Energy-based Adaptive Coverage Path Planning with Multiple Robots

Benjamin Sportich[1], Alessandro Renzaglia[1], and Sara Bernardini[2]

[1] Inria, INSA Lyon, CITI, 69621 Villeurbanne, France
benjamin.sportich@inria.fr, alessandro.renzaglia@inria.fr
[2] Department of Computer Science, University of Oxford, Oxford, United Kingdom
sara.bernardini@cs.ox.ac.uk

Abstract. In this paper, we address a dynamic version of the multi-robot Coverage Path Planning (mCPP) problem where relevant changes in the robots' state (i.e., energy levels, robot failure, etc.) or in the environment definition occur during the mission. In real-world applications, the actual energy consumption can deviate significantly from the initial prediction and can also vary from robot to robot depending on different external disturbances. Similarly, changes to the area to be covered may be necessary during the mission in case of sudden no-go zones or extension of the area of operation. While existing standard mCPP solutions define the robot paths offline and are not able to react to such deviations from the expected conditions, our proposed approach can adapt to new information and provide updated coverage paths online. In particular, by modifying a state-of-the-art approach, the DARP algorithm, our solution is also able to find a new plan that minimizes the total unexploited energy. Furthermore, the computational time required to find a solution is significantly reduced compared to the original version of the DARP algorithm to facilitate the online replanning process. Finally, we provide simulation results to show the effectiveness of the proposed approach and evaluate its performance against two possible alternative solutions.

Keywords: Multi-Robot Systems · Path Planning · Coverage

1 Introduction

Collecting data exhaustively in a predefined area has always been a major focus of robotics. Known as Coverage Path Planning (CPP) [11], its scope has become increasingly prominent in recent years with very diverse applications such as inspection tasks, vacuum cleaner robots, automated lawnmowers, and patrolling security robots. Most of the effort has been directed to challenging projects where the enhanced capabilities of robots made missions possible in complex environments that are too large or hostile for humans. Complex scenarios, sometimes involving multiple coordinated robots, such as planetary exploration, underwater missions for seabed mapping, demining in old war zones, or post-disaster search and rescue, are gradually becoming approachable [10,6,3].

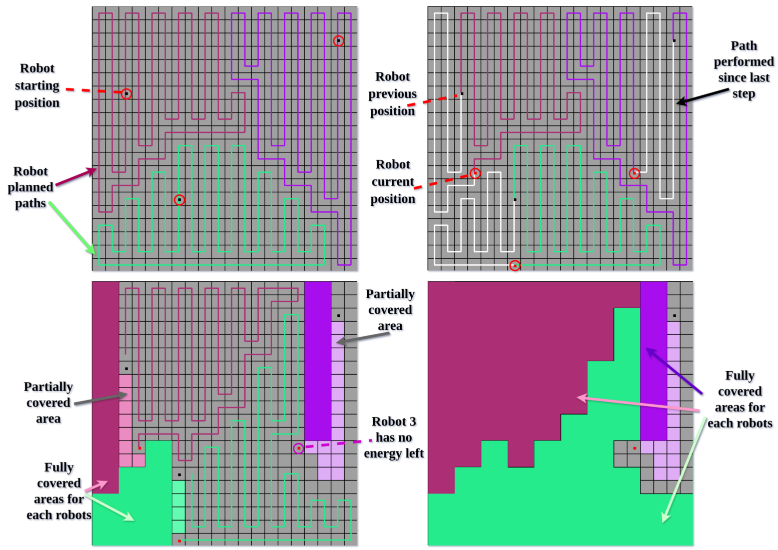

Fig. 1. Evolution of the paths of 3 robots when one of the robots (robot 3) has a failure mid-mission and has to stop.

However, existing multi-robot CPP (mCPP) solutions often make simplifying assumptions that do not reflect the complexity of real systems. All robots are usually expected to have the same characteristics, energy is rarely taken into account although it is critical in real applications, and more generally, a computed offline plan is assumed to be flawlessly performed by the robots in the real world. These unrealistic expectations, which ignore the dynamic aspect of robotic problems, necessarily induce a severe performance loss.

To cope with these limitations, mCPP missions can only be executed in controlled and predictable environments and can hardly be modified online when new information arises without prior significant ad hoc engineering. If the settings become too different from the initial conditions, the original plans must either be canceled or remain unaltered with substantial performance degradation, as the robots cannot benefit from new additional information during the execution of the plan. Furthermore, energy consumption is usually modeled in a very conservative manner to limit running out of energy during the mission, especially in extreme environments where deployment, recovery, and charge are costly or even impossible. This could leave much of the energy unused.

This paper deals with the dynamic nature of the coverage mission via the gradual integration of context changes in a previously computed offline plan. We focus on changes in energy consumption, team composition, or operational area definition that can hinder the correct execution of a plan (as in Figure 1). The proposed adaptive approach has been developed in the context of the Marine Robotics Project SoaR [21] for the inspection of offshore wind farms and aims at more accurately capturing the challenges of the multirobot coverage problem in the real world. Marine robotic missions cannot easily be interrupted due to the high cost of deployment, energy consumption is difficult to estimate due to the unpredictability of water currents and the effect of the tide, and sudden no-go zones can arise due to boats unexpectedly traversing the zone of operation.

Our main contributions are as follows: (a) We present a new framework for mCPP missions called *dynamic mCPP*, which captures the challenges of real-time robotic coverage applications such as energy fluctuations, mid-mission robots failures, and map changes; (b) We propose the first solution that efficiently adapts online the coverage paths to dynamic contexts based on dynamic mCPP; (c) We provide comparative results proving a higher success rate, faster convergence, and improved performance of our method with respect to two adaptations of a standard mCPP solution.

2 Related Work

The CPP problem has been extensively studied over the years and many possible solutions have been proposed [11], [16]. The first proposed approaches, initially focusing on the single-robot problem, considered regular paths as boustrophedon and back-and-forth motions as a way to efficiently visit all the cells of the environment [4], [17]. A seminal work was then presented in [9] where the Spanning Tree Coverage (STC) algorithm was proposed. This solution guarantees to find, in linear time, a minimum spanning tree to visit all the cells exactly once, thereby guaranteeing the non-backtracking property. The extension of the CPP problem to the multi-robot scenario, known as mCPP, has been proven to be an NP-hard problem [28]. An initial reference solution was presented in [12], where the STC algorithm was adapted for handling multi-robot teams. However, this solution demonstrated a high dependency on the initial positions of the robots. A poorly chosen spanning tree could result in a total exploration time nearly equivalent to that of a single robot. To address this issue, the same authors later proposed an enhanced solution in [2], introducing a heuristic to mitigate the impact of initial positions. Nevertheless, no performance guarantees were proven. Kapoutsis et al. then presented the DARP algorithm in [14]. This approach reduces the multi-robot case into n single-robot problems by partitioning the environment into n regions, assigning each to a different robot. The result is complete coverage with no backtracking and a minimum coverage path. A weighted version of this solution, which also handles non-uniform workload distributions over the environment, was later presented in [13]. More recently, a dynamic spanning tree coverage approach able to generate local modifications to an initial plan, but without taking into account energy constraints, has been presented in [20].

Energy consumption has been considered in the literature in relation to CPP problems as it represents a critical factor in real-world applications. However, variations in the energy consumption during the execution are never considered. Most works provide a static plan or restrict the energy aspect to a back-and-forth problem between coverage of an area with a limited energy budget and a charging station at a fixed location. Some approaches, such as [24], are purely offline whereas in [26] and [27], the authors propose a constant-factor approximation online algorithm valid for contour-connected areas. [23] addresses the same kind of setting of a robot and a charging station through an online cellular decomposition based on the obstacles shapes, saddle curves and corridor. A similar problem, but considering unknown environments, is studied in [7] and [22] with a particular attention to the return travel to the charging station for the latter and the number of recharges for the former. Contrary to our problem, these works make the assumption that the energy consumption throughout the mission is known, uniform and a fixed proportion of the distance traveled.

mCPP solutions with energy considerations in known environments are even more scarce: [15] considers multiple UAVs and a UGV for charging but again, the energy aspect is targeted on charging rendez-vous between the UGV and UAVS while minimizing path lengths. The algorithm also requires pre-assigning a partition of the area to the robots and is performed offline, which is incompatible with our context. [1] provides another offline multi-robot solution for a continuous space by transforming a coverage path planning problem into a line covering problem with some capacity constraints. In [5], a different kind of offline energy-sensitive solution is proposed, where the goal is to generate the trajectories of a fleet of aerial vehicles that minimize total energy consumption. But again all these offline mCPP methods consider fixed energy budgets not subject to variations. [18] is one of the only online multi-robot solutions but considers a set of homogeneous myopic robots to cover an unknown environment. Prioritizing the prevention of path conflict, the online centralized algorithm does not provide optimality guarantees or bounds. The same authors modified the algorithm in [19] to obtain additional theoretical guarantees. However, despite being online with a dynamic fixed horizon, they do not consider any energy aspect.

To the best of our knowledge, this work represents the first mCPP approach able to adapt the initial plan efficiently in response to unexpected changes in conditions such as energy consumption, operational area definition (new no-go zones and/or areas of interest), and number of robots in the team.

3 Dynamic mCPP Model

Limited energy in robotic missions is a critical efficiency factor where a defined amount of energy represents a quantitative measure of how much can be achieved by the involved robotic platforms. However, the expected energy capacity never corresponds to the actual tasks performed at the end of the mission. Typically, in uncertain environments, energy can be spent on collision avoidance, extra paths due to localization errors, fighting current flows in marine settings, and imperfect energy models. Under our coverage objective, we define the expected coverage (before launch) and the actual coverage achieved by the robots at the end of the mission. Our approach aims at minimizing the difference of the two.

3.1 Standard mCPP

We consider a set $R = \{r_1, r_2, ...r_n\}$ of n cooperative robots and a grid $\mathcal{G}$ of predefined square-shaped cells. The cell size corresponds to the smallest field of view of the sensors used by the robots and the goal is to cover the area $\mathcal{L}$, which is a finite subset of contiguous elements of $\mathcal{G}$. There is no restriction on the shape of $\mathcal{L}$, which can be non-convex. An obstacle is a cell in $\mathcal{G}$ that the robot cannot traverse. We call $\mathcal{B} \subset \mathcal{G}$ the set of forbidden cells in $\mathcal{G}$, and we have $\mathcal{L} \subseteq \mathcal{G} \setminus \mathcal{B}$.

A robot path $S_r = [(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)]$ is considered valid if (a) $\forall(x, y) \in S_r,\ (x, y) \in \mathcal{L}$ and (b) $(x_i, y_i), (x_{i+1}, y_{i+1}) \in S_r$ implies that (x_i, y_i) and (x_{i+1}, y_{i+1}) are adjacent. We call two paths intersecting if at least one cell is common to their respective cells set.

The original mCPP problem consists in determining the set of non-intersecting paths $S = \{S_r,\ r \in R,\ S_r \cap S_{r'} = \emptyset,\ \forall r, r' \in R\ r \neq r'\}$ that visit all the cells in $\mathcal{L}$, i.e. $\bigcup\limits_{r \in R} S_r = \mathcal{L}$. Thus, the mCPP problem can be described as:

$$\underset{S}{\text{minimize}} \quad \max_{r \in R} |S_r| \tag{1}$$

where $|S_r|$ denotes the length of the path S_r.

In [14], the authors propose a formulation of the mCPP problem based on optimal partitioning. The goal is to find an equal distribution of the cells to cover among the n robots, thus minimizing the following function:

$$F_1 = \frac{1}{2} \sum_{r \in R} (k_r - p_r)^2 \tag{2}$$

where k_r and p_r are, respectively, the portion of the total cells allocated to the robot r and its target allocation $p_r = \frac{1}{|R|}$.

3.2 Introducing dynamicity

The modeling of the dynamic conditions introduces a new set of challenges. Changes happening during the mission require distinguishing between areas where partial information was collected, unexplored zones, and forbidden areas. Preventing revisiting already inspected areas can lead the map to have disjoint components. Furthermore, accounting for online replanning requires reasoning about the computational load and the applicability of the approach. We propose a new model of the *dynamic mCPP* problem enriching the original mCPP model with the dimensions of used energy, partial information, and changes made to the goal of the mission during the execution.

The dynamic mCPP uses timestamped states to model the mission execution of a solution to a regular mCPP problem. Each state is defined by a timestamp k and a mission status (set of values for the mission variables) tuple $< k, R_k, \mathcal{L}_k, \chi_r(k), m_k, \eta_r(k) >$:

- t_k, $k > 0$ is the time when the mission status is obtained. These timestamps are not evenly distributed across the mission duration. The discrete set of these timestamps is $T = \{t_0, t_1, ...t_{end}\}$.
- $R_k = \{r_1, r_2, ...r_n\}$ is the set of operating robots at time step k. R_k can be different from or identical to R_{k-1} as robots can be added or removed.

- $\mathcal{L}_k$ is the area of interest updated with new information at the timestamp $t_k \in T$. $\mathcal{L}_k$ can be bigger (addition of a new cells from $\mathcal{G}$) or smaller (the set of forbidden cells has changed) than $\mathcal{L}_{k-1}$. Still, it remains a subset of $\mathcal{G}$, including all the cells where the robots in operation are currently positioned.
- $\chi_r(k) \in \mathcal{L}_k$ is the position of the robot r at time t_k.
- $\forall t_k \in T$, $k > 0$, $|m_k| = \sum_r |m_{r,k}|$ new cells have already been visited by the robots since the last step t_{k-1}, with $m_{r,k}$ being the individual ordered set of cells of $\mathcal{L}$ visited by the robot r since the previous step. We have $m_k \cap m_{k'} = \emptyset\ \forall k, k'$with $k \neq k'$. The full path of ordered visited cells performed by the robot r is $S_r = [m_0, m_1, ..., m_{end}]$. $m_0 = |R|$ because at the beginning of the mission the cells already covered are the ones where the robots start.
- $\eta_r : T \to \mathbf{R}^+$ is the function measuring each robot energy level. At the timestamp $t_k \in T$ each robot r has an energy level $\eta_r(t_k) \leq \eta_r(t_{k-1}) \leq \eta_r(t_0)$. For simplification, we will note $\eta_r(t_k) = \eta_r(k)$.

At the timestamp $t_k \in T$, $\mathcal{L}_k \cap m_{k-1} = \emptyset$. $\mathcal{L}_k$ is not necessarily compact but each component is compact and must contain the position of at least one robot.

For the dynamic mCPP, we adapt (2) to be based on the available energy of the robots. Originally, the function can be used with an unequal distribution p_r to reflect the heterogeneity in the team, as long as the constraint $\sum_r p_r = 1$ is respected. Thus, we introduce a new objective function F_2 to minimize:

$$F_2 = \frac{1}{2} \sum_{r \in R} (|S_r| - S_r^*)^2 \tag{3}$$

where S_r^* is the maximum number of cells that the robot r can visit, considering we have perfect knowledge of its future energy consumption and variation. $S_r = [m_0, m_1, ..., m_{end}]$ is the path effectively performed by the robot r during the mission. However, computing (3) requires perfect knowledge of the problem. In practice, $|S_r|$ and S_r^* cannot be known a priori, they are only known once the mission is completed. The function in (3) can be interpreted as the distance to the optimal distribution of cells to visit for the robots or, in other words, a measure of the unexploited energy. Note that, when the available energy is evenly distributed between the robots and no unexpected variations in the energy consumption occur during a mission with static forbidden cells, (3) and (2) are identical because the dynamic mCPP would only have one timestamp and hence coincides with a classical mCPP. Although in general the function (3) cannot be computed before the mission ends, we will show how it can be approximated by using the available information multiple times during the mission.

Theoretically, the timestamps state space is infinite for one mission. However, in practice, we only want to consider timestamped states that induce a deviation considered significant enough to trigger a replanning.

4 Solution Approach

Our proposed approach allows the multi-robot system to react dynamically to disruptions in the mission plan and to efficiently modify the paths. Reconstructed trajectories both maximize coverage based on available energy and minimize changes from initial trajectories despite the perturbations. This approach supports predictability, which is a key factor in the success of robotic missions. Trying to minimize energy use and maximize coverage leads us to avoid revisiting the same areas of the map. However online replanning respecting the non-backtracking constraint may produce a map with disconnected uncovered components separated by already visited areas, which requires a more sophisticated coverage strategy than existing ones. The proposed algorithm is hierarchical and divided in three successive stagess. First, we use an updated version of the partitioning algorithm used in the traditional DARP algorithm. Secondly, we provide an adaptation of the classical Kruskal algorithm to build new paths upon the existing ones, preventing revisiting already covered areas. Finally, a repair strategy deals with disjoint portions of the maps created by forbidden areas or non-revisit constraints.

4.1 Architecture

Our approach is designed to adapt the coverage paths plan online based on the feedback from the execution and the new environmental conditions. Thus it is dependent on an external architecture handling the messages between the robots and our algorithm. We assume that the general status of the system (energy available, robot status, change in the map) can be measured at any time. Our algorithm is designed in the context of a marine multi-robot project for infrastructure inspection and relies on message updates sent directly by the robots to a central station (our algorithm) that plans the operation offline and oversees their executions by the robot teams. Figure 2 presents the architecture of the system. Robot status information (position, energy level, system failure) is encapsulated in a message status update sent by the robots individually at predefined regular intervals. A set of independent dedicated modules called reactors located on the central station, analyze the incoming messages and check for any possible energetic deviations or robot failure (more information can be found in [21]). If the threshold of what is an acceptable deviation from the original plan is crossed, a new timestamp is created in the dynamic mCPP model. The creation of a new timestamp can also be triggered by human intervention through the GUI if new conditions arise (variations in the operational area or changes in the robot team). We refer to the k-generated subproblem as the subproblem associated with the dynamic mCPP at the new timestamp t_k. The latest timestamp is integrated into the knowledge base through an update of a PDDL (Planning Domain Definition Language) [8] problem. Once our algorithm converges to a solution, it is used with the knowledge base and a PDDL domain description to produce a robot-understandable action sequence.

The algorithm is designed to solve a sequence of k-generated subproblems, each describing an evolution of the previous configuration. To solve it, we follow the DARP approach of partitioning first and then planning the paths. Thanks to the Kruskal algorithm [25], an optimal polynomial solution exists for single-agent coverage path planning. Thus, instead of reasoning on the paths of the

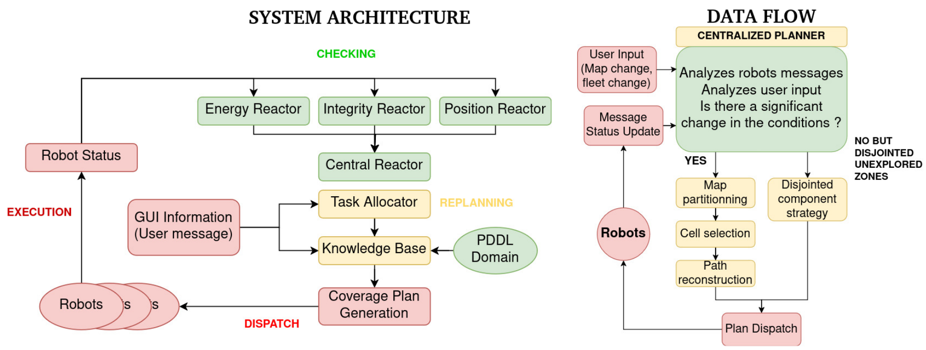

Fig. 2. Full system architecture (left) and data flow (right)

multiple robots, the focus is on optimally partitioning the area and then finding the individual Minimum Spanning Trees over each area to reach a solution. The final path will circumnavigate around the Minimum Spanning Trees. If the zone of operation is updated, we use an adapted version of the DARP partitioning method to provide a cell allocation. Tailored strategies are used for cell selection (when allocations exceed the robots' energy capacity) and for unconnected areas. Finally, paths are created using an ad-hoc version of Kruskal's algorithm to conserve the shape of the initial paths at each iteration.

4.2 Partitioning process

Goal distribution Within DARP, the cell distribution is made based on a modified Voronoi partitioning. Cells are rated according to a metric considering the distance to robots stored in an assignment matrix. Then, the assignment matrix is updated using cyclic coordinate descent on (2) to converge to a fair partition, while at each iteration disconnected components within temporary partitions are penalized. To generate the new partition at each timestamp, we also use cyclic coordinate descent for the assignment matrix but we adopt relaxed convergence criteria and a fine-tuned objective function for the k-generated subproblem to incorporate the updated robots' energy levels. Due to the use of Kruskal's algorithm that finds the minimum spanning tree on the grid, each existing cell is subdivided into four equal subcells. The path that circumnavigates the minimum spanning tree is a sequence of these smaller size cells. In our framework, a cell of the original grid map is called covered when all four of its subcells have been visited. If at least one of these subcells has been visited, but not all, the cell is called partially covered. We define $\tilde{m}_r \in m_r$ as the set of cells partially covered by robot r.

At each timestamp, partially covered cells are excluded from the pool of cells to be redistributed, as we want them to remain assigned to the robot that visited them first and already has partial information. The segments related to these cells in the minimum spanning tree of the previous plan are preserved and fed into the Kruskal algorithm as the initial state for the new path solution. Our assignment formula reflects that and we define:

$$p_{r,k} = \begin{cases} \frac{C_{r,k}}{\sum_{r=0}^{n} C_{r,k}} \cdot (m - m_r) & \text{if } \sum_{r=0}^{n} C_{r,k} \geq (m - m_r) \\ C_{r,k} & \text{else} \end{cases} \tag{4}$$

where $C_{r,k} = \eta_r(t_k) - \tilde{m}_{r,k}$ with $\tilde{m}_{r,k}$ the partially covered cells by the robot r at the timestamp t_k.

Optimum and convergence criteria An optimal solution to the mCPP problem can be defined as a partition $L_1, L_2, ..., L_n$ of $\mathcal{L}$ satisfying:

- 1: $L_r \cap L_{r'} = \emptyset$, $r, r' \in 1, .., n$ $r \neq r'$
- 2: $L_1 \cup L_2 \cup ... \cup L_n = \mathcal{L}$ (completeness)
- 3: $\mid L_1 \mid \approx \mid L_2 \mid \approx ... \approx \mid L_n \mid$ (fairness)
- 4: L_r is connected $\forall r \in 1, .., n$
- 5: $\chi_r(t_0) \in L_r$

These criteria can be used both as a definition of the optimal solution of the mCPP problem and as termination requirements for the partitioning process. In our case, we adopt a relaxed version of these criteria to provide a new definition of the solution for the k-generated subproblems.

Firstly, when L_r is not compact, we define the i-th non-connected components $\ell_{r,i}$ such that $\cup_i \ell_{r,i} = L_r$ and $\cap_i \ell_{r,i} = \emptyset$. $\ell_{r,0}$ is the component containing the starting position at this timestamp, i.e. $\chi_r(t) \in \ell_{r,0}$. We call $\ell_{r,0}$ the main component of L_r. These non-connected components exist only when the algorithm has not yet converged: when convergence is reached, the partition of the cells set is composed of compact parts.

In our context, the previous criteria concerning completeness and fairness are no longer adequate to the problem. The remaining energy of the robots may be insufficient to cover the whole area if one or more robots consume more energy than what was initially expected. Even in case of under-consumption, the area of interest could also have been extended by adding more cells. As a result, the variation in remaining energy across all robots does not allow for a fair distribution either. We thus establish the following new set of criteria to define a satisfying solution for the dynamic mCPP.

- 1: $L_r \cap L_{r'} = \emptyset$, $r, r' \in R$ $r \neq r'$
- 2: $\forall r \in R, \mid \ell_{r,0} \mid \geq C_{r,t}$
- 3: $\ell_{r,0}$ is connected $\forall r \in R$
- 4: $\chi_r(t) \in \ell_{r,0}$

Except for the backtracking property, which remains identical, we relaxed and adapted the other criteria to fit our domain and allow faster convergence. This relaxation is justified by the behavior of the algorithm: the partitioning aims at converging to a fair distribution between robots while penalizing non-connected components of a partition. Typically, the unconnected secondary components are progressively discarded or linked to the main one. In our context, not all the cells need to be distributed in a partition as not all of them will be covered due to insufficient energy capacity. Thus, we restrict ourselves to consider only the cells that can be covered by each robot with respect to the available energy and exploit this feature of the problem to speed up convergence.

4.3 Cell selection process

Once the partitioning has converged to an allocation L_r, if the number of cells assigned to a robot is superior to the energy capacity, a selection process is applied to determine the final set of cells $\bar{L}_r$ such that $|\bar{L}_r|$ equals to the number of cells selected for the robot r. To define the selection process, we consider two different types of coverage objectives: the first is the extensive exploration of an area for mapping or precise data collection, and the other is to localize an object or a person (typically in search and rescue applications). In both scenarios, ideally, all cells are visited, but when this is not possible (as in energy insufficient capacity), a different priority order can be devised. This is reflected in the following function used for the cell selection process:

$$U = \sum_{c \in \bar{L}_r} d(c, M(\bar{L}_r)) \tag{5}$$

where d is the Euclidean distance and $M(\bar{L}_r)$ the center of mass of the set $\bar{L}_r$. Depending on the application, the function U can be maximized or minimized either to increase the spatial spread of the paths or to make the paths as compact as possible respectively. To do so, all cells are temporarily considered and sorted by distance to the center of mass. Then, in the minimization case, cells that are not necessary to the graph connectivity are discarded greedily in the sorting order. In the maximizing case, the set of selected cells starts with the robot position and expands by the greedy addition of the furthest connected cell.

4.4 Path reconstruction

Once the set of cells assigned to each robot has been refined, the coverage paths are built for each robot independently. We extend the algorithm to build the minimum spanning tree to take into account partially covered cells and to minimize the change of shape of the original planned paths. The first step is to extract and sort the edges of the spanning tree from the previous timestamp t_{k-1} depending on their type. Edges that touch fully covered cells are discarded while the ones fully included in partially covered cells or that only touch them on one side are respectfully inserted in an inside edge and border edge sets. Edges fully included in unexplored portions of the map are called outside edges. Inside edges are used as the initial set of edges for the new tree as it corresponds to the part of the trajectory that is incomplete and brings the robot back to its initial location. Border edges, outside edges, and the rest of the edges of the map are inserted in that order in the edges queue. Then the Kruskal algorithm

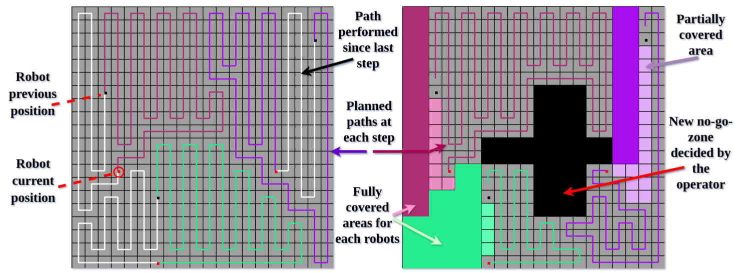

Fig. 3. Evolution of the paths of 3 robots between 2 timestamps when the operator introduces a new no-go zone in the middle of the map.

is performed drawing edges from the queue. This imposes that the tree solution of t_k is built giving priorities on the edge that were used for the tree of t_{k-1}, ensuring minimal changes in the shape of the tree even in the case of significant modification in the cells set to visit. It also ensures that partially covered cells will be visited again accordingly and that the end position of the path stays at the original starting position of the mission. A visual example of how the algorithm works can be seen in figure 3, where the operator introduces a new no-go zone mid mission and the paths of the robots are adapted online, retaining their shape, accounting for pre-covered cells, and respecting the original end position.

4.5 Disjoint component dealing strategy

Specific geometric configurations and the constraints on previously visited cells can lead to situations where uncovered parts of the maps are inaccessible to one or more robots. To prevent these portions of the map from being left unexplored, the unconnected components are temporarily discarded in the cells allocation process until at least one robot has no unexplored cells accessible. A robot in this status is called available. When a robot is available, the shortest path through the already covered cells to the closest unconnected component is computed through a wavefront algorithm. If its energy is sufficient, the available robot will then travel to the unexplored component of the map. Once this robot reaches the component, a new k-generated subproblem/timestamp is created with the new robot's position. If an available robot finds no path to an unconnected component, it stays available until the next replanning. This process is repeated until all cells are covered, robots have no energy left or robots do not have enough energy to travel to an uncovered disconnected region.

5 Simulation Results and Analysis

In this section, we evaluate the effectiveness of our approach in solving dynamic mCPP problems. We first focus on a qualitative assessment. We provided two visual examples of how the algorithm behaves in two mid-mission situations: first a robot failure (see figure 1) and second a no-go zone introduction decided by

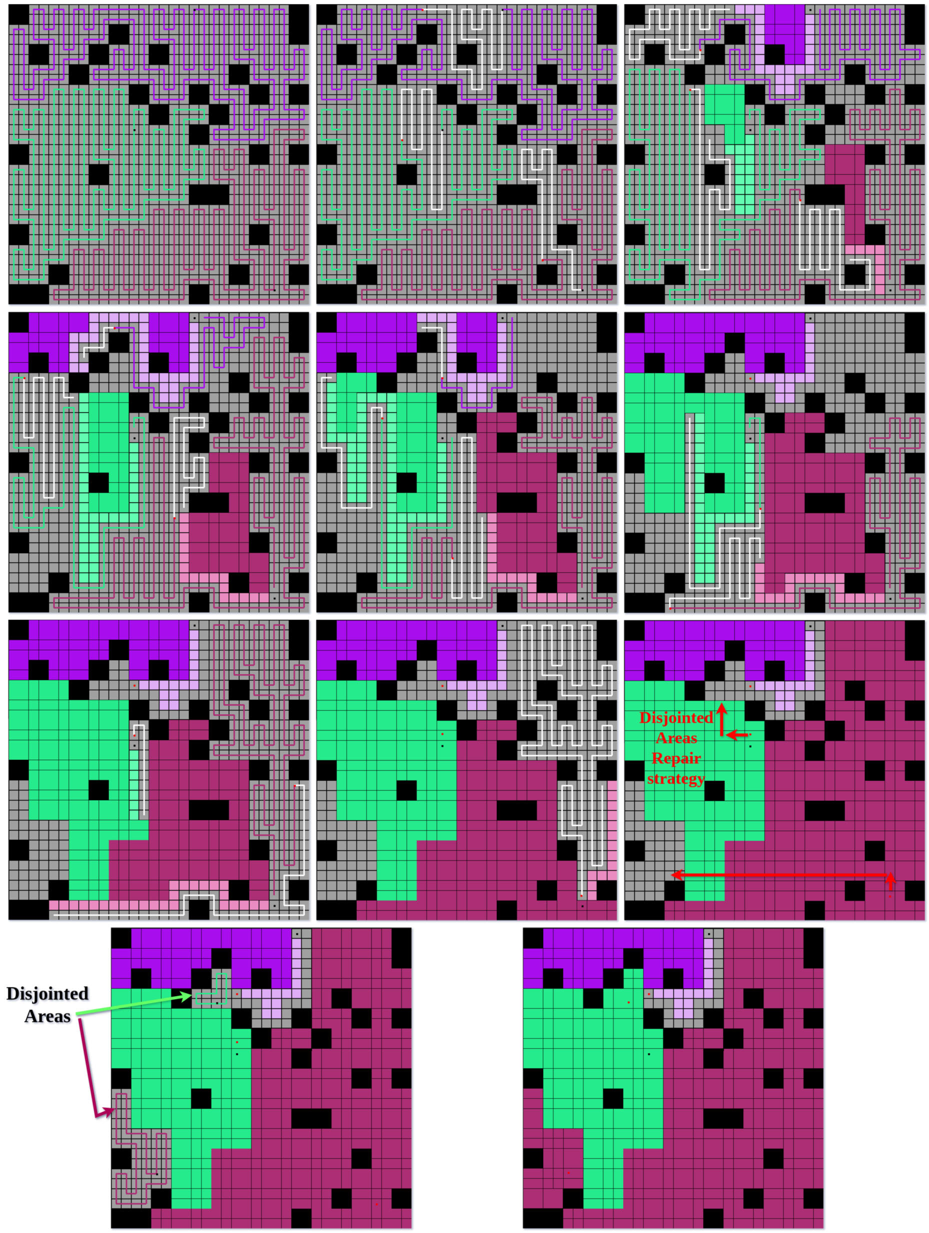

Fig. 4. Evolution of the paths of 3 robots from the initial plan until the final coverage. Every image represents a timestamp with the corresponding planned paths based on the current conditions, the performed path since the last stamp and pre-covered/fully-covered cells.

the operator (see figure 3). In figure 1, the paths of robot 1 and robot 2, in excess of energy, take over the portion of the map that robot 3 was supposed to visit because it ran out of energy, while in figure 3 our algorithm immediately adapts to the presence of a massive new forbidden area. These scenarios demonstrate the quality of our approach in terms of robustness and predictability: in two very typical use cases, the algorithm modifies the robots' trajectories online while conserving the original mission constraints (starting/end position). These qualities are also validated in a comprehensive step-by-step evolution of a full replanning scenario that we provide in Fig. 4, which shows the changes in the paths during a 3-robot mission. During the mission, the allocated parts of the map expand or contract on the basis of the fluctuation of the robots' energy consumption. Despite the number of replanning and the robots moving away from their original position, both the partition and the paths retain a similar shape. One robot has a very high consumption and a slower speed, which explains why the other robots take over its area in the course of the replanning. In the last images, we can notice a disjoint portion of the map, which is dealt with in the last stage of our algorithm (as explained in Section 4.5).

For a more quantitative analysis, we initially evaluate our convergence criteria for the partitioning stage. We created 1000 instances of the same grid size with 5 robots positioned at random and a rate of 3 to 7% obstacle cells. The energy capacity is generated randomly but we consider cases where it is insufficient to cover the whole map. The number of iterations until convergence of the partitioning algorithm is averaged over the 1000 different instances for our approach and the traditional DARP algorithm. We consider the algorithm fails for an instance if no optimal partition (each algorithm has its own optimality defined in Section 4.2) is found in 100000 iterations. Considering that DARP is designed to find solutions where all cells are allocated, we normalize the energy distribution so that the total energy capacity corresponds to the total amount of cells. The results, presented in Table 1 for different grid sizes, prove that our approach converges at least one order of magnitude faster in complicated small instances and that the gain scales with the size of the map. Our approach also has a better success rate, even in the smallest grid and dense instances.

Providing extensive comparative results is in our case difficult given the absence of state-of-the-art algorithms able to address the dynamic mCPP problem. To circumvent this problem and analyze the performance of our solution, we establish a comparative evaluation against a conventional mCPP approach, the DARP algorithm [14], adapted with two different strategies to face the online context. These strategies, called DARP-OR (Online Revisit) and DARP-ON (Online Non-revisit), will be later detailed. To show the importance of online replanning, we also compare our solution to the static offline plan provided by the original DARP algorithm. Designing a comprehensive comparative evaluation is, however, challenging due to the wide range of potential context changes (some potential examples are provided in figures 1 and 3). In the case of map variations, such as an expansion of the coverage area, the original DARP offline plan tends to perform poorly depending on the size of the added zone. On the other

Gride size	10x10	20x20	30x30
DARP	14369.7	9743.0	7016.4
	92.8 %	94.6%	95.7 %
Ours	**1069.7**	**655.7**	**40.6**
	98.7 %	**99.2%**	**100%**

Table 1. Mean iterations count and success rate for 5 robots and different grid sizes. Results correspond to the mean over 1000 runs, each one corresponding to random robots' initial positions and environment configurations.

hand, when new no-go zones are introduced, the offline plan is simply inadequate to adapt to these new conditions. Therefore, we focused our evaluation on scenarios involving energy over-consumption and reduced speed. To generate these scenarios, we perturbed the behavior of one robot in the fleet with respect to its expected behavior. We then compared our approach to both the unmodified execution of the original static offline plan and the re-execution of the DARP algorithm at each timestamp with updated initial conditions. Since the traditional DARP algorithm is not inherently designed for continuous replanning, we employed two possible adaptations: one treating pre-covered cells as forbidden cells (DARP-ON) and the other treating pre-covered cells as uncovered (DARP-OR). The n robots start the mission each with $\frac{1}{n-1} \cdot |\mathcal{L}|$ units of energy where one unit represents the amount required to cover one cell of the map. Thus, each robot has the initial energy capacity to explore $\frac{1}{n-1}$ of the map. The k timestamps are uniformly sampled during the execution and the robot is expected to move along its trajectory for a pre-defined distance of $\frac{1}{k \cdot n} \cdot |\mathcal{L}|$ and consume one unit of energy per cell traveled. To model the perturbations, the distance traveled and the energy consumed between each timestamp are obtained by multiplying the expected traveled distance and the expected energy consumed by a traveling factor α and a consumption factor β respectively. In our instance, when the robot has normal behavior, α and β are both limited to a maximum variation of 5%. When the robot has abnormal behavior, we set $\beta = 1.5$ and α is randomly generated between 0.25 and 0.75. Fig. 5 shows that our approach clearly outperforms the other alternatives. We can note a significant performance gain over the original DARP offline plan and DARP-ON, the latter being rapidly stuck in one of the disjoint components created by the non-revisit constraints. The difference is smaller for DARP-OR, as the performance loss accumulates over time since robots have no memory of pre-covered cells.

6 Conclusion

This paper introduced a novel algorithm devised to adapt to evolving conditions encountered during a multi-robot coverage mission. Specifically, we presented a new problem formulation along with a modified version of a state-of-the-art mCPP approach able to take into account dynamic energy constraints and potential changes in the environment and the system. We showed how our proposed solution can effectively leverage updated information to refine online the ini-

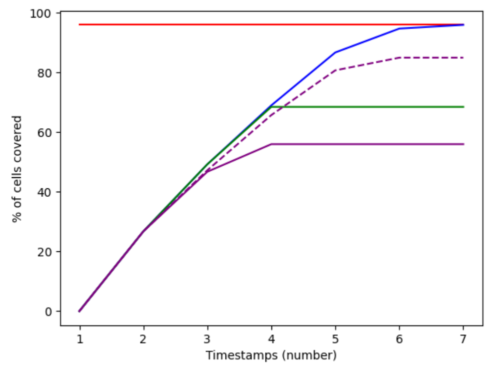

Fig. 5. Evolution of covered cells for a 3-robot team where only 2 out of 3 robots behave as expected. Optimum is in red, our solution in blue, purple is DARP-OR (dashed line) and DARP-ON, and offline DARP in green. Average on 200 instances.

tial plan and obtain an improved overall performance by minimizing redundant coverage with already visited areas and optimizing the total team energy consumption. Comparative results finally demonstrated that this approach outperforms alternative strategies also in terms of success rate and number of iterations needed to converge. In the future, we intend to further improve the strategy by linking disconnected areas already at the partitioning stage.

Acknowledgment

This work has been partly supported by the Agence Nationale de la Recherche (ANR), grant ANR-22-CE33-0004.

References

1. Agarwal, S., Akella, S.: Area coverage with multiple capacity-constrained robots. IEEE Robotics and Automation Letters **7**(2), 3734–3741 (2022)
2. Agmon, N., Hazon, N., Kaminka, G.A.: Constructing spanning trees for efficient multi-robot coverage. In: IEEE International Conference on Robotics and Automation. pp. 1698–1703 (2006)
3. Cho, S., Park, H., Lee, H., Shim, D., Kim, S.Y.: Coverage path planning for multiple unmanned aerial vehicles in maritime search and rescue operations. Computers & Industrial Engineering (2021)
4. Choset, H.: Coverage of known spaces: The boustrophedon cellular decomposition. Autonomous Robots **9**, 247–253 (2000)
5. Datsko, D., Nekovar, F., Penicka, R., Saska, M.: Energy-aware multi-uav coverage mission planning with optimal speed of flight. IEEE Robotics and Automation Letters (2024)
6. Dogru, S., Marques, L.: Improved coverage path planning using a virtual sensor footprint: a case study on demining. In: IEEE International Conference on Robotics and Automation. pp. 4410–4415 (2019)

7. Dogru, S., Marques, L.: ECO-CPP: Energy constrained online coverage path planning. Robotics and Auton. Systems **157** (2022)
8. Fox, M., Long, D.: PDDL2.1: an extension to PDDL for expressing temporal planning domains. CoRR **abs/1106.4561** (2011)
9. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Annals of mathematics and artificial intelligence **31**, 77–98 (2001)
10. Galceran, E., Carreras, M.: Efficient seabed coverage path planning for ASVs and AUVs. In: International Conference on Intelligent Robots and Systems (2012)
11. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous Systems **61**(12), 1258–1276 (2013)
12. Hazon, N., Kaminka, G.A.: Redundancy, efficiency and robustness in multi-robot coverage. In: IEEE International Conference on Robotics and Automation (2005)
13. Idir, O., Renzaglia, A.: Multi-robot weighted coverage path planning: a solution based on the DARP algorithm. In: 17th International Conference on Control, Automation, Robotics and Vision (ICARCV) (2022)
14. Kapoutsis, A.C., Chatzichristofis, S.A., Kosmatopoulos, E.B.: DARP: divide areas algorithm for optimal multi-robot coverage path planning. Journal of Intell. & Robotic Systems **86** (2017)
15. Karapetyan, N., Asghar, A.B., Bhaskar, A., Shi, G., Manocha, D., Tokekar, P.: AG-CVG: Coverage planning with a mobile recharging UGV and an energy-constrained UAV. In: IEEE International Conference on Robotics and Automation (2024)
16. Kumar, K., Kumar, N.: Region coverage-aware path planning for unmanned aerial vehicles: A systematic review. Physical Communication p. 102073 (2023)
17. Mannadiar, R., Rekleitis, I.: Optimal coverage of a known arbitrary environment. In: IEEE International Conference on Robotics and Automation (2010)
18. Mitra, R., Saha, I.: Scalable online coverage path planning for multi-robot systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2022)
19. Mitra, R., Saha, I.: Online on-demand multi-robot coverage path planning. In: IEEE International Conference on Robotics and Automation (2024)
20. Mo, W., Lin, Z.: MDSTC: A dynamic approach to multi-robot coverage path planning. In: 18th International Conference on Control, Automation, Robotics and Vision (ICARCV). pp. 1100–1105. IEEE (2024)
21. Salavasidis, G., et al.: Squads of adaptive robots (soar): Project overview and simulation results. OCEANS (2023)
22. Sharma, G., Dutta, A., Kim, J.H.: Optimal online coverage path planning with energy constraints. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. p. 1189–1197. AAMAS '19 (2019)
23. Shnaps, I., Rimon, E.: Online coverage of planar environments by a battery powered autonomous mobile robot. IEEE Transactions on Automation Science and Engineering **13**(2), 425–436 (2016)
24. Strimel, G.P., Veloso, M.M.: Coverage planning with finite resources. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2950–2956 (2014)
25. Tarjan, R.: Data structures and network algorithms. SIAM (1983)
26. Wei, M., Isler, V.: Coverage path planning under the energy constraint. In: IEEE International Conference on Robotics and Automation. pp. 368–373 (2018)
27. Wei, M., Isler, V.: A log-approximation for coverage path planning with the energy constraint. In: International Conference on Automated Planning and Scheduling. vol. 28, pp. 532–539 (2018)
28. Zheng, X., Jain, S., Koenig, S., Kempe, D.: Multi-robot forest coverage. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2005)