Robot Path Planning for Multiple Target Regions

Shu Ishida\(^1\), Marc Rigter\(^1\), Nick Hawes\(^1\)

Abstract—Optimal path planning to point goals is a well-researched problem. However, in the context of mobile robotics, it is often desirable to generate plans which visit a sequence of regions, rather than point goals. In this paper, we investigate methods for planning paths which visit multiple regions in a specified order, whilst minimising total path cost. We propose Multi-Region A\(^*\), an extension to the A\(^*\) algorithm with an admissible heuristic for traversing multiple target regions. The heuristic is used to trim sub-optimal paths from the search, thereby reducing the computation time required to find the optimal solution. Additionally, we extend this method to create the Windowed Multi-Region A\(^*\) which plans through overlapping sequences of regions. This provides a mechanism to trade-off optimality against computation time. We evaluate the performance of the proposed methods against point-to-point A\(^*\) planning methods using a simulation of a wheeled office robot. The evaluation shows that Multi-Region A\(^*\) with search pruning produces an optimal path, and the Windowed Multi-Region A\(^*\) with a small window size gives a good approximate solution without compromising the total navigation time, in addition to providing robustness to dynamic obstacles.

I. INTRODUCTION

Path planning is integral to mobile robot navigation, enabling robots to navigate through their environment whilst avoiding obstacles. Plans are typically created to optimise a function, e.g. the time or energy required to complete tasks. Most path planning methods operate between a set of point goals or waypoints \([1], [2], [3]\). However, in some application domains, mobile robot paths need only to pass through regions of space, rather than visit the exact waypoints. For example, in the context of a mobile robot performing an inspection task in an office environment \([4]\), we may require the robot to visit a number of rooms, but do not wish to specify a precise waypoint within each room. By planning a path which visits each room region, rather than specific waypoints (e.g. the centre of each room), the resulting path may be made shorter, or require less time or energy to follow.

Additionally, planning paths through target regions (Fig. 1) as opposed to waypoints results in a more robust navigation system in dynamic environments. During operation, robots may encounter dynamic obstacles such as humans. Waypoint-based approaches may fail in dynamic environments, because if a waypoint coincides with a dynamic obstacle, the planner will be unable to generate a valid path.

In contrast, a planner which plans between target regions is more likely to be able to generate a valid path, as it can plan to avoid dynamic obstacles, whilst still satisfying the constraint that the path enters the specified regions.

The problem of visiting multiple regions has been addressed in the past \([5], [6], [7], [8]\), but existing methods assume polygonal target regions and obstacles. Additionally, existing methods minimise the path length alone. In robotic navigation, we often wish to minimise the cost of a path over a costmap \([9]\), which gives a grid representation of the environment. Each cell in the grid is associated with a cost density, which could penalise paths near obstacles. A reliable transition between the cells is typically assumed for 4-directions (adjacent cells) or 8-directions (both adjacent and diagonal cells). There is no existing research on optimal path planning for multiple target regions on costmaps.

This research focusses on path planning methods in two dimensions for a fully or partially known environment. We assume that the path planner is to be used for a robot which could assume any 2D orientation. For a point source \(s\), a point destination \(t\) and a sequence of \(n\) convex intermediate regions \(Z_i\) \((i \in \{1, ..., n\})\), we present a path planning approach to visiting the regions in order, which minimises the path cost, given by the sum of its length and the path integration over the costmap.

![Fig. 1: Map of 24 octagonal regions with varying shapes. Map size is 47.3 m × 71.0 m, and resolution of grid is 0.05 m. Path shown traverses from a start node to a goal node, and is constrained to visit 3 randomly selected intermediate target regions (it may visit more regions if necessary).](image)
II. RELATED WORK

Popular path planning methods for a fully or partially known environment, such as PRM [10], RRT [11] and RRT* [12], all address the problem of point-to-point path planning. However, paths for multiple target regions as opposed to target points cannot be planned by a direct extension to the above techniques, since connecting shortest paths between adjacent regions may not yield an optimal overall path.

In the Travelling Salesperson Problem, the goal is to find the order in which to visit a set of waypoints to minimise the total cost. The Touring Polygon Problem is a variant of the Travelling Salesperson Problem, in which the waypoints are replaced with regions [13], [5], [6]. More specifically, the problem in which the agent must visit the edge of the region but is not allowed to enter it is often referred to as the Zookeeper Problem [7], and the problem in which the agent can freely cross the region is called the Safari Problem [8].

A Self-Organising Map [14] is one approach to solving the Touring Polygon Problem. Instead of giving an exact solution, the algorithm samples a set of points $V$ from the map, with path segments between them forming a ring connection. The algorithm iteratively updates the positions of points $ν \in V$ so that they move closer to the centers of the neighbouring target region, until all points are within a certain tolerance from the target regions, thereby giving an approximate solution to the problem.

Although this approach does not require the obstacles and targets to be polygonal for the shortest path to be calculated, the path is updated solely based on the positions of the centres of the target regions. Moreover, for convergence this solution requires repeated evaluation of the path cost for every point $ν \in V$ to its neighbouring goal. This is infeasible for online planning in realistically sized maps. We show that the algorithm we develop requires only a single computation of the path to get an optimal solution.

The Zookeeper Problem can be solved using a floodlight tree [15]. It captures the information of the vertex the path last visits before it visits the target region, similar to a visibility graph [16]. Unlike a visibility graph, it also considers edges of the polygonal region as well as the vertices and computes a reflection of the path when multiple targets are taken into consideration. The computational complexity of this algorithm is $n \log(n)$. Algorithms of $O(n^3)$ [17] and $O(kn \log(n))$ [5] have been proposed to solve the Safari Problem, using similar geometric arguments, where $k$ is the number of polygonal obstacles in the environment and $n$ is the total number of vertices specifying the polygons.

The above solutions to the Zookeeper Problem assume that all obstacles and target regions can be described as polygons for the geometric argument to hold. This restricts us from using more general forms of representing obstacles i.e. costmaps. Additionally, all of the variants to the Touring Polygon Problem assume that all regions must be visited once in any order. We assume a fixed order is given (e.g. by a task specification or higher level planner), and that order could include repeated visits to the same region.

The A* algorithm [18] is widely used for path planning on costmaps. It computes an optimal solution to the minimal cost problem in a graph, similar to Dijkstra’s algorithm [19], but with a more efficient search of the graph. The next vertex $ν$ with location $x_ν$, explored by A* is the one minimising $f(ν) = g(ν) + h(ν)$, where $g(ν)$ is the path cost from the starting location, $x_{start}$, to $x_ν$, and $h(ν)$ is the estimated remaining cost to the goal, referred to as the heuristic.

Optimality is guaranteed when the heuristic function $h(ν)$ is admissible, i.e. it never overestimates the actual minimal cost to the goal. A* is compatible with optimal path planning on costmaps. The costmap is represented by a function $W(x)$, such that $W(x)$ is the cost density for location $x$. In robot path planning applications, locations covered by obstacles typically have infinite cost, locations nearby obstacles have a finite penalising cost, and locations far from obstacles have zero cost. The total cost of a path, $g(ν)$, is the sum of the Euclidean path length and the path integral of $W(x)$ along the trajectory $(1)$

$$g(ν) = \int_{x_{start}}^{x_ν} (1 + W(x))||dx||. \quad (1)$$

Several variations of A* have been investigated to solve problems of visiting multiple targets. In [3], a heuristic function was proposed for the problem of visiting as many targets as possible within a resource constraint. The heuristic gives an estimate of the cost per goal for each search direction, and guides the search to a subgraph where the maximal number of goals are expected to be found. [1] proposes another method to solve multi-target problems, considering both settings where an agent is only required to reach one of the targets and settings where the agent has to reach all of the targets. [1] suggests an incremental method, which runs a shortest path search $n(n + 1)/2$ times to reach a solution, and a conversion method, which converts the problem into a graph search problem on which the A* algorithm has to be run only once. The search space, however, expands by $2^m$ compared to the original graph. These methods are based on Adaptive A* [20], which is an incremental heuristic search algorithm to solve a series of related path planning problems using information from previous searches. Neither approach, however, considers the problem where target regions must be visited.

III. MULTI-REGION PLANNING METHODS

This paper addresses the problem of planning a path that passes through multiple target regions. In the following sections we describe how we solve this problem with our Multi-Region A* algorithm and its sliding window variants.

A. Multi-Region A*

We propose the Multi-Region A* planner that is guaranteed to produce an optimal path by directly extending the capability of the A* algorithm. A typical application of the A* algorithm uses either a L1 or L2 norm as a heuristic function to solve a path planning problem to a target point.
Here, we define a new heuristic function that satisfies the two criteria of a heuristic function:

(a) the heuristic underestimates the remaining path cost,
(b) the heuristic decreases as the path visits the target regions in order, guiding the path towards each of the target regions in turn.

![Path to point P and heuristics (a) before reaching a region (h(p, 0)), and (b) after passing through a region (h(p, 1)). The orange line shows the shortest path to point P, and the purple line shows h(p, k), the lower bound of the shortest path to the goal. The circles show the smallest enclosing circles of the regions.](image)

Fig. 2: Path to point P and heuristics (a) before reaching a region (h(p, 0)), and (b) after passing through a region (h(p, 1)). The orange line shows the shortest path to point P, and the purple line shows h(p, k), the lower bound of the shortest path to the goal. The circles show the smallest enclosing circles of the regions.

For a point source \(s\), a point destination \(t\) and a sequence of \(n\) convex intermediate regions \(Z_i\) \((i \in \{1, ..., n\})\), each with a smallest enclosing circle \(C_i\), we define \(c_i\) and \(r_i\) to be the centre and radius of \(C_i\) respectively for \(i \in \{1, ..., n\}\), and \(c_{n+1} = t\) and \(r_{n+1} = 0\). We define the heuristic function for the Multi-Region A* as (2):

\[
h(x, k) = ||x - c_{k+1}|| - r_{k+1} + \sum_{i=k+1}^{n} (||c_i - c_{i+1}|| - r_i - r_{i+1})
\]

where \(x\) is the location for which the heuristic is to be evaluated, and \(k \in \{0, ..., n\}\) is the number of target regions already visited. Fig. 2 illustrates this heuristic.

We construct a graph \(G\) from vertices \(v \in V\). Each vertex in the graph has attributes of a position \(x \in X\) on the costmap \(W\), and the number of regions which have been visited, \(k\). We represent such vertex as \(v_{x,k}\). For every \(k\), we assume a 8-direction connection for the graph; there is an edge between \(v_{x,k}\) and \(v_{x',k}\) if and only if \(x \notin Z_{k+1}\) and \(x\) and \(x'\) are adjacent or diagonal cells in the grids, with an edge weight of \(w(v_{x,k}, v_{x',k}) = ||x - x'||(1 + W(x))\). There also exists a directed edge with a weight of \(0\) from \(v_{x,k}\) to \(v_{x,k+1}\) if and only if \(x \in Z_{k+1}\). We refer to a set of vertices \(L_k = \{v_{x,k} | x \in X\} (k \in \{0, ..., n\})\) as the \(k\)-th level in \(G\).

The search is illustrated in Fig. 3. Vertices in \(L_k\) keep track of the cost and the path segment between \(Z_k\) and \(Z_{k+1}\). \(L_0\) is for the path segment between the source \(s\) and \(Z_1\), and \(L_{n+1}\) is for the path segment between \(Z_n\) and the point destination \(t\). The only connections between \(L_k\) and \(L_{k+1}\) lie within \(Z_{k+1}\). As soon as the head of the path reaches \(Z_{k+1}\), the path head moves to \(L_{k+1}\) and continues exploring the graph, without returning to the previous level.

To find an optimal path which passes through the regions in the specified order, we perform A* search over \(G\) using the heuristic defined by (2). We propose this path planning method as a strong candidate solution to the research problem as it is guaranteed to give the optimal path.

**B. Multi-Region A* with search pruning**

While the Multi-Region A* method is guaranteed to provide the optimal path, the search is computationally expensive because the graph to be searched is larger than the initial two dimensional grid, hence more vertices have to be explored before a target is reached.

**Algorithm 1 Multi-Region A* with search pruning**

**Require:** Graph \(G\), source \(s\), target \(t\), regions \(Z_i\) \((i \in \{1, ..., n\})\)

**Ensure:** \(v_{x,0} \in G, v_{t,n} \in G, \exists\) path from \(v_{x,0}\) to \(v_{t,n}\)

1: for all \(v \in V[G]\) do
2: \(\quad g[v] \leftarrow +\infty\)
3: \(\quad parent[v] \leftarrow \text{undefined}\)
4: \(\quad g[v_{x,0}] \leftarrow 0\)
5: \(\quad v_{x,0}.\text{priority} \leftarrow 0\)
6: \(\quad Q \leftarrow \{v_{x,0}\}\)
7: \(\quad k* \leftarrow 0\)
8: while \(u \neq v_{t,n}\) do
9: \(\quad u \leftarrow Q.\text{pop\_min\_priority}()\)
10: if \(k_x > k*\) then
11: \(\quad k* \leftarrow k_x\)
12: \(\quad \text{cost\_to\_region}[k_x] \leftarrow g[u]\)
13: if \(k_x < k*\) and \(\text{cost\_to\_region}[k_x + 1] + 3r_{k_x + 1} \leq g[u] + ||x_u - c_{k_x + 1}||\) then
14: \(\quad \text{continue}\)
15: for all \(v\) connected to \(u\) do
16: \(\quad c \leftarrow g[u] + w(u, v)\)
17: if \(c < g[v]\) then
18: \(\quad g[v] \leftarrow c\)
19: \(\quad v.\text{priority} \leftarrow g[v] + h(v, t)\)
20: \(\quad \text{parent}[v] \leftarrow u\)
21: \(\quad Q.\text{insert}(v)\)

Fig. 3: Multiple level of search space, each level corresponding to a path segment between two consecutive regions

To solve this issue, we introduce search pruning (Algorithm 1, lines 10-14), which reduces the number of vertices in the graph explored by Multi-Region A*. Search pruning occurs when the A* algorithm explores a vertex in the graph. For an unexplored vertex \(v_{x,k}\) in the priority queue, if its immediate target region \(Z_{k+1}\) has already been visited with
minimum cost of \( C_{k+1} \), and the total of current cost and estimated remaining cost from \( v_{x,k} \) to \( Z_{k+1} \) is \( C_{k+1} + 2r_{k+1} \) or more, the vertex is discarded and is not explored (Fig. 4).

Fig. 4: Pruning path candidate whose expected cost to its immediate target region \( Z_{k+1} \) from the source is greater than the shortest path cost to \( Z_{k+1} \) by \( 2r_{k+1} \) or more.

We now show that search pruning does not affect the optimality of the Multi-Region A* algorithm. If there exists a better point of entry to \( Z_{k+1} \) compared to that of the current best path so far, the best path so far could reach that point of entry with less than \( 2r_{k+1} \) extra cost, assuming convex shape and zero costmap cost within the regions, i.e. regions do not contain any obstacles and the only cost of navigating within the region arises due to path length. Therefore, any path that has \( 2r_{k+1} \) or more distance to the \( Z_{k+1} \) (\( 3r_{k+1} \) to the centre of the region) will not contribute to the optimal overall path. Therefore, optimality is maintained regardless of search pruning.

C. Windowed Multi-Region A*

Multi-Region A* with search pruning is guaranteed to find an optimal path. However, this method may require an unacceptably long planning time in real-world problems where the environment is large and many intermediate target regions have to be visited.

In practice, we rarely need to know the full path for the robot to plan its initial motion. To minimise the total navigation time, it would be ideal if we could produce an initial path for the first few regions for the robot to execute, and update the path as the robot progresses, thereby reducing the wait time before the initial path is provided.

We devised a Windowed Multi-Region A* which allows the path planner to produce a path with only a subset of the regions. As illustrated in Fig. 5, the path planner takes the first \( m \) regions to produce a path from the current location to the centre of the \( m \)-th target region (\( c_m \)). Once the robot enters \( Z_m \), the execution of the path is preempted, i.e. the motion of the robot is terminated, and a new path using the next \( m \) regions is produced from the point of preemption. Looking at extreme cases, a Windowed Multi-Region A* planner with window size of \( m = 1 \) results in centre-to-centre A* path planner with preemption, and that with \( m = n + 1 \) results in a Multi-Region A* path planner.

D. Analysis of performance of Windowed Multi-Region A*

As can be seen in Fig. 5, there is a slight difference between the optimal path produced by Multi-Region A* and the Windowed Multi-Region A*, but the difference is negligible for an environment with sparse obstacles. Since the Windowed Multi-Region A* is performed on every \( m \) sequence of regions, the path planned for every \( m \) sequence is optimal, apart from the fact that the path is planned to the last region’s centre.

The worst case performance is given when there are obstacles in the environment so that for every sliding window applied, the planned path segment terminates on the opposite side across the target region to the overall optimal path, forcing the sub-optimal path to pass through the entire target region. An example of such a configuration is illustrated in Fig. 6. Small protruding obstacles force centre-to-centre A* to choose an overall sub-optimal path for each of the regions. Assuming that the difference in path segment lengths caused by the small protrusion is negligible, the optimal path is shorter by the diameter of the last region for each path segment, which is \( 2r_{lm} \) for the \( l \)-th path segment (\( l \in \{1, ..., \lfloor \frac{n}{m} \rfloor \} \)). Therefore, the upper bound to the sub-optimality of the path produced by the Windowed Multi-Region A* is \( \sum_{l=1}^{\lfloor \frac{n}{m} \rfloor} 2r_{lm} \).

This is a significant improvement from centre-to-centre A* planning with preemption (equivalent to the Windowed Multi-Region A* with \( m = 1 \)), since the upper-bound of the sub-optimality is reduced by a factor of \( m \). We could adjust the window size \( m \) for different obstacle settings.

IV. EVALUATION METHOD

To evaluate the performances of the Multi-Region A* and the Windowed Multi-Region A* planners, we compared the path cost and the planning time against point-to-point A* planners. We also measured the total navigation time for
different planners in simulation, using a differential drive office robot and a floor map of a building (Fig. 1).

A. Point-to-point planners for comparison

a) Centre-to-centre A*: A baseline method that returns the shortest path between the centres of adjacent target regions using the A* algorithm.

b) Centre-to-centre A* with preemption: Computes the shortest path to the centre of the next target region as soon as the path enters the previous target region.

B. Evaluation metrics

To compare the performance of the planners, we recorded the costs of the paths generated, and the computation time for each method. To assess the trade off between computation time, and more efficient paths, we use the sum of computation time and path execution time as a third metric.

C. Implementation of the simulation

We integrated our path planners into a topological navigation system, developed as a part of the STRANDS project [4]. The system defines a topological map on which regions of various size can be placed, and regions are connected by edges. A high-level planner plans the order in which the robot visits regions to achieve a task [21]. The path planners which we have described are used to compute a path which satisfies the requirement while minimising the cost along the path. We simulated a SCITOS robot [22] with a speed of 0.55 m s$^{-1}$ to navigate around an office building for which a global costmap is given. The path planners were all run in a Robot Operating System (ROS) implementation on an Intel® Xeon® E3-1505M v6 @ 3.0GHz laptop.

V. RESULTS

We evaluated the path planners on 50 trials by randomly selecting a start node, $n = 5$ intermediate target regions and a goal node from 24 topological nodes in Fig. 1, which represent key locations on the map. For each trial, we measured the planning time and the path cost difference compared to the path produced by the Multi-Region A* planner, which we expect to be optimal, and total navigation time of the simulated robot including planning time.

Fig. 7 (a) shows that all planners apart from the Multi-Region A* without preemption have similar planning time.

Comparing the average planning time (Table I) shows that planning for a larger window marginally increased the planning time. We observe that, although Multi-Region A* and the Windowed Multi-Region A* considers a set of point goals $\{v_{x,k} | x \in Z_k\}$ for every region $Z_k$, as opposed to the point-to-point A* methods which only consider a single point goal $v_{x,k}$, the planning time is only marginally impacted, unlike the Multi-Target Adaptive A* [1], for which the planning time increases quadratically with the number of points sampled from the target regions.

The average path cost for the Multi-Region A* was 1641.2. We plotted the relative cost difference for each of the planners in Fig. 7 (b), and indicated its average across 50 trials in Table I. Multi-Region A* produced the best possible path, regardless of search pruning; Windowed Multi-Region A* with larger window size produced better paths than that with smaller window size; these were followed by centre-to-
centre A* with preemption. This is expected since centre-to-centre A* with preemption is a special case of the Windowed Multi-Region A* planner with window size $m = 1$. All methods significantly outperformed centre-to-centre A*.

This confirms our hypothesis that search pruning for Multi-Region A* improves planning time while preserving optimality. Windowed Multi-Region A* further reduces planning time, with a slight compromise of optimality.

The average total navigation time for centre-to-centre A* with preemption was 233.6s. We compare the relative navigation time difference for each of the planners using box plots in Fig. 8. We observe that all Multi-Region A* variants with pruning outperform centre-to-centre A*. We also observe that the navigation time for Windowed Multi-Region A* is comparable to centre-to-centre A* with preemption, occasionally outperforming it. We may conclude that path planners which utilise the knowledge of regions can compensate for the extra time spent on planning by allowing the robot to traverse a more efficient path.

VI. CONCLUSION

In this research, we proposed Multi-Region A* and the Windowed Multi-Region A* for planning paths which visit multiple target regions in a specified order, with the objective of minimising path cost.

Geometric analysis and evaluation of the performances on an office building simulation with a differential drive robot showed that the Multi-Region A* planner produces an optimal path regardless of search pruning, and that search pruning improves the computation time. It was also shown that the Windowed Multi-Region A* planner gives a compromise between path cost and planning time, and changing the window size could allow us to dictate how much importance we assign to each; a small window size will allow faster computation but by compromising the path cost.

Multi-Region A* and its sliding window variants could be advantageous for large target regions, problems with many regions to visit, or when the robot is slow or costly to move. Moreover, region-aware planners add robustness to the navigation system by relaxing the constraints upon the path to be produced, as opposed to point-to-point path planners which are prone to failure in the face of dynamic obstacles within regions.

REFERENCES


