CHiPS: Composing Hierarchical Pareto Solutions for Scalable Planning in Multi-Objective MDPs

Carl Hentges[1,2], Matthew Budd[1], Andrew Platt[1], Bruno Lacerda[1], David Parker[1] and Nick Hawes[1]

***Abstract*— We present a hierarchical planning methodology for approximating Pareto-optimal policies in Multi-Objective Markov Decision Process (MOMDP) models. These models describe missions in which a mobile robot must navigate an environment and perform actions at specific locations. Our approach relies on clustering the state space of the full MOMDP into hierarchical subproblem MOMDPs. We then build the set of Pareto-optimal policies for these sub-problem MOMDPs, and treat them as macro-actions in a high-level MOMDP which selects the policy to use for each of the sub-problems, as well as the order in which to address them. Our bottom-up approach synthesises approximations of Pareto-optimal policies for large problems while providing precise performance guarantees. We empirically evaluate our method, showing it achieves substantial scalability gains over a non-hierarchical approach while preserving high-quality solutions.**

I. Introduction

In robotics applications, it is often necessary to reason about several, distinct objectives that are critical to a mission's success. If the relative importance of these objectives cannot be determined *a priori*, then algorithms may instead produce a Pareto-optimal set of solutions, where each element represents a solution in which the performance of one objective can only be improved at the cost of another. However, computing or approximating the Pareto-optimal set is computationally expensive. In this paper, we focus on multi-objective sequential decision-making, in particular the problem of finding Pareto-optimal policies for multi-objective Markov decision processes (MOMDPs).

We propose *Composing Hierarchical Pareto Solutions* (CHiPS), a novel *hierarchical* technique that improves the scalability of planning for multi-objective problems, whilst simultaneously delivering high quality solutions. CHiPS targets common mobile robot problems where a robot must navigate under uncertainty in order to service target locations, trading off mission cost (e.g., time or energy) against the reward obtained by executing service actions. Examples of such problems include service robotics [1], area disinfection [2], and space rover planning [3]. In these robotics problems, states include a spatial component which is used in CHiPS to subdivide the full (often intractable) multi-objective problem into (tractable) subproblem MOMDPs. Each subproblem MOMDP is solved to yield a set of Pareto-optimal policies, and then each sub-problem policy is used to define an action in a high-level MOMDP. Solutions to this high-level MOMDP dictate which Pareto-optimal policy to use for each subproblem, as well as the order in which subproblems should be completed. To the best of our knowledge, CHiPS is the first multi-objective planning method that combines Pareto-optimal sub-problem policies to solve a larger multi-objective problem. Furthermore, it does so while maintaining precise policy values at both levels. This enables the synthesis of hierarchical policies with precise guarantees on performance.

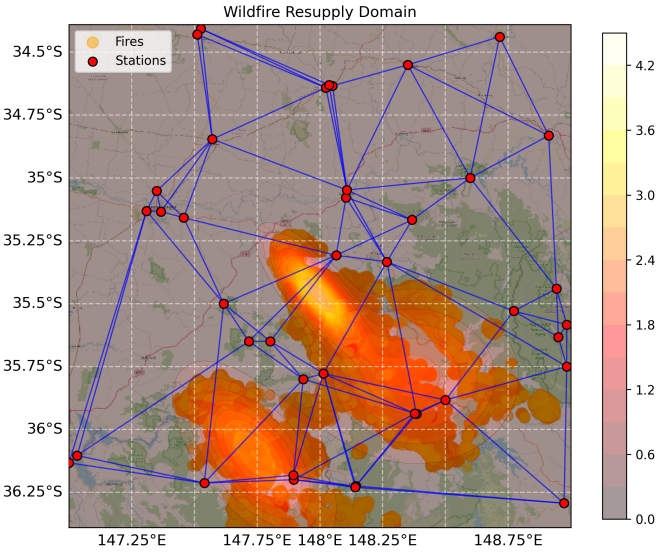

Fig. 1. Visualisation of the wildfire resupply domain in Kosciuszko National Park featuring 40 NASA GISTEMP v4 stations [4] resupplied based on proximity to wildfire predictions from historical NASA FIRMS data [5].

We evaluate our approach using three problems: an Autonomous Underwater Vehicle (AUV) in a data harvesting domain and in an offshore wind turbine inspection domain, and an Autonomous Aerial Vehicle (AAV) in a wildfire resupply domain (see Figure 1). We demonstrate that CHiPS substantially improves scalability whilst synthesising high quality policies.

II. Related Work

Multi-objective decision-making under uncertainty [6], [7] is highly relevant in robotics, where the inherent uncertainty of autonomy in the real-world must be tackled, whilst several, often conflicting, objectives must be managed. Examples include our proposed AUV data harvesting domain, based on the (single-objective) work in [8], battery-charge scheduling [1], and resource-performance trade-offs for rovers [9]. Scaling up to realistic applications is one of the crucial challenges of MOMDP planning, and has been tackled in a variety of ways. Lacerda et al. proposed a two-stage approach

[1]University of Oxford, Oxford, UK.
[2]Corresponding Author. Email: carl.hentges@eng.ox.ac.uk.

for approximating Pareto-optimal policies that involve time-bounded objectives [10]. The first stage optimises simpler, expected-time objectives, which are then used to prune actions from the second stage, where the time-bounded objectives are considered. However, this approach operates on the full state-action space in the first stage, and its pruning can lead to highly suboptimal results. Additionally, its applicability is limited to cost-bounded problem specifications. In contrast, CHiPS uses a divide-and-conquer approach which provides better scalability and high quality solutions, and is applicable to any problem where sub-problems can be clustered.

CHiPS operates on the model of the multi-objective planning problem, breaking it into several smaller problems that can be solved using off-the-shelf multi-objective solution algorithms. An orthogonal approach to increasing scalability is to focus on the solution algorithm, for example adapting scalable algorithms such as Monte-Carlo tree search [11], [12], [13] or heuristic search [14] to the multi-objective setting. These works could be used within our approach to approximate Pareto-optimal policies at both high and low levels. This would further increase the scalability of our approach, albeit at the cost of potential further suboptimality.

Hierarchies and other forms of abstractions have been widely used to improve the scalability of single-objective planning [15], [16]. Our definition of entry and exit locations (see Section V) mirrors the entrance and exit periphery in [15], while our subproblem policies parallel the options framework [16] but focus on spatial sub-MDPs rather than action sequences. Similar to abstract MDPs [17], our approach solves subproblems within actions. However, abstract MDPs use a top-down approach, solving only subproblems relevant to the higher-level solution, which improves scalability but sacrifices exact performance guarantees.

The above uses of hierarchy and abstraction in MDPs are limited to the simpler single-objective case, whereas CHiPS is one of the very few algorithms to use abstraction in multi-objective planning. This makes selecting abstract actions more challenging, as their impact on the performance across all objectives must be considered. We do this by using multi-objective solutions at both levels of abstraction.

III. Preliminaries

MDPs are a standard model for planning under uncertainty, and MOMDPs extend this by supporting a vector of objective values as opposed to a single reward value. A *MOMDP* is a tuple $\mathcal{M} = \langle S, \bar{s}, A, T, \mathbf{R} \rangle$, where S is the set of possible states; $\bar{s} \in S$ is the initial state; A is the set of actions; $T : S \times A \times S \to [0, 1]$ is the probabilistic transition function; and $\mathbf{R} = \{R_1, \ldots, R_{|\mathbf{R}|}\}$ is a set of reward functions $R_k : S \times A \to \mathbb{R}_{\geq 0}$. A policy $\pi : S \to Dist(A)$ maps states to distributions over actions. Given π, we can define the *value function* $V^{\pi}_{\mathcal{M},R} : S \to \mathbb{R}_{\geq 0}$ for R under π as:

$$V^{\pi}_{\mathcal{M},R}(s) = \mathbb{E}\left(\sum_{i=0}^{\infty} R(s_i, a_i) \mid \pi, s_0 = s\right),$$

Random variables s_i, a_i represent the i-th state and action selected, respectively, following π from initial state s.

We consider two types of objective for elements of $\mathbf{R}$. For a *reward objective* $R \in \mathbf{R}$, the goal is to maximise $V^{\pi}_{\mathcal{M},R}(\bar{s})$. For *cost objectives* $C \in \mathbf{R}$, the goal is to minimise $V^{\pi}_{\mathcal{M},C}(\bar{s})$. For both types, we assume that there is a goal state in the MOMDP which can be reached with probability 1 from any state, and from which no further rewards or costs can be accumulated. Furthermore, we assume there is a finite amount of reward that can be accumulated in the MOMDP, and thus that the value functions are always finite. Note that we chose to present these definitions of *undiscounted* reward maximisation and cost minimisation objectives because they met our robot mission planning setting more closely. However, the approach presented in this paper can be straightforwardly adapted to the commonly used setting of discounted reward maximisation.

A solution for a MOMDP is a set of *Pareto-optimal* policies where each policy represents an optimal compromise between competing objectives. This set is often called the *Pareto front* and contains only *undominated* policies. Without loss of generality, we formally describe this in the context of a vector $\mathbf{R}$ containing only reward objectives. We say that π *dominates* π' if $V^{\pi}_{\mathcal{M},R}(\bar{s}) \geq V^{\pi'}_{\mathcal{M},R}(\bar{s})$ for all $R \in \mathbf{R}$. π *strictly dominates* π' if it dominates π' and there exists $R \in \mathbf{R}$ such that $V^{\pi}_{\mathcal{M},R}(\bar{s}) > V^{\pi'}_{\mathcal{M},R}(\bar{s})$. A policy is *Pareto-optimal* if it is not strictly dominated by any other policy. The Pareto front for MOMDP $\mathcal{M}$ can be represented as a finite set of deterministic policies $\Pi(\mathcal{M})$ which are not dominated by any other policies. This induces an infinite set of Pareto-optimal stochastic policies, which can be obtained as mixtures of the deterministic policies in $\Pi(\mathcal{M})$.

IV. Problem Formulation

Navigation MDP: We assume a discrete set of locations L which the robot can navigate between. Navigation is stochastic, meaning the robot may reach $l' \neq l$ with some probability when aiming for $l \in L$. A *navigation MDP* is defined as $\mathcal{M}_{\text{nav}} = \langle L, \bar{l}, A_{\text{nav}}, T_{\text{nav}}, C_{\text{nav}} \rangle$, where:

- $L = \{l_1, \ldots, l_{|L|}\}$ is a set of locations;
- $\bar{l} \in L$ is the initial location;
- $A_{\text{nav}} \subseteq L \times L$ is the action set, where $(l, l') \in A_{\text{nav}}$ indicates the robot can navigate directly from l to l' without traversing other locations;
- $T_{\text{nav}} : L \times A_{\text{nav}} \times L \to [0, 1]$ is the transition function such that $T_{\text{nav}}(l, (l, l'), l'')$ is the probability of moving from l to l'' after executing action (l, l'); and
- $C_{\text{nav}} : L \times A_{\text{nav}} \to \mathbb{R}_{\geq 0}$ represents the cost $C_{\text{nav}}(l, (l, l'))$ of attempting to navigate from l to l'.

We assume $T_{\text{nav}}(l, (l, l'), l') > 0$, i.e., there is a non-zero chance of reaching l' when attempting to do so. Furthermore, any location in the navigation MDP is assumed to be reachable with probability 1 from any other location. These assumptions are reasonable, based on our experience deploying mobile robots in a range of environments, e.g. [8], [18], [19].

Navigation and Service MOMDP: We extend the navigation MDP to consider the cost and reward associated with servicing locations by adding a *service reward* function $R_{\text{service}} : L \to \mathbb{R}_{\geq 0}$ that represents the value of servicing l;

and a *service cost* function $C_{\text{service}} : L \to \mathbb{R}_{\geq 0}$ that represents the cost of servicing l. Given $L_{end} \subseteq L$ where the mission can be finished, the multi-objective *Navigation and Service MOMDP* (NS-MOMDP) is $\mathcal{M} = \langle S, \bar{s}, A, T, \mathbf{R} \rangle$ where:

- $S = (L \times \{0,1\}^{|L|}) \cup \{done\}$ consists of the location of the robot and Booleans representing whether each location has been serviced, and a $done$ state indicating that the robot has finished its mission;
- The robot starts in its initial location, with all locations unserviced, i.e., $\bar{s} = (\bar{l}, 0, \ldots, 0)$;
- $A = A_{nav} \cup \{service,\ finish\}$, i.e., we extend A_{nav} with a service and a mission completion action;
- For $s = (l, \mathbf{b})$ and $s' = (l', \mathbf{b}')$ with $\mathbf{b}, \mathbf{b}' \in \{0,1\}^{|L|}$ and action $a \in A$, the transition function is:

$$T(s,a,s') = \begin{cases} T_{nav}(l,a,l') & \text{if } a \in A_{nav} \wedge \mathbf{b} = \mathbf{b}' \\ 1 & \text{if } a = service \wedge l = l' \wedge \\ & \quad \mathbf{b}(l) = 0 \wedge \mathbf{b}'(l) = 1 \wedge \\ & \quad \forall_{l'' \neq l} \mathbf{b}(l'') = \mathbf{b}'(l'') \\ 1 & \text{if } a = finish \wedge \\ & \quad l \in L_{end} \wedge s' = done \\ 0 & \text{otherwise;} \end{cases}$$

  Intuitively, navigation actions have the same dynamics as T_{nav}; the service action can only occur in an unserviced location and changes its state to serviced; and the finish action can only occur in an end location;
- The reward vector is $\mathbf{R} = (C, R)$, where $C : S \times A \to \mathbb{R}_{\geq 0}$ is the cost for action execution, comprised of the cost for navigation and the cost for servicing locations; and $R : S \times A \to \mathbb{R}_{\geq 0}$ is the reward for servicing locations. For $s = (l, \mathbf{b}) \in S$ and $a \in A$:

$$C(s,a) = \begin{cases} C_{nav}(l,a) & \text{if } a \in A_{nav} \\ C_{service}(l) & \text{if } a = service \\ 0 & otherwise; \end{cases}$$

$$R(s,a) = \begin{cases} R_{service}(l) & \text{if } a = service \\ 0 & otherwise. \end{cases}$$

We address the problem of finding the set of Pareto-optimal policies for the NS-MOMDP. We focus on two objectives for simplicity, but our method can be extended to handle multiple costs and rewards.

V. Hierarchical Multi-Objective Planning

To address the exponential blow-up of the state space as more service locations are added, we propose *Composing Hierarchical Pareto Solutions* (CHiPS), which decomposes the NS-MOMDP into smaller subproblem MOMDPs by clustering locations spatially. The solutions to these subproblems are then used as macro-actions in a high-level MOMDP.

Subproblem MOMDP: We obtain a set of *subproblem MOMDPs* by partitioning the location set L into *location clusters* $\mathcal{L} = \{K_1, \ldots, K_{|\mathcal{L}|}\}$, $K_i \subset L$. For location $l \in L$, K_l represents the cluster that contains l. A *connection* exists from K to K' if $(l, l') \in A_{nav}$ for some $l \in K$ and $l' \in K'$. The locations $l \in K$ and $l' \in K'$ with minimal cost $C_{nav}(l, l')$ are the *exit location* from K to K', and the *entry location* to K' from K, respectively. For cluster $K \in \mathcal{L}$, we define $in_K \subseteq K$ as the locations in K which are an entry location to K from some $K' \in \mathcal{L}$; and $out_K \subseteq K$ as the subset of locations in K which are an exit location from K to some $K' \in \mathcal{L}$. The *neighbour locations* of a cluster K are defined as $\mathcal{N}(K) = \{l' \in L \setminus K \mid T_{nav}(l, a, l') > 0 \text{ for some } l \in K \text{ and } a \in A_{nav} \cap (K \times K)\}$. We denote a cluster plus its neighbours as $K^+ = K \cup \mathcal{N}(K)$. We only consider actions that try to navigate inside K, i.e., actions in $K \times K$. K^+ are the locations that have some probability of being the outcome of such actions.

We can now define the subproblem MOMDP for cluster K, $l_{in} \in in_K$ and $l_{out} \in out_K$, which constrains the robot to start at l_{in}, navigate towards and service locations in K only, and finish execution in l_{out}. We also introduce return actions to handle accidental transitions to $\mathcal{N}(K)$, allowing the robot to return to K. Formally, we define $\mathcal{M}_K(l_{in}, l_{out}) = \langle S_K, \bar{s}_K, A_K, T_K, \mathbf{R}_K \rangle$ where:

- $S_K = (K^+ \times \{0,1\}^{|K|}) \cup \{done\}$, i.e., states comprise the location of the robot within K or its neighbours, and a Boolean representing whether each location in K has been serviced. We also add a $done$ state indicating the robot has finished subproblem K;
- $\bar{s}_K = (l_{in}, 0, \ldots, 0)$, i.e., the robot starts in the entry location, with all locations unserviced;
- $A_K = (A_{nav} \cap (K \times K)) \cup \{return, service, finish\}$, i.e., we only allow navigation actions that target locations in K and add an action which returns to K if the robot navigates to one of its neighbours;
- The transition function T_K is constrained to K, also imposing that the finish action can only be executed in l_{out}. Furthermore, if $l \in \mathcal{N}(K)$, then the robot can execute an action which returns it to location $l' \in K$ with probability 1 and at minimum expected cost. This action is a macro-action which represents executing the shortest path policy $\pi_{l, closest_K(l)}$, where $closest_K(l) \in K$ is the location in K that has minimum expected cumulative cost from l. This location, and the corresponding policy, are found in a preprocessing step by solving, for each $l'' \in K$, a stochastic shortest path problem (i.e. minimise the expected cost to reach a goal state) in the navigation MDP, with start state l, goal state l'', cost function C_{nav}, and taking the minimum. Thus, for $s = (l, \mathbf{b})$ and $s' = (l', \mathbf{b}')$ with $\mathbf{b}, \mathbf{b}' \in \{0,1\}^{|K|}$ and $a \in A_K$:

$$T_K(s,a,s') = \begin{cases} T(s,a,s') & \text{if } a \in A_K \cup \{service\} \wedge \\ & \quad l \in K \\ 1 & \text{if } a = finish \wedge \\ & \quad l = l' = l_{out} \wedge \mathbf{b} = \mathbf{b}' \\ 1 & \text{if } a = return \wedge \\ & \quad l \in \mathcal{N}(K) \wedge \\ & \quad l' = closest_K(l) \wedge \mathbf{b} = \mathbf{b}' \\ 0 & \text{otherwise;} \end{cases}$$

- The reward vector $\mathbf{R}_K$ is defined as (C_K, R_K), where:

$$C_K(s,a) = \begin{cases} C(s,a) & \text{if } l \in K \text{ and } a \neq \textit{return} \\ c(l, \textit{closest}_K(l)) & \text{if } l \in \mathcal{N}(K) \text{ and } a = \textit{return} \\ 0 & \text{otherwise,} \end{cases}$$

where $c(l, \textit{closest}_K(l))$ is the expected cumulative cost of the optimal policy from l to $\textit{closest}_K(l)$.

$$R_K(s,a) = \begin{cases} R(s,a) & \text{if } l \in K \text{ and } a = \textit{service} \\ 0 & \text{otherwise.} \end{cases}$$

Solving a subproblem MOMDP yields a set $\Pi(\mathcal{M}_K(l_{in}, l_{out}))$ of deterministic Pareto-optimal policies which will form macro-actions in the high-level MOMDP.

High-Level MOMDP: The *high-level MOMDP* operates over the same state space as the NS-MOMDP but removes service actions. Instead, we add macro-actions that represent the Pareto-optimal policies obtained by solving the subproblem MOMDPs. We abstract the execution of these macro-actions, considering only their initial and final state, and a cost and reward defined by the values of the corresponding policy on the subproblem MOMDP. Formally, for NS-MOMDP $\mathcal{M} = \langle S, \bar{s}, A, T, \mathbf{R} \rangle$ and location partition $\mathcal{L}$, the *high-level MOMDP* is defined as $\mathcal{M}^+ = \langle S^+, \bar{s}^+, A^+, T^+, \mathbf{R}^+ \rangle$, where:

- $S^+ = (L \times \{0,1\}^{|\mathcal{L}|}) \cup \{\textit{done}\}$, i.e., we consider servicing of subproblems rather than specific locations;
- $\bar{s}^+ = (\bar{l}, 0, \ldots, 0)$, i.e., the robot starts in its initial location, with all subproblems unserviced;
- Actions are the navigation and finish actions from the NS-MOMDP, plus a set of Pareto-optimal policies for each entry and exit pair for each subproblem:

$$A^+ = A_{nav} \cup \{\textit{finish}\} \cup \bigcup_{K \in \mathcal{L}} \bigcup_{l_{in} \in in_K} \bigcup_{l_{out} \in out_K} \Pi(\mathcal{M}_K(l_{in}, l_{out}));$$

- The transition function has the same format as the transition function of the NS-MOMDP, except the single location service actions are replaced with the subproblem service macro-actions from the subproblem policies. These macro-actions can be executed from their entry location, have a deterministic outcome corresponding to their exit location, and update the subproblem state to serviced. If a cluster has already been serviced then it is only possible to navigate through the cluster, additional reward cannot be gathered. Thus, for $s = (l, \mathbf{b})$ and $s' = (l', \mathbf{b}')$ with $\mathbf{b}, \mathbf{b}' \in \{0,1\}^{|\mathcal{L}|}$:

$$T^+(s,a,s') = \begin{cases} T(s,a,s') & \text{if } a \in A_{nav} \cup \{\textit{finish}\} \\ 1 & \text{if } a \in \Pi(\mathcal{M}_K(l_{in}, l_{out})) \\ & \quad \wedge l = l_{in} \wedge l' = l_{out} \wedge \\ & \quad \mathbf{b}(K_{l_{in}}) = 0 \wedge \\ & \quad \forall_{K \neq K_{l_{in}}} \mathbf{b}(K) = \mathbf{b}'(K) \\ 0 & \text{otherwise;} \end{cases}$$

- The reward vector $\mathbf{R}^+$ is defined as (C^+, R^+), where:

$$C^+(s,a) = \begin{cases} C(s,a) & \text{if } a \in A_{nav} \cup \{\textit{finish}\} \\ V^{\pi}_{\mathcal{M},C}(s) & \text{if } a \text{ is a macro-action } \pi; \end{cases}$$

$$R^+(s,a) = \begin{cases} 0 & \text{if } a \in A_{nav} \cup \{\textit{finish}\} \\ V^{\pi}_{\mathcal{M},R}(s) & \text{if } a \text{ is a macro-action } \pi. \end{cases}$$

Note that the expected values of the macro-actions that are used for the costs and rewards of the high-level MOMDP are computed when building the Pareto-optimal policies for the subproblem MOMDPs.

Composing Hierarchical Pareto Solutions (CHiPS): To summarise, our approach proceeds as follows:

1) Partition the set of locations L into $\mathcal{L}$;
2) For each $K \in \mathcal{L}$, calculate the sets of entry and exit locations, in_K and out_K, then build and solve the subproblem MOMDP for each entry/exit pair, yielding a set of Pareto-optimal policies to be used as macro-actions for the high-level MOMDP;
3) Build and solve the high-level MOMDP $\mathcal{M}^+$, yielding a set of high-level Pareto-optimal policies for the full MOMDP;
4) Choose a high-level Pareto policy and execute it. Macro-actions are taken by executing the corresponding Pareto-optimal subproblem MOMDP policy.

CHiPS is a divide-and-conquer approach, solving subproblems whose solutions result in macro-actions for the high-level MOMDP. This can be seen as first building multi-objective *options* and then building and solving a multi-objective options MDP [16]. As such, it is a suboptimal approach: the CHiPS high-level MOMDP may not include actions needed for policies on the Pareto front of the full NS-MOMDP, removing the ability to model all globally possible cost/reward trade-offs. However, since the macro-actions are locally Pareto-optimal (i.e., Pareto-optimal for the subproblem MOMDP they are computed), we are still able to synthesise performant policies. Furthermore, the hierarchical approach allows us to scale to much larger environments. Note that, whilst we presented a two-level hierarchy, the general approach can be extended to a multi-level hierarchy.

Whilst suboptimal, CHiPS is able to provide precise guarantees for the policies it synthesises. Specifically, since the costs and rewards for the macro-actions are defined by the values of the corresponding policies over the subproblem MOMDP, the values associated with the Pareto-optimal policies for the high-level MOMDP are exact. In other words, the expectations over cumulative cost and reward of a policy computed for the high-level MOMDP correspond to the expectations of that policy for the full NS-MOMDP as well.

Proposition 1: For a policy π of the high-level MOMDP: $V^{\pi}_{\mathcal{M}^+,R^+}(\bar{s}^+) = V^{\pi}_{\mathcal{M},R}(\bar{s})$ and $V^{\pi}_{\mathcal{M}^+,C^+}(\bar{s}^+) = V^{\pi}_{\mathcal{M},C}(\bar{s})$.

Proof: (sketch) The high-level MOMDP navigation and finish actions are the same as in the NS-MOMDP. The high-level MOMDP macro-actions correspond to executing navigation and service actions in a subproblem. The cost and rewards for these actions are exact, as they are obtained

from solving the subproblem MOMDPs. Thus, the value of policies is the same for both models. ■

VI. Experiments

We evaluate CHiPS on two synthetic AUV Data Harvesting domains, as well as on two domains incorporating real-world data. For all domains, we define the cost function as the time to complete the mission and compute travel time between nodes (i.e., the cost function C_{nav}) using Euclidean distance. For an action that moves the AUV between two locations distance d apart, the probability of reaching the intended target is given by $1 - \frac{d}{d_{\max}}$, with the remaining probability mass distributed among all nodes within distance d of the starting node (including the target node). Here, $d_{\max}$ is a parameter specified for each domain in advance ($d_{\max} = 1500$ for the AUV Data Harvesting and Wind Turbine Inspection domains and $d_{\max} = 222{,}240$ for the Wildfire Resupply domain).

AUV Data Harvesting: The AUV Data Harvesting domain is based on the model presented in [8]. Sensor beacons at known locations contain data, and an AUV must collect as much of this data as possible while minimising the time spent doing so. Both the duration of data-collection actions and the associated reward depend on the beacon's data volume, where the reward is equal to the amount of data gathered, and one unit of data requires one unit of time to collect. To complete a mission, the AUV must surface at a predefined end location.

To generate problem instances, we start with a predefined list of obstacles and then randomly sample beacon locations on a rough grid, ensuring they do not lie within obstacles, until a given problem size n is reached. These beacon locations are then adjusted to ensure that beacon-to-beacon distances are not too small.

We generated two synthetic domain settings differing in data distribution. In *Synthetic Domain 1*, each beacon's data volume decreases according to a normal distribution based on proximity to a data source. In *Synthetic Domain 2*, most locations have minimal data (1 unit), with a few sparse locations assigned significantly larger values (1000 units).

Navigation actions are determined by connecting each beacon to its nearest neighbours, ensuring that each location has at most four and at least two neighbours. We generate a set of problem instances by varying the number of locations ($5 \leq n \leq 40$) and the number of clusters ($3 \leq k \leq 5$).

Wind Turbine Inspection: We base the offshore Wind Turbine Inspection domain on the London Array wind farm [20], located about 20 km off the coast of Kent. We use the turbine locations to define obstacles and manually specify places of interest near these turbines as locations for the NS-MOMDP. We also manually define the subproblem partition for CHiPS.

The agent gathers reward for inspecting these locations, with shallower areas yielding greater reward (20 units) than deeper areas (10 units). Each inspection action costs 100 time units. Our problem instances include 15, 20, or 30 locations.

n	Method / k	Synth. Domain 1		Synth. Domain 2	
		Time (s)	Opt. ratio	Time (s)	Opt. ratio
10	Baseline	25.54	1.00	27.41	1.00
	3	**24.18**	0.99	**13.77**	1.00
	4	39.35	0.99	22.20	1.00
	5	34.13	0.99	23.84	1.00
15	Baseline	2309.08	1.00	1736.39	1.00
	3	71.03	0.98	57.91	0.99
	4	**51.02**	0.99	**52.65**	0.99
	5	76.74	0.99	63.77	0.99
20	3	58.11	-	97.65	-
	4	**56.92**	-	**67.00**	-
	5	61.64	-	68.92	-
30	4	**215.82**	-	541.71	-
	5	325.69	-	**191.39**	-
40	4	2067.41	-	1955.61	-
	5	650.93	-	670.73	-

TABLE I
TIMING/PERFORMANCE FOR AUV DATA HARVESTING SYNTHETIC DOMAINS.

Wildfire Resupply: The Wildfire Resupply domain is inspired by the domain presented in [21], based on the locations of NASA GISTEMP v4 monitoring network stations [4] in the Kosciuszko National Park, specifically all stations within the coordinates 36.39S–147.00E to 34.39S–149.00E. These 40 stations are interconnected so that each station has a minimum of 2 and a maximum of 4 neighbours.

The reward for resupplying a station is informed by the risk of wildfire spread to that location. We model this by training a Gaussian mixture model on historical fire data from NASA Fire Information for Resource Management System (FIRMS) satellite fire observations [5], spanning 26-12-2019 to 14-01-2020. Each resupply action costs 100 time units. A visualisation of this domain is shown in Figure 1.

Evaluation: To demonstrate the performance of our method, we compare it to a non-hierarchical baseline that solves the NS-MOMDP directly, finding a Pareto-optimal set of solutions. To find the solutions both for this baseline and for the subproblem and high-level MOMDPs, we use the approach of Forejt et al. [22], implemented in the PRISM model checker [23]. All experiments were run on a machine with an Intel i5 @ 2 GHz CPU and 16 GB RAM; the solver was allocated 8 GB of RAM and a 30-minute timeout.

Results: Our results are summarised in Tables I, II and III for varying problem sizes of n locations and different numbers of clusters k for CHiPS vs. the baseline. We compare runtimes, marking the fastest in bold. For CHiPS, the measured time includes all stages: subproblem construction, subproblem solving, high-level problem creation, and high-level problem solving. We also show optimality of the solution with respect to the baseline (see below). Note that results only shown where data available, i.e., runtimes are only provided for configurations that did not exceed the timeout and optimality is only reported when the baseline solution could be computed within the timeout.

CHiPS outperforms the baseline and, as problem sizes grow, provides significantly lower computation times, with up to

n	Method / k	Time (s)	Opt. ratio
15	Baseline	1938.98	1.00
	3	44.45	0.92
	4	**38.63**	0.92
	5	43.63	0.95
	6	50.18	0.98
20	3	88.40	-
	4	64.62	-
	5	**53.19**	-
	6	66.20	-
30	4	243.66	-
	5	**163.06**	-
	6	157.51	-

TABLE II

TIMING/PERFORMANCE FOR THE WIND TURBINE INSPECTION DOMAIN.

n	k	Time (s)
40	4	336.75
	6	**332.92**
	8	407.29

TABLE III

TIMING RESULTS FOR THE WILDFIRE RESUPPLY DOMAIN.

a 50-fold reduction at higher subproblem counts. Furthermore, CHiPS can solve problems that the baseline approach cannot handle due to memory constraints. For instance, the Wildfire domain is too large for the baseline and thus Table III only shows runtimes and not optimality statistics. Moreover, in Table I, problems with more than $n = 15$ nodes exceed the allowed time when using the baseline solver.

To assess solution quality, we compare resulting Pareto fronts by calculating the area under the curve of both the optimal and CHiPS approximations—extending each to the maximum time coordinate—and taking their ratio for problems solvable by the baseline. On average, the CHiPS solutions achieve approximately 99% of the optimal baseline's area in the Synthetic domain (Table I) and 94% in the Wind Turbine Inspection domain (Table II).

The suboptimality of CHiPS arises from restricting location visitation order across subproblems. In the optimal baseline, the solver can freely choose an ordering across the entire set of locations, whereas CHiPS enforces an order within subproblems and then an order over subproblems. Notably, if $k = 1$ or $k = n$, CHiPS can recover the baseline solution. For intermediate k, solution quality depends on both the problem specifics (e.g., distribution of locations and connectivity) and how cluster boundaries affect connectivity.

Our results demonstrate that CHiPS improves scalability and efficiency, significantly reducing computation time and memory usage compared to the non-hierarchical baseline, while producing solutions that remain close to optimal.

VII. CONCLUSION

In this paper we introduced CHiPS, a hierarchical approach for multi-objective policy synthesis under uncertainty, and demonstrated its effectiveness in the context of AUV mission planning. CHiPS uses a divide-and-conquer approach, first solving sub-problem MOMDPs, and using the solutions to build a high-level MOMDP. This approach enables better scalability and computes a set of policies with precise expectations over the objectives.

In future work, we will investigate approximate solution methods, and top-down approaches that do not require solving all the sub-MOMDPs before solving the high-level MOMDP.

ACKNOWLEDGEMENTS

Lacerda and Hawes supported by the EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1).

REFERENCES

[1] M. Tomy, B. Lacerda, N. Hawes, and J. L. Wyatt, "Battery charge scheduling in long-life autonomous mobile robots via multi-objective decision making under uncertainty," *RAS*, 2020.

[2] M. Staniaszek, L. Brudermüller, R. Bhattacharyya, B. Lacerda, and N. Hawes, "Difficulty-aware time-bounded planning under uncertainty for large-scale robot missions," in *ECMR*. IEEE, 2023.

[3] A. J. Coles, A. I. Coles, M. M. Munoz, O. E. Savas, T. Keller, F. Pommerening, and M. Helmert, "On-board planning for robotic space missions using temporal pddl," in *IWPSS*, 2019.

[4] GISTEMP Team, "GISS Surface Temperature Analysis (GISTEMP), version 4," https://data.giss.nasa.gov/gistemp/, 2025.

[5] "NASA Fire Information for Resource Management System (FIRMS)," https://firms.modaps.eosdis.nasa.gov/, 2025.

[6] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *JAIR*, 2013.

[7] D. M. Roijers, S. Whiteson, R. Brachman, and P. Stone, *Multi-objective decision making*, 2017.

[8] 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 *IROS*, 2022.

[9] M. Lahijanian, M. Svorenova, A. A. Morye, B. Yeomans, D. Rao, I. Posner, P. Newman, H. Kress-Gazit, and M. Kwiatkowska, "Resource-performance tradeoff analysis for mobile robots," *IEEE RAL*, 2018.

[10] B. Lacerda, D. Parker, and N. Hawes, "Multi-objective policy generation for mobile robots under probabilistic time-bounded guarantees," in *ICAPS*, 2017.

[11] W. Chen and L. Liu, "Pareto Monte Carlo tree search for multi-objective informative planning," in *RSS*, 2019.

[12] M. Painter, B. Lacerda, and N. Hawes, "Convex hull monte-carlo tree search," in *ICAPS*, 2020.

[13] C. F. Hayes, M. Reymond, D. M. Roijers, E. Howley, and P. Mannion, "Distributional monte carlo tree search for risk-aware and multi-objective reinforcement learning," in *AAMAS*, 2021.

[14] D. Z. Chen, F. Trevizan, and S. Thiébaux, "Heuristic search for multi-objective probabilistic planning," in *AAAI*, 2023.

[15] M. Hauskrecht, N. Meuleau, L. P. Kaelbling, T. L. Dean, and C. Boutilier, "Hierarchical solution of markov decision processes using macro-actions," in *UAI*, 1998.

[16] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *AI*, 1999.

[17] N. Gopalan, M. desJardins, M. Littman, J. MacGlashan, S. Squire, S. Tellex, J. Winder, and L. Wong, "Planning with abstract markov decision processes," in *ICAPS*, 2017.

[18] B. Lacerda, F. Faruq, D. Parker, and N. Hawes, "Probabilistic planning with formal performance guarantees for mobile service robots," *IJRR*, 2019.

[19] M. Staniaszek, T. Flatscher, J. Rowell, H. Niu, W. Liu, Y. You, R. Skilton, M. Fallon, and N. Hawes, "Autoinspect: Towards long-term autonomous industrial inspection," 2024.

[20] "Navionics — navionics.com," https://www.navionics.com/gbr/, [Accessed 26-04-2024].

[21] A. Stephens, B. Lacerda, and N. Hawes, "Planning for long-term monitoring missions in time-varying environments," in *IROS*, 2024.

[22] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *ATVA*, 2012.

[23] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, 2011.