2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Multi-Goal Path Planning in Cluttered Environments with PRM-Guided Self-Organising Maps

Benjamin R. Davis, Edward Bray, and Graeme Best

***Abstract*— We consider the problem of multi-robot, multi-goal path planning in cluttered environments, motivated by scenarios including surveillance, object search, and package delivery in crowded office spaces and urban environments. While many solutions have been proposed for related vehicle routing problems, they typically do not generalise well for cluttered environments with obstacles due to the introduction of non-Euclidean point-to-point distances. We consider a Self-Organising Map (SOM) algorithm due to its versatility in optimising waypoints within region-based goals. Since standard SOM heavily relies on Euclidean distance-based operations, we propose a generalised SOM with several new innovations: Probabilistic Roadmap (PRM)-guided adaptation and winner selection rules, a two-level path representation for effective routing between goals, and caching operations to overcome the increased computational demands. We present simulation experiments in office and maze environments with one to three robots that show that our approach significantly outperforms standard SOM algorithms as it explicitly reasons over collision avoidance. These results demonstrate the viability of our PRM-guided SOM algorithm for tasks including surveillance in cluttered environments.**

I. Introduction

Many multi-robot tasks including surveillance [1], target search [2], and package delivery [3] can be formulated as multi-goal path planning problems. These problems involve reasoning over the assignment of goals to robots, efficient goal ordering, and selection of waypoints within goal regions. To compound the difficulty of multi-goal path planning, many real-world environments are cluttered, meaning they contain obstacles that robots must safely navigate around. Furthermore, utilising several agents in a multi-robot system could increase the speed at which goals are visited through parallel operation, but only if they can coordinate effectively. To fully realise the potential of multi-robot systems in real-world applications, the robots should be able to coordinate to visit goal regions in a computationally-efficient manner, while explicitly reasoning over obstacles in the environment.

Multi-goal path planning is akin to vehicle routing problems including the Travelling Salesman Problem (TSP), Orienteering Problem (OP), and their numerous variants and generalisations [4]. For special cases where Euclidean distances can be used to measure the cost of travelling between goals, many effective heuristic algorithms exist [5]. The *Self-Organising Map* (SOM) has emerged as one such algorithm that is particularly well-suited to scenarios with region-based (rather than point-based) goals [6], which is highly relevant for robots performing range sensing [7], for example. SOMs have also been shown to generalise well for scenarios with online-discovered goals [7], decentralised coordination [8], and prize collecting [9]. The SOM learning procedure involves cycling between: assigning waypoints to goal regions, adapting waypoints using a topological-distance measure, waypoint regeneration for efficient routing, and cooling the learning over subsequent iterations.

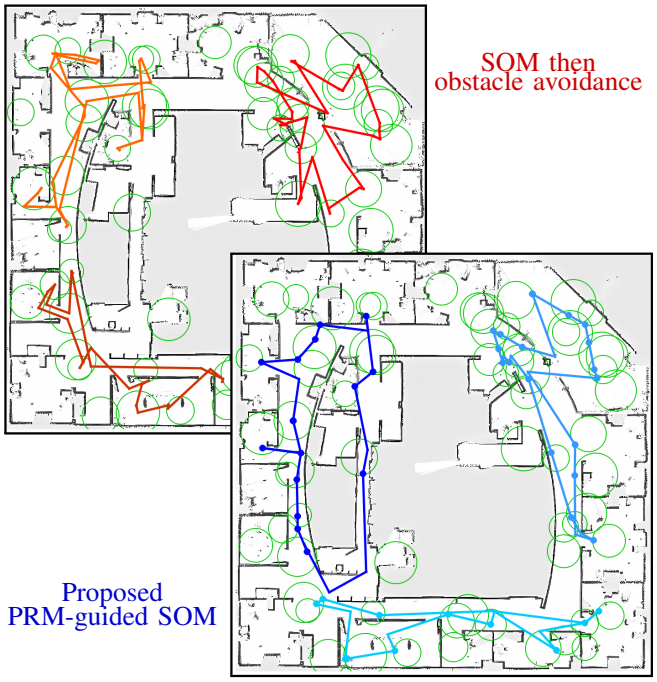


Fig. 1. Illustration of the multi-robot multi-goal path planning problem in a cluttered office environment with two different solutions. The SOM plans paths for three robots to visit the most goal regions (green circles), while avoiding obstacles and with constrained path lengths. Standard SOM does not consider obstacles, and therefore obstacle avoidance is considered afterwards (red), resulting in inefficient and over-budget paths. Our proposed PRM-guided SOM explicitly incorporates obstacle avoidance within the learning procedure, yielding higher reward and efficient routing.

However, in cluttered environments such as office spaces and urban environments, SOM algorithms fail due to the heavy reliance on Euclidean distance-based operations that do not generalise to obstacle-rich environments. This is clearly evident in the example shown in Fig. 1, where failing to adequately consider path distances results in inefficient back-and-forth behaviour over the environment. In general, these multi-goal problems are considered to be

This research is supported in part by the Centre for Advanced Defence Research in Robotics and Autonomous Systems, Australia.

Authors are with the Robotics Institute, University of Technology Sydney, NSW, Australia. benjamin.r.davis-1@student.uts.edu.au, {edward.bray, graeme.best}@uts.edu.au.

fundamentally harder than the Euclidean special cases [10]. Approximate algorithms exist that consider goal points in cluttered environments [11], [12], [13], [14], [15], but to date no algorithm has been developed that considers multi-goal path planning with goal regions in non-Euclidean space.

We propose a generalised Self-Organising Map algorithm specifically designed for multi-robot, multi-goal path planning in cluttered environments. Our key innovation is to introduce a Probabilistic Roadmap (PRM) [16] to guide every step of the SOM algorithm in place of all Euclidean distance-based operations, especially the winner selection, path adaptation, and path regeneration steps. To facilitate the integration between SOM and PRM, we also introduce a two-layer path representation to enable the algorithm to jointly reason over goal selection and route between goals. We also propose caching of repeated operations to improve computational efficiency. Our algorithm enables centralised coordination of multi-robot paths through the waypoint–goal assignment step, without requiring predefined or explicit partitioning of the environment. The algorithm is anytime, converges in polynomial time, and the convergence rate can be tuned via user-specified constants.

Our proposed PRM-guided SOM algorithm is tested in a range of scenarios with three maps, including a real-world map of a cluttered office [17]. We show that our algorithm is able to converge to a good set of robot paths with reasonable computational cost. Our algorithm is compared against other SOM-based approaches and is found to plan shorter paths while efficiently navigating around obstacles and visiting more goals than other methods. This shows the viability of our method for multi-goal, multi-robot path planning in cluttered environments with region-based goals.

II. Related Work

The TSP poses the problem of finding the shortest route that visits a given set of goals. A multiplicity of TSP variants have emerged as common problems in robotics [4]. The OP is closely related to the TSP, but instead requires maximising the number of visited goals while being constrained to a maximum path length [18].

Of particular interest in robotics scenarios are TSP/OP variants where each goal is defined by a set rather than a single point. This class of problems provides options for how a robot may fulfil each goal, such as on object of interest may be observed from many possible viewpoints [7]. One such variant is the Generalised TSP [19], where each goal is a finite set of points, only one of which should be visited. A related challenge is to consider continuous goal regions, as is done in the Travelling Salesman Problem with Neighbourhoods (TSPN) [20] and the Close Enough Orienteering Problem [21].

SOMs were first applied to the TSP in [22], but are generally less efficient than other heuristic methods for the standard TSP [23]. However, they have become one of the leading methods for multi-goal path planning with continuous goal regions due to their inherent ability to adapt in continuous space. An algorithm for solving the TSPN with varying rewards for each goal using SOMs was proposed in [24], and later generalised to a multi-robot case with limited travel budgets [7]. However, these approaches are only applicable in obstacle free environments.

Multi-goal path planning in cluttered environments is considerably harder than in obstacle-free spaces as it requires reasoning over non-Euclidean space [10]. Several algorithms have been developed that address this problem for point-goals, such as by growing multiple search trees from each goal [25], combining variable neighbourhood search with PRM* [11], and using a TSP solver in conjunction with LazyPRM* [12].

For multi-goal path planning with goal regions, SOMs are a promising solution, although they are challenging to apply to non-Euclidean spaces. One way of using SOMs in such scenarios is to approximate the robot workspace with polygons, which has shown promise for a standard TSP formulation with obstacles [14], as well as the multi-robot TSP with minmax objective [13]. Another approach is to use rapidly-exploring random graphs (RRGs) [26] to find the paths around obstacles between goal regions and use this to influence the SOM [15]. This has been shown to be effective, but the RRG algorithm must be called frequently to find paths between locations, so is computationally expensive. In all of these cases, only goal points are considered instead of goal regions, despite the inherent suitability of SOMs to the latter case. We build on these ideas to explicitly plan for goal regions by guiding the SOM with a PRM, which admits efficient multi-query path planning, and effective caching of paths found with A* for later reuse.

III. Problem Formulation

We consider a multi-robot, multi-goal path planning problem with goals defined as regions and an environment with obstacles that must be avoided. We formally define this problem as follows; we follow a similar formulation to [7] with the main difference being the introduction of obstacles.

We consider a team of R robots $\mathcal{R} = \left\{r^1, r^2, ..., r^R\right\}$ operating in a bounded environment $\mathcal{X} \subset \mathbb{R}^2$. The environment $\mathcal{X}$ is divided into free space, $\mathcal{X}_{free}$, and space occupied by an obstacle, $\mathcal{X}_{occupied}$. The path of robot r^i is a sequence of waypoints within this environment $X^i = \left(x^i_1, x^i_2, x^i_3, ..., x^i_1\right)$ where $x^i_j \in \mathcal{X}_{free}$. Each robot begins in a random location x^i_1, where it also finishes to form a loop, although this constraint can be readily relaxed. Robots are assumed to travel in a straight line between consecutive waypoints, so therefore a path is feasible only if all line segments bounded by each pair of consecutive waypoints $\left(x^i_j, x^i_{j+1}\right)$ are entirely within $\mathcal{X}_{free}$. The path cost c^i is defined as the sum of the cost of all pairs of consecutive waypoints, with the cost of each pair defined as the Euclidean distance between the two waypoints. A path is feasible only if the path cost c^i does not exceed the cost budget $b^i > 0$ of the robot.

There also exists a known set of goal regions $\mathcal{G} = \left\{g^1, g^2, ..., g^G\right\}$, with $g^k \subset \mathcal{X}, \forall k$. In this work we assume that each goal region is a circle with known centre and radius, although other definitions such as polygons can be readily

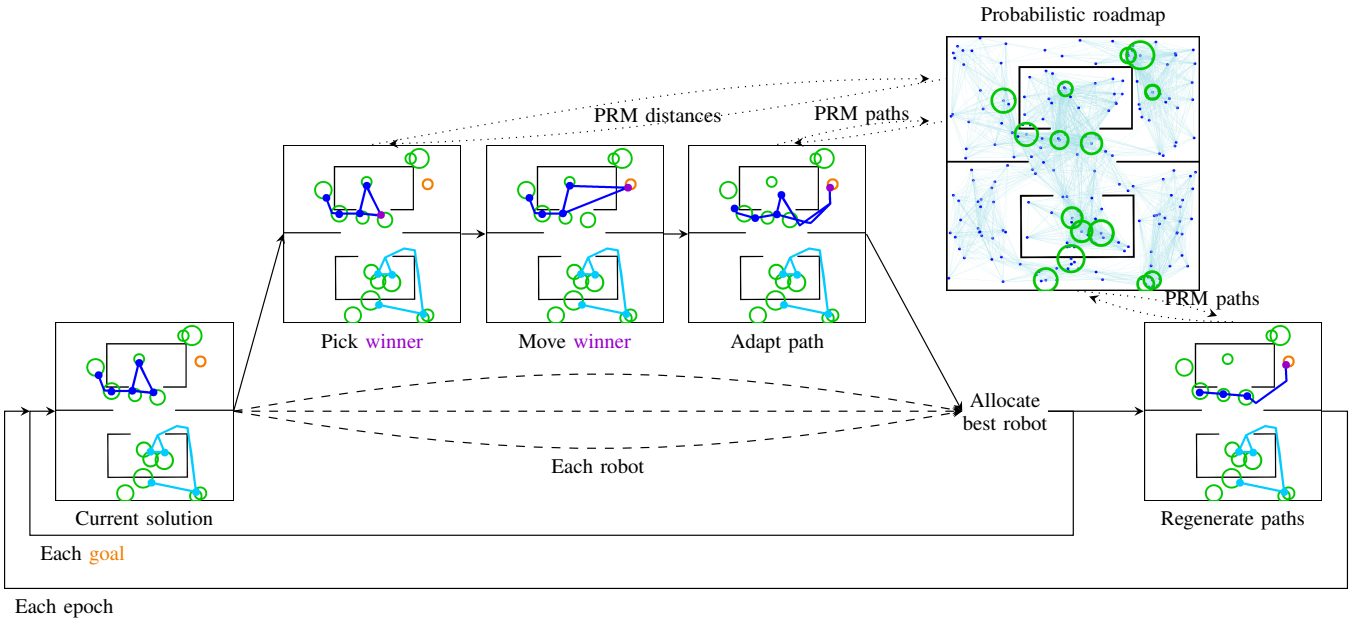


Fig. 2. Overview of the proposed PRM-guided SOM learning procedure, as described in Sec. IV-A. The paths of several robots (blue) are optimised to maximise the number of goal regions (green) visited while avoiding obstacles and respecting a path length budget. In each *epoch*, each goal is considered in turn (orange), and the effect of allocating each robot to it is considered. The closest node on the robot's current path (the *winner*, purple) is found using a *probabilistic roadmap* (PRM) and moved to the this goal, then the path is *adapted* towards it along the PRM. At the end of each epoch, paths are *regenerated* for efficiency, using the PRM for obstacle avoidance.

admitted [7]. A goal region g^k is visited by robot r^i if one or more of its waypoints are within the goal region. A goal region g^k is visited by the team if the goal region is visited by one or more robots; this is denoted by a binary indicator variable $o^k \in \{0, 1\}$, with 1 meaning visited and 0 meaning not visited.

Given these definitions, the problem is posed as follows: find the set of *feasible* paths $\{X^i\}$ for the team of robots that maximises the collective number of goal regions visited, $\sum_{g^k \in \mathcal{G}} o^k$. The paths must be feasible such that the above-mentioned obstacle avoidance and budget constraints are satisfied. This problem differs from the TSP family as not all goal regions must be visited.

IV. PRM-Guided Self-Organising Map

We propose a new Self-Organising Map algorithm that is guided by a PRM and is specifically designed for multi-goal planning in cluttered environments. This section describes our proposed algorithm, beginning with an overview in Sec. IV-A, followed by a detailed description of all components, and a brief analysis of computational complexity.

A. Algorithm Overview

1) Key Steps: An illustration of the algorithm is provided in Fig. 2 and pseudocode is provided in Alg. 1. Each robot is initialised with a random path, and in each iteration of the outer loop (*epoch*) the paths are improved until they converge to a solution. The improvement process involves iterating through each goal and considering the effect of allocating each robot to it in turn. The closest waypoint (termed the *winner*) to the goal for each robot is found and moved to the goal, then the rest of this robot's path is adapted towards the "winner" using an adaptation procedure. The most suitable robot is assigned to this goal and keeps the adapted path, while the other robots discard the adaptation. At the end of each epoch, the paths are regenerated to filter out unnecessary waypoints and ensure efficient routing between waypoints.

2) PRM and Path Planning: The key difference between our approach and the SOM algorithm in [7] is that we use a PRM to guide the learning procedure and plan collision-free paths between goals, which needs to be considered in almost every step of the algorithm. The PRM $\mathcal{P} = (V, E)$ is defined as a graph consisting of vertices $V = \{v_1, v_2, ...\}$ with $v_j \in \mathcal{X}_{free}$, with some pairs of vertices connected by edges $E = \{e_1, e_2, ...\}$.

Several steps of the algorithm require computing the shortest path between two points in the environment. For each path planning query, we find the closest vertex that is straight-line obstacle free to the start and end points, and use A* to efficiently find a path along E linking these two vertices. An important benefit of using a PRM over of tree-based approach such as RRGs [15] is that the PRM can be efficiently re-queried many times without needing to be regenerated. We further reduce computation time by caching the results of path planning queries to be used in any subsequent identical queries.

3) Primary and Secondary Waypoints: To enable the algorithm to jointly optimise for both goal selection and routing between goals, we divide the path X^i of each robot into primary waypoints X^i_α and secondary waypoints X^i_β. The main purpose of a primary waypoint is to visit an

Alg. 1: Self-organising map algorithm over a PRM.

Input : A set of robots $\mathcal{R} = \{r^i\}$ with associated travel budgets $\{b^i\}$
A set of goals $\mathcal{G} = \{g^k\}$
Output : Planed path for each robot $\{X^i\}^*$

1 $\mathcal{P} \leftarrow$ PRM across the environment;
2 $X^i \leftarrow$ initial path $\forall r^i \in \mathcal{R}$;
3 **for** $n = 1, ..., |epochs|$ **do** // Each epoch
4 perm $\leftarrow$ random permutation of $\mathcal{G}$;
5 **for** $g^k \in \mathcal{G}$ *in order* perm **do** // Each goal
6 **for** $r^i \in \mathcal{R}$ **do** // Each robot
7 $X^{i\prime} \leftarrow$ ADAPTATION$(X^i, g^k, \mathcal{G}, \mathcal{P}, n)$;
8 $c^{i\prime} \leftarrow$ travel cost of path $X^{i\prime}$;
9 $r_i \leftarrow \underset{r^i \in \mathcal{R}, c^{i\prime} \leq b^i}{\operatorname{argmin}} \left(\frac{c^{i\prime}}{b^i}\right)$; // Select robot
10 $X^i \leftarrow X^{i\prime}$; // Save path of selected robot
11 $\{X^i\} \leftarrow$ regeneration of $\{X^i\}$; // Regen paths
12 $F \leftarrow \sum_{g^k \in \mathcal{G}} o^k$; // Evaluate objective
13 **if** $F > F^*$ **then**
14 $\{X^i\}^* \leftarrow \{X^i\}$; // Save best plan

associated goal region. Secondary waypoints between each pair of consecutive primary waypoints facilitate feasible routing between goal regions; they are defined as the PRM vertices along the shortest path from a primary waypoint to the next primary waypoint. The primary waypoints are adapted and deleted by the main SOM learning procedure, while the secondary waypoints are regularly regenerated and may be promoted to primary waypoints.

B. Initialisation

Prior to commencing the main loop, the algorithm begins with the following initialisations.

1) PRM: Firstly, the PRM $\mathcal{P}$ is generated in Alg. 1, Line 1. While the rest of the algorithm is agnostic to how the PRM is generated, a typical procedure would be to randomly sample a fixed number of vertices in $\mathcal{X}_{free}$, and also include additional vertices at the centre and around the boundary of each goal region to ensure the PRM sufficiently covers the goal regions. An edge is created between each pair of vertices where the line segment bounded by the two vertices is entirely within $\mathcal{X}_{free}$ and the Euclidean distance between them is less than a specified distance.

2) Paths: Each path is initialised by performing a random walk of fixed length within a random goal region (Alg. 1, Line 2). Primary waypoints are sampled from the goal region, with feasibility ensured by discarding waypoints that would result in crossing occupied space. We note that the algorithm is insensitive to this initialisation procedure since these initial waypoints are quickly replaced in the early epochs.

Alg. 2: Adaptation step of the SOM algorithm, guided by the PRM.

Function: ADAPTATION
Input : Path X^i of robot r^i
Goal region g^k
Set of all goal regions $\mathcal{G}$
PRM $\mathcal{P}$ across the environment
Epoch number n
Output : An adapted path X^i for robot r^i

1 $x^\star \leftarrow$ closest $x^i_j \in X^i$ to g^k, using visibility check through $\mathcal{X}_{free}$ or plan over $\mathcal{P}$; // Pick winner
2 Promote $x^\star$ to a primary waypoint, if not already;
3 Move $x^\star$ to closest point in g^k; // Move winner
// Adapt path
4 Replan secondary waypoints either side of $x^\star$;
5 **for** $x^i_j \in \left\{X^i_\beta \cap \mathcal{G}\right\}$ **do**
// Promote secondary waypoints in goal regions
6 $X^i_\alpha \leftarrow X^i_\alpha \cup \{x^i_j\}$;
7 $X^i_\beta \leftarrow X^i_\beta \setminus \{x^i_j\}$;
8 **for** $x^i_{\alpha,j} \in X^i_\alpha$ **do**
// Move primary waypoints towards goal
9 $d \leftarrow$ distance from $x^i_{\alpha,j}$ to g^k over $\mathcal{P}$;
10 $m \leftarrow$ cardinal distance (primary waypoints only) from $x^i_{\alpha,j}$ to $x^\star$;
11 Move $x^i_{\alpha,j}$ along shortest path in $\mathcal{P}$ towards g^k by distance $f(d, m, n)$ (1)
12 $X^{i\prime}_\beta \leftarrow$ replan A* paths between consecutive X^i_α;
13 **for** $g^k \in \mathcal{G}$ *that is not visited* **do**
// Insert primary waypoints where path intersects unvisited goal regions
14 Insert x^i_α at an intersection point between edges in X^i and g^k;

C. Learning Epochs

During each learning epoch, the path of every robot is adapted to determine the effect of assigning each robot to each goal. These paths are then used to assign the most suitable robot to each goal, and finally regenerated to remove unnecessary waypoints. This will result in convergence to a suitable set of paths after sufficient epochs. These steps are described in more detail below.

1) Path Adaptation: The crux of the SOM learning procedure is the adaptation of the path of each robot towards each goal. These goals are considered in a random order to reduce the sensitivity on the initial conditions and help escape local minima (Alg. 1, Lines 4–5). The path of each robot is adapted towards the goal under consideration g^k, and the cost of doing so calculated (Alg. 1, Lines 6–8).

The adaptation step is described in more detail in Alg. 2. The distance from each waypoint in the path x^i_j to g^k is first calculated, and the closest waypoint saved (Alg. 2, Line 1). If there are no obstacles directly between x^i_j and g^k then the

Euclidean distance is used, otherwise A* is used to find a path along $\mathcal{P}$ and the total length of this path used instead.

The closest waypoint in the path is then moved to g^k to consider the case that this robot is assigned to this goal (Alg. 2, Line 3). To link this modified waypoint with the remainder of X^i, the path from it to the primary waypoints on either side must be replanned: A* is again used to find this path through $\mathcal{P}$, and the selected vertices become secondary waypoints (Alg. 2, Line 4). Any secondary waypoints that are within unvisited viewpoint regions are then converted to primary waypoints so they are not lost in subsequent path regeneration operations (Alg. 2, Lines 5–7).

The algorithm then moves each primary waypoint $x^i_{\alpha,j} \in X^i_\alpha$ towards g^k (Alg. 2, Lines 8–11). In previous algorithms [7], waypoints were moved along a straight line towards g^k, but such an approach does not account for obstacles in the environment. Our algorithm moves waypoints along the shortest path through $\mathcal{P}$ to g^k found using A* instead, which is guaranteed to be collision-free.

To ensure the path remains efficient, the closest neighbouring waypoints to the current closest waypoint should move a larger fraction towards g^k than the distant neighbours. To do so, we calculate two additional metrics quantifying the distance between $x^i_{\alpha,j}$ and g^k: the length d of the shortest collision-free path along $\mathcal{P}$, and the cardinal distance m (the number of intermediary primary waypoints) to the currently selected closest waypoint to g^k. We also define the epoch counter n, which is used to ensure neighbour attraction decreases in later epochs. The distance $x^i_{\alpha,j}$ is moved along the shortest path is calculated as:

$$f(d, m, n) = d \times C_1 \exp(-\frac{mn}{C_2}), \tag{1}$$

where C_1 and C_2 are constants that control the rate of convergence. The effect of this formulation is to move the nearest neighbours of the current closest waypoint a larger fraction towards the current closest waypoint, while waypoints that are several hops away will move a small fraction. These fractions also decrease as the learning progresses, such that the earlier iterations have large adaptations, and the later iterations have smaller local refinements.

After moving the primary waypoints, the modified path is given a final clean. This starts by recalculating the secondary waypoints using A* to ensure the modified primary waypoints are linked in the most efficient manner (Alg. 2, Line 12). Additional primary waypoints are inserted where the path edges intersect viewpoint regions that are not already visited, to ensure each goal the robot passes through is not lost during regeneration and is accounted for in the evaluation of the objective function (Alg. 2, Lines 13–14).

2) Robot–Goal Allocation: The effect of allocating each robot to g^k has now been considered. The actual robot allocation is done in a greedy manner, where the robot that reaches g^k with the lowest cost fraction of its budget used is assigned to the goal (Alg. 1, Line 9). The winning robot's path is updated to the adapted path accordingly (Alg. 1, Line 10), and the other robots' adapted paths are discarded.

3) Path Regeneration: After each goal has been considered, a regeneration procedure cleans the path by removing unnecessary primary and secondary waypoints (Alg. 1, Line 11). It consists of three stages:

1) Delete primary waypoints that are not in goal regions.
2) Replace secondary waypoints with new collision-free paths through $\mathcal{P}$ found using A*.
3) Promote secondary waypoints to primary waypoints if they are in goal regions that are not already visited by other primary waypoints.

The resulting path will have an equal number or fewer primary waypoints than before, ensuring that the size of the path does not grow unbounded. The new path should also have reduced or equal path length, ensuring that the cost budget is being used efficiently.

4) Solution Evaluation and Cooling: The objective function is evaluated, and the best set of paths found so far is saved (Alg. 1, Lines 12–14). Finally, the epoch count n is incremented to reduce the attraction between neighbouring primary waypoints in later epochs. This ensures the algorithm can initially make large (global) changes to explore the solution space, before making smaller (local) refinements in later epochs.

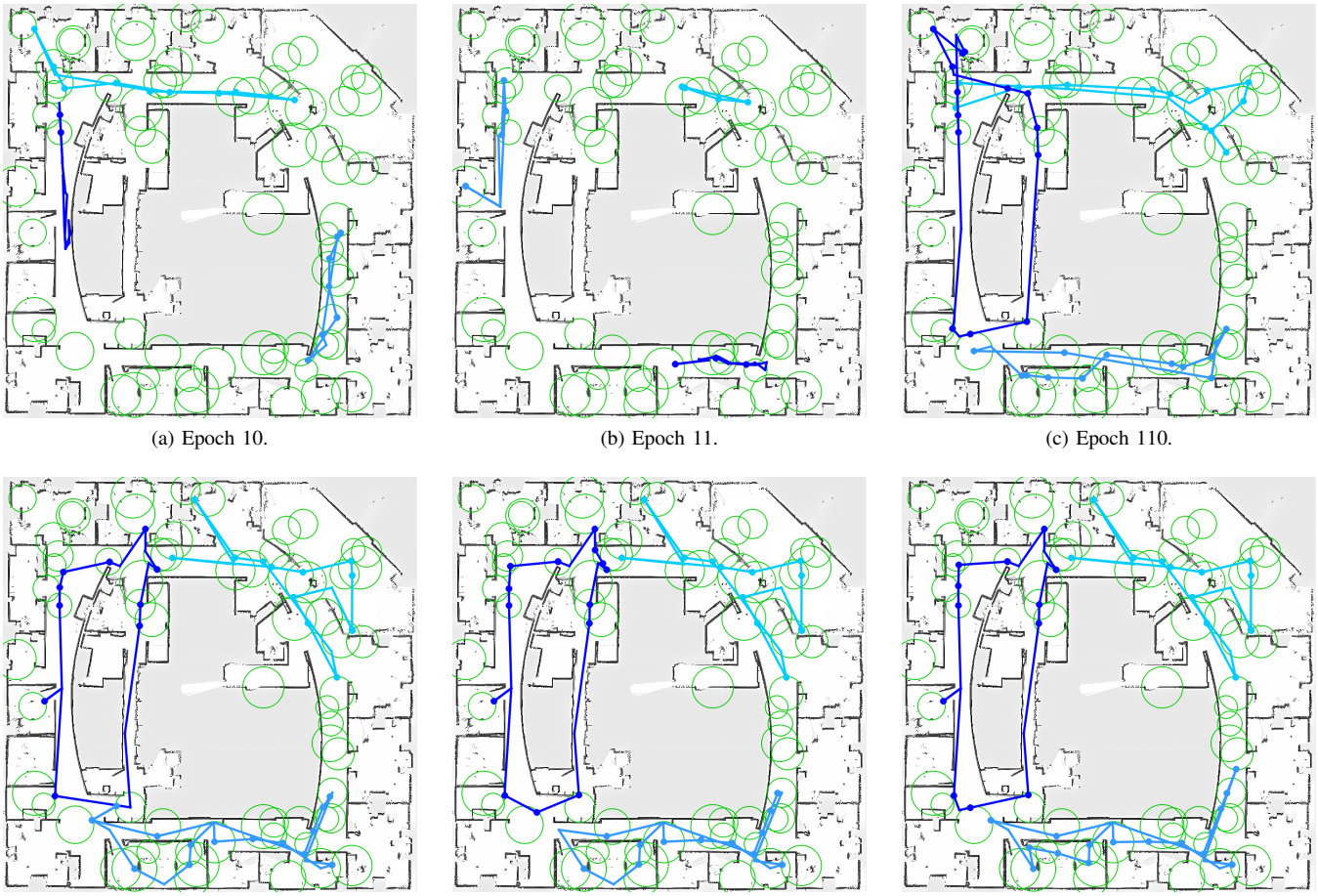
D. Analysis

Self-organising map algorithms are heuristic algorithms with tunable computation time and are anytime. Following the analysis of [7], our SOM algorithm has computational complexity of $O(|epochs| \cdot |\mathcal{G}| \cdot |X|)$, where $|\mathcal{G}|$ is the fixed number of goal regions, $|X|$ is an upper bound on the number of waypoints for the team of robots, and the number of epochs can be controlled by specifying the adaptation parameters C_1 and C_2. The number of waypoints varies between epochs but is bounded: the number of primary waypoints is upper bounded by $|\mathcal{G}|$ due to their removal during regeneration, and the number of secondary waypoints is bounded and dependent on the resolution of the PRM.

Each iteration of the inner loop performs several calls to the PRM-based path planner; therefore, it is important that this is efficient, and this motivates our proposed use of caching for efficient repeated queries. Waypoint-to-vertex obstacle checking is also called regularly, which can be performed efficiently using an adaptive step size based on a precomputed distance transform map. Although our proposed addition of path planning and obstacle checking steps increases computation time, we show empirically that these are necessary for successfully and efficiently navigating through obstacle-rich environments.

V. Experimental Results

To evaluate and demonstrate the performance of our proposed algorithm, we ran simulations in a range of scenarios. We considered three environments: (1) a publicly-available laser scan 2D map of a cluttered office environment [17], (2) a small arena inspired by a bug trap designed to exaggerate the differences between Euclidean and collision-free distances, and (3) a complex maze environment that further



(a) Epoch 10. (b) Epoch 11. (c) Epoch 110.

(d) Epoch 300. (e) Epoch 400. (f) Epoch 500.

Fig. 3. The evolution of the paths produced by our proposed algorithm across epochs for three robots in the office environment with 65 goal regions (green circles). The earlier epochs have big jumps seeking to find the main global structure of the three paths, which settles from around epoch 110 (c) onwards. Later epochs usually have smaller local adaptations to most efficiently route through the selected goal regions.

exaggerates these differences. We compare to several SOM variants that consider obstacle avoidance in different ways, to highlight the advantages of our proposed approach.

A. Experimental Setup

In each map, a fixed number of goal regions of random radii are distributed at random: these represent regions of interest that the team of robots should visit. The bug trap arena contained 50 goals, the office environment contained 65 goals, and the maze contained 20 goals. We tested the algorithm for a single robot and a small multi-robot system consisting of two or three agents. We ran ten simulations with new randomised goal regions for each scenario in the bug trap and office maps, and a single simulation on the maze map.

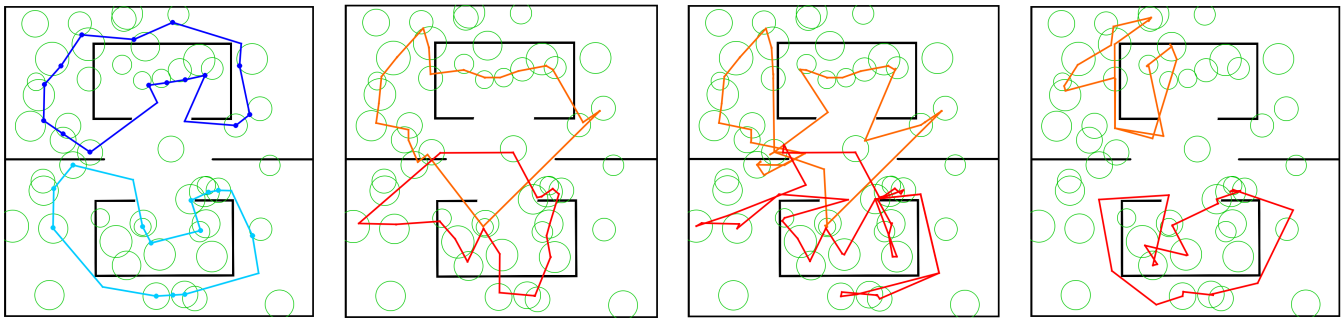
Our experiments compare several SOM variants that handle obstacle avoidance in different ways:

- *PRM-guided SOM*: Our proposed method.
- *Standard SOM*: The SOM approach in [7] that does not consider obstacle avoidance.
- *SOM then obstacle avoidance*: Post-processing *Standard SOM* by adding A* paths between the planned waypoints, which may result in exceeding the cost budget.
- *Adjusted-budget SOM*: Similar to SOM then obstacle avoidance, but with a smaller path cost budget to account for the post-processing exceeding the budget. Note that the smaller budget value was chosen empirically for each scenario, but does not guarantee the solution satisfies the original budget constraint.

For the comparison methods that use a PRM, a PRM was generated with 400 uniform-random vertices, and 5 additional vertices within each goal region. The adaptation cooling parameters were selected as $C_1 = 0.5$ and $C_2 = 100$, values chosen such that convergence was reached in approximately 500 epochs.

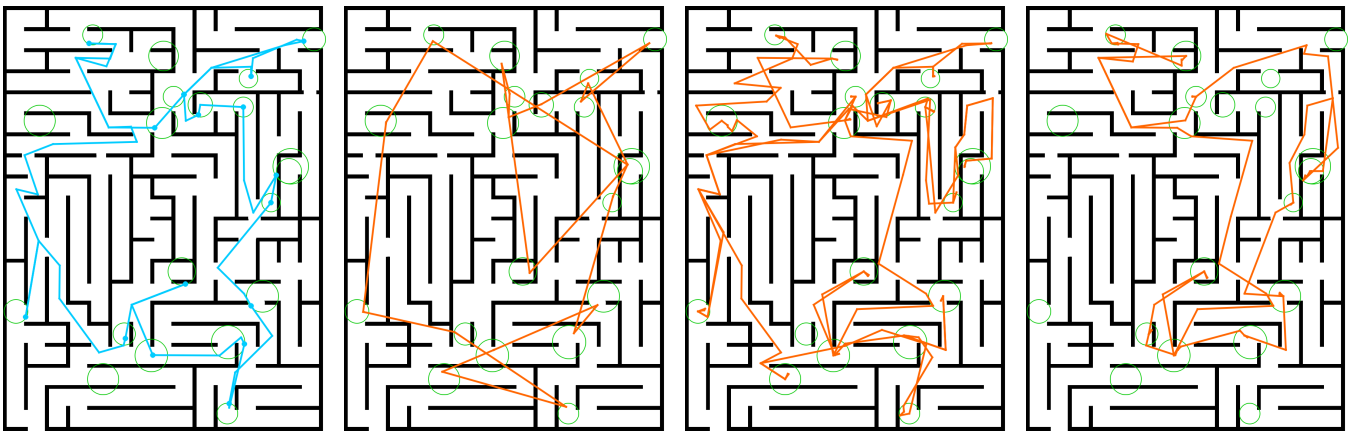
B. Results

The evolution of the paths produced during one trial of our proposed PRM-guided SOM algorithm is shown in Fig. 3. In early epochs, the paths change greatly as they are pulled a long way in the adaptation step to explore a range of different options until a general global path structure is settled upon. In later epochs, smaller changes are made to refine the paths



(a) *PRM-guided SOM*. (b) *Standard SOM* – collisions with obstacles (c) *SOM then obstacle avoidance* – cost budget exceeded. (d) *Adjusted-budget SOM*.

Fig. 4. Example solutions for each of the comparison methods in the bug trap environment where Euclidean distance costs are a poor approximation for path costs. Two robots are planned for, with the aim of maximising the number of goal regions (green circles) visited, while avoiding obstacles and subject to a path length constraint. Our method (a) yields high-reward paths that efficiently navigate around obstacles.



(a) *PRM-guided SOM*. Cost: 5926, Score: 17 (b) *Standard SOM* – collisions with obstacles. Cost: 5466, Score: 20 (c) *SOM then obstacle avoidance* – cost budget exceeded. Cost: 12234, Score: 19 (d) *Adjusted-budget SOM*. Cost: 5712, Score: 12

Fig. 5. Example solutions for each of the comparison methods in the maze environment where Euclidean distance costs are a poor approximation for path costs. One robot is planned for, with the aim of maximising the number of goal regions (green circles) visited, while avoiding obstacles and subject to a path length constraint. Our method (a) yields high-reward paths that efficiently navigate around obstacles.

and find efficient routes between the selected goal regions. Convergence to the final set of paths follows similar trends to the SOM approach of [7] without obstacle avoidance.

Fig. 4 shows example paths produced by each algorithm on the bug trap map to contrast their behaviour. Our *PRM-guided SOM* algorithm (Fig. 4a) is able to efficiently enter and exit the internal rooms on this map, and reaches a large number of goal regions. The *standard SOM* (Fig. 4b) ignores the obstacles, so produces paths that are not feasible. *SOM then obstacle avoidance* produces collision-free paths, but they are not efficient (Fig. 4c): note how the paths enter and exit the internal rooms several times as goal regions are selected that are close in Euclidean space, but are separated by walls. These inefficiencies cause the paths to significantly exceed the cost budget constraint. *Adjusted-budget SOM* provides paths that are feasible (Fig. 4d), but due to the inefficient selection of goal regions, fewer goals are visited than by our algorithm.

These differences in behaviour between the algorithms are even more stark in environments with more obstacles. Fig. 5 shows a solution generated by each algorithm with a single robot in the maze environment. Once again, we see the *PRM-guided SOM* produces collision-free paths between goal regions that are selected in an efficient order. *Standard SOM* generates paths that travel through walls, and *SOM then obstacle avoidance* removes the collisions but exceeds the cost budget by looping back on itself many times. Carefully tuning *adjusted-budget SOM* can keep the solutions in budget, but they are not as efficient as in *PRM-guided SOM* so fewer goal regions are visited.

A summary of the numerical results across all trials on the bug trap and office maps is shown in Fig. 6. In all cases, our *PRM-guided SOM* algorithm is able to reach the most goal regions while avoiding obstacles and remaining in budget. The *standard SOM* approach visits more goal regions, but does not produce collision-free paths. During *SOM then obstacle avoidance* the robots reach the same goals, but their paths are now over the cost budget so are also

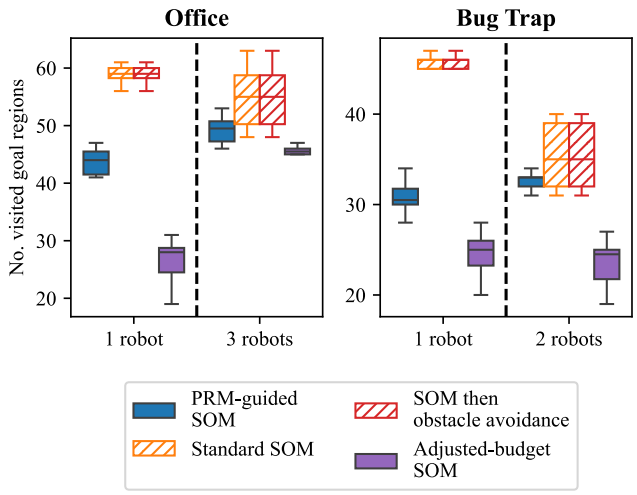


Fig. 6. Experimental results. Each boxplot shows the results of 10 trials for the given number of map, number of robots, and algorithm. Algorithms that lead to non-feasible paths are shown hatched as they cannot be directly compared to the other results: *standard SOM* produces paths that pass through obstacles, while *SOM then obstacle avoidance* exceeds the allowable cost budget.

not feasible. Iteratively adjusting the budget to compensate for this reduces the number of goal regions that are visited.

Regarding computation time, our *PRM-guided SOM* took on the order of 1 to 3 minutes on a 2.8 GHz Intel Core i7 processor. We note that runtime in our implementation is dominated by collision checking, taking 42 % of the time, which could be further optimised. The standard SOM approaches took approximately 15 seconds, but this efficiency improvement is due to the naïve consideration of obstacle avoidance by these alternative methods.

VI. Conclusion and Future Work

We presented a new generalisation of Self-Organising Maps that enables solving multi-robot, multi-goal path planning in obstacle-rich environments. Our method addresses the challenge of obstacle avoidance by guiding each step of the algorithm with a PRM, and this is demonstrated to significantly outperform standard SOM approaches. Our approach can readily incorporate other generalisations for SOMs proposed in other work, including decentralised planning [8], online goal discovery [7], polygonal goal regions [7], and other definitions of reward [9]. Other interesting avenues for future work include incorporating informed roadmap generation [27], and generalisations for dynamic obstacles, high-dimensional state spaces, and kinodynamic constraints [28].

References

[1] C. Reardon and J. Fink, "Air-ground robot team surveillance of complex 3d environments," in *Proc. of IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 320–327.

[2] M. Meghjani, S. Manjanna, and G. Dudek, "Multi-target search strategies," in *Proc. of IEEE Int. Symp. on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 328–333.

[3] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[4] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.

[5] S. Arora, "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems," *Journal of the ACM (JACM)*, vol. 45, no. 5, pp. 753–782, 1998.

[6] J. Faigl and G. A. Hollinger, "Autonomous data collection using a self-organizing map," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1703–1715, 2017.

[7] G. Best, J. Faigl, and R. Fitch, "Online planning for multi-robot active perception with self-organising maps," *Autonomous Robots*, vol. 42, pp. 715–738, 2018.

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

[9] J. Faigl and G. A. Hollinger, "Self-organizing map for the prize-collecting traveling salesman problem," in *Proc of International Workshop on Advances in Self-Organizing Maps and Learning Vector Quantization*, 2014, pp. 281–291.

[10] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: An overview of applications, formulations, and solution approaches," in *Traveling Salesman Problem: Theory and Applications*. InTech, 2010.

[11] R. Pěnička, J. Faigl, and M. Saska, "Physical orienteering problem for unmanned aerial vehicle data collection planning in environments with obstacles," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3005–3012, 2019.

[12] C. Chen, J. Frey, P. Arm, and M. Hutter, "Smug planner: A safe multi-goal planner for mobile robots in challenging environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7170–7177, 2023.

[13] J. Faigl, "An application of self-organizing map for multirobot multi-goal path planning with minmax objective," *Computational intelligence and neuroscience*, 2016.

[14] J. Faigl, M. Kulich, V. Vonásek, and L. Přeučil, "An application of the self-organizing map in the non-Euclidean traveling salesman problem," *Neurocomputing*, vol. 74, no. 5, pp. 671–679, 2011.

[15] J. Faigl, "On self-organizing map and rapidly-exploring random graph in multi-goal planning," in *Advances in Self-Organizing Maps and Learning Vector Quantization*, 2016, pp. 143–153.

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

[17] D. Hähnel, "Intel research lab," Pre-2014 Robotics 2D-Laser Datasets. [Online]. Available: https://www.ipb.uni-bonn.de/datasets/index.html

[18] P. Vansteenwegen and A. Gunawan, "Orienteering problems," *EURO Advanced Tutorials on Operational Research*, vol. 1, 2019.

[19] B. Charrow, *Information-theoretic active perception for multi-robot teams*. University of Pennsylvania, 2015.

[20] J. Faigl, "GSOA: growing self-organizing array-unsupervised learning for the close-enough traveling salesman problem and other routing problems," *Neurocomputing*, vol. 312, pp. 120–134, 2018.

[21] P. Štefaníková, P. Váňa, and J. Faigl, "Greedy randomized adaptive search procedure for close enough orienteering problem," in *Proceedings of the 35th annual ACM symposium on applied computing*, 2020, pp. 808–814.

[22] B. Angéniol, G. D. L. C. Vaubois, and J.-Y. Le Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, no. 4, pp. 289–293, 1988.

[23] K. Helsgaun, "An effective implementation of the Lin–Kernighan traveling salesman heuristic," *European journal of operational research*, vol. 126, no. 1, pp. 106–130, 2000.

[24] J. Faigl and G. A. Hollinger, "Unifying multi-goal path planning for autonomous data collection," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 2937–2942.

[25] J. Janoš, V. Vonásek, and R. Pěnička, "Multi-goal path planning using multiple random trees," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4201–4208, 2021.

[26] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[27] M. Saroya, G. Best, and G. A. Hollinger, "Roadmap learning for probabilistic occupancy maps with topology-informed growing neural gas," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4805–4812, 2021.

[28] N. T. T. Le, E. Bray, K. M. B. Lee, and G. Best, "Adaptive trajectory library planner for fast outdoor robots," in *Proc. of Australasian Conf. on Robotics and Automation (ACRA)*, 2023.