Published in Proceedings of 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 14-16 June, Belfast, Northern Ireland, which should be cited to refer to this work.
DOI: 10.1109/WoWMoM54355.2022.00023

Multi-Agent Data Collection in Non-Stationary Environments

Nhat Nguyen*, Duong Nguyen*, Junae Kim†, Gianluca Rizzo‡, Hung Nguyen*
*The University of Adelaide, Australia †Defence Science and Technology Group, Australia
‡HES SO Valais, Switzerland, and University of Foggia, Italy
Emails: nhatdaoanh.nguyen@adelaide.edu.au, duong.nguyen@adelaide.edu.au, junae.kim@dst.defence.gov.au, gianluca.rizzo@hevs.ch, hung.nguyen@adelaide.edu.au

***Abstract*—Coordinated multi-robot systems are an effective way to harvest data from sensor networks and to implement active perception strategies. However, achieving efficient coordination in a way which guarantees a target QoS while adapting dynamically to changes (in the environment, due to sensors' mobility, and/or in the value of harvested data) is to date a key open issue. In this paper, we propose a novel decentralized Monte Carlo Tree Search algorithm (MCTS) which allows agents to optimize their own actions while achieving some form of coordination, in a changing environment. Its key underlying idea is to balance in an adaptive manner the exploration-exploitation trade-off to deal effectively with abrupt changes caused by the environment and random changes caused by other agents' actions. Critically, outdated and irrelevant samples - an inherent and prevalent feature in all multi-agent MCTS-based algorithms - are filtered out by means of a sliding window mechanism. We show both theoretically and through simulations that 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 on the problem of underwater data collection, showing on a set of different models for changes that our approach greatly outperforms the best available algorithms for that setting, both in terms of convergence speed and of global utility.**

I. Introduction

Over the last few years, multi-robot cooperative data harvesting, based on unmanned aerial vehicles (UAVs) or underwater autonomous vehicles (AUVs) has attracted widespread attention [1]. Autonomous vehicles (AVs) are widely used for efficient data communication as wireless relays. Thanks to their high mobility, they can move in proximity of IoT devices to harvest data and relay them to a fusion center. This allows avoiding multi-hop or long range transmissions, greatly prolonging IoT devices lifetime. In particular, decentralized collaborative multi-robot systems for active perception have become popular due to their robustness and scalability, which makes them suitable for industrial and military applications, as well as in environmental monitoring, in search and rescue missions, or in online object recognition and tracking [1]–[3]. The use of AVs is particularly beneficial when sensors/tracked objects are too far apart from each other and from data sinks, and when sensors location, tracked objects, and the environment around change over time in an unpredictable manner, such as in post-disaster (e.g. flood, earthquakes) scenarios. It is this variability over time which 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 AoI or latency – a key issue still largely unsolved [4]–[6].

As these solutions show, the state-of-the-art approach in multi-robot path planning is based on Decentralized Monte Carlo Tree Search (Dec-MCTS) [6] and on an exponentially decreasing forgetting factor to handle the changes in rewards distribution caused by other agents' actions and to encourage exploration of new paths. However, recent works have highlighted significant issues with such approach [7]–[9], 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 elaborate a first approach to tackle the above-mentioned issues. We propose a new algorithm for efficient decentralized multi-robot path planning for data harvesting and active perception in volatile environments, designed to cope with complex and unexpected changes in the rewards associated to each data collection task, in which agents coordinate by periodically sharing a compressed form of their search trees. Our strategy allows each agent to optimize its own actions by maintaining a probability distribution over plans in the joint action space. Moreover, it adopts a sliding window mechanism to force the MCTS algorithm to ignore outdated and irrelevant samples, in a way which effectively accounts for changes due to the environment, as well as for those due to agents' choices at each round of the data collection process. Specifically, our main contributions are:

- We formulate the problem of optimal multi-robot path planning with time varying rewards, where changes are due to both dynamics of the environment and to 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 which balances exploration-exploitation trade off, to deal effectively with abrupt changes in the environment. Our algorithm admits a general class of objective functions, optimizes actions over an arbitrarily long planning horizon, it is anytime, and it is robust with respect to limitations in the frequency and amount of information exchange among agents e.g. due to energy budget constraints for 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 convergence rates 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 method using simulations in a realistic setup, by applying it to the problem of multi-drone path planning for underwater data harvesting. Results suggest that our solution performs substantially better than the best existing ones both in static settings and in presence of frequent changes in sensor position and availability, achieving faster convergence and higher resource efficiency, even when decreasing the frequency of coordination exchanges among agents.

II. Background and Related Work

Information gathering problems using robots are often modeled as sequential decision making problems [10]. In the simplest setting with one 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. Numerous approaches for the multi-agent planning problem have been proposed over the last few decades.

Some early heuristic solutions include myopic solvers, that minimize the objective function over a limited time horizon [11], [12]. When the objective function is submodular, the myopic methods can achieve near-optimal performance [13]. When the objective function is not submodular, non-myopic decentralized coordination algorithms for multiple agents exploiting problem-specific characteristics are proposed (e.g. in [14], [15] and literature therein).

In more general settings, decentralized active information gathering can be viewed as a partially observable Markov decision process (POMDP) in a decentralized form [16], [17]. The most dominant approach to Dec-POMDP is to first solve the centralized, offline planning over the joint multi-agent policy space and then push these policies to agents who then execute them online in a decentralized way [17]. These online-offline approaches are however not applicable to dynamic environments where the state of the environment is not known ahead of time.

When uncertainties in the environment are prevalent, simultaneous decentralized planning and execution can be performed for Dec-POMDP using solutions such as [18]. Standard Dec-POMDP methods, however, suffer from significant memory cost and high computational complexity due the requirements to compute and store the reachable joint state estimations of all agents [19]. Recently, decentralized planning using the *anytime* Monte Carlo Tree Search algorithm has gain 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 [20]–[23]. 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 as a multi-armed bandit (MAB) problem [24], [25]. These algorithms seek to find the most rewarding move regarding 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 [6], a Decentralized MCTS algorithm (Dec-MCTS) allows each agent to optimize its own actions by maintaining a probability distribution over plans in the joint action space. In order to deal with the uncertainty deriving from changes in the environment, [20] and [22] propose a version of Dec-MCTS which exploits heuristic-based prediction methods, based on predefined models of other teammate agents. MCTS with coordination graph is used in [23] for multi-agent planning. However, being based on previous knowledge, they are unsuitable for robot motion planning in settings which change in an unpredictable fashion, as it is often the case in realistic scenarios, e.g. in a disaster environment.

Within the larger domain of multi-armed bandit (MAB) approaches, recent works [26]–[29] have focused on applications in non-stationary environments. Some of these approaches are based on the assumption that the dynamics of the changing rewards over time are known [26]–[28], while others assume no previous knowledge on patterns of reward variation [30]. However, how to apply these results to multi-robot planning via MCTS in nonstationary settings is a non-trivial problem which 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 tackle 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).

III. System Model and Problem Definition

A. Basic assumptions

We consider a set of S wireless sensors, each with a communication radius 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 sea port 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.

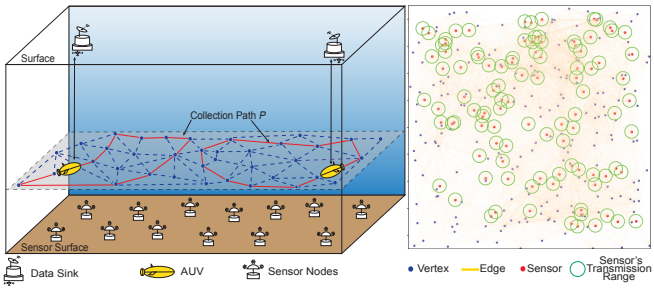

Fig. 1: Example of data collection scenario for the case of an underwater WSN (left) and representation of the sensor plane and of the motion graph (right).

We assume N static wireless data sinks are present in the same volume. 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 too distant among themselves and from data sinks to be able to relay data to other sensor nodes and/or directly to data sinks. Thus, we assume that in the given volume a set of M homogeneous agents (modeling such robots as UAVs, or underwater drones) periodically harvest data from sensors and relay them to data sinks. Let m be the label of the $m-$th agent.

Without loss of generality, we assume the trajectories of agents are 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 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. We assume each sink is connected to the motion graph via two overlapping edges, one per direction. A schematic representation of the system, and an example of the path model are illustrated in Fig. 1. All agents know the motion graph for the considered volume.

We assume that each agent communicates with other agents when it is at a sink, via the data sink itself. In what follows we assume the likelihood of two agents to get close enough to be able to exchange directly information to be negligible. The data harvesting process takes place over a time horizon of T time intervals (denoted as *rounds*), each of same duration. At the beginning of every round, each agent is at the location of a data sink. During a round each agent moves at constant speed (equal for all agents) along the motion graph, harvesting data from all sensors for which it gets within their transmission range. The path of each agent in a round is chosen in such a way as to terminate at a sink by the end of the round itself, in order to allow the agent to relay the harvested data to the sink, and to exchange information with the other agents. For ease of analytical treatment, we assume any exchange of information (e.g. from sensors to agents, from agents to data sinks, and among agents via data sinks) to be instantaneous. Note however that our approach can be easily extended to account for finite exchange duration, as well as to agents moving at different speeds.

In round t, with $v^t_{i,m}$ we denote the i-th vertex that the m-th agent has visited since the beginning of the round. The *path* of the m-th agent in round t, denoted as p^t_m, is an ordered list of vertices $p^t_m = (v^t_{1,m}, \ldots, v^t_{n_m,m})$ of length n_m, such that two adjacent vertices in the path are the two vertices of an edge of the motion graph G, and in which only the first and the last vertex in a path are sink nodes. Let B denote the maximum path length during a round. Such a maximum value is derived from node speed and round duration, but it may capture also various constraints, e.g. due to finite storage capacity, finite per-round energy budget, among others. We assume B is the same for each agent and in each round. We assume the round duration is larger than the time taken by any agent to traverse a maximum path length B, and that possibly it also accounts for the time required to recharge the agent and/or to exchange data at the sink. To any path p^t_m on the motion graph we associate a cost $b(p^t_m)$, equal to its length.

We consider that the first time in a round that a sensor finds an agent within its transmission range, it sends its own data, removing them from memory. We denote this event as a *successful harvesting* event. All subsequent events of contact between that sensor and agents in that round do not bring to any data transfer (*failed harvestings*). At the beginning of a round, all sensors have data to transfer, and the amount of that data remains the same independently on whether data has been harvested from a given sensor in the previous round, or not.

We assume that at the beginning of the first round, position and availability of sensors are unknown to the agents. At the beginning of each round, every agent knows the location of each collected sensor and its operating status thanks to the exchange of information among agents. However, this information might be outdated due to changes in the environment occurring after that location and status of each sensor have been assessed.

B. Problem formulation

Our goal is to find, at each round, a path for each agent such that the cumulative utility of the data collection process over T rounds is maximized. The utility function is defined as follows. To every harvesting event at the s-th sensor in round t we associate an utility, which is equal to w^s_t in case of success, and zero otherwise. w^s_t is in principle different for each sensor, and potentially varying at every round.

Let r^t_m denote the total utility associated to the traversal of path p^t_m. It is equal to the sum of the utility of all the successful harvesting events which the m-th agent has performed while traversing it. It is a quantity which, for a same path, may be in principle different at each round (e.g. due to sensor movement,

to changes in sensor availability and/or harvesting utility). Moreover, given that failed harvestings yield zero utility, it depends not only on the path of the m-th agent, but also on those of all the other agents in that round. We denote its expected value as $\bar{r}_m^t$.

With P_m^t, we denote the set of all possible paths which start at agent m's starting sink at the beginning of the t-th round. Let $\mathbf{P^t} = (P_1^t, ..., P_m^t, ..., P_M^t)$ denote the collection of sets of paths for all agents at the t-th round, and let $\mathbf{P} = (\mathbf{P^1}, ..., \mathbf{P^t}, ..., \mathbf{P^T})$. We define then the following problem:

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

$$\underset{\mathbf{P}}{\text{maximize}} \sum_{t=1}^{T} \sum_{m=1}^{M} \bar{r}_m^t \tag{1}$$

Subject to, $\forall p_m^t \in \mathbf{P}$,

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

Constraint (2) derives from imposing that the total path length for the m-th agent in the t-th round to be less than the travel budget B available to each agent in a single round. Such an optimization problem cannot be solved efficiently. Indeed it is easy to see that Problem 1 is a variant of the well known NP-hard travelling salesman problem.

In the next section we provide an adaptive and distributed learning solution to Problem 1, that learns from the environment and from the actions of other agents, updating planning decisions accordingly in each round.

IV. The Sliding Windows MCTS Algorithm

A. Algorithm overview

Our algorithm, denoted as *sliding window MCTS (SW-MCTS)* implements a distributed planner which aims at finding the best course of action (CoA) for each agent in each round t, i.e., to determine for each agent a path which globally maximizes the utility function in Problem 1. At each round the distributed planner determine these paths (one per agent), executes them, updates the information used for path planning (such as sensor position and availability, which might have changed during the round), and replans before starting a new round. 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 a vertex of the motion graph, and the *actions* at such node are the edges starting from that vertex. Specifically, our algorithm extends the well known UCT bandit algorithm for tree search [24] to nonstationary settings. In order to find the optimal equilibrium between exploration and exploitation in presence of changes in the reward scheme, the move selection problem at each internal tree node is modeled as a separate multi-armed bandit, in which the arms correspond to the possible moves 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 at the t-th round is shown in Algorithm 1. To each node v of the motion graph we associate, for each round $u \in [1, t-1]$, the following parameters (denoted as *node statistics*):

- The *rollout score* F_u^v, which is the average of the utility of all paths which, at round u, traversed node v;
- For every adjacent vertex v' of v, the number of times it has been visited from vertex v during the u-th round, denoted as $\alpha_v(v', u)$.

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

At the beginning of each round, each agent samples the paths taken by other agents by drawing at random a set of paths, one for each of the other agents, using the probability mass function given. These sampled paths are then used by the agent in planning its own optimal path, assuming the other agents are taking these sampled paths in the current round. Then, starting from the sink node, each agent at the i-th node of its path chooses the next node to visit using the following strategy:

- While the current node is fully explored (i.e., all actions 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 action. Then the agent *selects* the action with the highest score and visits the

Algorithm 1 *SW-MCTS algorithm* for agent m in round t

Input: Motion graph G, travel budget B, set of feasible paths and probabilities of other agents $(\hat{P}_{(m)}^t, q_{(m)}^t)$

Output: p_m^t

1: $p_{(m)}^t \leftarrow$ *Sample* $(\hat{P}_{(m)}^t, q_{(m)}^t)$
2: $i = 1$
3: **while** $v_{i,m}^t$ is fully explored AND residual budget allows AND sink not reached
4: $\quad v_{i+1,m}^t \leftarrow$*SW-UCT*$(v_{i,m}^t)$
5: $\quad i = i + 1$;
6: **end while**
7: **while** sink not reached AND residual budget allows
8: $\quad v_{i+1,m}^t \leftarrow$ *Random policy* $(v_{i,m}^t)$
9: $\quad i = i + 1$;
10: **end while**
11: *Backpropagation* $(p_m^t, p_{(m)}^t, \hat{E}_m^t)$
12: $(\hat{P}_{(m)}^{t+1}, q_{(m)}^{t+1}) \leftarrow$ *Update and Communicate* $(\hat{P}_m^{t+1}, q_m^{t+1})$

corresponding next hop. This continues until either the agent reaches a sink, or the agent reaches a node which is not fully explored. In the latter case, the agent chooses randomly one untried child node to *expand* the tree search. We use $\hat{E}^t_m(\subset p^t_m)$ to denote a set of chosen actions and the newly expanded node of the agent m by using the *SW-UCT* policy in round t.

- Then, the agent applies the *random policy*, randomly picking one action at every node and visiting the corresponding next hop until a sink is reached. Note that, in any case, an action cannot be chosen if it does not allow getting back to the starting sink with the residual travel budget. This condition is checked at every node along the path of the agent, thus ensuring that constraint Equation (2) is always satisfied.

In the *backpropagation* phase, the rollout score for p^t_m is computed, which is set equal to the utility of the agent's path at round t. This utility is derived using a *local utility function* as in [6], and by estimating the effect of the possible path choices of other agents, through $(\hat{P}^t_{(m)}, q^t_{(m)})$. Such information is updated for every nodes in $\hat{E}^t_m$ to use for calculating the SW-UCT scores in the following rounds.
After that, each agent elaborates $\hat{P}^{t+1}_m$ and q^{t+1}_m by accounting for p^t_m, and it shares them with the other agents. Finally, it updates the node statistics, and it starts a new round.

B. 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 parametrized by a sliding window constant $\tau \geq 0$ which tunes the weight which the results of past explorations should have in computing payoffs. The algorithm is based on computing, at round t and node v, an *upper confidence bound* $U_{j,t}$ on the value of each child j of the given node. That is, an upper bound on the potential payoff of choosing that child node as 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 $I_{v,t}$.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 [26]. Specifically, the upper bound 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

$$X_{j,t}(\tau) = \frac{1}{N_t(\tau, j)} \sum_{u=t-\tau+1}^{t} F^j_u \tag{3}$$

$N_t(\tau, j)$ is the number of times the child node j within the last τ iterations has been visited. The average empirical reward accounts for the results of the past explorations, and it thus represent 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:

$$H_{j,t}(\tau) = 2C_p \sqrt{\frac{\log(t \wedge \tau)}{N_t(\tau, j)}}, \tag{4}$$

where $t \wedge \tau = \min(t, \tau)$. $C_p > 0$ is an *exploration constant*, and it is used to tune the relative weight which 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 which have been visited less in the past, and it thus pushes the algorithm towards exploring new path choices. Critically, while the general structure of this bound follows the typical one of UCT algorithms [24], in the sliding window version we propose here both exploration and exploitation terms are functions of the sliding window constant τ, which thus models the "memory" of the system.

V. 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 [24], 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.

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 bandit 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.

A few notations are needed for the analysis of step 1, SW-UCB. Let $I_t \in \{1, \ldots, K\}$ 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 $\{X_{i,t}\}_t$, $i = 1, \ldots, K$ 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

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

where the empirical reward $\bar{X}_{i,t}(\tau)$ is given by (3) and the exploration bonus $H_{i,t}(\tau)$ is given by (4).

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 $\mathcal{F}_{i,t+1}, \mathcal{F}_{i,t+2}, \ldots$ 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 break point 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 $i \in \{1, \ldots, K\}$.

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 $\mu_{i_t^*,t} = \max_{i\in\{1,\ldots,K\}\mu_{i,t}}$. 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 $\Delta_{i,t} = \min_{u\in\{1,\ldots,t\}}\{\mu_{i_u^*,u} - \mu_{i_u,u} : i \neq i_u^*\}$. 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. Then for any arm $i \in \{1, \ldots, K\}$ and $T > 1$,

$$\mathbb{E}_\tau\left[\tilde{N}_i(T)\right] \leq O\left(\sqrt{\mathbb{E}[\Upsilon_T]T\log(T)}\ (C_p^2 + T_0(\epsilon) + T_p)\right). \tag{5}$$

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

Remark: Compared to the bounds for the Dec-MCTS in [6] 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 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 pay-off 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,

$$\mathbb{E}_\tau\left[\bar{X}_T - \mu^*\right] \leq |\delta_t^*| + O\left(\sqrt{\frac{\mathbb{E}[\Upsilon_T]\log(T)}{T}}\ (T_0(\epsilon) + T_p)\right).$$

The following lemma provides a bound on the concentration of the actual payoff $\bar{X}_t$ about the expected payoff. 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 $0 < \epsilon \leq 1$ and let $\Gamma_t = 9C_p\,\mathbb{E}[\Upsilon_t]t\sqrt{2\log(2/\epsilon)}$. Then, under the assumptions of Lemma 1, for $t \geq O\left(\sqrt{\mathbb{E}[\Upsilon_T]T\log(T)}(C_p + T_0(\epsilon) + T_p)\right)$ the following bound holds:

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

The proofs for Lemma 2 and 3 follow the steps in the proofs of Lemma 2 in [6] and Theorem 5 in [24] but require adjustments for the sliding window. Due to space constraints, we skip the detailed proofs in this version.

The previous three lemmas are for SW-UCB. We will 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 $t \geq 1$. 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 4. 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 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,

$$|\bar{F}_{i_0,T_{i_0}} - \mu_{i_0}^*| = O(KD\sqrt{E[\Upsilon_T]\log(T)/T})$$

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

The proof is an induction on D using results from Lemmas 1-4.

VI. Evaluation

In this section, we evaluate the performance of our SW-MCTS algorithm as a function of the main system parameters. The scenario we considered consists of an UWSN in which 100 sensors are randomly distributed in a 2000 m $\times$ 2000 m plane, with a transmission radius of 50 m (typical of underwater acoustic communications for UWSN, e.g. [31], [32]). Each agent in the scenario travels at a constant speed of 5 m/s. The motion graph is built using a probabilistic roadmap (PRM) with a Dubins path model [33]. This model uses curves for smoothing straight-lines between waypoints and it has been widely used to model motion constraint of vehicle-like nonholonomic robots such as AUVs. This constraint implies that each AUV agent can only travel along smooth curvatures without reversing direction, and that its trajectory must satisfy geometric continuity [34]. The resulting motion graph has 200 vertices, and a maximum edge length of 500 m. Unless otherwise stated, we observe the performance of our algorithm over a time interval of $T = 10^4$ rounds, and for an AUV travel budget of 4.5 km (typical of many current implementations of AUVs [31]).

We benchmark our SW-MCTS algorithm with the discounted method (Dec-MCTS [6]) which, although not being designed for varying environments, is the state-of-the-art decentralized multi-agent planning method for information gathering. In addition, we consider an ideal greedy algorithm (*greedy planner*), in which every agent at each step is able to forecast perfectly the reward associated to visiting a given edge of the graph, and in which every agent chooses the action which delivers the highest immediate reward.

The discounting factor of Dec-MCTS is 0.5, i.e. at the lowest value recommended [6] to ensure the strongest forgetting effect, and thus the fastest adaptation to changes in the environment. Unless otherwise stated, the window size of SW-MCTS has been set to 40 rounds, agents have communication exchanges with their fleet every 10 rounds and the exploration parameter C_p for both algorithms is equal to 1.8×10^{-3}. Those values are chosen to enable a high convergence speed in the vast majority of scenarios considered in our experiments. We assumed utility of successful harvesting events to be the same for all sensors and at any time. Three different models of changes in the environment have been considered:

- *Static:* Position and availability of sensors remain unchanged throughout the time interval T. This setting corresponds to a conventional decentralized offline planning problem with known reward distributions.
- *Location changes (LC):* The location of all sensors change randomly and abruptly at a given point in time. This scenario models the case in which sensors are drifted away due to a storm, or to a sudden change in water currents, e.g. due to ocean tides.
- *Availability changes (AC):* All sensors lying on the agents' optimal paths at round $T/2$ (i.e., the most likely at round $T/2$ among all the feasible paths, for the MCTS algorithms) are assumed to be unavailable starting from round $T/2+1$. This setup models the One Hot Region scenario, in which sensors that are located near the agents' trajectories tend to be out of battery faster, as they communicate more often than less visited nodes.

We use the following metrics to evaluate the performance of the algorithms:

- *Instantaneous reward coverage (IRC)* at round t: Fraction of available sensors covered at that round. A stable instantaneous reward indicates that the algorithm has converged.
- *Average per round reward coverage (ARC):* Fraction of available sensors covered per round, averaged across all rounds in the considered time interval. That is, $ARC = \frac{\sum_{t=1}^{T} IRC(t)}{T}$.

A. Performance Benchmarking

In order to perform a first evaluation of our algorithm, we considered the static scenario with 2 AUVs. As Fig. 2a shows, while the greedy planner performance does not improve over time, both MCTS-based algorithms quickly outperform it, as MCTS agents are able to discover more potential high-reward areas by reaching deeper levels of their search trees. Indeed, as visible from the overlay of a sample mission in Fig. 2d, decentralized path planning in SW-MCTS is very effective in avoiding overlapping paths while covering as many distinct sensors as possible within each round. Moreover, these results suggest that SW-MCTS, although not being designed for static settings, actually outperforms Dec-MCTS (i.e, the best available algorithm for AUVs coordination in such settings) thanks to its substantially faster convergence rate. Thus, our proposed algorithm is also a good fit for multi-robot coordinated information gathering in stable settings.

To evaluate the impact on performance of changes in the environment, we considered the *LC* and *AC* scenarios, assuming the changes to occur after $5,000$ rounds. As Figs. 2b and 2c respectively show, the greedy agents are able to react instantly after the break point in both setups. Instead, MCTS agents take slightly more time to adapt to these changes, but they quickly find better paths to cover the sensors than those chosen by greedy agents. Although both SW-MCTS and Dec-MCTS are able to handle the changes, SW-MCTS performs significantly better than the discounted method, achieving higher reward coverage before and after the break point, and converging substantially faster in both test cases.

B. Impact of parameter settings

In this section, we investigate the impact of the key parameters of the system and of the SW-MCTS algorithm on the performance of our solution, with a special focus on its scalability and robustness. Specifically, we consider the *LC* setting, and we evaluate the impact of travel budget, of the number of AUVs, of the duration of the communication cycle, and of the number of times a change in node position takes place over T rounds. Fig. 3 illustrates the impact of these

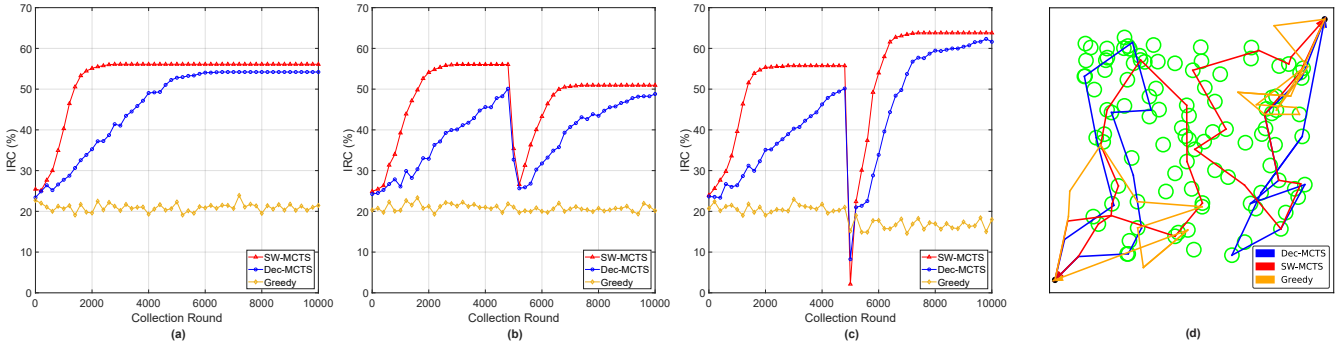

Fig. 2: Evolution over time of the Instantaneous Reward Coverage (IRC) in the *Static* setting (a), in the *LC* scenario (b), and in the *AC* scenario (c), averaged over 20 runs. Sample overlay mission of AUVs for underwater data collection, for the three algorithms considered, at the last round of the considered time horizon in the static setting (d).

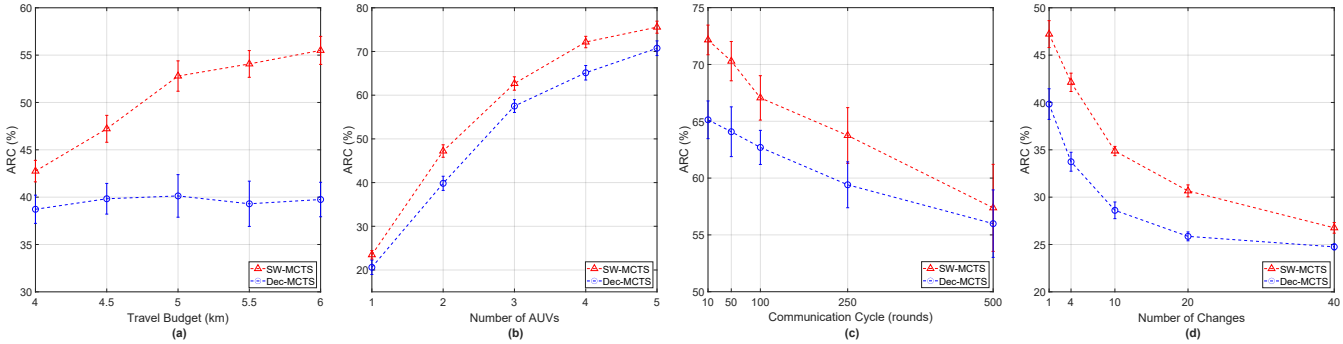

Fig. 3: Impact of different parameters on the algorithms performance. (a) Travel budget; (b) Number of AUVs; (c) Time between two consecutive events of synchronization among agents; (d) Number of changes per time window T in the LC scenario, for $T = 10,000$ rounds. Results are with 95% confidence interval.

parameters on ARC, and for a default number of AUVs equal to 2.

As Fig. 3a shows, with small values of B, the difference in performance between the two algorithms is marginal, but it increases quickly to 15% with growing travel budget. Indeed, with larger travel budgets the space of possible choices for path planning 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 Fig. 3b shows, the improvement of our approach with respect to the discounted method is substantial when increasing the number of AUVs. Naturally, when trying to achieve large values of coverage (larger than 70% in the considered setting) 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 to low reward, e.g. because of being far away from the bulk of the other sensors.

Among the factors affecting the AUVs energy budget a key role is played by the frequency of information exchanges among AUVs, taking place when AUVs communicate at data sinks. Potentially, more frequent information exchanges imply a faster adaptation of the SW-MCTS algorithm to environmental changes, at the cost of decreasing the amount of energy available for data harvesting (e.g., as a travel budget). As a key step towards characterizing such tradeoff, in Fig. 3c we have studied the impact of the duration of the communication cycle T_c on average per round rewards coverage, in the *LC* setup, with 4 AUVs. As expected, both SW-MCTS and Dec-MCTS drop in performance as synchronization among agents becomes less frequent. Indeed, the less agents synchronize, the less each agent is aware of the changes in the paths of other agents, thus potentially visiting sensors that have been covered already by other AUVs. Moreover, for very large communication cycles, the performance of SW-MCTS drops close to the discounted algorithm. This is due to the fact that, as explained in Section V for a given value of communication cycle and of frequency of changes, there is a range of values of window size τ which guarantee satisfactory behavior for SW-MCTS algorithm. Thus in the considered setup, a window of size $\tau = 40$ is a suboptimal choice for a communication cycle of 500 rounds or more.

In the *LC* setup, we have also assessed the impact of the rate of changes in the environment, with 2 AUVs. Specifically, we have assumed sensor position to vary n times within the time period T in a periodical fashion, with a new spatial

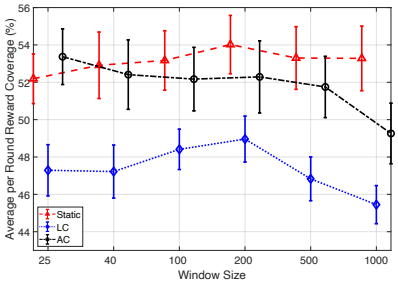

Fig. 4: Impact of window size on Average Reward Coverage (ARC) for the SW-MCTS algorithm. Results are with 95% confidence interval.

configuration of the sensors every T/n rounds. As shown in Fig. 3d, ARC decreases as the rate of changes increases. Indeed, a higher frequency of changes imply that less time is available for the algorithms to converge and adapt to the new configuration (and its induced reward distribution). However, thanks to its faster convergence rate our SW-MCTS algorithm substantially outperforms the discounted approach, showing a better ability to adapt to changing environments. Specifically, this result shows that in the considered setting, SW-MCTS with a 40 round sliding window allows agents to adapt and ignore irrelevant (pre-change) observations much more effectively than a Dec-MCTS configured for maximum forgetting effect.

Finally, we investigate the relationship between window size τ and ARC for our algorithm, for the three settings considered in a scenario with 2 AUVs. Results in Fig. 4 suggest that in each scenario performance improves when increasing window size from a value of a few rounds. However, in all settings ARC peaks at a specific window size, generally different for each setting, and starts decreasing monotonically for larger window values. Indeed, for SW-MCTS window size sets the tradeoff between, on one side, allowing more information to be available for the agent to estimate the reward of each action (and hence to plan the optimal path), and avoiding the inclusion of irrelevant past information (i.e. related to an outdated sensor layout, or to inherent randomness in the choices of agents). For instance, results suggest that in settings such as *AC*, in which environmental factors alter sensors' availability in the considered area in a very rapid fashion, a smaller window is required to allow the algorithm to quickly remove the outdated observations due to the change.

VII. 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 general approach to tackle this issue, based on carefully balancing the exploration-exploitation trade-off and on appropriately weighting the contribution of past information on planning decisions. We have shown on a practical scenario that our algorithm performs substantially better than the best approach available in the literature, achieving faster convergence and higher resource efficiency despite implementing a loose form of synchronization among agents, even in settings with frequent changes. As a followup, 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.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

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

[19] 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.

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

[21] 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.
[22] A. Czechowski and F. A. Oliehoek, "Decentralized mcts via learned teammate models," in *IJCAI*, 2020, pp. 450–456.
[23] S. Choudhury, J. K. Gupta, P. Morales, and M. J. Kochenderfer, "Scalable anytime planning for multi-agent mdps," in *AAMAS*, 2021, pp. 341–349.
[24] L. Kocsis, C. SzepesvÁRi, and J. Willemson, "Improved monte-carlo search," *Univ. Tartu, Estonia, Tech. Rep*, vol. 1, 2006.
[25] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Proc. 5th Int. Conf. Comput. Games*. Springer, 2006, pp. 72–83.
[26] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM Journal On Computing*, vol. 32, no. 1, pp. 48–77, 2002.
[27] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Algorithmic Learning Theory*. Springer, 2011, pp. 174–188.
[28] 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.
[29] 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.
[30] 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.
[31] 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.
[32] 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.
[33] 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.
[34] 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.

APPENDIX

A. Proof of Lemma 1

Step 1: For suboptimal arm i, let $A_0(t,\epsilon,\tau) = \min\{N_t(\tau,i) | c_{i,t}(\tau) \leq (1-\epsilon)\Delta_{i,t}/2\}$. From the definition of $c_{i,t}(\tau)$, $A_0(t,\epsilon,\tau) = \frac{16C_p^2\log(t\wedge\tau)}{(1-\epsilon)^2\Delta_{i,t}^2}$. Let $A(t,\epsilon,\tau) = \max(A_0(t,\epsilon,\tau), T_0(\epsilon), T_p)$. Then the number of times a suboptimal arm is played is

$$\tilde{N}_i(T) = 1 + \sum_{u=K+1}^{T} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)<A(u,\epsilon,\tau)\}} + \sum_{u=K+1}^{T} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)\geq A(u,\epsilon,\tau)\}} \quad (6)$$

From Lemma 1 in [27], the second term in the right hand side is bounded by

$$\sum_{u=K+1}^{T} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)<A(u,\epsilon,\tau)\}} \leq \lceil T/\tau\rceil A(u,\epsilon,\tau).$$

Let $D(\tau) = \frac{\log\left((1/\tau)C_p^2\log(K(1-1/\tau))\right)}{\log(1-1/\tau)}$, and denote by $\mathcal{T}(\tau)$ the set of all indices $t \in \{K+1,\dots,T\}$ such that for all integers $s \in]t-D(\tau), t]$, for all $j \in \{1,\dots,K\}, \mu_s(j) = \mu_t(j)$, then as shown in [27]

$$\sum_{u=K+1}^{T} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)\geq A(u,\epsilon,\tau)\}} \leq \Upsilon_T D(\tau) + \sum_{u\in\mathcal{T}(\tau)} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)\geq A(u,\epsilon,\tau)\}}.$$

Putting everything together, we then have

$$\tilde{N}_i(T) \leq 1 + \lceil T/\tau\rceil A(u,\epsilon,\tau) + \Upsilon_T D(\tau) + \sum_{u\in\mathcal{T}(\tau)} \mathbb{I}_{\{u:(I_u=i\neq i_u^*), N_u(\tau,i)\geq A(u,\epsilon,\tau)\}}.$$

Step 2: There are three conditions under which a suboptimal arm will be played when $N_u(\tau,i) \geq A(u,\epsilon,\tau)$ (following a breakpoint), as follows:

$$\{u : (I_u = i \neq i_u^*), N_u(\tau,i) \geq A(u,\epsilon,\tau)\} \subseteq \begin{cases} \{u : (\mu_u^* - \mu_{i,u} < 2c_{i,u}(\tau)) \wedge (N_u(\tau,i) \geq A(u,\epsilon,\tau))\} \\ \cup\{u : \bar{X}_{i_u^*,u}(\tau) \leq \mu_u^* - c_{i_u^*,u}(\tau)\} \\ \cup\{u : \bar{X}_u^*(\tau) \geq \mu_{i,u} + c_{i,u}(\tau)\}. \end{cases}$$

Let's consider the first case, using Theorem 2 in [24]. Since $N_t(\tau,i) \geq A(t,\epsilon,\tau) \geq A_0(t,\epsilon,\tau)$

$$c_{i,t}(\tau) \leq 2\sqrt{C_p^2\log(t\wedge\tau)/A_0(t,\epsilon,\tau)} \leq \frac{\Delta_{i,t}}{2}\ .$$

The first case, therefore, cannot occur when $N_t(\tau,i) \geq A_0(t,\epsilon,\tau)$. When $N_t(\tau,i) \geq T_0(\epsilon)$, we have that $|\delta_{i,t}| \leq \epsilon\Delta_{i,t}/2$. Since $\mu^* - \mu_i \geq \Delta_{i,t}, t = 1,2,\dots,$ we have that

$$\begin{aligned}\mu^* - \mu_i - 2c_{t,t_i}(\tau) &\geq \Delta_{i,t} - |\delta_t^*| - \delta_{i,t} - 2c_{i,t}(\tau) \\ &\geq \Delta_{i,t} - \epsilon\Delta_{i,t} - (1-\epsilon)\Delta_{i,t} = 0\end{aligned}$$

Step 3 and 4 of the proof provide bounds for the probabilities of the last two events in Step 2. Recall that $M_i(t)$ denotes the number of pulls of arm i after the most recent breakpoint. Then, under Assumption 1, when $M_i(t) \geq T_p \leq A(t,\epsilon,\tau)$ we can then use Theorem 4 in [27] to show that

$$\mathbb{P}\left(\bar{X}_{i,t}(\tau) \geq \mu_{i,t} + c_{i,t}(\tau)\right) \leq \tau - K + \left\lceil \frac{\log(\tau)}{\log(1+\nu)} \right\rceil \frac{t}{\tau(1-(1-1/\tau)^\tau)}$$

for any positive number ν.

Step 5 Now combining everything together in (6) and taking expectation, we obtain

$$\mathbb{E}_\tau\left[\tilde{N}_i(T)\right] \leq C(\tau)\frac{T\log(\tau)}{\tau} + \tau\,\mathbb{E}[\Upsilon_T] + \log^2(\tau), \quad (7)$$

where

$$C(\tau) = \frac{4C_p^2}{(1-\epsilon)^2\Delta_{i,t}^2}\frac{\lceil T/\tau\rceil}{T/\tau} + \frac{2}{\log(\tau)}\left\lceil\frac{\log(\tau)}{\log(1+\nu)}\right\rceil$$

Finally, choosing $\tau = 2C_p\sqrt{T\log(T)/\,\mathbb{E}[\Upsilon_T]}$ yields

$$\mathbb{E}_\tau\left[\tilde{N}_i(T)\right] = O\left(\sqrt{\mathbb{E}[\Upsilon_T]T\log(T)}(C_p + T_0(\epsilon) + T_p)/T\right).$$