

# Difficulty-Aware Time-Bounded Planning under Uncertainty for Large-Scale Robot Missions

Michał Staniaszek, Lara Brüdermüller, Raunak Bhattacharyya, Bruno Lacerda, Nick Hawes

**Abstract**—We consider planning problems where a robot must visit a large set of locations to complete a task at each one. Our focus is problems where the *difficulty* of each task, and thus its duration, can be predicted, but not fully known in advance. We propose a general Markov decision process (MDP) model for *difficulty-aware* problems, and propose variants on this model which allow adaptation to different robotics domains. Due to the intractability of the general problem, we propose simplifications to allow planning in large domains, the key being constraining navigation using a solution to the travelling salesperson problem (TSP). We build a set of variant models for two domains with different characteristics: UV disinfection, and cleaning, evaluating them on maps generated from real-world environments. We evaluate the effect of model variants and simplifications on performance, and show that our models outperform a rule-based baseline.

## I. INTRODUCTION

Many real-world mobile robot applications such as cleaning, visual inspection, and environmental monitoring involve missions where the robot must execute tasks to *service* each of a set of locations within a time bound.

In this paper, we identify and model a general class of such problems where the duration of actions required to make progress on a specific task is influenced by its *difficulty*. Because the factors which typically make a robotic task difficult, such as environment dynamics or state estimation uncertainty, can only be observed at execution time, we use probabilistic models to predict the difficulty at each location. This yields a problem which requires *planning under uncertainty*. To keep within the time bound, the system must take into account the fact that difficulty affects task duration. Time bounds may be imposed by various sources, such as operational requirements to complete tasks before a certain time, times when humans are in the environment, or weather. We refer to *difficulty-aware* planning problems as those where 1) difficulty of tasks can be modelled probabilistically, 2) task duration depends on difficulty and can also be modelled probabilistically, and 3) a location’s task difficulty can be observed online when the robot reaches it.

The class of difficulty-aware problems naturally captures a wide range of current robotics applications: 3D reconstruction of human spaces, where difficulty is due to scene complexity and dynamics [1]; underwater asset inspection, due to uncertain communication and currents moving the vehicle

as it captures data [2], [3]; and UV disinfection, due to localisation uncertainty [4], [5].

To solve a difficulty-aware planning problem, the planner has to jointly consider two problems: the order in which to visit locations (ordering), and how much time to spend servicing each location (time allocation). The ordering problem requires the planner to implicitly solve a TSP, so planning for difficulty-aware missions is computationally challenging even for small problems. As originally proposed by Lane and Kaelbling [6], we exploit the fact that these two problem aspects can be decoupled to apply a hierarchical planning approach. We solve the TSP to produce a tour of locations, following that tour while solving the difficulty-aware time allocation problem. Decoupling ordering in this way reduces both the action and state spaces of the MDP and allows our models to scale to much larger problem instances.

Our main contribution is a novel model for difficulty-aware time-bounded planning problems under uncertainty, which allows a wide range of mobile robot missions to be expressed as a reward maximisation problem in an acyclic MDP with a set of terminal states. We present variants of the model to indicate its adaptability to robotics problems, and use state space reduction methods to scale to large problem instances of over 250 tasks, showing that using a TSP to constrain ordering has the greatest effect. We validate our approach through a systematic evaluation of planning and simulated execution in two robotics domains.

## II. RELATED WORK

The core of our paper is a time-bounded planning problem under uncertainty. Such problems are commonly formulated as finite-horizon MDPs [7]. We consider a variation where we include time in the state and allow actions to take variable amounts of time. We are specifically interested in problems where reward-generating actions have stochastic duration, but the duration distribution for each action depends on the task difficulty, which is an observable state factor. Prior work has formulated related time-bounded planning problems using MDPs, such as an approach for speeding up the solution of time-bounded planning problems with multiple objectives, minimising time taken while maximising reward [8]. While we do not consider multiple objectives, our model could encode their example and handle larger problem instances. When action duration distributions are not known in advance, a Gaussian process (GP) can be used to maintain a belief over environmental features which influence the distributions, and sampled from when planning [9]. As we assume difficulty is

a discrete set of values which do not influence each other, a GP is overly complex for our needs.

Including uncertainty as a state factor is a way of avoiding formulating problems as a more complex and harder to solve partially observable Markov decision process (POMDP), instead creating an *Augmented MDP* [10], [11]. Those works consider localisation uncertainty, and we represent difficulty a similar way in our UV disinfection example domain. We discretise a distribution on the location variance and use it as a state factor representing different degrees of uncertainty.

Visiting a set of locations under a time-bound has been investigated in the literature on the orienteering problem (OP) [12]. In the OP, the goal is to visit a subset of a given set of locations that provides the maximum reward given a time bound. Probabilistic extensions to the OP have also been investigated where the associated reward with each location is stochastic [13], [14]. Both robotic exploration [15] and persistent monitoring problems [16] have been approached using solutions to the OP. However, our problem differs in that the reward is not obtained simply by visiting each location. It depends on the actions performed, and consequently the time spent, at a location.

Prior work has used a TSP to simplify practical robotics navigation problems by forcing an ordering of tasks [6]. They use a TSP to enforce ordering on a set of navigation macro-actions, considering uncertainty only at the macro-action level, and do not consider time as a factor in the model. We show that the same technique can be useful without the macro action decomposition, and in MDPs which must consider uncertainty in time coming from multiple sources. More formal hierarchical planning methods can be used to reduce the state space of a large MDP by breaking it into sub-problems. Some methods use new algorithms to convert and solve a problem in a hierarchical manner [17], [18], while others use standard MDP solution methods on hand-designed or learned hierarchies [19]–[21]. Both approaches impose an additional burden on the model designer, requiring either implementation of non-standard solution algorithms, or manual design and building of hierarchies. Our models are standard MDPs, with domain-specific simplification approaches.

### III. PRELIMINARIES

A *Markov decision process (MDP)* is a tuple  $\mathcal{M} = \langle S, \iota, A, T, R, \gamma \rangle$ , where  $S$  is a finite set of states;  $\iota$  is a probability distribution over the initial state;  $A$  is a finite set of actions;  $T : S \times A \times S \rightarrow [0, 1]$  is a probabilistic transition function returning the probability of arriving at state  $s'$  after taking action  $a$  in state  $s$ ;  $R : S \times A \rightarrow \mathbb{R}$  a function returning a reward for performing  $a$  in state  $s$ ; and  $\gamma \in (0, 1]$  is a discount factor. The aim for an MDP is to find an optimal *policy*  $\pi^* : S \rightarrow A$  that maximises the expected cumulative discounted reward:

$$\pi^* = \operatorname{argmax}_{\pi} \mathbb{E}_{\pi}^{\pi} \left[ \sum_{i=0}^{\infty} \gamma^i R(s_i, a_i) \right] \quad (1)$$

The optimisation objective may diverge when  $\gamma = 1$ , but if all runs of the MDP reach a terminal state from which no more reward can be gathered, the objective converges to a finite value regardless. All runs of our model reach a terminal state as mission duration is bounded, so we consider  $\gamma = 1$ .

### IV. PROBLEM FORMULATION

We consider a mobile robot that navigates in a discrete *topological map*. A topological map is a tuple  $\mathcal{T} = \langle V, E, dur_{nav} \rangle$ , where  $V = \{v_1, \dots, v_n\}$  is a set of locations in the environment represented by poses of the form  $(x, y, z, \theta)$  in a global frame;  $E \subseteq V \times V$  encodes a set of directed edges the robot can traverse; and  $dur_{nav} : E \rightarrow \mathbb{N}$  is a function which maps edges to travel durations. The goal is for the robot to navigate around the locations and *service* them by executing an action (e.g. clean or take a sensor reading). The action set available to the robot is to service its current location  $v$ , or traverse an edge  $(v_c, v') \in E$ .

Service actions increase the *service level* at the robot's current location. Service levels are denoted by  $l \in \mathbb{N}_L$ , where  $L \in \mathbb{N}$  and  $\mathbb{N}_L = \{0, \dots, L\}$ . These actions have a *difficulty*, modelled as a function  $diff : V \rightarrow \mathbb{N}_D$ , where  $D \in \mathbb{N}$ ,  $\mathbb{N}_D = \{0, \dots, D\}$ , which can vary according to the location. This function is unknown a priori, but we have access to a discrete distribution over the difficulty level at each location. For  $v \in V$  and  $d \in D$ ,  $P(diff(v) = d)$  is the probability of the difficulty at node  $v$  being  $d$ . There is also a *utility function*  $U : V \times \mathbb{N}_L \times \mathbb{N}_L \times \mathbb{N}_D \rightarrow \mathbb{R}_{\geq 0}$  such that  $U(v, l, l', d)$ , known a priori, is the utility of moving the service level at  $v$  from level  $l$  to level  $l'$ ,  $l' > l$ , given that the difficulty at  $v$  is  $d$ .

The difficulty level at a location impacts the *service duration*. We assume a discrete set  $\Lambda = \{\lambda_1, \dots, \lambda_{|\Lambda|}\} \subset \mathbb{N}$  of  $|\Lambda|$  representative durations, obtained by discretising a continuous distribution over possible durations. The exact mapping between difficulty and duration is unknown, but we assume that we have access to the probability of the robot taking duration  $\lambda$  to change the service level at  $v$  from  $l$  to  $l'$  under difficulty  $d$ , where  $v \in V$ ,  $l, l' \in \mathbb{N}_L$  with  $l' > l$ ,  $d \in \mathbb{N}_D$  and  $\lambda \in \Lambda$ . We denote this as  $P(dur_{serv}(v, l, l', d) = \lambda)$ .

Note that we make two core assumptions, neither of which reduce applicability to practical robotics problems. 1) The deterministic duration set  $\Lambda$  requires that possible durations be known in advance. 2) Difficulty distributions are independent, so difficulty at one location cannot influence that at another.

*Mission Goal:* Given an initial node  $\bar{v} \in V$  and time bound  $B \in \mathbb{N}$ , find a policy which decides the navigation and servicing actions to execute to maximise the expected sum of gathered utility  $U$ , while ensuring the robot returns to  $\bar{v}$  within  $B$  units of time.

### V. MODELLING

#### A. General MDP Model

We start by defining a general MDP model of the time-bounded mission described above. We define  $n = |V|$  and, for set  $X$ , denote the Cartesian product of  $X$  with itself  $n$  times as  $X^n$ . Given a set of discrete durations  $\Lambda$  and a time

bound  $B$ , we define  $\Lambda_B^+$  as the set of all sums of elements of  $\Lambda$  that are less than or equal to  $B$ .

*States:* The state space is of the form  $S = V \times \mathbb{N}_L^n \times \mathbb{N}_D \times \Lambda_B^+$ . A state  $s = (v_i, l_1, \dots, l_n, d_i, \tau)$  means that robot is at location  $v_i$ , for some  $1 \leq i \leq n$ ; the service level at each location  $v_j$  is  $l_j$ , for each  $1 \leq j \leq n$ ; the difficulty level at the current location  $v_i$  is  $d_i$ ; and the elapsed time since the start of the mission is  $\tau$ . The probability of the initial state being  $(\bar{v}, 0, \dots, 0, d, 0)$  is  $P(\text{diff}(\bar{v}) = d)$ , i.e. the robot starts at initial location  $\bar{v}$ , the service level at every location is initialised as 0 and the mission starts at time 0; the difficulty at  $\bar{v}$  is defined according to  $P(\text{diff}(\bar{v}) = d)$ .

*Actions and Transitions:* In state  $s$ , the robot can move between locations using a topological map edge with action  $a_e$ . Taking  $a_e$  to traverse  $e = (v_i, v') \in E$ , the current location changes to  $v'$ , and the time component of the state changes to  $\tau' = \tau + \text{dur}_{\text{nav}}(e)$ . The difficulty changes stochastically to  $d'$ , according to  $P(\text{diff}(v') = d')$ .

The robot can increase the service level at its current location to some  $l' > l_i$  with action  $a_{l_i, l'}$ . Taking service action  $a_{l_i, l'}$  with target service level  $l'$ , the current service level  $l_i$  changes to  $l'$ , and the time component is updated to  $\tau' = \tau + \lambda$  stochastically, according to  $P(\text{dur}_{\text{serv}}(v, l_i, l', d) = \lambda)$ . Service actions can only increment the service level, such that for all  $a_{l_i, l'} \in A$ ,  $l' = l_i + 1$ .

To ensure the robot returns to  $\bar{v}$  within time bound  $B$ , we disallow actions that have some probability of transitioning to a state from which it is not possible to return to  $\bar{v}$  in time. For some action  $a \in A$ , let  $\lambda_{\max}^a$  denote the maximum possible time increment according to the duration model of  $a$ ; and let  $\lambda(v^a, \bar{v})$  be the time to travel to  $\bar{v}$  from the location the robot will be at after executing  $a$ , according to the shortest path in  $\mathcal{T}$ . If  $\tau + \lambda_{\max}^a + \lambda(v^a, \bar{v}) > B$ , then  $a$  is not enabled in  $s$ . States with no action enabled have a *return to start* action which moves the robot from  $v_i$  to  $\bar{v}$  by the shortest path in  $\mathcal{T}$ , and puts the model into a terminal state from which no more reward can be gathered.

*Reward Function:* The reward function returns a reward based on the utility  $U$ , known a priori, of service actions, which may have different utility depending on the difficulty, service level, or location. Other actions do not give a reward.

$$R((v_i, l_1, \dots, l_n, d_i, \tau), a) = \begin{cases} U(v_i, l_i, l', d_i) & \text{if } a = a_{l_i, l'} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

*Optimisation Objective:* Optimise the expected cumulative reward on the proposed MDP according to Equation 1. We construct the MDP to ensure all policies return to the start node within the time-bound  $B$ , after which no more reward can be gathered. Thus, maximising the cumulative reward matches the mission goal stated in the problem formulation.

## B. Variants

Our model makes assumptions about how difficulty should be tracked and how the time component is incremented. We now discuss some variations on those assumptions which can be used to model different types of problems.

*Endogenous/Exogenous Difficulty:* The difficulty of servicing a task can be internal or external to the robot. For example, the difficulty of tasks based on accurate localisation depends on the robot's quantification of its localisation uncertainty, while the difficulty of cleaning an area can be related to the amount of dirt detected there. The former is an example of *endogenous* difficulty and the latter of *exogenous* difficulty. Assuming a static world, once the robot observes a level of exogenous difficulty at a given location, that level remains the same until the end of the mission. Difficulty in endogenous processes changes whenever the robot observes the value of that internal process again. The model in Section V-A assumes endogenous difficulty, so the state only tracks the difficulty value at the current node. Exogenous difficulty can be specified by adding a difficulty state feature for each location, replacing  $d_i$  by a set of state factors  $\{d_1, \dots, d_n\}$ . This changes the state factor for difficulty from  $\mathbb{N}_D$  to  $\mathbb{N}_D^n$ . The difficulty for each location is initialised to 0, indicating unknown difficulty. When the robot visits the location for the first time, the difficulty is set to the value retrieved from  $\text{diff}(v)$ . After the difficulty for a location is set, it does not change.

*Informative Action Durations:* In our general model, we assume that the execution of a service action at a node  $v$  does not provide extra information on the duration of further service actions on  $v$ . However, there are situations where the duration of a service action in a node informs the duration of further actions there. For example, when applying UV radiation to a surface, how long it takes to apply a certain dose depends on the pose of the robot and is stochastic. Once the time taken is observed, the time to apply the dose again in the same pose does not change. In these cases, the value of  $P(\text{dur}_{\text{serv}}(v, l, l', d) = \lambda)$  depends on the duration values previously observed when servicing  $v$ . We consider a special case of this, where the observed duration  $\lambda$  of an initial service action determines the duration for subsequent service actions at a location. One additional state factor is required, which starts as 0 to indicate an unknown duration, and is set to  $\lambda$  once a service action is executed and its duration observed. If the uncertainty over the duration is exogenous, it is necessary to keep a factor for each location. The value of the factor is the deterministic duration for further service actions there. In the endogenous case this value is reset when revisiting a location. In the exogenous case, its value can be fixed once it is observed as it is a feature of the environment.

## VI. EXAMPLE DOMAINS

We define two domains to show how our proposed models can be adapted to robotics problems with different properties.

*UV disinfection:* In the UV disinfection problem, the aim is to use a robot to disinfect a series of locations by irradiating surfaces with UVC light in the 100-280nm range to apply a dose which will achieve log reductions in microbial activation [4]. A 1-log reduction indicates 90% inactivation of a microbial colony, 2-log 99%, and so on.

We assign four service levels, 0 indicating the location has not been cleaned, the other levels corresponding to 1,

2, and 3-log reductions in activation. The dose applied to a surface is proportional to the inverse square of distance to the UV source, which means that small variations in the robot's position can have a significant effect. As such, we define the difficulty as the robot's metric localisation uncertainty, discretised into three levels representing high (0), medium (1), and low (2) confidence in its location.

To generate difficulty distributions  $diff(v)$  we use empirical localisation uncertainty data from navigation of a Scitos X3 robot in Oxfordshire County Library (the library map in Fig. 1). We cluster the data using a Gaussian mixture model for each confidence level. A location's distribution is based on how often the localisation uncertainty there is matched by each mixture model. This data is used over all maps by randomly sampling from the set of location distributions.

The service duration distribution  $dur_{serv}$  is generated by sampling poses from the GMM and using an irradiation simulation to determine the duration required to apply the dose required to reach service level 1 for each difficulty. We fit a categorical distribution to these samples with a discretisation of 5s. As the service levels are log-linear, we can multiply the durations for  $dur_{serv}(v, 0, 1, d)$  by two to get the distribution for level 2, and multiply by 4 to get the durations for level 3. This domain has rewards with diminishing returns, to encourage higher coverage. 100 reward is given for service level 1, 50 for 2, and 25 for 3. For further details, see the supplementary material<sup>1</sup> or our previous work [22].

*Cleaning:* In the cleaning domain, the robot must fully clean floors in various locations, which may have different sizes and dirtiness levels. There are only two service levels, as a location is either clean or not clean. A location's dirtiness corresponds to the difficulty, affecting how long the robot takes to clean it. We track the difficulty for each location, as it is exogenous. A location can have no dirt (0), or low (1), medium (2) or heavy (3) dirt. The robot only knows how dirty a location is after visiting it, and could be determined using computer vision or human input. During model building  $diff(v)$  for each location is sampled from a set of 3 artificially generated distributions by uniform random selection. Each location is randomly designated as a small, medium, or large room.  $dur_{serv}$  is deterministic, found by multiplying the duration for the difficulty by the location's size. For servicing a location with difficulty  $d$ , the system receives  $100d$  reward.

## VII. STATE SPACE REDUCTION STRATEGIES

As our model has state factors which depend on the number of locations, the resulting MDP is impractical to solve for anything more than trivial problems. Combining simple strategies can greatly reduce the state space.

### A. Fixed Navigation

The most drastic state space reduction comes from limiting navigation on the topological map to a tour generated by solving a TSP [6]. Let  $V_{next} : V \rightarrow V$  be the mapping from a location to the next location to visit. It is defined by a

TSP tour constrained to the topological graph, generated by an optimal TSP solver [23], which visits all locations. This prunes possible edge transition actions  $a_e$  at location  $v_i$  by making the only valid edge  $e = (v_i, V_{next}(v_i))$ . It can be the case that  $(v_i, V_{next}(v_i)) \notin E$ , as to get from  $v_i$  to  $V_{next}(v_i)$  requires traversing through an intermediate location in the topological map. We redefine  $dur_{nav} : V \times V \rightarrow \mathbb{N}$  to give the shortest duration path between any two locations. This simulates a fully connected map and means that  $a_e$  can be used to jump directly from one location to another without having to traverse individual edges along the shortest path. The tour takes time  $B_{TSP}$ , based on the sum of  $dur_{nav}$  for the traversed edges. The remaining budget to service all locations is  $B_{MDP} = B - B_{TSP}$ . By pruning actions according to  $V_{next}$ , it is impossible for the model to return to any previous location, so the state only needs to track state factors relevant for the current location. The service level state factors for all nodes go from  $\mathbb{N}_L^n$  to  $\mathbb{N}_L$ . The state  $s = (v_i, l_1, \dots, l_n, d_i, \tau)$  becomes  $s = (v_i, l_i, d_i, \tau)$ , greatly reducing the number of dimensions in the Cartesian product.

### B. Single Service Action

We change the action  $a_{l,l'}$  such that  $l$  must be 0. This compresses actions for incrementally increasing the service level at a location into a single action to go from service level 0 to any service level  $l'$ .

The probability distributions  $P(dur_{serv}(v, l, l', d))$  for incremental actions must be collapsed to represent a transition from service level 0 to  $l'$  without reaching intermediate service levels. This can be done by taking the product of the probability distributions for each incremental service level, i.e.  $P(dur_{serv}(v, 0, l', d)) = \prod_{l=0}^{l'-1} P(dur_{serv}(v, l, l+1, d))$ . This leads to a reduction of choices available at each location. With incremental actions there is a choice at each service level increment to either continue servicing or leave. With single actions, it is only necessary to choose which (if any) service level to achieve, without having to make the same decision again at each increment. This is limiting, as the policy cannot choose to abort servicing early if the service time observed from  $dur_{serv}(v, l, l', d)$  is unfavourable, and if service times are favourable it cannot exploit the extra time to return to locations for further servicing.

The state factor for the service level at each location  $l_i$ , which contributes  $\mathbb{N}_L^n$  to the state space, can become Boolean  $\mathbb{N}_L = \{0, 1\}$  instead of an integer value  $\mathbb{N}_L = \{0, 1, \dots, L\}$ , greatly reducing the model's state space. The outcome of any action  $a_{0,l'}$  is that  $l_i = 1$ . The simplification relies on implicit encoding of the service level  $l'$  in the action. We maintain the reward by summing values of  $U$  for each increment.

## VIII. EXPERIMENTAL EVALUATION

We will refer to models based on which variants and reduction strategies they use: incremental (I) or single (S) actions; stochastic (ST), informative action duration (AD), or deterministic (D) service duration distributions (in the deterministic case, for a given level of difficulty,  $dur_{serv}(v, l, l', d)$  has a single outcome  $\lambda \in \Lambda$  with probability 1); and free (FN)

<sup>1</sup>robots.ox.ac.uk/~michal/papers/difficulty-aware-sup.pdf

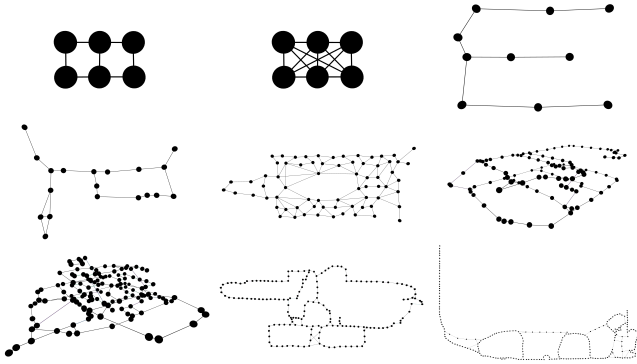


Fig. 1. Topological maps used in experiments. From left to right and top to bottom: tiny (6 locations), tiny\_fc (6), warehouse (10), foyer (19), library (70), hall (107), oilrig (138), plant (172), fence (257). See supplementary material for examples of environment pointclouds.

or fixed (TSP) navigation. For example, the model described in Section V-A has incremental service actions, stochastic action transitions, and free navigation, so is written as I-ST-FN. For the UV domain there are six models with endogenous difficulty. Single action and informative action duration models are mutually exclusive, so we have three free (I-AD-FN, I-ST-FN, S-ST-FN), and three fixed navigation models (I-AD-TSP, I-ST-TSP, S-ST-TSP). The cleaning domain has only a single service level, and its service duration distribution is deterministic, so it has two models, S-D-FN and S-D-TSP, with exogenous difficulty. Models are evaluated by solving their MDP with the PRISM model checker [24], which generates an optimal policy using value iteration. All models were solved using 64GB RAM, 16GB of swap space, and an Intel Core i7-8700 CPU at 3.20GHz. Models are evaluated on the maps in Fig. 1, with all except the tiny map generated from visits to real environments.

#### A. Domain Baselines

We compare our models to a rule-based *baseline behaviour* to quantify performance and highlight the benefits of planning.

1) *UV Baseline*: Allocates uniform service duration  $\beta = B_{MDP}/|V|$  to all locations in the map, and follows the TSP tour. With no stochasticity in difficulty or time, constant service time per node should always reach the same service level. In practice this is not the case and the level reached at different locations will vary.

2) *Cleaning Baseline*: Follows the TSP tour, greedily cleaning each location for the deterministic service time computed from its size and dirtiness, while ensuring it can return to the start location within the time bound.

#### B. Time Bound Selection

Models are evaluated with three different time bounds  $B = B_{TSP} + B_{MDP}$ .  $B_{MDP}$  is defined by  $\alpha_1$ ,  $\alpha_2$ , and  $\alpha_3$ , where  $\sum_i \alpha_i = 1$ , as described below. The purpose of the  $\alpha$  values is to generate time bounds proportional to the number of nodes in a map, and to use knowledge of the service levels and expected times to define how constraining the bound is on the service level and number of nodes that can be serviced.

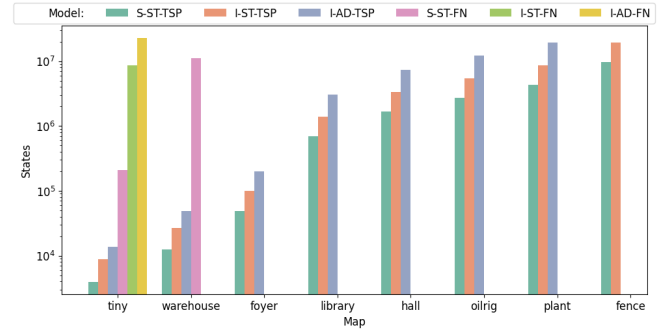


Fig. 2. Number of states in MDPs for the UV domain across topological maps of different sizes. Models which we are unable to solve due to memory constraints are omitted.  $B$  for each map is defined by  $\alpha_2 = \alpha_3 = 0.5$ . tiny\_fc is omitted as it is identical to tiny, except I-AD-FN is unsolvable.

1) *UV Time Bound*: Values of  $\alpha_s$  correspond to what proportion of locations should be serviced to level  $s$  in the ideal case, e.g. if  $\alpha_1 = 0.5$ , and  $\alpha_2 = 0.5$  we expect half of nodes should be serviced to level 1 and half to level 2.  $t_s$  is the time to reach level  $s$  at all locations assuming a deterministic setting. The bound is then  $B_{MDP} = \sum_{s=1}^3 \alpha_s t_s$ .

2) *Cleaning Time Bound*: The time to service a location depends on the difficulty. We compute the expected number of locations for each difficulty level, and the expected time to clean a location with a specific difficulty regardless of the size of the location. The dot product of the resulting vectors gives the total expected time  $\mathbb{E}(t_s)$  to clean the expected number of nodes at each difficulty. The bound is then  $B_{MDP} = \sum_{s=1}^3 \alpha_s \mathbb{E}(t_s)$ , with values of  $\alpha$  controlling the bound according to the proportion of locations each difficulty we might expect to be serviced, e.g. setting  $\alpha_1 = 0.5$  and  $\alpha_2 = 0.5$ , the model should have enough time to clean half of locations with low dirt, and half with medium dirt.

#### C. Results

1) *State Space and Solution Time*: Figure 2 shows the states for models across all maps in the UV domain. The number of states required to represent the I-AD-FN on the smallest map is larger than that required for S-ST-TSP on the largest map. We were unable to generate solutions for any free navigation models for maps larger than 19 nodes due to memory limits. In the cleaning domain, with the largest time bound on the tiny map, S-D-FN has around  $1.02 \times 10^7$ , while S-D-TSP has 457. Even in the 257 node map with the largest bound S-D-TSP has only  $1.17 \times 10^6$  states. The growth of the state space for free navigation models appears approximately log-linear in the number of nodes, but also depends on the average degree of the topological map.

TABLE I  
SOLUTION TIME IN SECONDS FOR FIXED NAVIGATION MODELS ACROSS  
SELECTED MAPS WITH HORIZON  $\alpha_2 = \alpha_3 = 0.5$ .

Model	foyer	library	hall	oilrig	plant	fence
I-AD-TSP	14	773	2767	6378	16431	N/A
I-ST-TSP	4	125	555	974	1956	7540
S-ST-TSP	2	74	269	570	1181	4074

Table I shows solution time across selected maps. The number of states is closely related to solution time of the MDP. In the map with 6 locations, for the longest horizon PRISM required 30m to solve I-AD-FN, the most complex model, but less than 0.2s to solve S-ST-TSP, the simplest. The disparity between the fixed navigation models becomes clearer on the library map, where the horizon is around 50 minutes. For the largest map of 257 locations, with the time bound defining 3 hours of robot operation, solution time for S-ST-TSP is a little over an hour. Computation time on the tiny\_fc map for free navigation models is over twice that of the same models on the tiny map, as a result of increased choices and transitions in the MDP.

2) *Service levels*: We evaluate our models by executing the policies they generate in a discrete event simulator which evolves the state according to the dynamics of a model without simplifications. Any variation in reward should show the effects of the simplifications which do not directly map to the behaviour of the world represented by the unsimplified model. To represent the world, we use the *world model* I-AD-FN for the UV domain, and S-D-FN for the cleaning domain, as they are closest to how we expect the real world to evolve. The simulator does not require solution of the MDP of the world model, so we are able to run simulations of even the largest maps. We run policies generated from other models on the simulator. We can only compare all models on the tiny map as we are unable to solve the full set of free navigation models on larger maps due to memory constraints.

Fig. 3 shows the difference in service levels achieved between the baseline and models. In the UV domain, models always achieve a higher average service level across all nodes, and are much closer to achieving the service levels we would expect based on  $\alpha$ . Compared to the rule-based baseline, planning always achieves higher average service levels across all nodes in the UV domain. In a deterministic setting, with time bound from  $\alpha_1 = 1$ , the baseline should always reach service level 1, but the majority of locations are not serviced as the time allocated is insufficient. Planning achieves cleaning level 1 in almost all locations.

The cleaning domain has only two service levels, so we show the average number of nodes at each dirtiness level after the policy has been executed (Fig. 4). The baseline and models have access to the same service duration information, but after policy execution the S-D-TSP model receives more reward on average for shorter time bounds. As the time bound increases, there is less difference between planning and the baseline, as slack in the budget allows the baseline to gather reward at every location regardless of uncertainty.

3) *Rewards*: Table II shows the mean reward for various models on the library map for the UV and cleaning domains over 1000 simulations. Our models always outperform the baselines, even in the cleaning domain with deterministic service action duration. On the 6 node map, mean rewards across all models are within approximately 1.5% of each other. This is also the case for the subsets of models we can solve for larger maps. This indicates that for small maps the state reduction strategies enable scalability while retaining

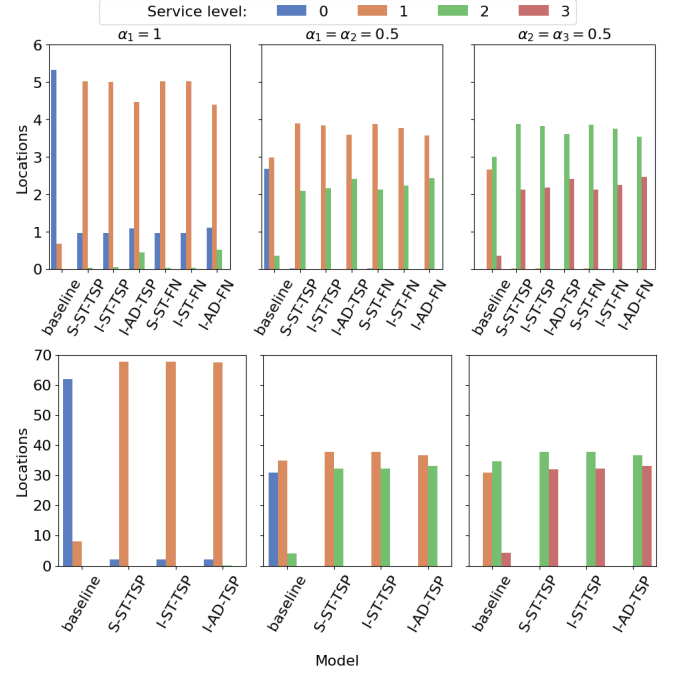


Fig. 3. Average number of locations at specific service levels achieved in the UV domain on the tiny (top) and library (bottom) maps. Other maps have similar results. 1000 policy executions per model on each time bound as specified by  $\alpha$  values. Standard error is less than 1% in all cases.

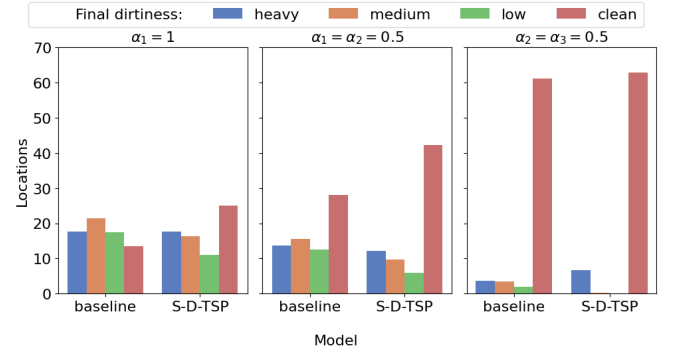


Fig. 4. Average number of locations with specific dirtiness after policy execution in the cleaning domain by baseline and S-D-TSP model on the library map. 1000 policy executions per model for each time bound specified by  $\alpha$  values. Standard error is less than 1% in all cases.

performance. We performed this evaluation with uniform and non-uniform probability distributions to test whether more uncertainty has a large effect. Uniform distributions show a slight increase in variation, but all models are still within approximately 2% of each other. The result of simulating policies gives very similar reward values to the expected policy value from PRISM. Over 1000 policy executions, the mean reward for all models in simulation is within 1% of the expected value of the policy, with the largest standard deviation equivalent to a difference of fully servicing at most two locations. On the tiny map, the only one where we can run all models, we see very little difference in reward values for fixed and free navigation variations. This is likely due to the simplicity of the map, where there is little advantage to free navigation models in making choices about navigation.



TABLE II

TOTAL REWARD ON THE LIBRARY MAP OVER 1000 POLICY EXECUTIONS.

UV	$\alpha_1 = 1$	$\alpha_1 = \alpha_2 = 0.5$	$\alpha_2 = \alpha_3 = 0.5$
Baseline	$804 \pm 261$	$4104 \pm 211$	$9058 \pm 107$
S-ST-TSP	$6781 \pm 78$	$8608 \pm 65$	$11302 \pm 32$
I-ST-TSP	$6783 \pm 75$	$8609 \pm 63$	$11305 \pm 34$
I-AD-TSP	$6784 \pm 79$	$8662 \pm 67$	$11329 \pm 30$
Cleaning	$\alpha_1 = 1$	$\alpha_1 = \alpha_2 = 0.5$	$\alpha_2 = \alpha_3 = 0.5$
Baseline	$2666 \pm 169$	$5534 \pm 298$	$12094 \pm 511$
S-D-TSP	$4615 \pm 117$	$8092 \pm 188$	$12151 \pm 302$

On the warehouse map, the mean reward for the S-ST-FN model over 1000 simulations for the shortest time bound was 1088, while the three fixed navigation models were between 905 and 912. For the foyer map, solving the MDP for the shortest time bound took 32 hours. The expected reward value was 1873, as opposed to the fixed navigation models having reward of approximately 1790. Both of these differences are significant according to a two-sided Kolmogorov-Smirnov test, which further indicates that models with free navigation may be able to get more reward on larger maps. Comparing results on the tiny maps, connectivity also affects reward distribution, with free navigation models performing up to 2% better for some models. We hypothesise that this difference may be clearer on larger maps, but we are unable to test this due to memory constraints on the free navigation models.

## IX. CONCLUSIONS

We propose a general MDP for robot task planning under uncertainty which considers task difficulty, and apply simplifications to solve it for large topological maps in two representative robotics domains. We greatly reduce the state space of the MDP by constraining the navigation with a TSP, decoupling the ordering and time allocation problems. On small maps, this simplification results in similar expected reward to policies generated by unsimplified models. We show that our models always outperform a rule-based baseline. Unconstrained navigation with the general model should outperform simplified models on larger maps, but even with our simplifications, evaluating this will require different solution approaches.

Future work will investigate the use of approximate techniques such as Monte Carlo tree search [25] and labeled real-time dynamic programming [26], or further simplification through hierarchical planning methods to find solutions. We aim to implement our models on a physical robot, where the time bound might be set using expected battery life. A potential difficulty is that real world task execution times may not easily map onto our assumption of a fixed set of possible task durations.

## REFERENCES

- [1] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the Art on 3D Reconstruction with RGB-D Cameras," *Computer Graphics Forum*, May 2018.
- [2] M. Budd, G. Salavasidis, I. Karnarudzaman, C. A. Harris, A. B. Phillips, P. Duckworth, N. Hawes, and B. Lacerda, "Probabilistic Planning for AUV Data Harvesting from Smart Underwater Sensor Networks," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.
- [3] G. A. Hollinger, A. A. Pereira, J. Binney, T. Somers, and G. S. Sukhatme, "Learning Uncertainty in Ocean Current Predictions for Safe and Reliable Navigation of Underwater Vehicles: Learning Uncertainty in Ocean Current Predictions," *Journal of Field Robotics*, 2016.
- [4] A. Pierson, J. W. Romanishin, H. Hansen, L. Z. Ya, and D. Rus, "Designing and Deploying a Mobile UVC Disinfection Robot," in *IROS*, 2021, p. 2.
- [5] L. Brudermüller, R. Bhattacharyya, B. Lacerda, and N. Hawes, "Time-bounded large-scale mission planning under uncertainty for uv disinfection," in *ICAPS '22 PlanRob Workshop*, 2022.
- [6] T. Lane and L. P. Kaelbling, "Approaches to macro decompositions of large markov decision process planning problems," in *Mobile Robots XVI*, 2002.
- [7] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*, 2012.
- [8] B. Lacerda, D. Parker, and N. Hawes, "Multi-objective policy generation for mobile robots under probabilistic time-bounded guarantees," in *ICAPS*, 2017.
- [9] P. Duckworth, B. Lacerda, and N. Hawes, "Time-Bounded Mission Planning in Time-Varying Domains with Semi-MDPs and Gaussian Processes," Tech. Rep., 2020.
- [10] N. Roy and S. Thrun, "Coastal navigation with mobile robots," in *NeurIPS*, 2000.
- [11] L. Nardi and C. Stachniss, "Uncertainty-aware path planning for navigation on road networks using augmented MDPs," in *ICRA*, 2019.
- [12] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," 2016.
- [13] E. Angelelli, C. Archetti, C. Filippi, and M. Vindigni, "The probabilistic orienteering problem," *Computers & Operations Research*, 2017.
- [14] —, "A dynamic and probabilistic orienteering problem," *Computers & Operations Research*, 2021.
- [15] O. Peltzer, A. Bouman, S.-K. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobue, M. J. Kochenderfer, M. Schwager, J. Burdick *et al.*, "Fig-op: Exploring large-scale unknown environments on a fixed time budget," in *IROS*, 2022.
- [16] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks," *IEEE Transactions on Robotics*, 2016.
- [17] J. L. Barry, L. P. Kaelbling, and T. Lozano-Perez, "Deth\*: Approximate hierarchical solution of large markov decision processes," in *ICJAI*, 2011.
- [18] B. Bakker, Z. Zivkovic, and B. Kröse, "Hierarchical dynamic programming for robot path planning," in *IROS*. IEEE, 2005, pp. 2756–2761.
- [19] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. Dean, and C. Boutilier, "Hierarchical solution of markov decision processes using macro-actions," in *Uncertainty in Artificial Intelligence*, 1998.
- [20] T. G. Dietterich, "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *JAIR*, 2000.
- [21] N. Gopalan, M. L. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, L. L. Wong *et al.*, "Planning with abstract markov decision processes," in *ICAPS*, 2017.
- [22] L. Brudermüller, R. Bhattacharyya, B. Lacerda, and N. Hawes, "Time-bounded large-scale mission planning under uncertainty for uv disinfection," in *ICAPS 2022 Workshop on Planning and Robotics (PlanRob)*, Jun. 2022.
- [23] D. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "Concorde tsp solver," <https://www.math.uwaterloo.ca/tsp/concorde/index.html>.
- [24] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International Conference on Computer Aided Verification*, 2011.
- [25] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games*, 2007.
- [26] B. Bonet and H. Geffner, "Labeled rtdp: Improving the convergence of real-time dynamic programming," in *ICAPS*, 2003.