2025 IEEE International Conference on Robotics and Automation (ICRA 2025)
May 19-23, 2025. Atlanta, USA

Hybrid Decentralization for Multi-Robot Orienteering with Mothership-Passenger Systems

Nathan L. Butler and Geoffrey A. Hollinger

***Abstract*— We present a hybrid centralized-decentralized planning algorithm for a multi-robot system consisting of a Mothership robot and multiple Passenger robots. In this system, the Passenger robots execute tasks while the Mothership provides support. This paper addresses the challenge of planning Passenger robot movements, framing it as a Stochastic Multi-Agent Orienteering Problem (SMOP) complicated by factors like stochastic operational efforts and disruptive events. We optimize the task completion efficiency of the system by combining centralized solutions from the Mothership with local plans from Passengers to enhance system resilience. Our contributions include defining the SMOP, developing a solution using Decentralized Monte Carlo Tree Search, presenting a hybrid algorithm that integrates centralized plans into the distributed framework, and evaluating the algorithm's performance in a simulated environment. Our results show that our hybrid approaches outperform fully centralized and fully distributed algorithms in highly-dynamic scenarios with up to a 26.6% increase in task completion efficiency over baseline methods.**

I. Introduction

Rising sea-levels threaten coastal communities worldwide with flooding and displacement. Melting polar ice sheets are key contributors to this issue, with the West Antarctic Ice Sheet alone predicted to cause over three meters of sea-level rise. Marine ice sheets are particularly unstable as the ice is warmed by both hot air above and by warm water below the ice shelf, triggering sudden, rapid collapse. Because of this, uncertainty persists in models predicting the melting rate and volume loss of receding marine ice sheets. A rich collection of under-ice data would help scientists improve these models and more accurately forecast sea-level rise. [1].

However, exploring the marine under-ice environment is challenging and dangerous. The ice shelf restricts surface access, so underwater vehicles may be unable to surface in emergencies and risk becoming trapped. This threat to human crew members prompts the use of an unmanned Autonomous Underwater Vehicle (AUV). Nevertheless, many AUVs rely on frequent surfacing to reduce localization error and exchange information. As this is not an option, error could accumulate, resulting in the vehicle developing poorly-informed plans that exceed its battery life or lead it into hazardous regions where turbulent ocean currents cause damaging collisions. Both events risk costly AUV damage and data loss.

One proposed concept for a more robust system employs a team of small, inexpensive Passenger AUVs to explore the under-ice environment while aggregating data on a large, capable Mothership AUV stationed far from high-risk areas. The Mothership first transports a team of Passengers under the ice shelf. The Passengers then deploy to perform multiple data collection tasks in parallel before attempting to return to the Mothership to exit the region. During operations, a team of support robots coordinate to maintain localization accuracy and facilitate communications between the Mothership and Passengers, as illustrated in Fig. 1.

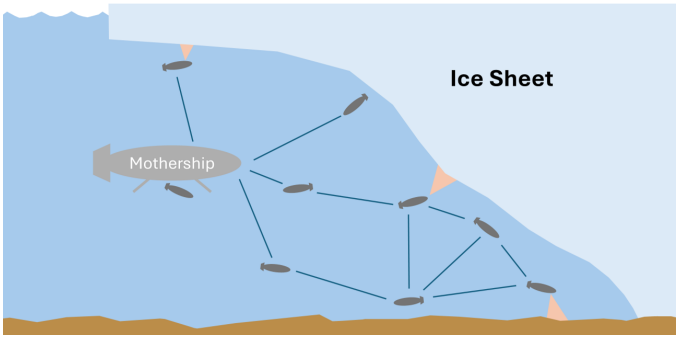


Fig. 1: Mission schematic for a Mothership-Passenger system exploring an under-ice environment, including hybrid communication network.

In this paper, we focus on planning the Passenger robots' movements between data collection tasks distributed through the environment. We assume that scientists define these tasks prior to the mission and that more can be added online as the Mothership processes incoming data. With this, we aim to generate an optimal set of schedules by 1) allocating tasks to individual robots and 2) planning each robot's tour through its task set such that the global reward is maximized and budget constraints (usually time or energy) are met. We frame this problem as a Multi-Agent Orienteering Problem (MOP) [2] made more difficult by stochastic factors common to the robotics domain such as: communication packet loss, unpredictable travel costs, and random equipment failure [3].

Previously published works have solved the MOP and related Multi-Robot System (MRS) scheduling problems using two distinct approaches: 1) centralized MRS networking, where planning is done at a global scale and 2) distributed MRS networking, where schedules are created locally [4]. However, to the best of our knowledge, no prior works explore ways to leverage a hybrid centralized-distributed network, such as that available in the Mothership-Passenger system, to solve complex orienteering problems.

We propose a hybrid solution that combines centralized and distributed planning. The Mothership uses its compu-

*This work is funded in part by NSF award 2322055

*Nathan Butler and Geoffrey Hollinger are with the Collaborative Robotics and Intelligent Systems (CoRIS) Institute, Oregon State University, Corvallis OR 97331, USA {butlnath, geoff.hollinger}@oregonstate.edu

tational power to develop centralized solutions, while each Passenger generates local plans using lightweight solvers. Passengers can request updated tours from the Mothership to compare against their local plans, sharing the best options across the network for dynamic rescheduling. This approach combines the high-quality solutions of centralized control with the resilience of distributed systems.

The contributions of this paper include:

1) Defining the Stochastic Multi-Agent Orienteering Problem (SMOP)
2) A distributed solution to this problem built on decentralized Monte Carlo tree search
3) A hybrid decentralized algorithm that incorporates plans from a centralized node into the distributed framework
4) Simulations conducted with high fidelity simulated under-ice tidal data

II. Related Work

We organize the related works into two sections to explore the background and current approaches in this area. The first category explores recent advancements in the Stochastic Team Orienteering Problem (STOP). The second covers solutions to the MOP used to coordinate a team of agents when a centralized solver is unavailable or infeasible.

The Stochastic Orienteering Problem extends the classic NP-Hard deterministic OP by introducing stochastic costs and rewards sampled from probability distributions. When solving schedules for a team of agents, we consider the STOP [2]. Recent research, such as Panadero et al.'s work, has focused on solving STOP using simheuristic methods, particularly the state-of-the-art simulation biased-randomized Variable Neighborhood Search (Sim-BRVNS) algorithm, which effectively combines simulations with metaheuristics in an anytime solver [5], [6]. Additionally, Juan et al. and Karunakaran et al. explore genetic algorithms for solving STOP variants. Juan et al. incorporate Monte Carlo simulation to handle stochasticity, while Karunakaran et al. use an island model to evolve diverse policies that are robust to varying scenarios [7], [8].

In distributed systems, the MOP is commonly used to coordinate team schedules, allowing agents to exchange information and converge on a global solution [2]. Recent works by Best et al. and Skrynnik et al. employ a decentralized Monte Carlo Tree Search (Dec-MCTS) method to solve tours for multiple agents in an anytime, distributed manner by communicating partial solutions between agents. Dec-MCTS's action distribution method has proven effective, providing anytime performance in lossy networks [9], [10]. Other researchers have explored MOP variants, such as the prize-collecting MOP, where Murray et al. proposed policies to minimize inefficiencies [11], and the robust multiple-path orienteering problem, for which Shi et al. introduced approximation-based and MCTS-based solutions [12].

Our work lies at the intersection of centralized STOP and distributed MOP approaches. By extending MOP to incorporate stochastic costs, we can use distributed MOP solvers to enable robust coordination among Passengers in uncertain environments. Introducing a Mothership robot into the MRS allows us to leverage powerful STOP solvers from the literature, developing centralized solutions that enhance the distributed system.

III. Problem Formulation

We consider a team of robots that must coordinate in an uncertain environment to complete spatially-distributed tasks. Each task can only be completed once, so revisiting a completed task offers no reward. The team must coordinate to create schedules, or tours, for each member to follow to multiple tasks. The energy expended by a robot moving between two locations is difficult to predict but can be modeled with a probability distribution. The objective is to select a set of schedules that maximize the reward gained from completed tasks. However, unpredictable disruptive events may occur during execution, motivating online rescheduling.

We begin formalizing the SMOP by defining a team of N robots $\mathbf{a} = \{a_1, a_2, ...a_N\}$, with each robot a_i assigned an energy budget b_i, a start task, and an end task. The environment contains V spatially-distributed tasks $\mathbf{v} = \{v^1, v^2, ...v^V\}$. Each task v^j is defined with a deterministic location, work cost c_w^j, and reward r^j, such that when robot a_i completes task v^j it reduces its budget by c_w^j and collects reward r^j. The stochastic cost incurred by a robot traveling from task v^j to task v^k is defined by $c_e^{j,k} > 0$, which follows a Gaussian distribution with mean $\mathbb{E}[c_e^{j,k}] > 0$.

To represent the problem, we model the environment as a complete graph $G = \{\mathbf{v}, \mathbf{e}\}$, where the set of tasks $\mathbf{v}$ forms the nodes and the edges $\mathbf{e}$ represent the travel costs between tasks. The travel costs are probabilistic and represented by distributions. We assume that the distribution of cost realizations is well-represented by a set of scenarios, $\mathbf{S}$, where each scenario $\mathbf{s} \in \mathbf{S}$ contains a set of sampled edge costs, denoted $\mathbf{s} = \{c_e^{0,1}, c_e^{0,2}, ...\}$.

We solve for a tour, or sequential ordering of nodes, for each robot to follow. A tour for a_i is denoted $\mathbf{t}_i = (v_{i,1}, v_{i,2}, ...)$, and the reduced set of tasks that are uniquely visited by a_i is denoted $\mathbf{u}_i = \{v_{i,1}, v_{i,2}, ...\}$. We consider $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N\}$ as a solution to the problem in solution space $\mathcal{T}$.

The goal is to maximize the total reward accumulated by all robots, considering the stochastic nature of travel costs. For each scenario $\mathbf{s}$, the reward depends on the number of unique tasks completed by each robot, provided their energy budgets are not exceeded. With this, we solve for the optimal set of tours $\mathbf{T}^*$ that maximizes task completion efficiency. The objective can be expressed as:

$$\mathbf{T}^* = \arg\max_{\mathbf{T} \in \mathcal{T}} \frac{1}{|\mathbf{S}|} \sum_{\mathbf{s} \in \mathbf{S}} \sum_{i \in N} f(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s}), \tag{1}$$

where $f(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s})$ represents the total reward gained from $\mathbf{u}_i$, derived as:

$$f(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s}) = l^{\mathbf{s}}(\mathbf{t}_i) \cdot \sum_{v_{i,j} \in \mathbf{u}_i} r^j. \tag{2}$$

The indicator function $l^{\mathbf{s}}(\mathbf{t}_i, \mathbf{s})$ enforces budget constraints by forcing the tour's reward to 0 if the sampled tour cost exceeds b_i:

$$l^{\mathbf{s}}(\mathbf{t}_i, \mathbf{s}) = \left\{ \begin{array}{ll} 1, & \text{if } b_i - \sum_{v_{i,j} \in \mathbf{t}_i} (c_e^{j-1,j} + c_w^j) > 0 \\ 0 & \text{otherwise} \end{array} \right\}. \tag{3}$$

In the multi-robot setting, each robot seeks the tour $\mathbf{t}_i^*$ that maximizes its individual contribution to the global reward. We consider the solution space of individual tours $\mathcal{P}$, such that $\mathbf{t}_i \in \mathcal{P}$. The utility of robot i's schedule is calculated as the difference in the reward with robot i participating and without robot i's contribution. With this, we optimize:

$$\mathbf{t}_i^* = \underset{\mathbf{t}_i \in \mathcal{P}}{\arg\max} \frac{1}{|\mathbf{S}|} \sum_{s \in \mathbf{S}} g(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s}), \tag{4}$$

where

$$g(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s}) = \sum_{a_i \in \mathbf{a}} f(\mathbf{t}_i, \mathbf{u}_i, \mathbf{s}) - \sum_{a_j \in \mathbf{a} \setminus a_i} f(\mathbf{t}_j, \mathbf{u}_j, \mathbf{s}). \tag{5}$$

IV. Hybrid Rescheduling

We explore two ways in which the Mothership supports a distributed team of Passenger robots: 1) through offline planning before deployment, and 2) by providing online schedules enhanced by aggregated data. This section details the algorithms used for scheduling on the Mothership and each Passenger, and how planning information is shared across the hybrid system. Fig. 2 provides an overview of our hybrid framework, which guides the following sections.

This framework governs the exchange of scheduling information between all robots in the MRS. The Mothership M runs a centralized solver to generate tours for one or more robots, sharing each tour $\mathbf{t}_i^m$ with the respective robot a_i. Passenger a_i solves a set of k tour options $\mathbf{T}_i^l = \{\mathbf{t}_{i,1}^l, \mathbf{t}_{i,2}^l, ..., \mathbf{t}_{i,k}^l\}$, which it compares against its stored set of tour options $\mathbf{T}_i = \{\mathbf{t}_{i,1}, \mathbf{t}_{i,2}, ..., \mathbf{t}_{i,k}\}$ and the received Mothership tour $\mathbf{t}_i^m$. The local schedule selection algorithm $UpdateScheduleDists$ reduces these to a set of the k best tours and updates a_i's stored set $\mathbf{T}_i$.

Each tour $\mathbf{t}_{i,j} \in \mathbf{T}_i$ is associated with a probability $p_{i,j}$, forming a distribution $\mathbf{p}_i = \{p_{i,1}, p_{i,2}, ..., p_{i,k}\}$ representing the likelihood that a_i will follow each tour in $\mathbf{T}_i$. When a_i's tour distribution $\theta_i = (\mathbf{T}_i, \mathbf{p}_i)$ is shared with other robots, it informs their sampling-based planners about a_i's likely task visits. Each robot stores a set of received schedule distributions $\boldsymbol{\theta}_i = \{\theta_1, \theta_2, ..., \theta_N\}$ for reference during planning.

A. Centralized Solver

We adopt the Sim-BRVNS algorithm from Panadero et al. to solve the STOP on the Mothership [5]. The $SimBRVNS$ function solves the STOP for a team of N robots by:

1) Converting the STOP into a deterministic problem using the mean values of edge cost distributions.
2) Solving this deterministic problem using a constructive heuristic
3) Evaluating the solution in a stochastic setting via Monte Carlo Simulation (MCS), where edge costs are sampled from the Mothership's graph G_m
4) Applying a biased-randomized variable neighborhood search to explore additional solutions

The solver iterates between steps 3 and 4 within a given time limit, returning the best-performing set of N schedules based on the evaluation function (1). For more details on this method, refer to [13], [6].

In the $SMOP\text{-}SimBRVNS$ function, we extend Sim-BRVNS to the multi-robot SMOP case using the schedule distributions from $\boldsymbol{\theta}_m$ stored on the Mothership. In this case, the Mothership solves a single tour for a_i. During each iteration of the MCS stage, the solver samples a set of tours for all robots except a_i. The sampled tours and the current solution are evaluated on G_m, and the utility reward for a_i's tour is calculated using (4). At the end of the solver's runtime, the highest-scoring solutions $\mathbf{T}_i^m$ are returned.

B. Distributed Solver

We select the $Dec\text{-}MCTS$ algorithm proposed by Best et al. to solve local tour solutions on each Passenger a_i [9]. The algorithm grows a search tree by cycling through four phases: selection, expansion, simulation, and backpropagation, while sharing partial solution information between robots. Each node in the tree represents a tour, with the root node corresponding to the Passenger's current location. The algorithm returns the set of the k best-performing tours $\mathbf{T}_i^l$. For further reading on Dec-MCTS, see [9].

We apply this algorithm to the SMOP by adding an additional sampling layer to the rollout phase. During this phase, multiple rollouts of the leaf node's tour are performed. The rollout process simulates adding valid tasks to the tour, maximizing the ratio of added reward to incurred travel cost. These travel costs are sampled from the local graph G_i. Nodes are considered valid only if they are not already included in tours sampled from $\boldsymbol{\theta}_i$ at the start of the rollout. The local utility reward of the simulated tour is then calculated against the sampled tours using (4).

C. Hybrid Scheduling Algorithms

Two distinct algorithms, Alg. 1 (Mothership Rescheduling) and Alg. 2 (Passenger Rescheduling) run on the Mothership and each Passenger, respectively. Alg. 1 enables the Mothership to solve either the STOP for team scheduling (line 2) or an individual robot's schedule for online SMOP (line 6). The STOP is used at the mission's outset to generate an offline mission plan for each robot, while tours for the SMOP are solved during runtime for a_i as requested. Basic communication functions $Send(Content, Receiver(s))$ and $Receive(Content, Sender)$ to share data between robots.

Alg. 2 manages rescheduling for each Passenger a_i. Upon startup, a_i receives an initial tour from the Mothership and begins executing it (lines 1-5). During runtime, a_i can request updated schedules from the Mothership (line 6). If a new schedule is received, a_i updates its schedule distribution

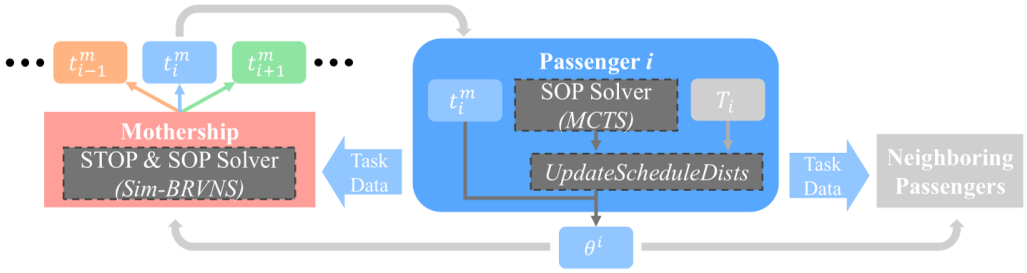


Fig. 2: Procedure for sharing schedule distributions within system. The Mothership's centralized solver generates set $\mathbf{T}^m$ of tours and sends $\mathbf{t}_i^m$ to Passenger a_i. On a_i, a MCTS solver produces set $\mathbf{T}_i^l$ tours with local information. New schedules on a_i are compared to stored set $\mathbf{T}_i$ schedules, with the highest-scoring subset of k schedules assigned as new $(\mathbf{T}_i, \mathbf{p}_i)$

Algorithm 1 Mothership Scheduling

Input: robots $\mathbf{a}$, task graph G_m, stored schedule distributions $\boldsymbol{\theta}_m$

1: **if** mission is beginning **then** ▷ Run on startup
2: $\quad\mathbf{T}^m \leftarrow SimBRVNS(G_m, \mathbf{a})$
3: $\quad Send(\mathbf{T}^m, \mathbf{a})$ ▷ Distribute tours
4: **end if**
5: **if** $Receive(request, a_i)$ **then** ▷ Run as requested by a_i
6: $\quad\mathbf{t}_i^m \leftarrow SMOP\text{-}SimBRVNS(G_m, a_i, \boldsymbol{\theta}_m)$
7: $\quad Send(\mathbf{t}_i^m, a_i)$ ▷ Return solved tour
8: **end if**

Algorithm 2 Passenger a_i Scheduling

Input: robots $\mathbf{r}$, task graph G_i, stored schedule distributions $\boldsymbol{\theta}_i$, stored local schedules $\mathbf{T}_i$
Output: new schedule $\mathbf{t}_i$

1: **if** mission beginning **then** ▷ Receive init. tours
2: $\quad\mathbf{t}_i \leftarrow$ extract $\mathbf{t}_i^m$ from $Receive(\mathbf{T}^m, M)$
3: $\quad\theta_i \leftarrow ([\mathbf{t}_i], [1.0])$ ▷ Init. dist. $(\mathbf{T}_i, \mathbf{p}_i)$
4: $\quad$**return** $\mathbf{t}_i \leftarrow \arg\max_{\mathbf{t}_i \in \mathbf{T}_i}[p_i(\mathbf{t}_i)]$
5: **end if**
6: $Send(request, M)$ ▷ Request tour from M
7: **if** $Receive(\mathbf{t}_i^m, M)$ **then** ▷ Use received tour
8: $\quad\theta_i \leftarrow ([\mathbf{t}_i^m], [1.0])$
9: **else** ▷ Else solve local tours
10: $\quad\mathbf{T}_i^l \leftarrow Dec\text{-}MCTS(G_i, a_i)$
11: $\quad(\mathbf{T}_i, \mathbf{p}_i) \leftarrow UpdateScheduleDist(\mathbf{T}_i \cup \mathbf{T}_i^l)$
12: $\quad\theta_i \leftarrow (\mathbf{T}_i, \mathbf{p}_i)$
13: **end if**
14: $Send(\theta_i, \mathbf{a} \cup M)$ ▷ Share new sched. dist.
15: **return** $\mathbf{t}_i \leftarrow \arg\max_{\mathbf{t}_i \in \mathbf{T}_i}[p_i(\mathbf{t}_i)]$

(lines 7-9). If no update is provided, the Passenger runs $Dec\text{-}MCTS$ to generate local solutions (line 10). These local options are then evaluated against stored schedules using the $UpdateScheduleDist$ function (line 11) as described in the next section. Finally, the Passenger shares its updated schedule distribution with the Mothership and nearby Passengers (line 14), and then returns the best-performing schedule from the new distribution for execution.

These algorithms equip the system to operate as a centralized network when Passengers have consistent connection to the Mothership by granting centrally-generated plans precedence over local tours. Additionally, Alg. 2 enables Passengers with strong connections to neighboring robots to develop plans as a distributed system by exchanging schedule distributions.

D. Updating Schedule Distributions

The $UpdateScheduleDist$ function evaluates both old and new schedules developed for a_i to determine the best subset to carry forward. Alg. 3 outlines the procedure used to assess and select the top schedules from a set of candidates. The algorithm iterates through each candidate schedule $\mathbf{t}$, first modifying $\mathbf{t}$ according to locally-stored information with $Prune$ (i.e. removing tasks that a_i knows have been completed).

A Monte Carlo Simulation is then performed using local budget constraints and graph information to assess the failure probability of $\mathbf{t}$ across multiple scenarios (line 4). Next, $LocalUtil$ evaluates the local utility of $\mathbf{t}$ using (4). The utility and failure probability are combined into a score α for each tour (lines 5 and 6). The algorithm stores each tour-score pair in a list (line 7). Tours are then sorted by α and reduced to a list of the top n candidates (line 9). Finally, the scores $\boldsymbol{\alpha}$ are normalized to create a probability distribution $\mathbf{p}$ for the selected schedules $\mathbf{T}'$.

V. Experiments and Results

We evaluate our Dec-MCTS SMOP solution and two versions of our hybrid Mothership-Passenger algorithms alongside a centralized, offline STOP solver, defined as follows:

- Sim-BRVNS: Solve STOP with Sim-BRVNS to generate a full team schedule offline prior to mission execution. No online optimization [5].
- Dec-MCTS: Use Dec-MCTS to solve local schedules without input from Mothership. No initial offline scheduling, no online hybrid rescheduling [9].
- 2-Stage (Ours): Solve initial offline schedules, then perform local-only rescheduling during deployment.
- 2-Stage Hybrid (Ours): Solve initial offline schedules, then reschedule using local solutions and requested solutions from Mothership (Alg. 1 and Alg. 2).

Algorithm 3 UpdateScheduleDist

Input: tours to evaluate $\mathbf{T}$, stored schedule distributions $\boldsymbol{\theta}_i$, budget b_i, task graph G_i
Output: new schedule distribution $(\mathbf{T}_i, \mathbf{p}^i)$

$pairs \leftarrow \emptyset$
for $\mathbf{t} \in \mathbf{T}$ **do**
  $Prune(\mathbf{t}, G_i)$ ▷ Prune completed tasks
  $rel \leftarrow MCS(\mathbf{t}, b_i, G_i)$ ▷ Eval. fail rate
  $rew \leftarrow LocalUtil(\mathbf{t}, \boldsymbol{\theta}_i)$ ▷ Eq. 4
  $\alpha \leftarrow rew \cdot rel$ ▷ Compute tour score
  $pairs \leftarrow pairs \cup (\mathbf{t}, \alpha)$ ▷ Add to pairs
end for
$(\mathbf{T}', \boldsymbol{\alpha}) \leftarrow SelectTopSchedules(pairs, n)$
$\boldsymbol{\alpha} \leftarrow Normalize(\boldsymbol{\alpha})$
return $(\mathbf{T}', \mathbf{p})$

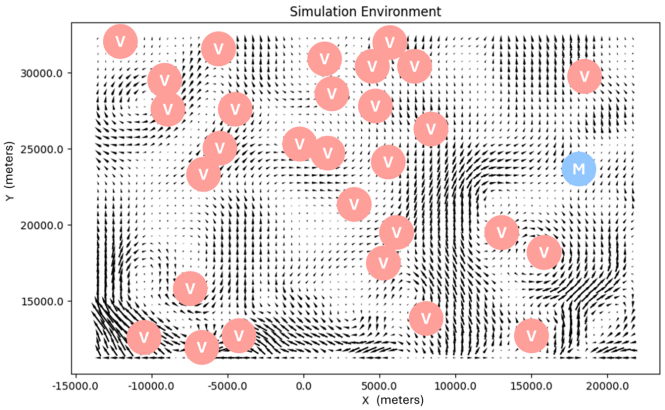


Fig. 3: Example simulation environment with sampled flow data, random task locations V, and random Mothership location M.

A. Evaluation Methods

We motivate our experiments within the context of an under-ice environment. The operational area is modeled as a 2D grid, utilizing simulated under-ice ocean current data generated by Si et al. [14]. These ocean currents are unknown to the robots at the start of the mission and influence vehicle movements by increasing energy consumption when traveling against strong flows. Robots traveling with the current experience less energy drain. All robots are assumed to move at a constant velocity. Given the uncertainty in energy cost predictions due to unknown currents, robots model movement costs as stochastic. Each test environment is randomly sampled from a pool of 20 flow datasets.

Communication between robots occurs at a fixed frequency, with message success rates impacted by underwater absorption and signal spreading. We approximate packet loss as an exponential decay function of distance between transmitting and receiving robots, with packet success probabilities reducing to 80% at 5000 meters [15]. Beyond this range, the success rate decreases rapidly to the point where messaging attempts are no longer viable.

The simulated MRS consists of a Mothership, Worker robots, and Support robots. Each Worker is provided with a team of Support robots, who provide infrastructure for communication and localization by distributing themselves evenly between the Mothership and the Worker. The Mothership and Workers focus on coordinating tours to randomly distributed tasks. The Mothership serves as both the starting and ending point for each Worker's schedule, and its position is randomly assigned at the start of each test. Fig 3 shows an example simulation environment, including the Mothership location (M) and the task locations (V's). During operation, Workers share completed task information with the Mothership and neighboring robots. In the online rescheduling cases, Workers update their tours at a fixed frequency during runtime.

We evaluate each algorithm's ability to coordinate teams of robots in this stochastic environment with increasing levels of disruption with the following tests:

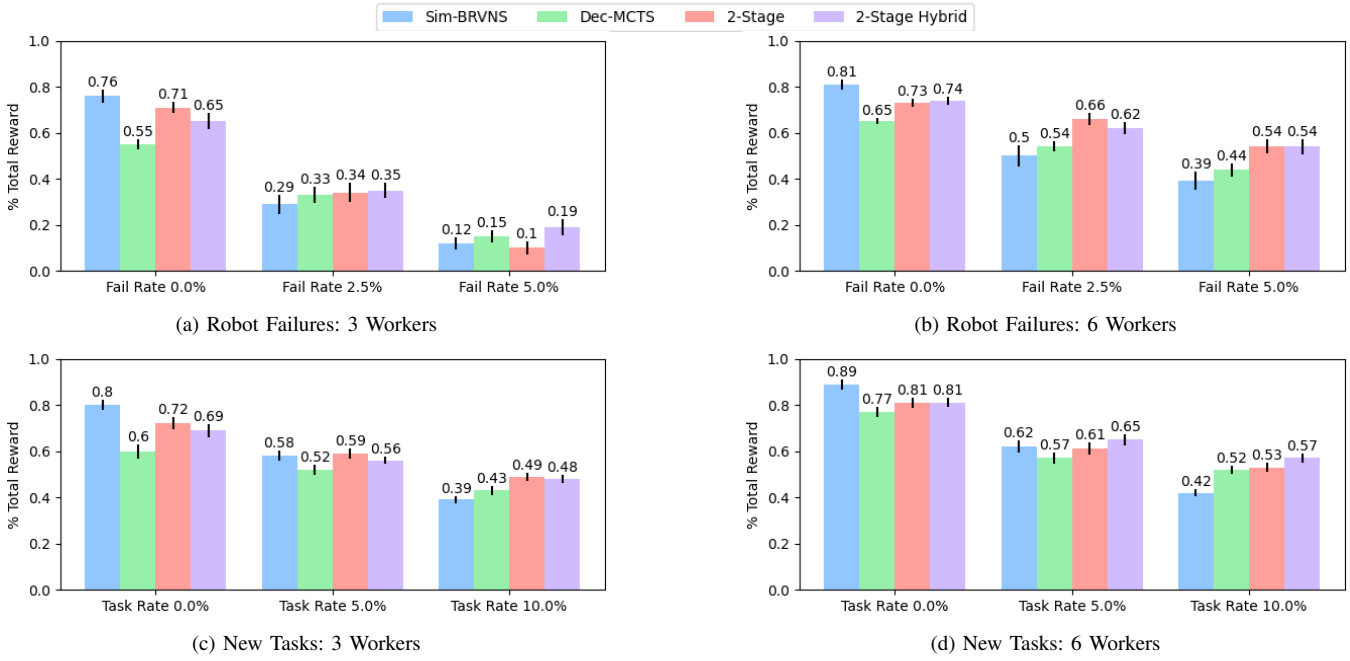
1) Robot Failures: Initialize 30 tasks. At each simulation step, there is a probability (0%, 2.5%, 5%) that a random worker will experience a failure, resulting in zero, some, or most robots dead at the end of runtime. This failure has a 50% chance of either fully immobilizing the worker or partially reducing its remaining battery life by 50%.
2) New Tasks: Initialize 20 tasks. At each simulation step, there is a probability (0.0%, 5.0%, 10.0%) that the Mothership will generate a new task, resulting in the problem size increasing by approximately 0%, 50%, or 100% throughout runtime.

Finally, we assess the algorithms' performance in each test with teams of 3 and 6 Workers to observe scalability trends between group sizes.

B. Results

For each test case, we ran each algorithm in 30 randomly-generated environments. Fig. 4 presents the task efficiency rate results, or the mean task rewards gained by all Workers that successfully completed tours through the environment and returned to the Mothership, per (1).

We observe the following high-level trends in the test results. First, in the baseline scenarios (0.0% robot failures or new tasks), the offline Sim-BRVNS solver achieves the highest mean reward, consistent with the results expected from the powerful STOP solver. Our 2-Stage and 2-Stage Hybrid algorithms follow, while Dec-MCTS performs the worst in every baseline test, highlighting the benefit of an initial planning stage. Next, as disturbances are introduced into the environment via failures and new tasks, the efficacy of the initial offline plan tends to decrease at a higher rate relative to the other methods. On average, the reward obtained by Sim-BRVNS declined by 60.0% between the baseline and high disturbance scenarios over all tests. The 2-Stage method experienced the next highest average decrease of 44.6%, followed by 2-Stage Hybrid at 39.5%, and Dec-MCTS at 32.4%. These results underscore the robustness provided by replanning in highly dynamic scenarios. Finally, in the high-disturbance cases (5.0% robot failures or 10.0%



(a) Robot Failures: 3 Workers

(b) Robot Failures: 6 Workers

(c) New Tasks: 3 Workers

(d) New Tasks: 6 Workers

Fig. 4: (a), (b) Algorithm comparison results for teams of 3 and 6 Worker robots operating in a 30 task environment with increasing per-timestep robot failure probability. (c), (d) Algorithm comparison results for teams of 3 or 6 Worker robots operating in a 20 task environment with increasing per-timestep new task introduction probability. Results show mean task completion efficiency with standard error over 30 trials. As environment dynamics increase, the utility of the initial offline Sim-BRVNS solution degrades while replanning approaches remain robust. 2-Stage and 2-Stage Hybrid achieve relatively high rewards in most instances, with 2-Stage Hybrid performing best in high-disruption scenarios where the utility of the mothership's information is greatest.

new tasks) we observe that 2-Stage Hybrid completes tasks most efficiently. In particular, 2-Stage Hybrid achieves a 26.6% mean reward improvement over Dec-MCTS in the 3 Worker 5.0% failure test and a 7.5% increase over 2-Stage in the 6 Worker 10.0% task rate test, while performing competitively with 2-Stage in the other high-disturbance scenarios. These results suggest that the hybrid rescheduling phase is effective in bolstering the quality of rescheduled solutions in highly dynamic scenarios.

The results also draw out differences between the offline STOP solution and our online SMOP approaches. In all baseline scenarios, performance declines from Sim-BRVNS to our 2-Stage rescheduling methods. Despite starting with a highly optimal offline plan, the 2-Stage approaches allow Workers to replan conservatively by selecting low-risk tours that skip high-risk tasks. Additionally, 2-Stage outperforms 2-Stage Hybrid in many low-to-medium disruption scenarios. This is likely the result of two aspects: 2-Stage following the initial offline solution longer than 2-Stage Hybrid due to low-quality Dec-MCTS solutions being rejected by Alg. 2, and 2-Stage Hybrid lacking sufficient information to produce Sim-BRVNS plans that improve on the initial solution due to communication limits. As a result, 2-Stage Hybrid performs best when environment dynamics diminish the utility of the initial plan and when the mothership is well-informed.

Overall, our results show that the 2-Stage and 2-Stage Hybrid algorithms consistently perform best in the presence of disruptive events. Both approaches scale well across different problem sizes, as their distributed nature allows computation to be performed in parallel across the mothership and all passengers. However, we anticipate diminishing performance with larger instances, as anytime solvers may struggle to find high-quality solutions given limited solving time. Initial offline planning improves performance in most scenarios, and online replanning maintains this advantage in dynamic conditions. In highly dynamic cases, 2-Stage Hybrid's distributed scheduling approach, which integrates centralized and local planning, achieves the best results.

VI. Conclusion

In this work we explored leveraging a Mothership-Passenger MRS for solving the Stochastic Multi-Agent Orienteering Problem. We formulated the problem, in which a team of robots must coordinate to solve tours through a stochastic environment while adhering to budget constraints. We presented a distributed solution to this problem based on Dec-MCTS and a hybrid centralized-distributed algorithm that reinforces the distributed algorithm with solutions inserted by a centralized Mothership. We then presented the results from two sets of experiments carried out in a simulated under-ice domain. The tests showed the resilience of our algorithms to a variety of disturbances, underscoring the coordination benefits provided by an initial offline scheduling phase and highlighting the utility of well-informed Mothership-provided plans during online rescheduling. Future work will seek to reduce the communication overhead in the system and explore ways to adapt our framework to more general problem formulations.

REFERENCES

[1] R. B. Alley, S. Anandakrishnan, K. Christianson, H. J. Horgan, A. Muto, B. R. Parizek, D. Pollard, and R. T. Walker, "Oceanic Forcing of Ice-Sheet Retreat: West Antarctica and More," *Annual Review of Earth and Planetary Sciences*, vol. 43, pp. 207–231, May 2015. Publisher: Annual Reviews.

[2] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering Problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, pp. 315–332, Dec. 2016.

[3] W. Cai, Z. Liu, M. Zhang, and C. Wang, "Cooperative Artificial Intelligence for underwater robotic swarm," *Robotics and Autonomous Systems*, vol. 164, p. 104410, June 2023.

[4] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot Task Allocation: A Review of the State-of-the-Art," in *Cooperative Robots and Sensor Networks* (A. Koubâa and J. Martínez-de Dios, eds.), Studies in Computational Intelligence, pp. 31–51, Cham: Springer International Publishing, 2015.

[5] J. Panadero, A. A. Juan, C. Bayliss, and C. Currie, "Maximising reward from a team of surveillance drones: a simheuristic approach to the stochastic team orienteering problem," *European Journal of Industrial Engineering*, vol. 14, pp. 485–516, Jan. 2020. Publisher: Inderscience Publishers.

[6] J. Panadero, A. A. Juan, E. Ghorbani, J. Faulin, and A. Pagès-Bernaus, "Solving the stochastic team orienteering problem: comparing simheuristics with the sample average approximation method," *International Transactions in Operational Research*, vol. 31, no. 5, pp. 3036–3060, 2024.

[7] A. A. Juan, A. Freixes, P. Copado, J. Panadero, J. F. Gomez, and C. Serrat, "A Genetic Algorithm Simheuristic for the Open UAV Task Assignment and Routing Problem with Stochastic Traveling and Servicing Times," in *2021 Winter Simulation Conference (WSC)*, pp. 1–12, Dec. 2021. ISSN: 1558-4305.

[8] D. Karunakaran, Y. Mei, and M. Zhang, "Multitasking Genetic Programming for Stochastic Team Orienteering Problem with Time Windows," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1598–1605, Dec. 2019.

[9] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Decentralised Monte Carlo Tree Search for Active Perception," in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics* (K. Goldberg, P. Abbeel, K. Bekris, and L. Miller, eds.), pp. 864–879, Cham: Springer International Publishing, 2020.

[10] A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. Panov, "Decentralized Monte Carlo Tree Search for Partially Observable Multi-Agent Pathfinding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 17531–17540, Mar. 2024. Number: 16.

[11] T. Murray, J. Garg, and R. Nagi, "Prize Collecting Multiagent Orienteering: Price of Anarchy Bounds and Solution Methods," *IEEE Transactions on Automation Science and Engineering*, vol. 19, pp. 531–544, Jan. 2022.

[12] G. Shi, L. Zhou, and P. Tokekar, "Robust Multiple-Path Orienteering Problem: Securing Against Adversarial Attacks," *IEEE Transactions on Robotics*, vol. 39, pp. 2060–2077, June 2023.

[13] A. A. Juan, Y. Li, M. Ammouriova, J. Panadero, and J. Faulin, "Simheuristics: An Introductory Tutorial," in *Proceedings of the Winter Simulation Conference (WSC)*, pp. 1325–1339, Dec. 2022. ISSN: 1558-4305.

[14] Y. Si, A. L. Stewart, A. Silvano, and A. C. Naveira Garabato, "Antarctic Slope Undercurrent and onshore heat transport driven by ice shelf melting," *Science Advances*, vol. 10, Apr. 2024.

[15] W. Kuperman and P. Roux, "Underwater Acoustics," *Springer Handbook of Acoustics*, p. 149, Jan. 2007.