Citation for published version (APA):
Strategic-Tactical Planning for Autonomous Underwater Vehicles over Long Horizons

Dorian Buksz¹, Michael Cashmore¹, Benjamin Krarup¹, Daniele Magazzeni¹, and Bram Ridder¹

Abstract—In challenging environments where human intervention is expensive, robust and persistent autonomy is a key requirement. AI Planners can efficiently construct plans to achieve this long-term autonomous behaviour. However, in plans which are expected to last over days, or even weeks, the size of the state-space becomes too large for current planners to solve as a single problem. These problems are well-suited to decomposition and abstraction planning techniques.

We present a novel approach in the context of persistent autonomy in autonomous underwater vehicles, in which tasks are complex and diverse and plans cannot be precomputed. Our approach performs a decomposition into a two-level hierarchical structure, which dynamically constructs planning problems at the upper level of the hierarchy using solution plans from the lower level. Solution plans are then executed and monitored simultaneously at both levels.

We evaluate the approach, showing that compared to strictly top-down hierarchical decompositions, our approach leads to more robust solution plans of higher quality.

I. INTRODUCTION

Persistent autonomy entails planning long-term behaviour for one or more autonomous systems achieving purposeful and directed activity over hours, days, or weeks without human intervention. This presents many challenges at execution time: robust execution, detection of errors, and recovery [1], [2]. However, there is a challenge that precedes execution: generating plans for missions that extend over hundreds or thousands of actions, within hours or days of activity.

We tackle this challenge in the context of managing Autonomous Underwater Vehicles (AUVs). These vehicles are tasked with maintaining a seabed facility unsupervised. The structures on the seabed must be inspected on a regular basis; the AUVs must interact with control panels within set time-windows to manage the site within time and resource constraints; and they must manage their energy budget and plan to recharge autonomously.

We build on our previous work in the context of the EU Project PANDORA², where we used temporal planning for AUV missions [2], and tested using the Girona 500 AUV (Figure 1). The main drawback of that approach, however, is that it is limited to short-horizon missions, as with longer horizons, the state space becomes too large for state-of-the-art planning systems.

In this paper we overcome this limitation, and introduce a novel decomposition approach for planning in the context of AUVs whose mission is to complete tasks over long horizons and within tight deadlines and energy constraints.

Our approach decomposes the problem into a tactical level and a strategic level. The tactical level is responsible for planning the actions needed for completing clusters of tasks (tactical plans). The strategic level is responsible for combining the tactical plans in such a way that mission time constraints and resource budget are satisfied.

Both levels take the model of the mission and use AI planning for generating the tactical and strategic plans before the mission begins. In this sense, plans for each task don’t need to be static and predefined by the user, as in the case of Hierarchical Task Network (HTN) Planning [3], [4].

The process of generating the strategic and tactical levels takes two steps: first, the mission tasks are clustered into a set of serializable subgoals [5]. A series of subgoals is serializable if there exists an ordering among the subgoals such that they can be solved sequentially without ever violating a previously solved subgoal in the order [6]. Second, the planner generates a tactical plan for each subgoal. Once these tactical plans are generated, the strategic level can make use of macro actions that encapsulate the tactical plans. Using these macro actions, the overall mission becomes simpler, and state-of-the-art planners are able to generate a strategic plan for the complete mission.

Once the mission begins, the tactical plans are regenerated online within the schedule of the strategic plan, when their initial states are known. Using this combination of offline and online reasoning, we are able to deal with very large state-spaces, while ensuring a robust execution.

To test this approach in the context of managing AUVs, we have built an underwater environment simulation (figure 2). The simulator is built in the Robot Operating System (ROS) [7]. The simulator possesses an inbuilt editor used to model missions with very long horizons, allowing us to experiment with multi-hour and multi-day missions, com-

¹Department of Informatics, King’s College London, London, WC2R 2LS firstname.lastname@kcl.ac.uk
²http://persistentautonomy.com
bining multiple tasks, such as inspection of a complex site and valve-turning, under deadlines and resource constraints. The simulation provides uncertainty about action durations, creates unpredictable features such as marine life, simulates currents, and allows the AUV to discover new features in the environment.

The paper is organized as follows. In Section II we place the contribution of this paper with respect to related work. In Section III we focus on the off-line planning phase, and define the strategic and tactical levels. In Section IV we describe the online execution of the mission plan. In Section V we present the framework and details of the architecture. In Section VI we describe the evaluation and discuss the experimental results. We conclude in Section VII.

II. RELATED WORK

Integrating planning systems with robotic systems for onboard planning in long-term missions is not a new concept, for example, NASA’s EUROPA planning framework was used for the EO-1 mission [8], the CoBot robots autonomously and robustly navigate buildings [9], the STRANDS project uses mobile service robots for long-term installations in security and care environments [10]. On the domain of planning AUVs, examples include: T-REX (Teleo-Reactive EXecutive) for reasoning onboard AUVs [11]; a planning system to track algae blooms [12]; using homotopy classes to guide the path planning [13]; using sampling-based motion planning to solve missions modelled in LTL [14]; and a planner integrated using ROSPlan with the COLA2 control architecture for AUVs in subsea intervention tasks [15], [16]. Alternative strategies for long-term autonomy typically focus on execution monitoring or onboard replanning (e.g. [11], [15], [17]).

In addition to integrating planning and robotics systems, there is a great deal of related work regarding formalisms and algorithms for plan execution onboard robots. For example, the Spacecraft Command Language (SCL) [8], Plan Execution Interchange Language (PLEXIL) and PLEXIL Executive [18], and the Petri-Net Planning formalism [19]. Executing plans onboard robots includes the dispatch of planned actions to controllers and execution monitoring, checking whether the conditions specified in the plan hold during execution. The system in which we implement strategic planning is agnostic to the plan execution system used.

The contribution of this paper is complementary to this body of work, as we focus on how to solve large planning problems through decomposition. Decomposing a goal into subgoals, in general, is not trivial. Planners such as REPLAN [20] have explored this in the past, decomposing a plan based on the number of resources available, e.g. creating a different plan for each available robot. In contrast, our system uses the decomposition and ordering of subgoals based on serializable goals [5], [6], and an observation of locality. The observation we exploit is that many long-term autonomy missions involve located executives interacting with their environment to achieve tasks. A task can be associated with the area in which operations will be performed, and with temporal constraints governing when it must be completed. These tasks can be clustered, geographically and temporally, into a set of discrete serializable subgoals.

Abstraction into a hierarchical structure is also exploited by HTN planners [3], [4], which rely on a top-down approach, exploiting pre-constructed plans to tackle separate component elements of the hierarchy. Similar to an HTN model, the strategic level in our approach contains macro actions that encapsulate plans at the tactical level. However, we focus on domains where tasks are complex and diverse, such that plans for them cannot be precomputed. We propose to use an AI planner to construct these plans automatically.

III. OFFLINE STRATEGIC-TACTICAL PLANNING

A. Mission Description

In the context of managing AUVs, the mission is to complete a set of tasks located at manifolds. A subsea manifold is a large metal structure, made up of pipes and valves and designed to transfer oil and gas from wellheads into a pipeline. Manifolds can be separated by large distances. Some of the tasks are associated with time-windows.

We consider two types of task: inspection and valve-turning. Inspection tasks require the AUV to inspect a set of joints and welds on the manifold, which involves choosing positions from which to best view the inspection points while saving time and energy. Valve-turning tasks require the AUV to set a series of valves on one or more panels to the correct angle. Valve-turning tasks are always associated with time-windows. We also consider the battery charge of the AUV. All operations decrease battery charge at different rates. The AUV is able to dock and recharge the battery at a recharging station, but this action is time-consuming and must be scheduled carefully. Formally, we define the AUV mission as a planning problem, as follows.

Definition 1: A Planning Problem is a tuple \( \Pi := \langle P, V, A, I, G, W \rangle \), where \( P \) is a set of Boolean variables; \( V \) is a vector of real variables; \( A \) is a set of actions; \( I(P, V) \) is a function over \( P \cup V \) which describes the initial state; \( G := \{ g_1, \ldots, g_n \} \) is a set of goals; \( W := \{ w_1, \ldots, w_m \} \) is a set of time windows.
States are described with Boolean and real variables. An action is applicable in a state if its preconditions are satisfied, and if applied, the state is updated according to the action effects. As an example, below is the action do_hover\(^3\) which is used to navigate the AUV between waypoints:

\[
\text{:(durative-action do_hover :parameters (?v - auv ?from ?to - waypoint) :duration ( = ?duration
(*) (distance ?from ?to)
(inverse_speed ?v))) :condition (and
(over all (vehicle_free ?v))
(at start (at ?v ?from))
(at start (>=(charge ?v)
(* (distance ?from ?to)
(discharge_rate ?v))))
(over all (connected ?from ?to)))

\]

Note that the duration of the action depends on the distance being travelled and the speed of the AUV. The numeric variable charge represents the current battery level of the AUV.

The time windows are defined using timed initial literals (TILs). A TIL \((t,p)\) describes that Boolean variable \(p\) becomes true at time \(t\). A TIL \((t,\neg p)\) describes that Boolean variable \(p\) becomes false at time \(t\).

Goals \(g_1,\ldots,g_n\) are Boolean variables or inequalities over real variables that represent what must be achieved. Temporal planners [22]–[24] take the problem description as input and search for a sequence of actions that satisfy the goals.

In the following we describe how a planning problem representing the AUV mission is used by both the tactical and the strategic levels.

**B. Tactical Level Planning**

The tactical level is responsible for planning the sequence of actions needed for completing a set of tasks, which we call tactical plans. A goal in the tactical level is a Boolean variable that represents the completion of a single task. A tactical plan includes navigation actions, valve turning, inspection actions, etc., and is associated with a duration and an amount of battery charge consumed. Formally, they are defined as follows.

**Definition 2:** A Tactical Plan is a tuple \(p := (\pi, D, C)\), where \(\pi\) is a sequence of timestamped actions constituting the plan; \(D\) is the duration of the plan; and \(C\) is the amount of battery charge consumed by the plan.

An example of a simple tactical plan to correct a single valve, generated using the planner POPF, is shown in figure 3.

\[\begin{align*}
0: & (\text{do_hover auv wp0 wp1}) \quad [134] \\
134: & (\text{check_for_panel auv wp1 ip0}) \quad [10] \\
144: & (\text{correct_auv_position auv wp1}) \quad [10] \\
154: & (\text{check_valve_state auv wp1 p0}) \quad [10] \\
170: & (\text{do_hover auv wp0 wp2}) \quad [9] \\
179: & (\text{turn_valve auv wp2 p0 v0}) \quad [180] \\
359: & (\text{correct_auv_position auv wp2}) \quad [10]
\end{align*}\]

**Fig. 3:** A tactical plan for a simple valve-turning task.

In this case, the duration \(D\), is the time at which the final action in the plan completes, that is 369 seconds, while the resource consumption, \(C\), is the total amount of decrease in the battery charge, computed by accumulating the change from each action.

In our experiments, described in Section VI, we deal with missions with up to 300 distinct inspection points, and 100 valves. We show that it is not feasible to solve problems with this many tasks purely at the tactical level, and that the strategic level abstraction is needed.

**C. Strategic Level Planning**

The strategic level is responsible for combining the tactical plans in such a way that all tasks are achieved, while mission time constraints and resource budget are satisfied. The first step, however, is to decompose the whole set of tasks into clusters that can be dealt with by the tactical level.

**Goal Decomposition:** given the overall mission, the whole set of tasks is decomposed into a set of subgoals. Each subgoal represents the set of tasks that will be completed by a tactical plan. This decomposition of tasks produces a set of serializable subgoals \(G^s := \{g_1,g_2,g_3,\ldots\}\). A set of subgoals is serializable if it is guaranteed that a solution exists, achieving the subgoals in a specific order without violating previously achieved subgoals [5]. We generate the set of subgoals \(G^s\) from the set of tasks using the observation that the tasks can be clustered, geographically around manifolds and temporally based on time-windows. The resultant set of subgoals is serializable, as in our domain achieving one subgoal does not violate an achieved subgoal, and temporally constrained tasks are represented as a distinct subgoal as described below.

With respect to the geographical clustering, a fully connected, weighted waypoint graph \((V,E)\) is constructed. All tasks and manifold centers are represented by nodes, and edges are weighted by distance. The decomposition into subgoals is generated by an agglomerative hierarchical clustering of the graph based on supplied cluster centers [25]. Nodes representing manifolds are used as initial cluster centers. This results in a set of subgoal clusters and a function \(M : g \rightarrow m\) mapping the resulting subgoals \(g \in G^s\) to a manifold location \(m\) with which they share a cluster.

With respect to temporal clustering, some tasks can be associated with time-windows. A task with a time-window has an absolute lower and upper bound on the time during which the task must be completed. The lower bound is written \(t_l(g)\) and the upper bound \(t_u(g)\). These tasks are represented as a distinct subgoal \(g\).
Strategic Plan Generation: after the subgoals have been defined, a tactical plan for each subgoal is generated. Note that at this stage the tactical plans do not take time-windows and overall resource budget into account, as it is not yet known when the tactical plans will begin execution. The following step is for the strategic level to combine the tactical plans.

The strategic level extends the planning problem with a set of macro actions, each representing a tactical plan that achieves a subgoal. For each subgoal $g$, three new Boolean variables are generated to support the macro action: location$(g,w)$, active$(g)$, and complete$(g)$. These variables are used to represent the location of a subgoal, its time-window, and whether or not it has been completed. The duration of the tactical plan is used as an estimate of the duration of the macro action. The amount of resource used throughout the tactical plan is used as an estimate for the cost of the macro action. These macro actions, which we call in the following strategic actions, are defined below.

Definition 3: Given a subgoal $g$, achieved by the tactical plan $p(g)$, a Strategic Action is the tuple $a(g) := (\text{con}, \text{dur}, \text{eff})$, where $\text{con}$ represents the condition which must hold throughout the application of the action; $\text{dur}$ is the real-valued duration of the action, estimated as the duration of the tactical plan, $p(g)_D$; and $\text{eff} := \text{complete}(g) \land (\text{charge} = p(g)_C)$ represents the effect of completing the action, including the consumption of the battery charge.

The condition which must hold throughout the application of the strategic action is defined as:

\[
\text{con} := \text{executive}_{at}(g,M(g)) \land \text{location}(g,M(g)) \land (\text{charge} > 0) \land \text{active}(g)
\]

We use an existing Boolean variable (executive$_{at}$) that describes the location of an executor – in our context an AUV – in order to condition that the strategic action cannot be applied unless the executor is in the same location as the subgoal to be completed. We use an inequality over the numeric variable charge to ensure the battery remains charged until the end of the action.

The Boolean variable active$(g)$ is used to model the time-window within which a subgoal must be completed. For each plan $p(g)$ and its subgoal $g$ representing a task with associated time-window $[t_l(g), t_u(g)]$, two TILs are generated: $(t_l^*(g), \text{active}(g))$ and $(t_u^*(g), \neg \text{active}(g))$.

\[
t_l^*(g) = t_l(g) - c_t
\]

\[
t_u^*(g) = t_u(g) - c_a + p(g)_D
\]

Where $c_t$ (resp. $c_a$) is the lower (resp. upper) bound on the time at which the task is completed in the tactical plan. The set of all such TILs is named $T_p^*$. Example of Time-Window Generation: A simple valve-turning task has the time-window $[500, 800]$, during which the manipulation of the valves must begin and end. This task becomes a distinct subgoal, and a tactical plan is generated to achieve it. The tactical plan is shown in figure 3.

The tactical plan is encapsulated as a strategic action with duration 309. The valve-turning operation begins at $c_t = 179$ and completes by $c_a = 359$. Subsequently, the time-window within which the strategic action must begin and complete its execution is calculated to be: $[321, 810]$. This is illustrated in figure 4, which shows the earliest and latest times for which the strategic action can be applied.

The two TILs generated are: $(321, \text{active}(g))$ and $(810, \neg \text{active}(g))$. Given that active$(g)$ is a condition that must hold throughout the application of the strategic action, these TILs force the planner to find a solution that adheres to the time-windows.

The time-window for the strategic action, $[t_l^*(g), t_u^*(g)]$ will be valid if and only if:

\[
c_a - c_t \leq t_u(g) - t_l(g)
\]

Intuitively, if the interval within which the executor can choose to complete the task is of greater size than the time window within which it must be completed, then there is no valid interval for the strategic action. As our problem definition states that actions have deterministic duration, then restricting the interval within which the task can be completed does not invalidate the tactical plan. It suffices to set $c_t$ so that:

\[
c_t = \max(c_t, c_a - (t_u(g) - t_l(g)))
\]

By increasing $c_t$ in this way, the choices available to the executor are restricted such that the task is achieved no earlier than the lower bound of the task’s associated time window.

Given a set $G^*$ of subgoals, a set $A^*$ of strategic actions, set $P^*$ of supporting Boolean variables, and set $T_p^*$ of TILs, the strategic problem becomes:

\[
\Pi^* := \{ P \cup P^*, V, A \cup A^*, T_p \cup T_p^*, I, G^* \}
\]

A plan for the strategic problem is a set of actions from $A \cup A^*$, which achieve the goal set $G^*$.

The process is illustrated by Algorithm 5. For each subgoal, a tactical plan is generated (line 4). This plan is used to generate a strategic action, forming an estimate of its duration and resource cost (lines 9-10) and the time-window within which it is applicable (lines 14-15).

IV. Online Execution of Strategic Missions

Once a strategic solution is found, a monolithic plan at the tactical level could be constructed by replacing the abstract
1: procedure GENERATESTRATEGICPROBLEM(Π, G, M)
2: Πs = Π, Gs = ∅ ⊿ Strategic problem and goal
3: for every subgoal g ∈ G do
4:     p(g) ← generateTacticalPlan(Π, g)
5:     Πs ← ¬complete(g)
6:     Πs ← location(g, M(g))
7:     a(g) ← New strategic action
8:     a(g)_con ← active(g) ∧ location(g, M(g))
9:     a(g)_eff ← complete(g)
10:     a(g)_dur ← p(g)_D
11:     Πs ← a(g)
12:     Gs ← complete(g)
13:     if there exist temporal constraints then
14:         Πs ← (t_s^a(g), active(g))
15:         Πs ← (t_u(g), ¬active(g))
16:         Πs ← ¬active(g)
17: else
18:     Πs ← active(g)
19: end if
20: end for
21: Πs = Gs
22: end procedure

Fig. 5: Procedure for strategic problem generation.

strategic actions with the tactical plans they represent. Then, the complete plan could be passed to a single executor. However, action durations and resource costs used offline are estimates and when executing this plan in the real world inconsistencies can grow. When the time comes to execute a tactical plan later in the mission, the initial state might be far from what was expected:

1) depending on the execution of previous tactical plans, the current amount of available resource might differ from what was expected to be available;
2) the strategic action might be available to be dispatched earlier or later than was anticipated, which affects time-windows within the tactical problem.

Robustness in AUV persistent autonomy is a key requirement. Hence, in our approach we execute the strategic plan directly, but the tactical plans are regenerated when the associated strategic actions begin execution. Time-windows and resource constraints are taken into consideration when regenerating the tactical plan. In the experimental evaluation described in Section VI, this process takes < 10 seconds, and can be performed in parallel with the execution of other actions. Execution monitoring and replanning take place on both levels simultaneously. This provides a more efficient and robust execution strategy.

The execution of a tactical plan, including any tactical replanning, is handled by an onboard executive. The approach is embedded in ROSPlan [15], as described in Section V. We use a conservative model of action duration and cost, so we expect that most tasks will be completed within the estimated time. However, when executing plans onboard a physical platform there is always the chance that actions might fail, take longer than expected, or be accomplished more quickly.

Finally, after the replanning or completion of any strategic action, the strategic plan is revalidated using ROSPlan. It is possible that a tactical plan takes longer to complete than expected, and from the current time the strategic plan no longer respects the task time-windows. Similarly, a tactical plan might take more resource than accounted for by our conservative action model, and there is no longer enough resource to complete subsequent actions in the strategic plan.

To account for these points, we use the ROSPlan executive for the dispatch of the strategic, as well as tactical plan. More generally, execution monitoring techniques developed for tactical plans are used for the execution of plans at both levels in the hierarchy.

V. IMPLEMENTATION

The process described in Sections III and IV was implemented in ROS and embedded in ROSPlan [15], a framework for embedding AI Planning in ROS. The architecture is illustrated in figure 6.

Subgoal decomposition and strategic problem generation take place in the Strategic Controller node. This node implements Algorithm 5, in which the procedure generateTacticalPlan is completed as a remote procedure call to the planning interface in the tactical level (see figure 6 (a), (b)). Once the strategic problem is generated, it is passed to the strategic planner. The resulting strategic plan schedules the completion of each subgoal, respecting their associated time-windows (if any) as illustrated by figure 7.

The planner interfaces at the strategic and tactical level are separated, to enable the use of a different solver at each level. In our context, problems at both the strategic and tactical level were represented in PDDL2.1 [21], and solved using the temporal planner POF [23].

Plan execution at both levels includes the dispatch of planned actions to lower-level controllers and monitoring of invariant conditions and temporal constraints. Strategic actions are dispatched to a tactical controller node. This node generates a problem at the tactical level, using the original problem description (see figure 6 (d)). In this tactical problem, resource constraints and time-windows [t_i(g), t_u(g)] associated with the tactical level task are included. A new tactical plan is generated, and then executed. Execution monitoring ensures that if the plan fails or becomes invalid, then any executing actions are cancelled (or allowed to complete if they cannot be cancelled). In both the tactical and strategic level the response to plan failure is replanning.

Replanning at the tactical level means that the tactical controller node regenerates the problem with updated time-windows, and calls the planner interface again. If the problem is now unsolvable, due to time constraints or lack of resource, then the tactical controller returns action failure, triggering a replan at the strategic level.
Fig. 6: The architecture of the strategic and tactical planning system that was implemented in ROS. The planning domain and mission tasks are passed to the Strategic Controller node, which performs subgoal decomposition and strategic problem generation. During these processes tactical plans are generated (a) and solved (b) offline to form strategic actions. The Strategic Controller outputs the abstracted problem to be solved and executed (c). Strategic actions are executed by first regenerating a tactical plan (d).

Fig. 7: Two tactical plans are generated for subgoals $m01$ and $m02$. Their duration and resource requirements are used to generate the strategic plan below. The strategic plan schedules the tactical plans to satisfy the temporal constraints upon the subgoals.

Replanning at the strategic level means that the strategic control generates a new strategic problem with updated time windows. This does not require a new subgoal decomposition or the generation of new tactical plans. Existing strategic actions will be rescheduled to generate a new, valid strategic plan. If the strategic problem becomes unsolvable, then the system reports failure.

As the goal set is serializable, and any task with temporal constraints is treated as an independent subgoal, the system will not fail as a result of a specific subgoal decomposition. Outside of our context, this might not be the case. In general the subgoal decomposition and strategic problem generation could be repeated during a strategic replan, using additional constraints preventing the failed decomposition. Exploring the generation of these constraints and a more general strategic replanning process is left to future work.

VI. RESULTS

We tested this approach in the context of the managing a fleet of AUVs. To do this, we have built an underwater environment simulation that allowed us to model missions with horizons of multiple days. Tasks which could be posed included the inspection of a complex site (figure 2) and valve-turning under time-windows and resource constraints. Our problems vary in size from 100 to 400 discrete tasks. The simulation provides uncertainty about action durations, creates unpredictable features such as marine life, simulates currents, and allows the AUV to discover new features in the environment.

We compare our approach with a selection of precomputed hierarchical abstractions, demonstrating that the tactical information gathered by the proposed approach leads to higher quality, and more robust solutions. We compare against a purely tactical approach to demonstrate that the scenario is too large to solve as a single planning problem, and that hierarchical decomposition is a possible solution.

A. Precomputed Hierarchical Abstractions

We used the subgoal decomposition described in Section III to generate a set of subgoals from tasks. We then found strategic planning solutions to these problems. To compare, we used the same set of subgoals in a precomputed approach. In this approach, macro actions are not generated using estimates computed by an AI planner. Instead their duration and resource usage is estimated using some prior knowledge based on similar plans. We used the duration and resource usage of the computed tactical plans as this $a\ priori$ knowledge, and used it in four different estimates:

- **mean**: a naive strategy that takes the mean resource use over all tactical plans, and assigns this as the cost of all macro actions;
- **conservative**: a conservative strategy that assigns to all...
macro actions the 80th percentile of resource use over all tactical plans;
- bucket-mean: the subgoals were divided into task sets of similar type – inspection, valve-turning, etc. – and size. Then, the mean resource use from each set was assigned as the estimated resource use for a macro action representing that set;
- bucket-conservative: the subgoals were divided in the same way as bucket-mean, but the 80th percentile from each set was used instead of the mean.

We used the AI planner POPF [23]. POPF is a temporal and metric planner, which allowed us to model the synchronisation aspects of our problems (including time-windows for interaction with valves), the constraints on energy over long missions, and to optimise plans based on a metric function. The metric to be optimised was the number of subgoals completed. POPF was given 1800 seconds and 8GB of RAM to find the best possible solution. In the strategic planning approach POPF was given 10 seconds per subgoal to perform initial tactical planning. The amount of time given to solving the strategic problem was reduced by the total tactical planning time.

Due to the extra information from the tactical level, we expected that the solutions generated using the strategic planning approach would be more efficient in terms of number of subgoals completed. Moreover, as the subgoals had already been tactically planned, the duration and resource usage of fewer subgoals would have been underestimated, and the solutions are expected to be more robust.

Table I compares the number of subgoals achieved by our strategic planning approach and the various top-down strategies in our simulation. As can be seen from the table, strategic planning achieves more subgoals than any of the top-down strategies. Table II shows the number of subgoals for which the resource usage is underestimated. In the decompositions generated by the top-down strategies, macro actions completing these subgoals have resource estimates lower than that derived from the tactical plan. As a result, these actions are very likely to run out of time and resource before successfully completing.

In the top-down approaches, the conservative approach performs the poorest, which is to be expected: the large uniform estimates for the resource usage of every subgoal leads to a solution with many unnecessary refueling actions and missed deadlines. Both mean and bucket-mean strategies lead to higher quality solutions, but are not robust, underestimating the resource use of 32% and 27% of subgoals respectively. The bucket-conservative approach performs best of the top-down approaches. The approach is the most robust, underestimating the resource requirements of only 14% of subgoals, and generating the highest quality solutions amongst all top-down approaches.

Strategic planning outperforms all top-down approaches, confirming our expectations. The variance in the resource usage between tasks of similar type and size is due to constraints that are only visible at the tactical level, and as such are not available prior to planning for a subgoal itself. With this information, the strategic solutions are the most robust – none of the tasks are guaranteed to fail. In addition, the strategic solution is of the highest quality.

<table>
<thead>
<tr>
<th>number of subgoals</th>
<th>Strategic Planning</th>
<th>Subgoals Completed</th>
<th>Underestimated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Conser -ervative</td>
<td>bucket-mean</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Table I: Comparing the number of subgoals completed by strategic planning and precomputed, top-down decompositions in missions of varying size.

<table>
<thead>
<tr>
<th>number of subgoals</th>
<th>Subgoals conser -ervative</th>
<th>Underestimated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>bucket-mean</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

Table II: The number of tasks likely to run out of time or resource during execution for different top-down representations.

B. Tactical Approach

Table III shows the results of attempting to plan missions without decomposing into a tactical/strategic hierarchy. The planner was given 1800 seconds to find the best possible plan, solving the problem purely at the tactical level. The number of tasks achieved by the best plan is shown. Strategic planning far outperforms a non-hierarchical approach. In the strategic planning approach, tactical plans have durations from 10 minutes to 3.5 hours depending upon the type and size of tasks. The longest strategic plans have durations of 72 hours. Given the need for onboard replanning, it is clear that a purely tactical approach is insufficient for these sizes of problems.
<table>
<thead>
<tr>
<th>tasks</th>
<th>Strategic Planning</th>
<th>Separate tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>106</td>
<td>106</td>
<td>9</td>
</tr>
<tr>
<td>106</td>
<td>106</td>
<td>9</td>
</tr>
<tr>
<td>106</td>
<td>106</td>
<td>9</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>9</td>
</tr>
<tr>
<td>150</td>
<td>150</td>
<td>9</td>
</tr>
<tr>
<td>212</td>
<td>212</td>
<td>9</td>
</tr>
<tr>
<td>212</td>
<td>212</td>
<td>9</td>
</tr>
<tr>
<td>212</td>
<td>212</td>
<td>9</td>
</tr>
<tr>
<td>265</td>
<td>258</td>
<td>11</td>
</tr>
<tr>
<td>265</td>
<td>250</td>
<td>9</td>
</tr>
<tr>
<td>265</td>
<td>265</td>
<td>9</td>
</tr>
<tr>
<td>310</td>
<td>270</td>
<td>8</td>
</tr>
<tr>
<td>310</td>
<td>190</td>
<td>9</td>
</tr>
<tr>
<td>310</td>
<td>270</td>
<td>10</td>
</tr>
</tbody>
</table>

TABLE III: Comparing the strategic planning subgoal decomposi-
tion against planning entirely at the tactical level, showing the
number of tasks achieved by the best plan.

VII. CONCLUSION

We have presented a novel approach to bottom-up abstrac-
tion of a planning problem into a two-level hierar-
chical structure in the context of persistent autonomy for
AUVs. Our approach is capable of solving complex planning
problems with horizons spanning multiple days. We have
demonstrated that our hierarchical approach is very robust
during execution, by simulating long-term missions by AUVs
in a dynamic environment.

In generating a strategic level there is a trade-off when
estimating the resource constraints of abstracted actions that
encapsulate collections of lower-level behaviour. A more
conservative approach is more robust, but over-estimation of
resource use leads to less efficient plans that do not utilise
all of the time and resource actually available. We show that
it is possible to use AI planning to find tighter bounds on
resource use that remain robust. We use this information in
the automatic generation of the strategic problem.

We described the subgoal decomposition used in the
context of AUV maintenance missions of a seabed facility.
This decomposition is based on geographical and temporal
clustering. This forms the foundation of a general strategic-
tactical approach. In future work we will investigate general
decompositions formed from interleaving separate subgoals.
In particular we will exploit recent integration between
planning and machine-learning techniques [26] to learn an
efficient decomposition. This will allow us to apply general
strategic-tactical planning in any of the diverse contexts in
planning for persistent autonomy.

ACKNOWLEDGEMENTS

We thank Maria Fox and Derek Long for their very useful
comments and suggestions.

REFERENCES

      Johansen, J. B. de Sousa, and K. Rajan, “Coordinating uavs and uavs
      for oceanographic field experiments: Challenges and lessons learned,”


      Preliminary approach, research challenges,” in Proceedings of 29th
      AAAI, 2015.

      recursive subgoals,” in Knowledge-Based Intelligent Information and


      R. Wheeler, and A. Y. Ng, “ROS: an open-source robot operating

[8] R. Sherwood, S. Chien, D. Tran, A. Davies, R. Castalio, G. Rabideau,
      D. Mandl, S. Frye, S. Shulman, and J. Szwaczkowski, Enhancing sci-
      ence and automation operations using onboard autonomy. Pasadena,
      CA: JPL, National Aeronautics and Space Administration, 2006.

      cobots over long-term deployments,” I. J. Robotics Res., vol. 32,

[10] N. Hawes, C. Burbridge, F. Jovan, L. Kunze, B. Lacerda, L. Mudrova,
      J. Young, J. Wyatt, D. Hebesberger, T. Kortner, R. Ambrus, N. Bore,
      J. Folkesson, P. Jensfelt, L. Beyer, A. Hermans, B. Leibe, A. Aldoma,
      T. Faulhammer, M. Zillich, M. Vincze, E. Chinellato, M. Al-Omari,
      P. Duckworth, Y. Gatsoulis, D. C. Hogg, A. G. Cohn, C. Dondrup,
      J. Pentanes, T. Krajnik, J. M. Santos, T. Duckett, and M. Hanheide,
      “The strands project: Long-term autonomy in everyday environments,”

      McEwen, “A deliberative architecture for AUV control,” in IEEE


topologically guided path planner for an auv using homotopy classes,”

to enhance autonomy of underwater vehicles operating in the littoral
zone,” in Workshop on Combining Task and Motion Planning at IEEE
ICRA, 2013.

      N. Palomeras, N. Hurtos, and M. Carreras, “Rosplan: Planning in the

      control architecture for uavs,” IEEE Journal of Oceanic Engineering,

[17] B. D. Smith, K. Rajan, and N. Muscettola, “Knowledge acquisition for
      the onboard planner of an autonomous spacecraft,” in EKAW, 1997.

[18] V. Baskaran, M. Dalal, T. Estlin, C. Fry, M. Iatauro, R. Harris,
      A. Jonsson, C. Pasareanu, R. Simmons, and V. Verma, “Plan exec-
      ution intermachine language (pxml) version 1.0,” in NASA Technical

      tical framework for robust decision-theoretic planning and execution


      additive heuristic for temporal and numeric planning,” in Proceed-
      ings of ICAPS, 2009.

[23] A. Coles, A. Coles, M. Fox, and D. Long, “Forward-chaining partial-

      the full PDDL+ language into SMT,” in Proceedings of ICAPS, 2016.

      An overview,” vol. 2, pp. 86–97, 01 2012.

[26] S. Krivic, M. Cashmore, D. Magazzini, B. Ridder, S. Szedmak, and
      J. Piater, “Decreasing Uncertainty in Planning with State Prediction,”
in International Joint Conference on Artificial Intelligence. IJCAI, 8