the number of robots used for mission execution. MRGA is a planner agnostic approach which support plan generation using different solvers depending on the domain definition and problem requirements. This strategy works for special cases such as problems that have lower numbers of goals than robots. The algorithm attempts to optimise the number of robots used to implement a mission by considering their capabilities. In practice, the algorithm tends to use the robots with more capabilities to solve multiple goals. The method improves the performance of benchmark planners for all problem types we explored (using heterogeneous robots). However, the strategy can fail to optimise the results if the fleet is homogeneous, there is one region, and the goal makespans are similar. This case is very specific and quite unlikely in missions with heterogeneous robot teams. In addition, the definition of the weighting factors γ, α, and β is an element to consider in order to achieve the best possible goal distribution.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we investigated task allocation and planning for missions involving multiple heterogeneous robots and complex models of the environment. We presented a novel task allocation strategy called Multi-Role Goal Assignment which improves the efficiency of plan generation in many situations using temporal planning. The approach attempts to decompose mission goals by considering their position in the environment, the task implementation time, the robot capabilities, and the redundancy of the robot sensory system. We demonstrate the applicability of our approach with multiple robots in a simulated offshore environment. Results from the experiments indicate that MRGA is a flexible strategy which improves the quality of goal allocation with respect to the underlying benchmark planners. An analysis of plan makespan and planning time validates the robustness of the approach. The method substantially reduces planning time with similar makespans compared with the tested temporal planners for small robot fleets. In addition, MRGA+TP shows good performance in terms of scalability for situations with a large number of goals and heterogeneous robots.

Future work involves studying plan deviation analysis to evaluate plan performance during execution. We intend to introduce multiple replanning strategies that improve the mission solvability rate. We are also working to introduce an online version of the strategy. We aim to adapt this approach to a more flexible online execution context. Finally, we aim to evaluate the performance of the approach using real robot platforms.

ACKNOWLEDGMENTS

We acknowledge the support of the BAE Systems Surface Ships Ltd along with the EPSRC ORCA Hub (EP/R026173/1, 2017-2021) and consortium partners.

REFERENCES

[1] Patrick Bechon, Charles Lesire, and Magali Barbier. 2019. Hybrid planning and distributed iterative repair for multi-robot missions with communication losses. *Autonomous Robots* (2019), 1–27.

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

[3] Robert Bogue. 2019. Robots in the offshore oil and gas industries: a review of recent developments. *Industrial Robot: the international journal of robotics research and application* (2019).

[4] Dorian Buksz, Michael Cashmore, Benjamin Krarup, Daniele Magazzeni, and Bram Ridder. 2018. Strategic-tactical planning for autonomous underwater vehicles over long horizons. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3565–3572.

[5] Rainer E Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment problems*. Springer.

[6] 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 *Proceedings of ICAPS*.

[7] Yaniel Carreno, Yvan Petillot, and Ronald P A Petrick. 2019. Multi-Vehicle Temporal Planning for Underwater Applications. In *Proceedings of ICAPS Workshop on Planning and Robotics 2019*.

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

[9] Michael Cashmore, Andrew Coles, Bence Cserna, Erez Karpas, Daniele Magazzeni, and Wheeler Ruml. 2018. Situated Planning for Execution Under Temporal Constraints. *AAAI Spring Symposium on Integrating Representation, Reasoning, Learning, and Execution for Goal Directed Autonomy* (2018).

[10] Michael Cashmore, Maria Fox, Tom Larkworthy, Derek Long, and Daniele Magazzeni. 2014. AUV mission control via temporal planning. In *Proceedings of ICRA*. 6535–6541.

[11] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. 2015. ROSPlan: Planning in the Robot Operating System. In *Proceedings of ICAPS*. 333–341.

[12] Isabel Cenamor, Tómas de la Rosa, Mauro Vallati, Fernando Fernández, and Lukáš Chrpa. 2018. TemPoRal: Temporal Portfolio Algorithm. *Proceedings of ICAPS International Planning Competition* (2018).

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

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

[15] Matthew Crosby and R Petrick. 2014. Temporal multiagent planning with concurrent action constraints. In *Proceedings of ICAPS Workshop on Distributed and Multi-Agent Planning (DMAP)*.

[16] Matt Crosby, Michael Rovatsos, and Ronald Petrick. 2013. Automated Agent Decomposition for Classical Planning. In *Proceedings of ICAPS*. 46–54.

[17] Patrick Eyerich, Robert Mattmüller, and Gabriele Röger. 2012. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Towards Service Robots for Everyday Environments*. 49–64.

[18] Enrique Fernandez-Gonzalez, Brian Williams, and Erez Karpas. 2018. ScottyActivity: Mixed Discrete-Continuous Planning with Convex Optimization. *JAIR* 62 (2018), 579–664.

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

[20] 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.

[21] John A Hartigan and Manchek A Wong. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28, 1 (1979), 100–108.

[22] Helen Hastie, Katrin Lohan, Mike Chantler, David A Robb, Subramanian Ramamoorthy, Ron Petrick, Sethu Vijayakumar, and David Lane. 2018. The orca hub: Explainable offshore robotics through intelligent interfaces. *ACM Human-Robot Interaction Conference. Paper presented at the Explainable Robotics Workshop, Chicago IL*. (2018).

[23] Andreas Hertle and Bernhard Nebel. 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)*. Springer, 86–98.

[24] Chih-Wei Hsu and Benjamin W Wah. 2008. The SGPlan planning system in IPC-6. In *Proceedings of ICAPS International Planning Competition*.

[25] Yichuan Jiang. 2015. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 2 (2015), 585–599.

[26] Jonas Kvarnström. 2011. Planning for loosely coupled agents using partial order forward-chaining. In *Proceedings of ICAPS*.

[27] Itziar Landa-Torres, Diana Manjarres, Sonia Bilbao, and Javier Del Ser. 2017. Underwater robot task planning using multi-objective meta-heuristics. *Sensors* 17, 4 (2017), 762.

[28] Christine Largouët, Omar Krichen, and Yulong Zhao. 2016. Temporal Planning with extended Timed Automata. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 522–529.

[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] Christian Muise, Nir Lipovetzky, and Miquel Ramirez. 2015. MAP-LAPKT: Omnipotent multi-agent planning via compilation to classical planning. *Competition of Distributed and Multi-Agent Planners (CoDMAP-15)* (2015), 14.

[31] Robin R Murphy, Karen L Dreger, Sean Newsome, Jesse Rodocker, Brian Slaughter, Richard Smith, Eric Steimle, Tetsuya Kimura, Kenichi Makabe, Kazuyuki Kon, et al. 2012. Marine heterogeneous multirobot systems at the great Eastern Japan Tsunami recovery. *Journal of Field Robotics* 29, 5 (2012), 819–831.

[32] Alexandros Nikou, Dimitris Boskos, Jana Tumova, and Dimos V Dimarogonas. 2018. On the timed temporal logic planning of coupled multi-agent systems. *Automatica* 97 (2018), 339–345.

[33] Ernesto Nunes and Maria L Gini. 2015. Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints.. In *AAAI*. 2110–2116.

[34] Ernesto Nunes, Mitchell McIntire, and Maria Gini. 2017. Decentralized multi-robot allocation of tasks with temporal and precedence constraints. *Advanced Robotics* 31, 22 (2017), 1193–1207.

[35] Èric Pairet, Paola Ardón, Xingkun Liu, José Lopes, Helen Hastie, and Katrin S Lohan. 2019. A Digital Twin for Human-Robot Interaction. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 372.

[36] Pedro Patrón, David M Lane, and Yvan R Petillot. 2009. Situation-aware mission planning using distributed service oriented agents in autonomous underwater vehicles. In *International Symposium on Unmanned Untethered Submersible Technology*.

[37] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P How, Matt Vavrina, and John Vian. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*. 3998–4003.

[38] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*.

[39] Oscar Sapena, Eva Onaindıa, and Alejandro Torreno. 2014. Combining heuristics to accelerate forward partial-order planning. *Constraint Satisfaction Techniques for Planning and Scheduling* 25 (2014).

[40] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. 2017. Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems. *The International Journal of Robotics Research* (2017), 0278364918774135.

[41] Eric Schneider, Elizabeth I Sklar, and Simon Parsons. 2017. Mechanism selection for multi-robot task allocation. In *Annual Conference Towards Autonomous Robotic Systems*. Springer, 421–435.

[42] Amit Shukla and Hamad Karki. 2016. Application of robotics in offshore oil and gas industry—A review Part II. *Robotics and Autonomous Systems* 75 (2016), 508–524.

[43] Marius M Solomon. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research* 35, 2 (1987), 254–265.

[44] Sarath Sreedharan, Yu Zhang, and Subbarao Kambhampati. 2015. A first multi-agent planner for required cooperation (MARC). *Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP'15)* (2015), 17–20.

[45] Alejandro Torreño, Eva Onaindia, Antonín Komenda, and Michal Štolba. 2018. Cooperative multi-agent planning: a survey. *ACM Computing Surveys (CSUR)* 50, 6 (2018), 84.

[46] Tony T Tran, Tiago Vaquero, Goldie Nejat, and J Christopher Beck. 2017. Robots in retirement homes: applying off-the-shelf planning and scheduling to a team of assistive robots. *JAIR* 58 (2017), 523–590.

[47] Lanyong Zhang, Lei Zhang, and Sheng Liu. 2019. Role-based collaborative task planning of heterogeneous multi-autonomous underwater vehicles. *International Journal of Advanced Robotic Systems* 16, 3 (2019), 1729881419858536.

[48] Ziying Zhang, Jie Wang, Dong Xu, and Yulong Meng. 2017. Task Allocation of Multi-AUVs Based on Innovative Auction Algorithm. In *Proc. of ISCID*, Vol. 2. IEEE, 83–88.