Decentralized Coordination for Multi-Agent Data Collection in Dynamic Environments

Nhat Nguyen, Duong Nguyen, Junae Kim, Gianluca Rizzo, and Hung Nguyen

Abstract—Coordinated multi-robot systems are an effective way to harvest data from sensor networks and implement active perception strategies. However, achieving efficient coordination in a way that guarantees a target QoS while adapting dynamically to changes (in the environment and/or in the system) is a key open issue. In this paper, we propose a novel decentralized Monte Carlo Tree Search (MCTS) algorithm for dynamic environments that allows agents to optimize their own actions while achieving some form of coordination. Its main underlying idea is to balance adaptively the exploration-exploitation trade-off to deal effectively with changes in the environment while filtering out outdated and irrelevant samples via a sliding window mechanism. We show both theoretically and through simulations that in dynamic environments our algorithm provides a log-factor (in terms of time steps) smaller regret than state-of-the-art decentralized multi-agent planning methods. We instantiate our approach to the problem of underwater data collection, showing in a variety of different settings that our approach greatly outperforms the best-competing approaches, both in terms of convergence speed and global utility.

Index Terms—Monte-Carlo Tree Search, Active Perception, Multi-Agent Systems, Underwater Sensor Networks, Autonomous Underwater Vehicles

1 Introduction

Over the last few years, there has been a growing interest in utilizing multi-robot for cooperative data harvesting, such as the use of unmanned aerial vehicles (UAVs) or autonomous underwater vehicles (AUVs) [1], [2]. In particular, decentralized collaborative multi-robot systems for active perception have gained widespread popularity due to their robustness and scalability, which makes them suitable for industrial applications, as well as in environmental monitoring, search and rescue missions, or online object recognition and tracking [1], [3]–[5]. In these schemes, autonomous vehicles (AVs) play the role of wireless relays, moving in the proximity of IoT devices to harvest data and relay them to a fusion center. This allows relieving IoT devices from energy-intensive long-range transmissions, greatly increasing their availability. This is particularly valuable when sensors/tracked objects are too distant from each other and from data sinks, and when the location of sensors and tracked objects, and the surrounding environment change over time in an unpredictable manner, such as in post-disaster scenarios (e.g. after a flood, or an earthquake).

It is this dynamic of the environment that makes the problem of how to effectively achieve coordination in a decentralized manner – emerging from the need to guarantee a target performance, e.g. in terms of Age of Information (AoI) or latency – a key issue still largely unsolved [6]–[8].

Decentralized Monte Carlo Tree Search (Dec-MCTS) [8] is the state-of-the-art approach in multi-robot path planning. A key factor in its performance is an exponentially decreasing forgetting factor to handle the changes in reward distribution caused by other agents' actions and to encourage the exploration of new paths. Nevertheless, recent works have identified significant issues with such approach [9]–[11], showing that it is unfit for addressing effectively realistic scenarios characterized by inherently unpredictable and dynamic changes in the environment, such as uncertain or unknown variations in sensor locations and/or availability.

In this paper, we perform a first step towards tackling the above-mentioned issues. We propose a novel approach for efficient decentralized multi-robot path planning for data harvesting and active perception in volatile environments. Our algorithm is able to cope effectively with complex and unexpected changes in the rewards associated with each data collection task, by having agents periodically share a compressed form of their search trees. Our strategy enables each agent to optimize its actions by maintaining a probability distribution over plans in the joint action space. Moreover, it adopts a sliding window mechanism to avoid accounting for outdated samples. This allows adapting efficiently to changes in the environment, as well as to those due to agents' choices at each round of the data collection process, without the need for a full restart of the search tree. Specifically, our main contributions are:

- We formulate the problem of optimal multi-robot path planning with time-varying rewards, where changes are due to the dynamics of the environment and the actions of each agent.
- We propose a novel sliding window decentralized Monte Carlo Tree Search algorithm (SW-MCTS), which allows each AUV to plan its own trajectory in a way that balances the exploration-exploitation trade-off, to deal effectively

- *N. Nguyen and H. Nguyen are with the School of Computer and Mathematical Sciences, The University of Adelaide, SA 5005, Australia. Email: {nhatdaoanh.nguyen, hung.nguyen}@adelaide.edu.au.*
- *D. Nguyen and J. Kim are with the Defence Science and Technology Group, Australia. Email: {duong.nguyen, junae.kim}@defence.gov.au.*
- *G. Rizzo is with the HES SO Valais, Switzerland, and the University of Foggia, Italy. Email: gianluca.rizzo@hevs.ch.*

with changes in the environment. Our algorithm admits a general class of objective functions and optimizes actions over an arbitrarily long planning horizon. It is anytime and robust with respect to limitations in the frequency and amount of information exchanged among agents.

- We prove formally that our algorithm provides a log factor (in terms of time steps) smaller regret than state-of-the-art decentralized solutions. By doing so, we show that our algorithm converges faster and achieves better performance than the best available decentralized MCTS planning algorithm for coordinated multi-robot data collection and active perception. We provide guarantees for the convergence rate to the optimal payoff sequence even when rewards change, based on an analytical relationship between the sliding window size and the rate of changes in the environment. To the best of our knowledge, our work is the first to provide theoretical bounds and performance guarantees for a decentralized MCTS algorithm with changing rewards.
- We evaluate the performance of our algorithm in a multi-drone path planning scenario for underwater data harvesting. Results suggest that our solution substantially outperforms the most relevant existing approaches, both in static settings and in the presence of frequent changes in sensor distribution. The numerical assessment of our approach shows that it consistently achieves faster convergence and higher resource efficiency than competing algorithms, even when decreasing the frequency of coordination exchanges among agents.

The paper is structured as follows. In Section 2 we review the state-of-the-art. Section 3 presents the system model and the problem formulation. Our sliding window MCTS algorithm is illustrated in Section 4. Section 5 presents some analytical results on the properties of our algorithm, which is assessed numerically in Section 6. Finally, Section 7 concludes the paper.

2 Background and Related Work

Information-gathering problems using robots are often modeled as sequential decision-making problems [12]. In the simplest setting with a single agent, the problem reduces to a typical traveling salesman problem - a well-known NP-hard search problem. When there are multiple agents, joint optimization of all agent actions explodes the state space and it results in an intractable problem even for a small number of agents. Some early heuristic solutions include myopic solvers, that minimize the objective function over a limited time horizon [13], [14]. When the objective function is submodular, these methods can achieve near-optimal performance [15]. When this is not the case, non-myopic decentralized coordination algorithms for multiple agents (e.g. [16], [17] and literature therein) have been proposed. However, they rely heavily on problem-specific assumptions, and they are thus not applicable outside of those settings, and in particular, in dynamic scenarios.

In more general settings, decentralized active information gathering is modeled as a decentralized version of a partially observable Markov decision process (POMDP) [18], [19]. The dominant approach to planning in these works consists of first solving the centralized, offline planning over the joint multi-agent policy space, and pushing these policies to agents who execute them online in a decentralized way [19]. These online-offline approaches are however not applicable to dynamic environments where the state of the environment is not known ahead of time. Simultaneous decentralized planning and execution solutions for Dec-POMDP exists [20], but they suffer from significant memory and computational demands to store the reachable joint state estimations of all agents [21]. Several works approximated the problem to a set of local POMDPs that are easier to solve and allow each agent to act independently [22], [23]. Alternatively, role-based abstraction was proposed to decompose the problem into a set of single-agent problems and an optimal task assignment [24], [25]. These techniques can improve computational efficiency and scalability but at the cost of losing global optimality [26].

Recently, decentralized planning using the *anytime* Monte Carlo Tree Search algorithm has gained significant traction due to its flexibility in trading off computation time for accuracy, and it has been applied to several decentralized collaborative planning problems for groups of robots [27]–[30]. The key idea in these solutions is to use MCTS with upper confidence bound (UCB) in order to find the best paths, treating node selection at each step as a multi-armed bandit (MAB) problem [31], [32]. These algorithms seek to find the most rewarding move relative to a given goal at a given time and a given state in a mission or game, based on a search tree with an arborescence that grows and evolves as it is used. In order to create cooperation between agents, these methods usually keep a predefined model of the teammates, which can be heuristic or machine learning trained [27], [29], [30]. However, as they are based on previous knowledge, they are unsuitable for robot motion planning in settings that change in an unpredictable fashion, as is often the case in many application scenarios, such as in post-disaster environments.

One way to coordinate agents without apriori knowledge about their behavior is by having them communicate the intended actions with each other and act accordingly [8], [28]. This introduces *breakpoints* where the reward distribution and optimal actions from one agent's perspective switch as the actions of other agents can vary over time. In [8], the Decentralized MCTS algorithm (Dec-MCTS) uses a discounting factor to handle such changes. However, this technique can be suboptimal in environments where the distributions of rewards change abruptly and independently of the agents' policy [33]. To deal with this uncertainty, Dec-MCTS allows for online replanning during execution by letting agents update their beliefs of the environment and even reset the search tree if changes are deemed substantial. However, determining the *significance* of a change event is often impractical and lacks a clear metric. Frequent resetting could also hinder the performance in scenarios with constrained planning time or computing resources.

Within the larger domain of multi-armed bandit (MAB) approaches, recent works [33]–[38] have focused on non-stationary environments. Some of these approaches are based on the assumption that the dynamics of the changing rewards over time are known [33]–[36], while others assume no previous knowledge on patterns of reward variation [37]–[39]. However, how to apply these results to multi-robot planning via MCTS in nonstationary settings is a non-trivial problem that is still open to date. Indeed, it involves modeling the evolution of multiple interdependent MABs and the way in which they dynamically impact each other's decisions. Our present work tackles these challenges, providing for the first time a sliding-window-based MCTS algorithm for nonstationary environments where changes cannot be forecasted, proving formally its convergence properties, and assessing its performance via simulation in a concrete scenario of underwater data harvesting by a set of autonomous underwater vehicles (AUVs).

3 System Model and Problem Definition

3.1 Basic Assumptions

We consider a set of $\mathcal{S}$ of S wireless sensors, each with a transmission range R, arbitrarily distributed within a volume of space, and let s be the label of the s-th element of the set. These sensors may model e.g. a WSN for underwater monitoring of a seaport exit, or for hydrogeologic measurements, among others. We assume sensors may move over time within the volume of reference, e.g. as in the case of wildlife monitoring, or in underwater monitoring when sensors are displaced by water currents, movement of marine species, and other factors.

We assume that, within the given volume of space, *data sinks* collect data from sensors and deliver them e.g. to the cloud for further elaboration. In water monitoring systems, data sinks are typically located on the surface of the water volume monitored, while in post-disaster communications they usually lie at the border of the monitored area.

We assume that, in general, sensor nodes are not able to relay data directly to a sink. Thus, in the given volume a set of $\mathcal{M}$ of M homogeneous *data harvesting agents* (e.g., modeling UAVs, or underwater drones) periodically collect data from sensors by moving within their transmission range and establishing a direct wireless connection with the sensors. Then the data harvesting agents head to one of the *data gathering points* of the volume, where they relay the collected data either directly to a data sink or (as is the case in many underwater scenarios) to a *transporting agent*, that carries the data to a sink. We assume each agent is endowed with a wireless communications interface for direct device-to-device exchange of data with sensor nodes and with transporting agents, and with the same transmission radius as sensor nodes. In addition, we assume that each agent is endowed with another wireless communication interface (e.g., cellular in the case of UAVs) for information exchange with all of the other agents in the given volume of space.

The path and schedule of each transporting agent are such that the data carried by each harvesting agent can be considered as transferred instantaneously to a transporting agent as soon as the harvesting agent gets to a gathering point. Let m be the label of the $m-$th harvesting agent. Without loss of generality, we assume the trajectories of the harvesting agents to be constrained on a *motion graph* G, i.e. a directed graph defined at system setup time, and which does not vary over time. The motion graph typically models constraints to agent trajectories due to e.g. morphology of the monitored volume, presence of obstacles, maximum distance from sensors sufficient for successfully harvesting

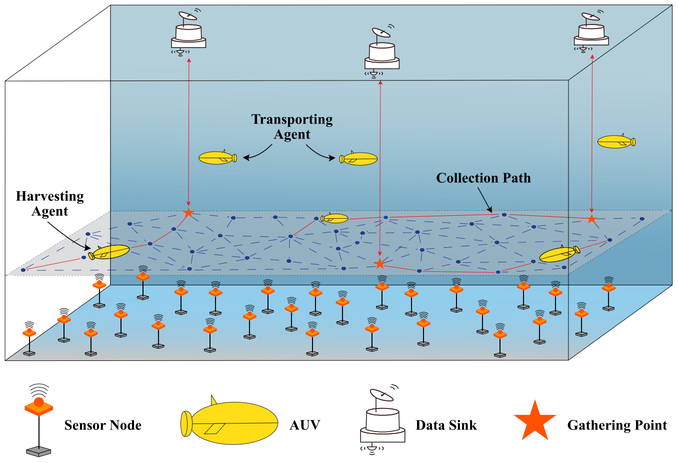

Fig. 1: Example of multi-agent data collection scenario for the case of an underwater WSN. Both harvesting agents and transporting agents are implemented by AUVs.

data, limitations in agent movements (e.g. on a road grid), legal constraints on agent paths, among others. The specific way in which the graph is derived is thus application- and context-dependent, and it is out of the scope of the present work. A schematic representation of the system and an example of the path model are illustrated in Fig. 1. All harvesting agents know the motion graph for the considered volume and the locations of the gathering points.

We assume the data harvesting process takes place as a sequence of *missions*, all of the same duration. At the beginning of a mission, each agent is located at a data gathering point, and all sensors have data to be harvested. During each mission, each agent moves along the graph G and collects data from sensors. By the end of the mission, each agent reaches a data gathering point, delivers the harvested data to a transporting agent, and waits for the beginning of the next mission. With p_m we denote a *path* of the m-th agent on the motion graph during a mission, consisting in an ordered list of edges $p_m = (e_m^1, e_m^2, \ldots)$, such that two adjacent edges in the path are connected by a vertex of the motion graph G. We denote the collection of paths of every agent in a mission as $p = (p_1, ..., p_m, ...p_M)$. The travel budget B is the maximum number of edges an agent can traverse in a mission, and it is assumed to be the same for all agents. Such a maximum value is a function of the duration of the mission, of the agent's speed, but it may also account for several other constraints, e.g. due to finite storage capacity, and finite energy budget at each mission.

Each mission takes place over a finite time horizon. At the beginning of a mission, each agent is located at a gathering point. All agents perform distributed *planning* together, to decide a path for each agent. Each agent then *executes* its planned path by moving along the first edge of the path, harvesting data from all sensors for which it gets within the transmission range, possibly updating its information on the position and availability of those sensors. We assume agents can observe the location and availability of sensors using localization techniques such as e.g., periodically broadcasting a beacon message to sensor

TABLE 1: Key notation used in the paper.

Notation	Description
$\mathcal{S}$	Set of S sensor nodes
s	Index of the s-th node $\in \mathcal{S}$
R	Transmission radius
$\mathcal{M}$	Set of M harvesting agents
m	Index of the m-th agent $\in \mathcal{M}$
G	Motion graph
B	Travel budget
p	Collection of paths for all agents, in a mission
p_m	Path of the m-th agent in a mission
w_s	Utility of the s-th sensor
P_m	Set of all possible paths of the m-th agent
P	Collection of sets of all possible paths for all agents
$\mathcal{T}_m$	Search Tree of the m-th agent
T	Planning time budget per action
τ	Maximal window size of SW-MCTS
C_p	Exploration parameter of SW-MCTS

nodes, with which each sensor can respond with its updated location, and unresponsive sensors are assumed to be unavailable [40]. This technique is widely applicable since beacon messages have a low bit rate and travel fast while the sensor's data is more complicated and requires close-range transmission [41]. After that, all agents exchange updates on sensor position and availability, and on those sensors whose data has already been harvested. Finally, they plan their residual path with the newly available information. We assume the planning, execution, and replanning phases to be synchronized across all agents. This *planning-executing* procedure repeats until the end of the mission, or until all agents' travel budget expires.

We assume at any time each agent can communicate with all the other agents. In scenarios with D2D communications, this may correspond to the case in which each agent is always in the transmission range of all of the other agents so that each agent implements this information exchange by periodically broadcasting to all others. Or, we may assume that each agent relays the information received by other agents. In this case, all-to-all communications are feasible if the connectivity graph of the agents is always connected. As we assume that all nodes do not move while exchanging information and planning, and as these phases are assumed to be synchronized across all nodes, the exchange duration does not affect the performance of our algorithm. For ease of analytical treatment, we assume that traversal time is the same for all edges. In addition, we assume any exchange of information (e.g. from sensors to agents, from agents to data sinks, and among agents) to be instantaneous. Note however that our approach can be easily extended to account for realistic exchange durations, as well as for different edge traversal times.

At the beginning of each mission, all sensors have data to be harvested. We consider that the first time in the mission that a sensor finds an agent within its transmission range, it sends its data, removing them from memory. We denote this event as a *successful harvesting* event. Therefore, all subsequent events of contact between that sensor and agents during the mission will not bring to any data transfer (*failed harvesting*). This satisfies the diminishing returns property, which frequently arises in data collection applications [42], [43], i.e., there is no additional gain to revisit a sensor that has already been collected. This utility formulation belongs to the class of submodular reward functions, which is broadly applicable to many planning problems (e.g., coverage control [44], informative path planning [45], resource allocation [46]). All data from sensors not collected during a mission is discarded at the end of the mission by the sensors themselves (e.g. since more recent data is made available). In practice, the reward associated with a path can be modeled as the number of unique sensors crossed by that path. This formulation is fast to compute and it naturally aligns with many practical applications such as object classification [8] or area coverage [47]. We assume that at the beginning of the mission, agents know the exact position and the operating status of the sensors. However, we assume that during a mission both the position and the operating status of each sensor may change. Note that mission duration can be set in such a way as to account for, e.g., the mean speed of agents, node distribution in the volume, and maximum acceptable worst-case age of the information collected, among others.

3.2 Problem Formulation

The goal of path planning is to find a path for each agent such that the joint utility of the data collection task is maximized. The utility function is defined as follows. To every harvesting event at the s-th sensor, we associate a utility, which is equal to w_s in case of success, and zero otherwise. w_s is in principle different for each sensor. This reward structure is appropriate for applications with homogeneous sensors such as in tectonic movement monitoring where each sensor covers a given seabed area. More specific reward structures where different sensors have different rewards and that rewards decay or improve with time can be developed for specific applications but will not fundamentally alter our solution for decentralized agent coordination.

Let $r(p_m)$ denote the total utility associated with the traversal of path p_m. It is equal to the sum of the utility of all the successful harvesting events that the m-th agent has performed while traversing it. For a same path, such quantity may in principle differ, e.g., due to sensor movement, changes in sensor availability, and/or harvesting utility. Moreover, given that failed harvesting events yield zero utility, r_m depends on the paths of all the other agents, in addition to that of the m-th agent. Moreover, to any path p_m on the motion graph, we associate a cost $b(p_m)$, equal to the number of edges of the path. Note however that our analysis can be easily extended to the case of different edge costs.

Let us consider a given mission, and the location of each agent at the beginning of such mission. With P_m we denote the set of all possible paths that start at agent m's starting position and end at a gathering point. Let $P = (P_1, ..., P_m, ..., P_M)$ denote the collection of sets of paths for all agents. We define the following problem:

Problem 1. (*Multi-robot path planning in non-stationary settings)*

$$\underset{p\in P}{\text{maximize}}\ \mathcal{R}(p)=\sum_{m\in\mathcal{M}} r(p_m), \tag{1}$$

Subject to, $\forall m$,

$$b(p_m)\leq B \tag{2}$$

Constraint (2) derives from imposing that the total path cost for the m-th agent is less than the travel budget B available to each agent.

Such an optimization problem cannot be solved efficiently in polynomial time. Indeed it is easy to see that in the simplest cases, it is a variant of the well-known NP-hard traveling salesman problem. In the next section, we provide an adaptive and distributed learning solution to Problem 1, that learns from the environment and the actions of other agents, updating planning decisions accordingly.

4 The Sliding Window MCTS Algorithm

4.1 Algorithm Overview

Our algorithm, denoted as *sliding window MCTS (SW-MCTS)* implements a distributed planner that aims at finding the best course of action (CoA) for each agent in each round, i.e., to determine for each agent a path from P_m which globally maximizes the utility function in Problem 1. Given a planning time budget T, the distributed planner determines these paths (one per agent), executes the first action (i.e., traverse the first edge), updates the information used for path planning (such as sensor position and availability, which might have changed), and replans. Each agent runs an independent instance of SW-MCTS.

The best CoA for each agent is determined using a Monte Carlo tree search, where a node of the tree corresponds to an edge of the motion graph, and a sequence of nodes starting from the root represents a valid travel path for agent m. Specifically, our algorithm extends the well-known UCT bandit algorithm for tree search [31] to non-stationary settings. Let $\mathcal{T}_m$ represent the MCTS tree of agent m, which is built on the set of all its possible paths P_m. In order to find the optimal equilibrium between exploration and exploitation in the presence of changes in the reward scheme, the action selection problem at each internal tree node is modeled as a separate multi-armed bandit, in which the arms correspond to the possible children nodes from each node, and the payoff to the result of the rollout episode that traverses the node. The pseudo-code of SW-MCTS for the m-th agent is shown in Algorithm 1.

For coordination between agents with SW-MCTS, each agent optimizes its own actions while accounting for other agents' choices by maintaining a probability distribution over its search tree. To this end, at each planning iteration t, every agent m exchanges with other agents a description of the set of best paths it has chosen so far, in the form of a set of paths $\hat{P}_m$ and of a probability mass function q_m which associates to every element of $\hat{P}_m$ a value of probability, which is function of how likely that path would be chosen by agent m. In the same way, to account for the effect of other agents' choices, every agent maintains a set $\hat{P}_{(m)}$ of paths that other agents might take, and its probability mass function $q_{(m)}$. To reduce the computation and communication requirements for agents, the set $\hat{P}_m$ may be built to include only those paths that are sampled the most thus far (Line 3).

Algorithm 1 SW-MCTS for agent m

```
Input: Planning time budget $T$, travel budget $B$, set of
    possible paths $\hat{P}_{(m)}$ and probability mass function $q_{(m)}$
    of other agents
Output: Travel path $p_m$ for agent $m$
 1: $\mathcal{T}_m \leftarrow$ Initialize the MCTS Tree
 2: while planning time budget not met do
 3:     $\hat{P}_m \leftarrow$ Select subset $(P_m)$
 4:     for fixed number of iterations, at iteration $t$ do
 5:         $p^t_{(m)} \leftarrow$ Sample paths for other agents $(\hat{P}_{(m)}, q_{(m)})$
 6:         Expanded Node $i \leftarrow$ SW-UCT Selection $(\mathcal{T}_m)$
 7:         $p^t_m \leftarrow$ Rollout policy $(i, B)$
 8:         Calculate rollout score $F^t_m$ using Eq. (3)
 9:         $\mathcal{T}_m \leftarrow$ Backpropagation $(\mathcal{T}_m, F^t_m)$
10:         $\hat{P}_{(m)}, q_{(m)} \leftarrow$ Update and Communicate $(\hat{P}_m, q_m)$
11:     end for
12: end while
13: return $p_m \leftarrow \arg\max_{p\in\hat{P}_m}[q_m(p)]$
```

In our algorithm, each agent does not know the deterministic action of others. Instead, it takes a sample from the set of shared intended actions $\hat{P}_{(m)}$ based on the associated probability distribution $q_{(m)}$ to predict the choice of others at the beginning of each planning iteration t (Algorithm 1, Line 5). These sampled paths, denoted as $p^t_{(m)}$, are then used by the agent in planning its own optimal path, assuming the other agents are taking these sampled paths. Then, each agent incrementally grows its search tree by adding a new leaf node each iteration via a four-step process: *selection*, *expansion*, *rollout*, and *backpropagation*. To each node i of the search tree, we associate the following parameters (denoted as *node statistics*):

- The *expected value*, which is the average of the utility of all paths which traversed node i;
- For every child node j of i, the number of times it has been visited from the parent node i.

In the *selection* phase, starting from the root node, while the current node is fully explored (i.e., all children nodes have been tried in the past at that node), the *sliding window UCT (SW-UCT)* algorithm (described in the following section) is used to compute a score for each child. Then the agent *selects* the node with the highest score and visits the corresponding next hop. This continues until either the travel budget expires, or the agent reaches a node that is not fully explored. In the latter case, the agent chooses randomly one untried child node to *expand* the tree search (Line 6). Then, the agent applies the *random policy*, randomly picking one child at every node and visiting the corresponding next hop until the travel budget expires (Line 7).

In the *backpropagation* phase, the rollout score F^t_m for p^t_m is computed, which is set equal to the utility of the agent's path at that planning iteration. This utility is derived using a *marginal contribution function* as described in [48] as follows:

$$F^t_m = r(p_m) = \mathcal{R}(p^t_m, p^t_{(m)}) - \mathcal{R}(p^t_{(m)}), \tag{3}$$

where $\mathcal{R}$ is the global objective function as defined in Eq. (1) and $\mathcal{R}(p^t_{(m)})$ is the joint utility of every agent except agent m (Line 8). Under this utility scheme, the actions of one agent can influence the contribution of others, which in turn affects their decisions. Such information is updated for every node selected by the *SW-UCT policy* for calculating the SW-UCT scores in the following iterations (Line 9).

After that, each agent elaborates $\hat{P}_m$ and q_m by accounting for p^t_m using a decentralized gradient descent algorithm [8], and it shares them with the other agents (Line 10). Finally, it updates the node statistics and starts a new planning iteration. When the planning time budget is met, the algorithm returns the path p_m that has the highest probability $q_m(p_m)$ to be executed (Line 13).

4.2 Sliding Window UCT Algorithm

In this section, we present our algorithm for computing a score for each of the actions available at a node, when the node has been fully explored. To incorporate the time-varying nature of reward distribution in nonstationary settings, a sliding window strategy is used to force the algorithm to "forget" outdated previous samples. Thus, the algorithm is parameterized by a sliding window constant $\tau \geq 1$ which tunes the weight that the results of past explorations should have in computing payoffs.

The algorithm is based on computing, at planning iteration t and node i, an *upper confidence bound* $U_{j,t}$ on the value of each child $j \in C(i)$ of the given node i. That is an upper bound on the potential payoff of choosing that child node as the next hop. The algorithm then selects the child node that maximizes this quantity over all children of the given node. We denote this optimal node by $I_{i,t}$, i.e.

$$I_{i,t} = \underset{j \in C(i)}{\arg\max}\ U_{j,t}.$$

The upper bound $U_{j,t}$ is derived as a combination of the empirical mean of rewards received at node j and a confidence interval derived from the Chernoff-Hoeffding inequality [34]. Specifically, at each planning iteration t, the UCB score $U_{j,t}$ for the children $j \in C(i)$ of the parent node i is calculated is given by

$$U_{j,t} = X_{j,t}(\tau) + H_{j,t}(\tau),$$

where $X_{j,t}(\tau)$ is the *average empirical reward* for choosing node j, given by

$$X_{j,t}(\tau) = \frac{1}{N_t(\tau, j)} \sum_{u=t-\tau+1}^{t} F^j_u\, \mathbb{I}_{\{I_{i,u}, j\}}, \tag{4}$$

where $\mathbb{I}_{\{I_{i,u}, j\}}$ denotes the indicator function, equal to one if the child node of i selected at iteration u is j, and $N_t(\tau, j)$ is the number of times the child node j within the last τ iterations has been visited, given by

$$N_t(\tau, j) = \sum_{u=t-\tau+1}^{t} \mathbb{I}_{\{I_{i,u}, j\}}.$$

The average empirical reward accounts for the results of the past explorations, and it thus represents the *exploitation* component of $U_{j,t}$, as it tends to favor the child with the best score in the recent past. $H_{j,t}(\tau)$ is instead the *exploration bonus* in a window of τ past iterations from t for node j:

$$H_{j,t}(\tau) = C_p \sqrt{\frac{2 \log N_t(\tau, i)}{N_t(\tau, j)}}, \tag{5}$$

where $N_t(\tau, i)$ is the number of times the parent node node i within the last τ iterations has been visited. $C_p > 0$ is an *exploration constant*, and it is used to tune the relative weight that the exploration bonus has with respect to the average empirical reward. As it can be seen, $H_{j,t}(\tau)$ is larger for child nodes that have been visited less in the past, and it thus pushes the algorithm towards exploring new path choices.

Critically, in the classical UCT algorithm [31], every sampled reward is considered when estimating the score of an action. For abruptly changing environments, the distributions of rewards change at unknown time instants, thus making the UCT scores slow or even unable to converge. Our work provides for the first time a sliding-window-based UCT algorithm for decentralized settings, in which exploration and exploitation terms are functions of the sliding window constant τ that models the "memory" of the system to limit outdated and irrelevant samples from before the abrupt changes. This allows the algorithm to estimate the UCT scores better and converge faster.

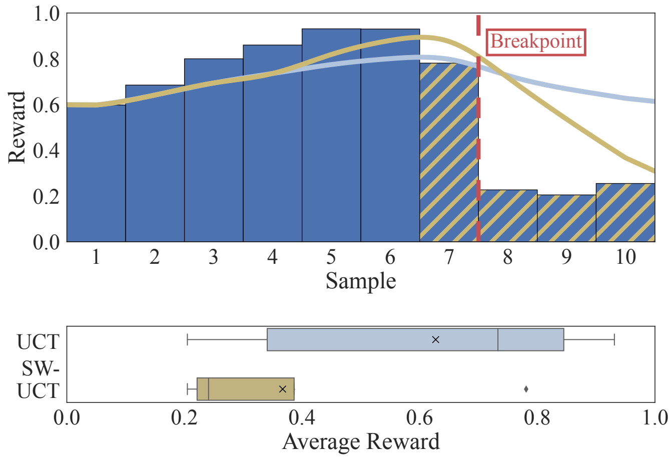

Fig. 2: Example of SW-UCT mechanism with a window size $\tau = 4$, and an abrupt change in reward value from sample 8th. SW-UCT (in *yellow*) only considers the most recent observations and has a better estimation of the new UCT score. In contrast, classical UCT (in *blue*) converges slower as it considers all past samples.

4.3 Online Replanning

SW-MCTS is designed as an online, and anytime algorithm. Given enough planning time budget T, the algorithm can return a solution that is arbitrarily close to optimal, that is the action sequence (i.e., the travel path p_m) that has the highest probability in the subset $\hat{P}_m$. Agent m then executes p_m, observes the changes in the environment (such as the sensor's new location and availability), updates such information with its teammates, and replans. Planning might require resetting the search tree if changes are significant, or else adapting the previous tree.

Remark: Our goal is to maximize the global utility, which is a function of the joint action sequence and is known by all agents. Specifically, we assume agents know the exact position and operating status of the sensors to calculate the utility associated with each action. However, in cases where agents do not have full knowledge of the reward distribution, our algorithm can be readily extended to optimize based on the agent's estimate of the environment, with the estimations being refined as agents traverse the graph and replan such as in [45].

5 Theoretical Bound on SW-MCTS performance

We provide in this section an upper bound on the regret of SW-MCTS, where the regret is generally defined as the difference between the actual payoff at the root node and the optimal payoff for the root node. In our particular application, the regret is the difference between the reward that an AV agent obtains by using SW-MCTS versus the optimal reward. SW-MCTS aims to minimize this regret.

Recall that the child node selection problem at each node in the tree is similar to the bandit problem [31], with the key difference that the payoff received on selecting an arm changes as we explore the search tree. The changes come from three sources:

- The reward for each internal node drifts as we discover more descendant nodes;
- The reward for each node in the MCTS tree of each agent changes as a result of other agents' actions; and
- The rewards for the nodes change as the environment changes. This last point has not been considered elsewhere in the literature and is our main theoretical contribution.

Our analysis of SW-MCTS therefore needs to take into account these three sources of changes and provide a thorough treatment of their combined impact. To handle the inter-dependencies between the sources of changes, we break the proof into three steps: (1) SW-UCB: Bounds on the regret when applying sliding windows to bandits with drifting changes; (2) Bounds on sliding window when applying to UCT as extensions of results in step 1; and (3) Bounds on SW-MCTS when there are multiple agents using results from step 2. We start with known results for SW-UCB when applying to the bandit problem [33]. From these results, we analyze the convergence of the actual to the optimal payoff sequence at the root node after some transitory periods.

5.1 Upper-bound on SW-UCB

A few notations are needed for the analysis of step 1, SW-UCB. Let $I_t \in \{1, \dots, K\}$ denote the arm pulled at round t, with K being the number of possible arms. After selecting the arm $I_t = i$, we receive a stochastic payoff $X_{i,t} \in [0,1]$. The sequence of payoffs generate the stochastic process $\{X_{i,t}\}_t$, $i = 1, \dots, K$ for $t \geq 1$. Let $\mu_{i,t}$ be the expected reward for arm i at time t.

The SW-UCB arm selection policy chooses the arm with the best UCB within a sliding window τ as

$$I_t = \underset{i \in \{1,\dots,K\}}{\arg\max} \{\bar{X}_{i,t}(\tau) + H_{i,t}(\tau)\},$$

where the empirical reward $\bar{X}_{i,t}(\tau)$ is given by (4) and the exploration bonus $H_{i,t}(\tau)$ is given by (5).

We make the following four key assumptions about the rewards.

Assumption 1. (Independence) Fix $1 \leq i \leq K$. Let $\{\mathcal{F}_{i,t}\}_t$ be a filtration such that $\{X_{i,t}\}_t$ is $\{\mathcal{F}_{i,t}\}_t$-adapted and $X_{i,t}$ is conditionally independent of $\mathcal{F}_{i,t+1}, \mathcal{F}_{i,t+2}, \dots$ given $\mathcal{F}_{i,t-1}$. Further, there exists an integer T_p such that for $t_i \geq T_p$ and $t < t_i$, $X_{i,t}$ is independent from $\mathcal{F}_{i,t}$.

Let Υ_t denote the number of breakpoints before time t, where a breakpoint is defined as the time instant where distributions of the rewards change.

Assumption 2. (Finite Number of Changes) The sequence $\{\Upsilon_t\}_t$ is known and bounded such that $\lim_{t\to\infty} \Upsilon_t = \sup_t \Upsilon_t < \infty$ and $\Upsilon_{t+1} \geq \Upsilon_t$.

We also assume that the expected payoff $\mu_{i,t}$ converges.

Assumption 3. (Convergence of means) The limit $\mu_i = \lim_{t\to\infty} \mu_{i,t}$ exists for all $i \in \{1, \dots, K\}$.

Let the difference between the two quantities be $\delta_{i,t} = \mu_{i,t} - \mu_i$. For any arbitrary time t, denote the optimal arm as t_{i^*}, and define the optimal expected payoff by $\mu_{i_t^*,t} = \max_{i\in\{1,\dots,K\}\mu_{i,t}}$. The average expected payoff up to time t is

$$\mu_t^* = \frac{1}{t}\sum_{u=1}^{t} \mu_{i_u^*,u}.$$

The minimum difference between the optimal reward and the instantaneous reward up to time t is defined as

$$\Delta_{i,t} = \min_{u\in\{1,\dots,t\}} \{\mu_{i_u^*,u} - \mu_{i_u,u} : i \neq i_u^*\}.$$

Let $M_i(t)$ be the number of times arm i is pulled following the most recent breakpoint. The following assumption requires that the drift $\delta_{i,t}$ is proportional to $\Delta_{i,t}$ after a finite burn-in period.

Assumption 4. (Small drifts) There exists an index $T_0(\epsilon)$ such that for any arbitrary $\epsilon > 0$ and $M_i(t) \geq T_0(\epsilon), |\delta_{i,t}| \leq \epsilon\Delta_{i,t}/2$ and $|\delta^*| \leq \epsilon\Delta_{i,t}/2$ for all i.

Given these assumptions, we first bound the number of times each sub-optimal arm is pulled. Let $\tilde{N}_i(t) = \sum_{u=1}^{t} \mathbb{I}_{\{I_u = i \neq i_u^*\}}$ be the number of times an arm i was played when it was not the best arm in the first t rounds. The following lemma gives a bound on $\tilde{N}_i(t)$.

Lemma 1. Consider the SW-UCB applied to a non-stationary, switching bandit problem where Assumptions 1, 2, 3, and 4 hold and let $\tau = \left\lceil \sqrt{16t\log(t)/\mathbb{E}[\Upsilon_t]} \right\rceil$. Then for any arm $i \in \{1, \dots, K\}$ and $t > 1$,

$$\mathbb{E}_\tau\left[\tilde{N}_i(t)\right] \leq O\left(\sqrt{\mathbb{E}[\Upsilon_t] t \log(t)}\, (C_p^2 + T_0(\epsilon) + T_p)\right). \quad (6)$$

This lemma is the cornerstone of all the theoretical analyses in this work and we present the proof in Appendix A.

Remark: Compared to the bounds for the Dec-MCTS in [8] where the number of suboptimal pulls is bounded by $O\left(\sqrt{\mathbb{E}[\Upsilon_t]t}(C_p^2\log(t)+T_0(\epsilon)+T_p)\right)$, our SW-MCTS performs better in terms of time steps by a factor of $\sqrt{\log t}$.

In the typical multi-arm bandit problem with fixed expected pay-offs, the bound on the number of suboptimal pulls leads directly to the bound on the expected regret. In UCT, as the expected payoff drifts over time, we need to prove that the expected payoff converges to the optimal payoff. The following lemma gives such guarantee.

Lemma 2. Let

$$\bar{X}_t = \sum_{i=1}^{K} \frac{N_i(t)}{t}\bar{X}_{i,t}$$

Under Assumptions 1-4,

$$\mathbb{E}_\tau\left[\bar{X}_t-\mu^*\right] \le |\delta_t^*| + O\left(K\sqrt{\frac{\mathbb{E}[\Upsilon_t]\log(t)}{t}}\left(C_p^2+T_0(\epsilon)+T_p\right)\right) \tag{7}$$

We still need to provide a bound on the concentration of the actual payoff $\bar{X}_t$ about the expected payoff. The following lemma provides that bound. The bound relies on a non-trivial assumption related to the number of suboptimal pulls. Let Z_t denote the indicator variable that a suboptimal arm was pulled at time t. From Assumption 1, for $t > T_p$, the indicator Z_t is independent of $Z_{t+1}, Z_{t+2}, \ldots,$ given $Z_1, \ldots, Z_{t-1}$. Thus, after T_p and T_0, the non-stationary bandit problem becomes equivalent to a stationary problem with high probability.

Lemma 3. For an arbitrary $0 < \epsilon \le 1$ and let

$$\Gamma_t = 9\,\mathbb{E}[\Upsilon_t]t\sqrt{2\ln(2/\epsilon)}$$

Then, under the assumptions of Lemma 1, for

$$t \ge O\left(\sqrt{\mathbb{E}[\Upsilon_t]t\log(t)}(C_p^2+T_0(\epsilon)+T_p)\right)$$

The following bound holds:

$$\mathbb{P}(t|\bar{X}_t-\mathbb{E}_\tau[\bar{X}_t]| \ge \Gamma_t) \le \epsilon.$$

Finally, we are interested in an upper bound on the failure probability of the algorithm. The proof relies on the assumption that the breakpoint sequence is known as monotone and bounded, thus making SW-UCB equivalent as UCB1 as t grows large.

Lemma 4. Under the assumptions of Lemma 1, it holds that

$$\lim_{t\to\infty}\mathbb{P}_\tau(I_t \ne i_t^* = i^*) = 0$$

The proofs for Lemma 2, 3, and 4 follow closely the steps in the proofs of Theorem 3 and Theorem 5 in [31] and Lemma 4 in [8] with adjustments for the sliding window, and are presented in Appendix B, C, and D respectively.

5.2 Upper-bound on SW-MCTS

We are now in the position to prove the main theorem for SW-MCTS. The previous three lemmas are for SW-UCB. We need to extend these results to a tree, where the node selection problem at each node in the tree is equivalent to the bandit problem, however with different assumptions on the payoff. For node i_d, after selecting node $I_{i_d,t} = j$, the tree search further down the tree and subsequent MCTS rollout yield a stochastic payoff $F_{j,t} = F_t \in [0,1]$ that is adapted to $\mathcal{F}_{j,t}$ as in Assumption 1. As nodes are slowly expanded in the search tree, the expected reward at any node higher up the tree slowly drifts until all nodes are explored in the subtree (Assumption 3).

The sequence of payoffs generates the stochastic process $\{F_{j,t}\}, \forall j \in C(I_d)$ and $t \ge 1$. Recall that $\bar{F}_{i_d,t_{i_d}}$ is the empirical mean and that $\bar{F}_{i_0,t_{i_0}}$ is the empirical mean at the root node. Let $\mu_{i_0}^*$ denote the optimal expected payoff at the root node and note that $t_{i_0} = t$.

Theorem 1. Consider algorithm SW-MCTS running on a tree of depth D and branching factor K. The payoff distributions of the leaf nodes are independently distributed and they can change at breakpoints. The sequence that gives the expected bound of breakpoints $E[\Upsilon_t]$ follows Assumption 2, and let $\tau = \left\lceil\sqrt{16t\log(t)/\,\mathbb{E}[\Upsilon_t]}\right\rceil$. Then, the regret of the payoff at the root node is given by

$$|\bar{F}_{i_0,t_{i_0}} - \mu_{i_0}^*| = O\left(KD\sqrt{\frac{E[\Upsilon_t]log(t)}{t}}\right) \tag{8}$$

Further, the probability of failure at the root node becomes zero as t grows large.

The proof of Theorem 1 is presented in Appendix E. In addition, we make the following observations that are important for the practical applications of the algorithm.

- Smooth rewards: it was observed in [49] that UCT can have exponential transitory periods and performs best when the rewards on different branches of the MCTS tree are similar (smooth). The smooth rewards are assumed in Assumption 4. In practice, if the assumption does not hold, the algorithm can take exponential time to converge.
- Optimal window size τ: from Theorem 1, given a known and fixed $\mathbb{E}_\tau[\Upsilon_t]$, the optimal value for τ can be determined precisely.

6 Numerical Assessment

6.1 Experimental Setup

In this section, we evaluate the performance of our SW-MCTS algorithm as a function of the main system parameters. To this end, we consider a reference scenario modeling an underwater wireless sensor network (UWSN), in which 200 sensor nodes are located at the bottom surface of a volume of water (sensor plane) of size 4000 m × 4000 m and at a depth of 200 m. Such choices have been made to have sensor nodes enough spaced among them for multi-hop data relaying not to be feasible. At the same time, the dimensions of the volume of water are compatible with the mean autonomy of available commercial AUVs [50]–[52]. Each node is connected through a cable of length equal for all nodes to a stationary anchor (see Fig. 3). Anchors

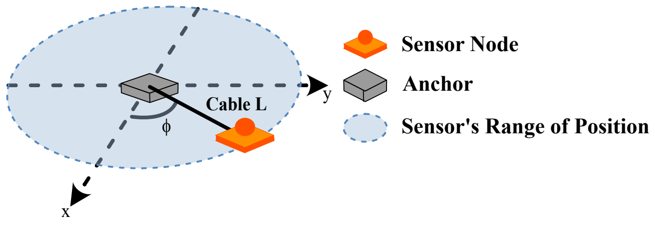

Fig. 3: Configuration of an underwater sensor node in the considered setup.

are distributed uniformly at random on the bottom surface of the given volume. This type of model is referred to as active-restricted underwater WSN [53], [54], and it allows modeling sensor drifts due to storms or to changes in water currents, e.g. due to ocean tides. With reference to Fig. 3, we assume that for each node, the angle ϕ is uniformly distributed $(0, \pi)$, while the distance of the node from the anchor varies between 0 and the cable length. The cable length has been set to 50 m.

We assume that each sensor node has a transmission radius R of 50 m and a transmission rate d of 5 kb/s, and the size of each data packet is 1 kb (i.e., typical for UWSN, e.g. [55]–[58]). Since the data is to be transmitted in a short distance, the water flow does not affect the transmission rate and quality. We assume sensors can send their data to an agent once it enters a sensor's transmission range. We denote the minimum time required for each sensor to transmit a data packet as $\tau_{trans} = S/d$. We assume every agent moves at a constant speed v of 5 m/s. Let $l_{(m,s)}$ be the length of the path segment of agent m that crosses the transmission range of sensor s. We denote the traversal time of agent m on this path segment as $\tau_{trave} = l_{(m,s)}/v$. We assume the average length of $l_{(m,s)}, \forall m, s$ equals the average chord length of a circle whose radius is the sensor's transmission radius. We assume the time to traverse this length, denoted as $\bar{\tau}_{trave} = 4Rv/\pi$, as the average time for an agent to traverse any sensor's transmission range. Fig. 4 shows different scenarios where an agent can visit and harvest data from a sensor. The harvest succeeds if the data transmission time τ_{trans} is less than or equal to the cross-over traversal time τ_{trave}.

We adopt a D2D all-to-all communication network in which each AUV can directly communicate with all other AUVs, e.g., using an acoustic modem whose effective range is several thousand meters [41]. With each communication attempt, we assume an agent only tries to transmit its message exactly once. For the sake of simplicity and to better examine the fundamentals of our proposed algorithm, we assume that the communication link remains robust and unaffected by changes in both spatial and temporal domains. We later relax this assumption and investigate the impact of nonidealities in information sharing on the effectiveness of our approach.

We assume that the motion graph lies on the same 2-dimensional plane as the sensor plane. The motion graph has been built using a probabilistic roadmap (PRM) with a Dubins path model [59]. This model uses curves for smoothing straight lines between waypoints and it has been widely used to model motion constraints of vehicle-like nonholonomic robots such as AUVs [60]. These constraints imply that each AUV agent can only travel along smooth curvatures without reversing direction and that its trajectory must satisfy geometric continuity. The resulting graph has 400 vertices and an average of 15500 edges. We assume that a single data gathering point and a single sink are present in the scenario. We assume the transporting agent only moves vertically between the gathering point and the data sink to transmit data. This configuration has proven to improve the energy efficiency of the overall data harvesting process [61]. The rewards for data harvesting from sensor nodes have been set to be the same for all nodes.

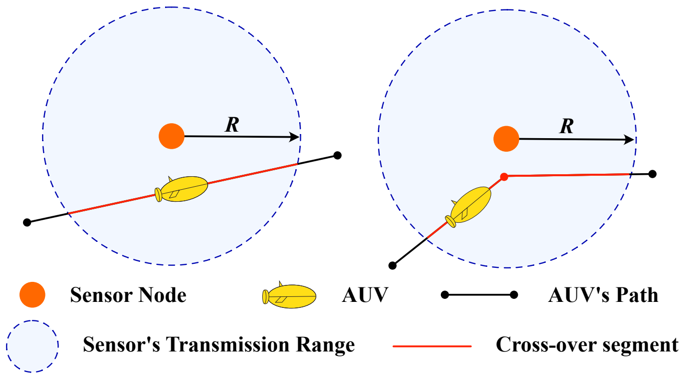

Fig. 4: Different examples of AUV harvest data from sensors.

To perform an accurate evaluation of the performance of our SW-MCTS scheme, we considered the following baseline algorithms:

- *Centralized MCTS (Central-MCTS)*: In this scheme, all agents are assumed to communicate directly with a central server, on which a single search tree is built for all of the M harvesting agents. Note that in such a tree the actions of agent m are at tree depth $(m, m+M, m+2M, ...)$.
- *Dec-MCTS* [8]: It is the state-of-the-art decentralized multi-agent planning method for information gathering. In it, agents build their own search tree with a discounting factor and adapt it when replanning after a change in the environment.
- *Dec-MCTS with reset (Dec-MCTS-Res)*. In this scheme, like Dec-MCTS, each agent builds its own search tree with a discounting factor. Whenever there are changes in the environment, the tree of each agent is reset [8].

Table 2 compares the key features of the considered algorithms, i.e., the forgetting mechanisms, and whether they handle varying environments via online replanning and/or tree reset.

TABLE 2: Main features of the considered algorithms.

Algorithm	Forgetting Mechanism	Online Replanning	Reset Tree
Central-MCTS	None	no	no
Dec-MCTS	Discounting factor	yes	no
Dec-MCTS-Res	Discounting factor	yes	yes
SW-MCTS	Fixed window	yes	no

In all these algorithms, the duration of the planning phase has been set to 500 iterations. For SW-MCTS, Dec-MCTS, and Dec-MCTS-Res, each agent exchanges with all the other agents a set of the 10 best possible paths every 50 planning iterations. These values have been chosen as they are commonly used in the literature for this family of algorithms (e.g., as in [8]). Moreover, we have verified experimentally that increasing either the number of paths exchanged and/or the frequency of these exchanges has no measurable impact on performance. The exploration parameter C_p for all algorithms has been set to 0.4, i.e. within the recommended range [8]. In realistic scenarios, determining the appropriate number of agents and travel budgets must account for the size of the area under surveillance and the communication range of the sensor nodes being utilized. Unless otherwise stated, in our experiments, we assumed 10 harvesting agents are deployed, each with a travel budget of 10 edges, as this choice has proven sufficient to enable full coverage of all sensors in the given area. The default value of the window size τ of SW-MCTS has been set to 300 iterations, and the discounting factor of Dec-MCTS and Dec-MCTS-Res to 0.9, i.e. within the range of values recommended in [8] to ensure an optimal balance between exploration and exploitation, and thus an effective adaptation to changes in the environment. The key performance metric we have used in our assessment is the *Instantaneous Reward coverage (IRC)*, given by the total amount of harvesting reward collected by all agents, expressed as a percentage of the total amount of harvesting rewards available.

For what concerns the changes in the environment, in our experiments, we considered two models:

- The *stationary model*, in which sensor nodes do not change their position over time.
- The *correlated model*, in which the location of all sensors changes one or more times during a mission. As for the location of sensor nodes after a change event, we adopted the underwater sensor mobility model from [62]. Specifically, after a change event, we assume that each node independently takes a new position satisfying the properties of the active-restricted underwater WSN layouts described above. We assume that there is no correlation between the values that the angles and the distance from the anchor take before and after a change. By varying the cable length, such a mobility model allows tuning the mean distance between the old and the new location of each sensor, and thus the degree to which the change event may actually bring a change in the trajectory choices made by each agent. At the same time, the assumption of independence (both across nodes, and between a node position before and after a change) aims at a worst-case assessment of the impact of changes in sensor location on the performance of the considered schemes.
- The *dynamic model*, in which a fraction of nodes change location at random within the sensor plane one or more times during a mission. We assume during each change event, each node has an equal probability $1/|\mathcal{S}|$ to change. This is interesting as a worst-case assessment of the ability of our algorithms to converge and achieve high IRC values.

For both *correlated* and *dynamic* models, we assume that the change events are distributed uniformly at random over the duration of the mission. In addition, we also assume that the sensor plane is wraparound to avoid border effects (i.e., sensors exit the sensor plane via one edge of the area would re-enter via the opposite edge).

6.2 Evaluation Results

6.2.1 Performance Benchmarking

To perform a first evaluation of our algorithm, we considered the stationary model, in which the location of each sensor does not vary over time. Fig. 5a shows the IRC at the end of the mission of our algorithm as well as the baseline for different planning times. It can be seen that SW-MCTS sustainably outperformed other methods and typically converged after 500 iterations. Fig. 5d shows the IRC of each algorithm throughout the mission for a default planning time of 500 iterations. While all algorithms perform similarly at the early stage of a mission, all decentralized algorithms quickly outperform the centralized counterpart, with our scheme achieving an average IRC that is 10% and 35% larger than that of Dec-MCTS and Central-MCTS, respectively. The centralized MCTS performs well initially because its rollout policy maximizes the global rewards by optimizing the action selection of each agent sequentially. However, because the joint action space grows exponentially with the number of agents, the central algorithm would require more iterations for efficient exploration. On the other hand, the decentralized methods take advantage of distributed and parallel computing while letting each agent grow its search tree over its respected action space. Thus, given the same number of planning iterations, the decentralized methods can reach deeper levels of their search trees and outperform the central one. Additionally, agents can discover more potential high-reward areas by performing online replanning after every action. Note that, as there are no changes in the environment, the tree is never reset, and the performance of Dec-MCTS-Res coincides with that of Dec-MCTS. These results suggest that SW-MCTS, although not designed for static settings, can outperform Dec-MCTS (i.e., the best available algorithm for AUVs coordination in such settings) even when there are no changes in the environment, thanks to its substantially faster convergence rate.

To evaluate the impact of changes in the environment, we performed a second set of tests in non-stationary scenarios, in which the location of all sensors changes according to the correlated and dynamic models respectively. Specifically, we observed the performances of our algorithm in the correlated model with a default cable length of 50 m, and in the dynamic model with all sensors changing their location, and for a number of changes per mission equal to 10. Fig. 5b and c show the IRC of SW-MCTS and other baseline algorithms at the end of the mission with different planning times, while Fig. 5e and f show the evolution of IRC with a planning time of 500 iterations of each algorithm over the mission for the correlated and dynamic model respectively. Similar to the stationary model, the result demonstrated that agents running SW-MCTS required approximately the same planning time to produce a sufficient output. This is thanks to they actively performed online replanning based on newly available information during the mission. This is

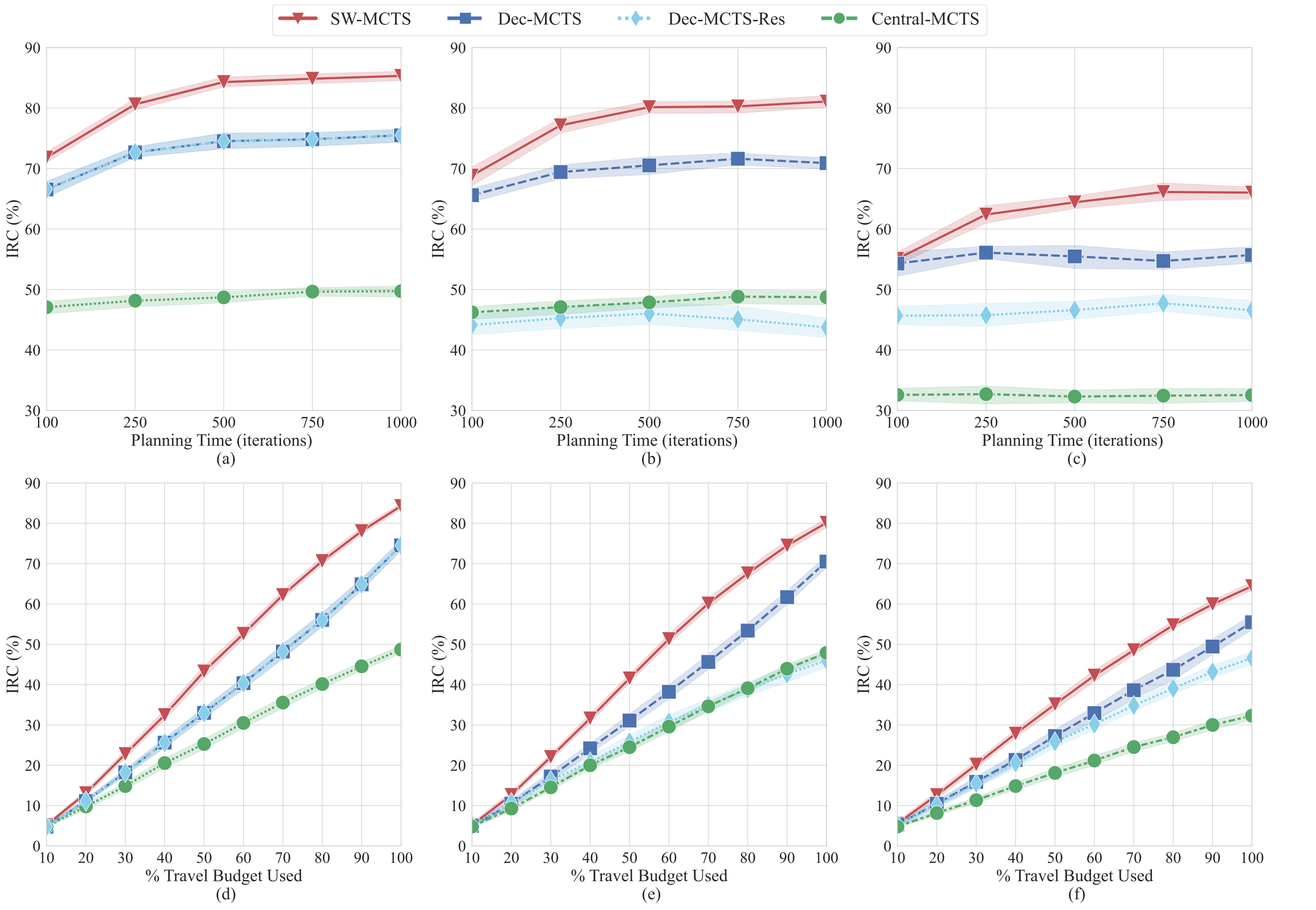

Fig. 5: Impact of planning time (top) on the algorithms' performance at the end of the mission, and evolution of the Instantaneous Reward Coverage (IRC) over the mission duration (bottom) for different environment models: *Stationary* (left); *Correlated* (middle); and *Dynamic* (right). The shaded areas denote the 95% confidence intervals.

ideal for applications such as underwater data collection, in which the state-action space can be extremely large and varies over time.

Results also suggested that the Dec-MCTS version that resets the search tree for replanning produced no benefits compared to those that adapted the same tree and even performed worse in the later stage of the mission. This is because at the start of the planning phase, MCTS agents need to spend some iterations to explore the environment, and incrementally add new nodes to the search tree. In this process, agents intentionally take random actions in order to improve their knowledge of the reward distribution. Thus, resetting the tree frequently causes the produced joint policy to be sub-optimal, as agents are stuck in such *transient* exploration phase. This is particularly sub-optimal if the change is minor, such as in the case of the correlated model as can be seen in Fig. 5e. These results suggest that when dealing with abrupt changes in the environment, the decision to reset the tree is beneficial the most given that the changes are significant and/or occur less frequently. Nevertheless, these factors are impractical to determine beforehand. Regardless of such difficulties, SW-MCTS performance is the best across all considered baselines. Specifically, our algorithm achieved a substantially better IRC than Dec-MCTS, as the sliding window of SW-MCTS allows agents to remove irrelevant samples faster than in the discounting factor technique.

6.2.2 Impact of Nonidealities in Communication

A key factor affecting the performance of multi-agent schemes for data collection is the quality of communications, particularly those among agents. Naturally, underwater communications can be very challenging due to the difficulties of channel modeling, limited bandwidth, and signal attenuation. To understand better the impact of such limitations, we evaluated the algorithms' performances under different successful communication probabilities. Specifically, we randomly dropped for each agent a percentage of its sent messages during a mission. As shown in Fig. 6, when agents cannot communicate at all, SW-MCTS performs the same as Central-MCTS in the stationary and correlated models. However, a success rate of 5% only could yield more than 30% improvement over the centralized algorithm. Furthermore, when the communication is severely unstable with 90% of the communication lost, there is still

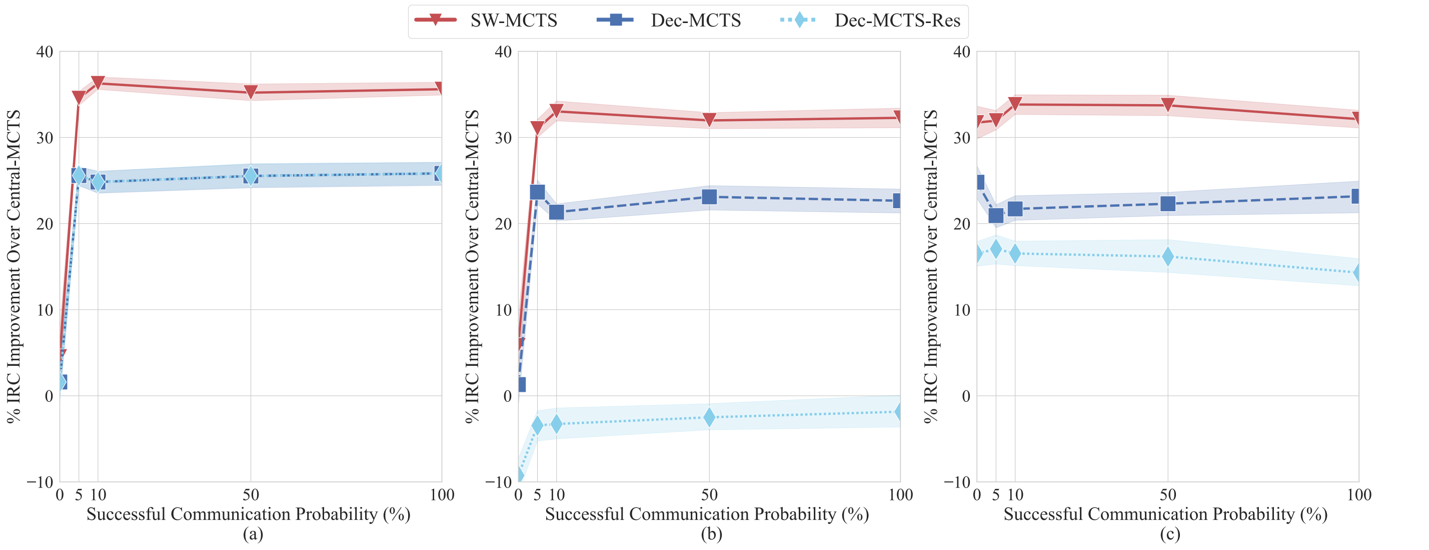

Fig. 6: Impact of nonidealities in communication on the algorithms' performance at the end of the mission for different environment models: *Stationary* (left); *Correlated* (middle); and *Dynamic* (right). The shaded areas denote the 95% confidence intervals.

no significant degradation in the average mission's reward coverage. These results show that under our proposed SW-MCTS paradigm, agents can cooperate efficiently in hostile environments even with restricted communication.

6.2.3 Impact of Parameter Settings

In this section, we investigate the impact of the key parameters of the system on the performance of our solution, with a special focus on its scalability and robustness. Specifically, we evaluate the impact of the travel budget, the number of AUVs, the ratio between the data transmission time τ_{trans} and the average traversal time $\bar{\tau}_{trave}$, and the number of sensor nodes. Fig. 7 illustrates the impact of these parameters on the instantaneous reward coverage at the end of the mission, for a default number of changes of 10, a default cable length of 50 m in the correlated setting, and all sensors changing in the dynamic setting.

As Fig. 7a-c shows, the improvement of our approach with respect to the discounted method also grows from 5% to 10% with the increasing travel budgets. Indeed, with larger travel budgets the depth level of the search tree reached at each planning iteration increases too, and so do the gains of a better planning strategy, such as the one of SW-MCTS. A similar behavior is exhibited by the system when we vary the number of AUVs as shown in Fig. 7d-f. However, when trying to achieve large values of coverage with the use of a larger number of AUVs, the performance advantage of a smarter planner is less evident. Indeed, achieving very high values of coverage requires reaching nodes associated with low reward, e.g. because of being far away from the bulk of the other sensors.

Another key factor affecting the performance of the data harvesting scheme is the ratio between the data transmission time τ_{trans} and the average traversal time $\bar{\tau}_{trave}$. Naturally, if the data transmission time is insignificant compared to the traversal time, it would afford agents greater flexibility in selecting edges to traverse, thereby potentially harvesting more sensors. Indeed, results from Fig. 7g-i show that as the $\tau_{trans}/\bar{\tau}_{trave}$ ratio increases, the performances of all algorithms decrease. Regardless, our SW-MCTS's improvement with respect to Dec-MCTS grows as we raise the ratio from 0 to 1.4, especially in the *Correlated* and *Dynamic* models.

In the three considered settings, we also assess the impact of the density of sensors on the algorithms' performance. Specifically, within the same area, we vary the number of sensor nodes and observe the instantaneous reward coverage at the end of the mission of each algorithm. As shown in Fig. 7j-l, the percentage of sensors covered declines as more sensors are introduced in the system. Naturally, with an increasing number of sensors the area that must be covered by agents expands as well. Nevertheless, SW-MCTS still managed to consistently outperform other baseline methods despite the difficulty of decentralized planning with a growing number of sensors. These outcomes thus demonstrate our proposed algorithm as a scalable and robust distributed planner for multi-agent systems.

6.2.4 Impact of the abruptness of change

In this section, we parameterized the abruptness of changes in the environment, including its intensity and frequency, and assessed the performance of SW-MCTS as a function of it. Specifically, we considered the two non-stationary scenarios with the frequency represented by the number of changes, and the intensity represented by the cable length L in the correlated model, by the number of changed sensors in the dynamic model. Fig. 8 shows the results in terms of instantaneous reward coverage at the end of the mission.

Among the factors affecting the intensity of changes in the correlated model, a key role is played by the cable length L connecting the sensor with its anchor. Potentially, a longer cable length implies a larger area in which the sensor's new location can be. Similarly, in the dynamic model, the intensity can be measured by the amount of changed sensors in each change event. As a key step towards studying such

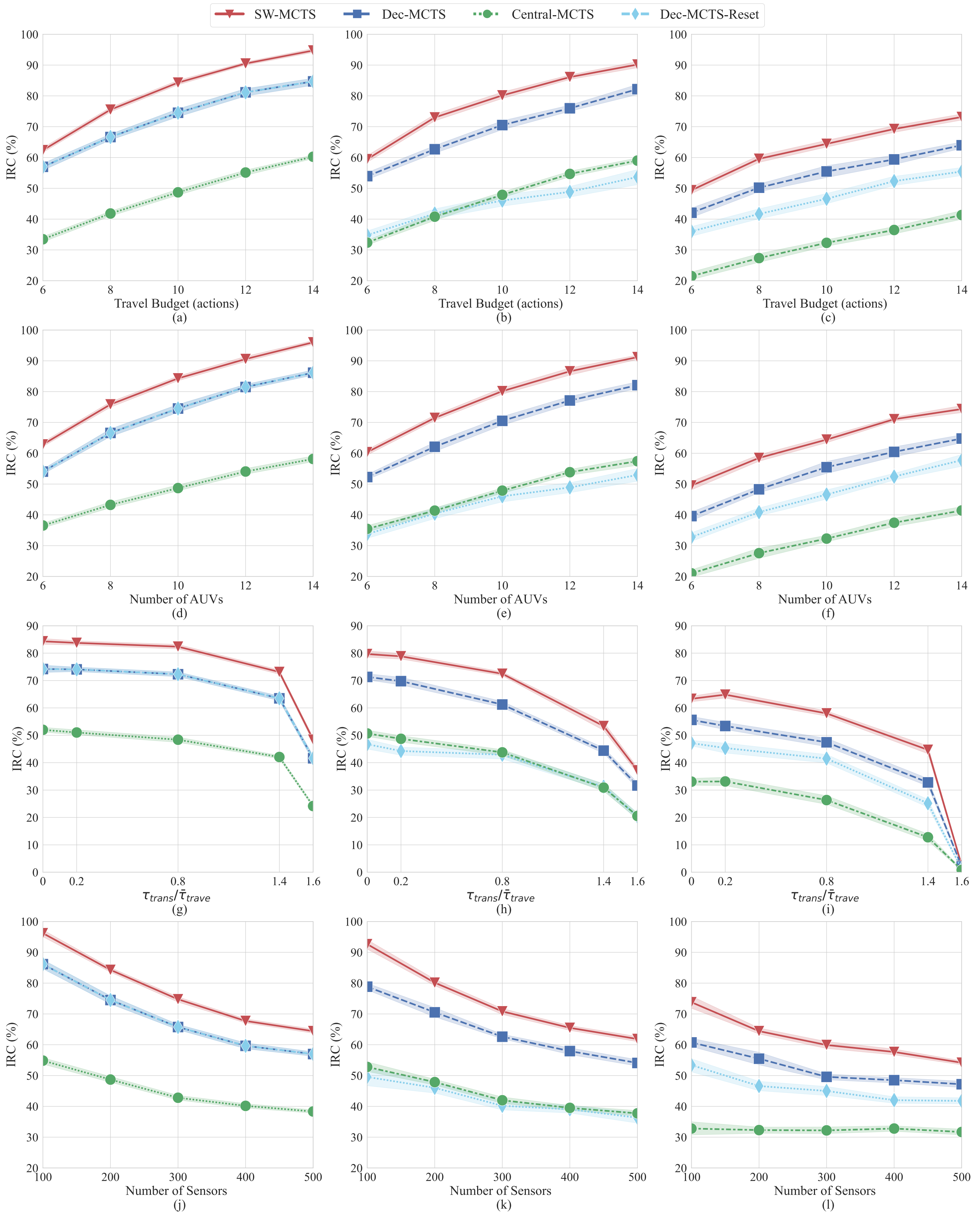

Fig. 7: Impact of travel budget (top row), number of AUVs (second row), ratio between τ_{trans} and $\bar{\tau}_{trave}$ (third row), and number of sensors (bottom row) on the algorithms' performance at the end of the mission for different environment models: *Stationary* (left); *Correlated* (middle); and *Dynamic* (right). The shaded areas denote the 95% confidence intervals.

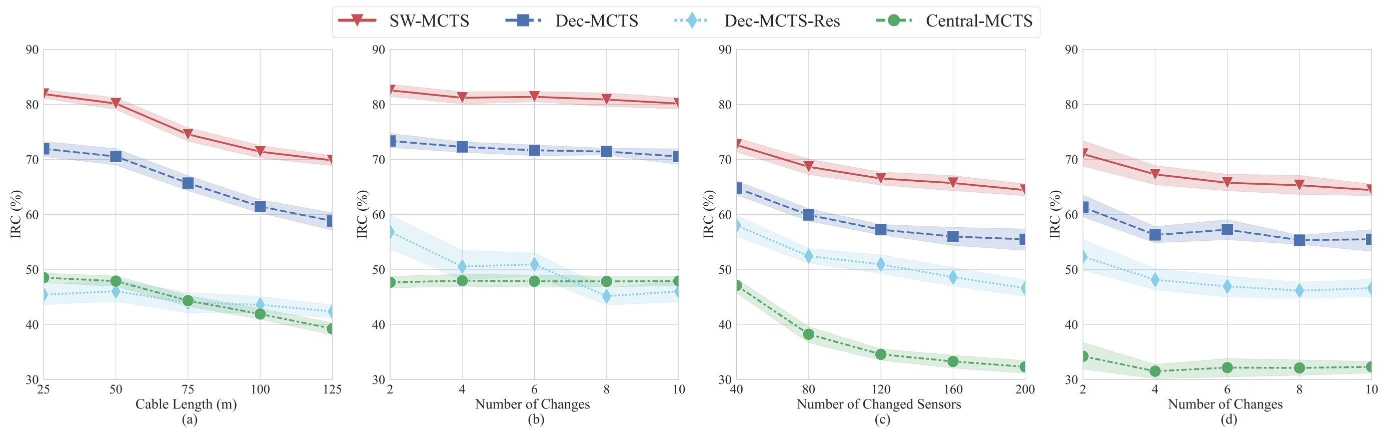

Fig. 8: Impact of changes intensity and number of changes on the algorithms' performance at the end of the mission for different environment models: *Correlated* (a-b) and *Dynamic* (c-d). The shaded areas denote the 95% confidence intervals.

impact, in Fig. 8a and c we observed how the average IRC at the end of the mission varied with different cable lengths and different numbers of changed sensors respectively, for a default number of changes of 10. As expected, both SW-MCTS and Dec-MCTS drop in performance as the intensity increases. Indeed, the further sensors drift away or the more sensors drift, the more severe the rewards associated with each edge change, thus potentially making a current optimal branch sub-optimal.

To assess the impact of the frequency of changes in the environment, we considered a default cable length L of 50 m in the correlated setting and a default for all sensors changed in the dynamic setting with different numbers of changes. A similar trend can also be observed from Fig. 6b and d as the reward coverage decreases when the number of changes increases. Indeed, a higher number of changes implies that less time is available for the algorithms to converge and adapt to the new configuration as well as its induced reward distribution. However, thanks to its faster convergence rate our SW-MCTS algorithms substantially outperform the discounted approach in both experiments, showing a better ability to adapt to changing environments regardless of the intensity and frequency of the abruptness.

7 Conclusions

Achieving efficient coordination in multi-robot planning for active perception is a hard open issue in practical settings, where available resources and environmental conditions vary over time, often in an abrupt and unpredictable fashion. In this work, we proposed a new decentralized Monte Carlo Tree search approach to tackle this issue that carefully balances the exploration-exploitation trade-off in a dynamic environment. Specifically, we proposed a new tree expansion policy based on a sliding window mechanism to appropriately weigh the contribution of past information on planning decisions. We prove that our algorithm provides a log factor (in terms of time steps) smaller regret than the state-of-the-art decentralized planning method. Numerical results also confirmed that our algorithm performs substantially better than the best approach available in the literature on a practical scenario of underwater data collection using multiple AUVs, achieving faster convergence and higher resource efficiency despite implementing a loose form of synchronization among agents, even in settings with frequent changes. As a follow-up, we intend to explore the impact of long transitory periods for SW-MCTS when rewards vary significantly, and of non-stationarity in the frequency and spatial distribution of changes. In addition, we plan to extend our approach to settings where the MCTS tree is imbalanced and the assumption of the smooth reward does not hold.

Another interesting avenue for future work is to incorporate other practical communication models into our approach. While an all-to-all communication model offers great benefits in coordinating the agents' actions, it is often limited in real-world applications. A possible consideration could be an opportunistic communication model such as [63] in which agents share information only as they are in range with one another. It could also be useful to integrate the value of communication into the path planning utility, as agents have to decide between collecting rewards or moving toward other agents to exchange information.

References

[1] G. Han, X. Long, C. Zhu, M. Guizani, Y. Bi, and W. Zhang, "An AUV Location Prediction-Based Data Collection Scheme For Underwater Wireless Sensor Networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 6, pp. 6037–6049, 2019.
[2] T. Cerquitelli, M. Meo, M. Curado, L. Skorin-Kapov, and E. E. Tsiropoulou, "Machine learning empowered computer networks," p. 109807, 2023.
[3] R. Ma, R. Wang, G. Liu, H.-H. Chen, and Z. Qin, "Uav-assisted data collection for ocean monitoring networks," *IEEE Network*, vol. 34, no. 6, pp. 250–258, 2020.
[4] F. Banaeizadeh and A. T. Haghighat, "An energy-efficient data gathering scheme in underwater wireless sensor networks using a mobile sink," *Int. J. Inf. Technol.*, vol. 12, no. 2, pp. 513–522, 2020.
[5] Z. Lv, L. Xiao, Y. Du, G. Niu, C. Xing, and W. Xu, "Multi-agent reinforcement learning based uav swarm communications against jamming," *IEEE Transactions on Wireless Communications*, 2023.
[6] M. Otte and N. Correll, "Any-com multi-robot path-planning: Maximizing collaboration for variable bandwidth," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 161–173.
[7] G. Best, J. Faigl, and R. Fitch, "Online planning for multi-robot active perception with self-organising maps," *Autonomous Robots*, vol. 42, no. 4, pp. 715–738, 2018.

[8] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 316–337, 2019.

[9] O. Gupta and N. Goyal, "The evolution of data gathering static and mobility models in underwater wireless sensor networks: A survey," *J. Ambient. Intell. Humaniz. Comput.*, pp. 1–17, 2021.

[10] X. Su, I. Ullah, X. Liu, and D. Choi, "A review of underwater localization techniques, algorithms, and challenges," *Journal Of Sensors*, vol. 2020, 2020.

[11] K. M. Awan, P. A. Shah, K. Iqbal, S. Gillani, W. Ahmad, and Y. Nam, "Underwater wireless sensor networks: A review of recent issues and challenges," *Wirel. Commun. Mob. Comput.*, vol. 2019, 2019.

[12] X. Yao, X. Wang, F. Wang, and L. Zhang, "Path following based on waypoints and real-time obstacle avoidance control of an autonomous underwater vehicle," *Sensors*, vol. 20, no. 3, p. 795, 2020.

[13] Z. Xu, R. Fitch, J. Underwood, and S. Sukkarieh, "Decentralized coordinated tracking with mixed discrete–continuous decisions," *Journal Of Field Robotics*, vol. 30, no. 5, pp. 717–740, 2013.

[14] S. K. Gan, R. Fitch, and S. Sukkarieh, "Online decentralized information gathering with spatial-temporal constraints," *Autonomous Robots*, vol. 37, no. 1, pp. 1–25, 2014.

[15] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[16] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings Of The IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[17] A. Sadeghi and S. L. Smith, "Heterogeneous task allocation and sequencing via decentralized large neighborhood search," *Unmanned Systems*, vol. 5, no. 02, pp. 79–95, 2017.

[18] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics Of Operations Research*, vol. 27, no. 4, pp. 819–840, 2002.

[19] F. A. Oliehoek and C. Amato, *A Concise Introduction To Decentralized Pomdps*. Springer, 2016.

[20] M. T. Spaan, G. J. Gordon, and N. Vlassis, "Decentralized planning under uncertainty for teams of communicating agents," in *AAMAS*, 2006, pp. 249–256.

[21] M. Lauri and F. Oliehoek, "Multi-agent active perception with prediction rewards," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 13 651–13 661.

[22] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2017–2023.

[23] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[24] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.

[25] H. Zhang, J. Chen, H. Fang, and L. Dou, "A role-based pomdps approach for decentralized implicit cooperation of multiple agents," in *2017 13th IEEE International Conference on Control & Automation (ICCA)*. IEEE, 2017, pp. 496–501.

[26] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2022.

[27] D. Claes, F. Oliehoek, H. Baier, K. Tuyls, and Others, "Decentralised online planning for multi-robot warehouse commissioning," in *AAMAS*, 2017, pp. 492–500.

[28] M. Li, W. Yang, Z. Cai, S. Yang, and J. Wang, "Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty." in *IJCAI*, 2019, pp. 450–456.

[29] A. Czechowski and F. A. Oliehoek, "Decentralized mcts via learned teammate models," in *IJCAI*, 2020, pp. 450–456.

[30] S. Choudhury, J. K. Gupta, P. Morales, and M. J. Kochenderfer, "Scalable anytime planning for multi-agent mdps," in *AAMAS*, 2021, pp. 341–349.

[31] L. Kocsis, C. SzepesvÁRi, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.

[32] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proc. 5th Int. Conf. Comput. Games*. Springer, 2006, pp. 72–83.

[33] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Algorithmic Learning Theory*. Springer, 2011, pp. 174–188.

[34] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The nonstochastic multiarmed bandit problem," *SIAM Journal On Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[35] H. Luo, C.-Y. Wei, A. Agarwal, and J. Langford, "Efficient contextual bandits in non-stationary worlds," in *Conf. On Learning Theory*. PMLR, 2018, pp. 1739–1776.

[36] W. C. Cheung, D. Simchi-Levi, and R. Zhu, "Learning to optimize under non-stationarity," in *22nd Internat. Conf. Artificial Intelligence Statist.* PMLR, 2019, pp. 1079–1087.

[37] Y. Cao, Z. Wen, B. Kveton, and Y. Xie, "Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 418–427.

[38] F. Liu, J. Lee, and N. Shroff, "A change-detection based framework for piecewise-stationary multi-armed bandit problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[39] Z. S. Karnin and O. Anava, "Multi-armed bandits: Competing with optimal sequences," *Proc. Adv. Neural Inform. Processing Systems*, vol. 29, pp. 199–207, 2016.

[40] J. Luo, Y. Yang, Z. Wang, and Y. Chen, "Localization algorithm for underwater sensor network: A review," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 126–13 144, 2021.

[41] R. Su, D. Zhang, C. Li, Z. Gong, R. Venkatesan, and F. Jiang, "Localization and data collection in auv-aided underwater sensor networks: Challenges and opportunities," *IEEE Network*, vol. 33, no. 6, pp. 86–93, 2019.

[42] M. Corah and N. Michael, "Efficient online multi-robot exploration via distributed sequential greedy assignment." in *Robotics: Science And Systems*, vol. 13, 2017.

[43] Y. Satsangi, S. Whiteson, F. A. Oliehoek, and M. T. Spaan, "Exploiting submodular value functions for scaling up active perception," *Autonomous Robots*, vol. 42, no. 2, pp. 209–233, 2018.

[44] M. Prajapat, M. Turchetta, M. Zeilinger, and A. Krause, "Near-optimal multi-agent learning for safe coverage control," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 998–15 012, 2022.

[45] G. Best and G. A. Hollinger, "Decentralised self-organising maps for multi-robot information gathering," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4790–4797.

[46] B. L. Nguyen, D. D. Nguyen, H. X. Nguyen, D. T. Ngo, and M. Wagner, "Multi-agent task assignment in vehicular edge computing: A regret-matching learning-based approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2023.

[47] C. Dornhege, A. Kleiner, A. Hertle, and A. Kolling, "Multirobot coverage search in three dimensions," *Journal of Field Robotics*, vol. 33, no. 4, pp. 537–558, 2016.

[48] D. H. Wolpert, S. R. Bieniawski, and D. G. Rajnarayan, "Probability collectives in optimization," *Handbook of Statistics*, vol. 31, pp. 61–99, 2013.

[49] P.-A. Coquelin and R. Munos, "Bandit algorithms for tree search," in *Proc. Conf. Uncert. Artif. Intell.*, ser. UAI'07. Arlington, Virginia, USA: AUAI Press, 2007, p. 67–74.

[50] J. Faigl and G. A. Hollinger, "Autonomous data collection using a self-organizing map," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1703–1715, 2017.

[51] X. Zhuo, M. Liu, Y. Wei, G. Yu, F. Qu, and R. Sun, "Auv-aided energy-efficient data collection in underwater acoustic sensor networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 010–10 022, 2020.

[52] J. Jiang, W. Tian, G. Han, and F. Zhang, "A medium access control protocol based on parity group-graph coloring for underwater auv-aided data collection," *IEEE Internet of Things Journal*, 2023.

[53] Y. Guo and Y. Liu, "Localization for anchor-free underwater sensor networks," *Computers & Electrical Engineering*, vol. 39, no. 6, pp. 1812–1821, 2013.

[54] W. Zhang, G. Han, X. Wang, M. Guizani, K. Fan, and L. Shu, "A node location algorithm based on node movement prediction

in underwater acoustic sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3166–3178, 2020.

[55] S.-H. Chang and K.-P. Shih, "Tour planning for auv data gathering in underwater wireless," in *Proc. IEEE 18th Int. Conf. Netw.-Based Inf. Syst.* IEEE, 2015, pp. 1–8.

[56] G. Han, Z. Tang, Y. He, J. Jiang, and J. A. Ansere, "District partition-based data collection algorithm with event dynamic competition in underwater acoustic sensor networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 10, pp. 5755–5764, 2019.

[57] S. Cai, Y. Zhu, T. Wang, G. Xu, A. Liu, and X. Liu, "Data collection in underwater sensor networks based on mobile edge computing," *IEEE Access*, vol. 7, pp. 65 357–65 367, 2019.

[58] M. Huang, K. Zhang, Z. Zeng, T. Wang, and Y. Liu, "An auv-assisted data gathering scheme based on clustering and matrix completion for smart ocean," *IEEE Internet Things J.*, vol. 7, pp. 9904–9918, 2020.

[59] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, 1996.

[60] B. Barsky and T. Derose, "Geometric continuity of parametric curves: Three equivalent characterizations," *IEEE Comput. Graph. Appl.*, vol. 9, no. 6, pp. 60–69, 1989.

[61] Z. Fang, J. Wang, C. Jiang, Q. Zhang, and Y. Ren, "Aoi-inspired collaborative information collection for auv-assisted internet of underwater things," *IEEE Internet of Things Journal*, vol. 8, no. 19, pp. 14 559–14 571, 2021.

[62] Z. Zhou, J.-H. Cui, and S. Zhou, "Efficient localization for large-scale underwater sensor networks," *Ad Hoc Networks*, vol. 8, no. 3, pp. 267–279, 2010.

[63] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "A gossip learning approach to urban trajectory nowcasting for anticipatory ran management," *IEEE Transactions on Mobile Computing*, 2023.

Nhat Nguyen received the BEng (Honours) degree in Electrical and Electronic at the University of Adelaide in 2020. He is currently pursuing the PhD degree with the School of Computer and Mathematical Sciences, the University of Adelaide. His research focuses on sequential decision-making algorithms for multiple-agent systems.

Duong D. Nguyen received the B.Sc. degree (Hons.) in electronic communication systems from the University of Plymouth, U.K., in 2008, the M.Sc. degree in mobile and personal communication from King's College London, U.K., in 2009, and the Ph.D. degree in engineering from The University of Adelaide, Australia, in 2018. From 2018 to 2023, he was a Postdoctoral Researcher at The University of Adelaide. In 2023, he joined as a Research Scientist at the Defence Science and Technology Group, Australia. His research interests include game theory models and decision-making algorithms for autonomous systems.

Dr Junae Kim earned her Ph.D. in Computer Engineering with a focus on computer vision and machine learning from the Australian National University. Since 2013, she has been serving as an AI specialist at DSTG (Defence Science and Technology Group) in Australia, where her research centres around artificial intelligence, machine learning and cybersecurity.

Gianluca Rizzo is Associate Professor of Computer Science at Università di Foggia , Italy, and Senior Research Associate at HES-SO Valais, Switzerland. Previously, he has been with Institute IMDEA Network, and Adjunct Professor at UC3M, Madrid. He received his M.Sc. in EE from Politecnico di Torino in 2001, and his PhD in Computer Science in 2008 from EPFL, Switzerland. His main research interests are in the performance evaluation of distributed systems.

Hung Nguyen is an associate professor in Computer Science at the University of Adelaide, Australia, where he has been since 2009. He obtained his PhD in computer and communication sciences from EPFL. His current research interest is in models and algorithms for improving network performance, security, and resiliency, especially for IoT and wireless networks.