Space-filling forest for multi-goal path planning

Vojtěch Vonásek and Robert Pěnička
Faculty of Electrical Engineering, Czech Technical University in Prague,
Technická 2, 166 27 Prague, Czech Republic,

***Abstract*—In multi-goal path planning, the task is to find a sequence to visit a set of target locations in an environment. The combinatorial part of the problem (finding the sequence) can be solved as an instance of Traveling Salesman Problem, which requires knowledge about collision-free paths (and distances) between the individual targets. Finding the collision-free paths between the targets is essential in this task. Sampling-based planners like Probabilistic Roadmaps (PRM) and Rapidly-exploring Random Tree (RRT) can be used to find these paths. However, PRM can be computationally demanding, as it attempts to connect each node in the roadmap to its neighbors, regardless of their later usage in the solution. Contrary, RRT is a tree-based planner, and one run can only provide paths starting in the root of the tree (a single target). In this paper, we propose a novel planner for multi-goal path planning. Multiple trees (forest) are constructed simultaneously from the targets and expanded by collision-free configurations until they touch each other or obstacles. Each tree, therefore, does not explore the whole configuration space (as in the case of RRT), and its construction is faster than PRM, as it uses lower number of edges. The efficiency of this new planning approach is demonstrated in the multi-goal path planning in 2D environment with tens of targets and with narrow passages.**

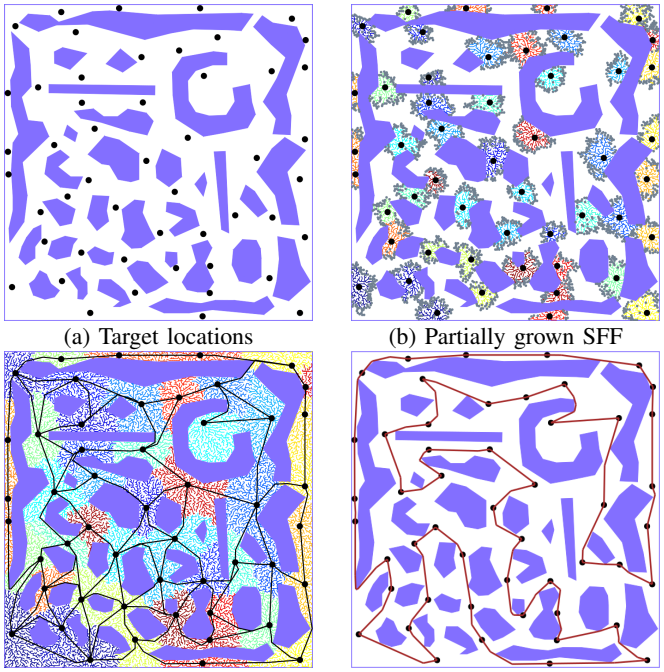
I. Introduction

In this paper, we consider the multi-goal variant of the robotic path planning problem. Typical use-cases of this task are in robotic surveillance missions [14] and data collection planning [15]. The classical multi-goal path planning is formulated as the Traveling Salesman Problem (TSP) [13], where a path visiting all the given target locations with minimal length is to be found (Fig. 1a,d). In the scenarios with obstacles, which are considered in this paper, this requires to find collision-free paths between all pairs of the targets before the TSP can be used to find the sequence.

Sampling-based motion planning can be used to find the collision-free paths between the targets. These planners explore the configuration space of the robot by random sampling [8]. The random samples are classified using collision detection and the free ones are stored into a roadmap. The roadmap approximates the collision-free region of the configuration space and a path in it relates to motion in the workspace. Widely used sampling-based planners are Rapidly Exploring Random Tree (RRT) [7] and Probabilistic Roadmaps (PRM) [6].

RRT represents the roadmap as a tree and therefore, it is suitable for finding paths from a single target to other targets. Paths between multiple targets need to be solved by building the RRT tree from each target, which is time consuming. Contrary, PRM builds the roadmap as a graph with cycles, where paths between different start/goal targets can be found at once. However, PRM attempts to connect each node of the roadmap with its neighbors, regardless if these edges are later used in a path. Another issue of sampling-based planners is the narrow passage problem [4], [10]: regions with low volume (in the comparison to the volume of the whole configuration space) are difficult to cover by the random samples. Their number has to be increased to sample the passage dense enough, which also increases the planning time.

This work is supported by the Czech Science Foundation (GAČR) under project No. 19-22555Y. Project CESNET LM2015042 enabled computations.



(a) Target locations (b) Partially grown SFF (c) Finished SFF with paths (d) Final sequence (TSP solution)

Fig. 1: Example of multi-goal path planning (a). The proposed SFF method grows trees from the target locations (b) until they cover space (c), which allows us to find the paths between the targets. The final sequence of visiting the targets (d).

In this paper, we propose a novel planning algorithm for multi-goal path planning, called Space-filling forest (SFF). The method simultaneously builds multiple trees that are rooted at the target locations. The trees are incrementally expanded by collision-free configurations, until they touch each other or until an obstacle is reached (Fig. 1b,c). In comparison to RRT, the proposed method can find paths between multiple targets at once. As space is explored using a tree structure, the number of edges tested during the planning phase is lower than in PRM, and the planning is therefore faster.

978-1-7281-0303-7/19/$31.00 ©2019 IEEE

II. Related work

Path planners construct a solution in the given order: from start to goal. In multi-goal path planning, also the order of visiting the targets has to be found which leads to the classical combinatorial routing problem. We overview the most relevant approaches for finding paths between individual targets, that can be further used for multi-goal path planning.

Path planning combined with routing has been mostly studied in the context of the Traveling Salesman Problem (TSP). The Physical TSP (PTSP) [16] combines TSP with real-time motion planning in video games. A multi-tree Transition-based RRT [2] has been proposed for creating a collision-free roadmap for the routing problem. Several approaches combining the routing problems with motion planning have been introduced for scenarios with Autonomous Underwater Vehicles (AUV), e.g., planning mine countermeasures missions with PTSP [11], the Clustered TSP [3] and high-level mission planning [12].

The multi-goal path planning requires knowledge about mutual reachability of the targets, which requires path planning to find collision-free paths between the targets. Sampling-based planners like PRM [6], RRT [7] and Expansive Space Trees (EST) [4] can be used to find these paths. Originally, PRM and RRT do not guarantee optimality of the path; the work [5] proposes asymptotically optimal RRT* and PRM*.

PRM first samples the configuration space, classifies the samples using collision-detection and stores the free-ones [6]. The stored samples are then connected, if possible, by a collision-free edge with their close neighbors. PRM is suitable for finding the paths between all pairs of targets, as these paths can be found in the single roadmap.

RRT iteratively builds a tree rooted in the initial configuration [7]. In each iteration, a random sample is generated in the configuration space and its nearest node in the tree is found. This node is then expanded towards the random sample using a local planner (e.g., using the straight-line expansion). The growth terminates if the tree approaches the goal. RRT is single-query tree-based planner, therefore it can find paths starting only from the given initial configuration. To obtain paths between different targets, the planner has to be run multiple times (starting from each target). RRT for multi-query motion planning in dynamic environments was presented in [9]. The configuration space is covered by a set of RRT trees, that have been built in the previous iterations. The trees are merged and pruned according to the changes in the environment. The work [17] also maintains multiple RRT trees; the new tree is established if the random sample is classified as being located in a narrow passage. The trees are merged if they approach each other.

However, the aforementioned multi-tree planners [9], [17] are not designed for path planning between different targets. The tree-based search brings several advantages in the comparison to PRM: the number of edges in the tree is lower than in the PRM roadmap, the tree can be expanded based on forward motion model, which allows to consider the dynamics and kinematics of the robot. This motivates us to design new tree-based planner for multi-goal path planning, which is described in the next section.

III. Multi-goal path planning

Let $\mathcal{C}$ denote the configuration space of the robot and let $\mathcal{C}_{free} \subseteq \mathcal{C}$ is the collision-free region, where the robot can move; $\varrho(\cdot,\cdot)$ is the distance between two configurations. The task of the multi-goal path planning is to find the shortest path connecting n given targets $c_i \in \mathcal{C}_{free}$, $i = 1, \ldots, n$.

A. Finding the sequence of targets

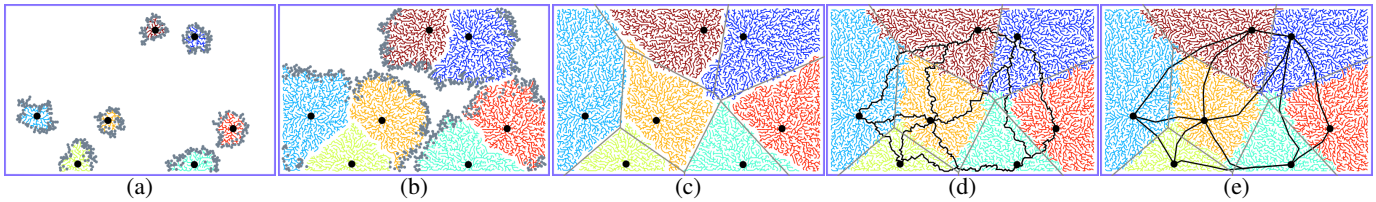
This requires to find the right permutation of targets described by vector of target indexes $\Sigma = (\sigma_1, \cdots, \sigma_n)$ assuming that the distances $\varrho(c_i, c_j)$ between the targets are known. Then, the finding of TSP path can be summarized as an optimization problem *minimize* $\sum_{i=2}^{n} \varrho(c_{\sigma_{i-1}}, c_{\sigma_i}) + \varrho(c_{\sigma_n}, c_{\sigma_1})$ where the sequence Σ is optimized to minimize closed-loop path over all targets.

The integral task of the sequence finding is the computation of the distances between the targets, which should be as minimal as possible. This requires to find collision-free paths between the targets, which we solved using the proposed method described in the next section.

B. Space-filling forest

The task of the path-planning part of multi-goal planning is to find collision-free paths between the targets $c_i \in \mathcal{C}_{free}$, $i = 1, \ldots, n$. We propose a novel method, called Space-filling forest (SFF), to find these paths. The idea of SFF is to grow random trees $\mathcal{T}_i$, $i = 1, \ldots, n$ rooted at the target locations c_i. Let O denote an open-list containing nodes of the trees that are available for the expansion. At the beginning of the algorithm, the open-list O contains the root nodes of all trees. The trees are expanded until they approach each other, or touch the obstacles.

The method is listed in Alg. 1. In each iteration, the algorithm attempts to expand the trees. A node q from the open-list is selected randomly; let $\mathcal{T}$ denote the tree containing the node q. The task of the expansion is to find new point q_c that can be added to the tree $\mathcal{T}$. A random candidate $q_c \in \mathcal{C}$ is generated around q at the distance d, $0 < d \leq R$, where R defines the length of the expansion (planning resolution). The candidate q_c is added to the tree $\mathcal{T}$ (and also to the open-list), if the following three conditions are met: a) q_c does not approach other node in the same tree, i.e., if $\varrho(q_c, q'_c) > \varrho(q, q_c)$, where q'_c is the nearest node in $\mathcal{T}$ towards q_c; b) distance of q_c from all other trees is higher than the threshold d_{tree}; and c) q_c does not collide with the obstacles. The first condition ensures that the tree does not grow to itself, but towards unexplored regions of the configuration space. The second condition ensures that two trees do not explore the same region. The expansion step at each node q is repeated at most k times. If no new point q_c can be generated around q, the node q is deleted from the open-list. The algorithm terminates if the open-list is empty. Example of the growth of the trees is illustrated in Fig. 2.



(a) (b) (c) (d) (e)

Fig. 2: (a,b): SFF growth from seven targets (black). Each tree is depicted in a different color. Nodes in the open-list are in gray. (c): The fully growth trees resemble Voronoi cells of the targets. (d): paths connecting neighboring trees. (e) smoothed paths ($w = 5$).

Algorithm 1: Space-filling forest

Input: targets $c_1, \ldots, c_n \in \mathcal{C}_{free}$, num. of expansion attempts k, distance between trees d_{tree}, expansion step R
Output: paths P_{ij} between the targets

1 $\mathcal{T}_i$.addNode(c_i), $i = 1, \ldots, n$; // trees are rooted at the targets
2 O.append((c_i, i)), $i = 1, \ldots, n$; // node and index of its tree
3 $P_{ij} = \emptyset$, $\forall i, j = 1, \ldots, n$;
4 **while** *O is not empty* **do**
5 q, i = select random item from O;
6 $succ = false$;
7 **for** $trial = 1 : k$ **do**
8 q_c = random conf. around q such that $\varrho(q, q_c) \leq R$;
9 d_i, q' = $\mathcal{T}_i$.distanceToNearest(q_c);
10 $d_j, q_j = \text{argmin}_{j \neq i} \mathcal{T}_j$.distanceToNearest($q_c$) ;
11 **if** $d_j > d_{\text{tree}}$ **then**
12 **if** $d_i > \varrho(q, q_c)$ **and** *canConnect*(q, q_c) **then**
13 $\mathcal{T}_i$.addNode(q_c);
14 $\mathcal{T}_i$.addEdge(q, q_c);
15 O.add((q_c, i));
16 $succ = true$;
17 **break**;
18 **else**
19 **if** $P_{ij} = \emptyset$ **and** *canConnect*(q, q_j) **then**
20 $P_{ij} = \mathcal{T}_i.path(c_i, q) \cup \mathcal{T}_j.path(q_j, c_j)$;
21 **if not** *succ* **then**
22 remove item (q, i) from O;

When two trees $\mathcal{T}_i$ and $\mathcal{T}_j$ approach each other (line 19 in Alg. 1) and their nearest nodes can be connected by a collision-free line, the path P_{ij} between the targets i and j is created. The SFF thus finds a graph with paths only between the neighboring targets (Fig. 2d). This is sufficient for TSP where the short connections between targets are preferred. The trees are constructed without any optimality criteria, moreover, the nodes are expanded towards a random direction. Therefore, the paths found in the trees have many zig-zags and need to be smoothed, e.g. by repeated removal (Fig. 2d). The Alg. 2 shows a simple, yet effective, method to smooth the paths. The idea of the smoothing is to remove a node q_i, if its neighboring nodes q_{i-w} and q_{i+w} can be connected by a collision-free line, where w is a smoothing parameter. Example of the path smoothing is shown in Fig. 2e.

The result of the SFF is the graph, where nodes are the targets and edges are defined by the found paths P_{ij}. The distances between the cities $\varrho(c_i, c_j)$ are found in this graph e.g. using Dijkstra's algorithm and used by the TSP solver to find the final sequence of visiting the targets (Sec. III-A).

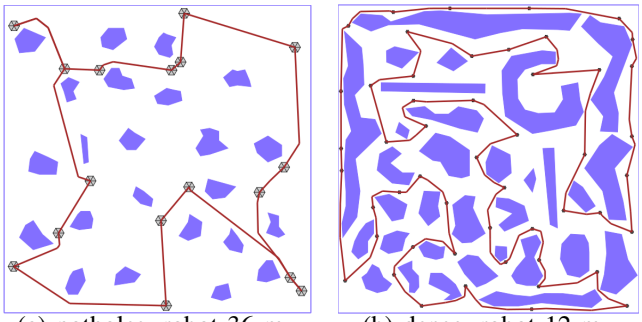
IV. Initial Results

The proposed SFF method was tested on *dense* and *potholes* 2D scenarios (Fig. 3), with 17 and 50 target locations, respectively, with approximate size of 2000×2000 map units (mu). The results were achieved using a single core of Intel Core i7-8650U CPU/8GB RAM. All instances were solved 10 times to obtain meaningful statistical results. Illustration of the SFF growth can be found at https://youtu.be/xTowHu7FYkM .

Algorithm 2: Path smoothing

Input: path $P = (q_1, \ldots, q_m)$, $q_i \in \mathcal{C}_{free}$ of m configurations, window size w
Output: smoothed path P as a sequence of configurations

1 **for** $i = w + 1 : m - w$ **do**
2 **if** *canConnect*(q_{i-w}, q_{i+w}) **then**
3 remove q_i from P;
4 $i = i + w$

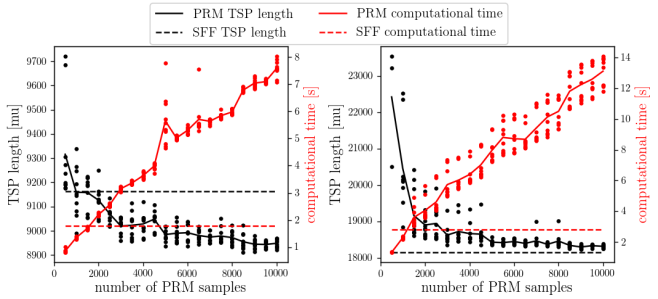


(a) potholes, robot 36 mu (b) dense, robot 12 mu

Fig. 3: Example TSP solutions with depicted size of hexagonal robot.

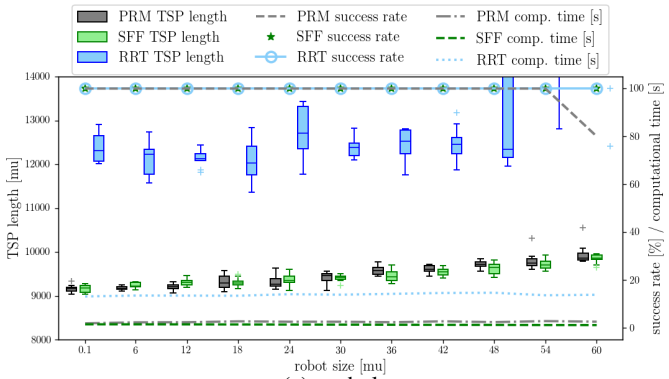
SFF was compared with asymptotically optimal PRM [5] and RRT [7] methods. SFF was run with parameters: $R = 1$ mu, $d_{\text{tree}} = 20$ mu, $w = 5$ and $k = 10$. RRT was used sequentially to obtain all collision-free paths between all pairs of target locations with the setup: planning resolution $\varepsilon = 1$ mu and number of iterations 10^4. The distances $\varrho(\cdot, \cdot)$ are measured using the 2D Euclidean metric. After the path planners find the paths between the targets, the TSP solution is achieved using the Concorde solver [1].

The first evaluation of the SFF focuses on the comparison with PRM with the hexagonal robot of size 0.1 mu for the increasing number of PRM samples. Figure 4 shows the comparison of runtime and TSP solution quality with displayed average performance of SFF method (constant with respect to PRM number of samples). The results show that for the relatively empty potholes scenario, the computational time of SFF is equal to the one of PRM with 1500 samples for which the TSP solution length of PRM and SFF are comparable. For the dense scenario with the higher density of obstacles and target locations, however, the SFF method outperforms PRM for all tested number of samples and even one order of magnitude higher runtime of PRM is not sufficient to find

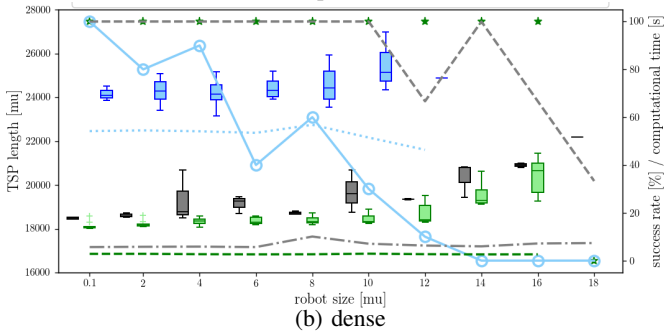


(a) potholes (b) dense

Fig. 4: Runtime and TSP length comparison for different PRM number of samples on both scenarios.



(a) potholes



(b) dense

Fig. 5: Runtime and TSP length comparison for different robot sizes.

better quality paths than in case of SFF.

The second evaluation compares the runtime, solution quality and success rates of SFF, PRM, and RRT for various robot sizes. The increased robot size induces narrow passages and thus highly influence both TSP solution lengths and success rate of considered methods. Figure 5a shows the results for the potholes scenario, where the PRM uses 1500 samples (requires same computational time as the SFF). The comparison in Fig. 5b on the dense dataset is performed for the same setup of both SFF and RRT, however, with 4000 samples of PRM to ensure sufficient success rate of PRM in dense scenario with narrow passages.

The comparison for different robot sizes shows that RRT method has high both computational time and TSP length compared to SFF and PRM. In the potholes scenario, SFF performs better only for the large robot sizes where the narrow passages start to appear. However, SFF outperforms other methods in the dense scenario. The computational time of SFF is always the smallest one and the success rate is the highest for all robot sizes with an exception of large 18 mu robot in dense scenario. Therefore, we consider the proposed algorithm as a valid method for quick finding of high-quality collision-free paths for multi-goal path planning such as the considered TSP, especially in the environments filled with the obstacles.

V. Conclusion

The paper presents a novel path planner for finding collision-free paths between multiple target locations. The planner simultaneously builds a forest of trees, each being rooted at one target. The trees are iteratively expanded until they touch each other or obstacles. The experiments have shown superior performance of our planner in environments with a high density of obstacles, and in scenarios with narrow passages (e.g., larger robots moving among the obstacles).

References

[1] Concorde tsp solver. http://www.math.uwaterloo.ca/tsp/concorde.html citied on 2019-06-07.

[2] D. Devaurs, T. Simon, and J. Corts. A multi-tree extension of the transition-based rrt: Application to ordering-and-pathfinding problems in continuous cost spaces. In *IEEE/RSJ IROS*, pages 2991–2996, 2014.

[3] S. Edelkamp, M. Pomarlan, and E. Plaku. Multiregion inspection by combining clustered traveling salesman tours with sampling-based motion planning. *IEEE Robotics and Automation Letters*, 2(2):428–435, April 2017.

[4] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *International Journal of Computational Geometry and Applications*, volume 3, pages 2719–2726, 1997.

[5] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Tr. on Robotics and Automation*, 12:566–580, 1996.

[7] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998. Technical report 98-11.

[8] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[9] T.-Y. Li and Y.-C. Shie. An incremental learning approach to motion planning with roadmap management. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3411–3416, 2002.

[10] S. R. Lindemann and S. M. LaValle. Current issues in sampling-based motion planning. In *Robotics Research: The Eleventh International Symposium*, pages 36–54, 2005.

[11] J. McMahon and E. Plaku. Autonomous underwater vehicle mine countermeasures mission planning via the physical traveling salesman problem. In *OCEANS - MTS/IEEE*, pages 1–5, 2015.

[12] J. McMahon and E. Plaku. Mission and motion planning for autonomous underwater vehicles operating in spatially and temporally complex environments. *IEEE J. of Oceanic Engineering*, 41(4):893–912, 2016.

[13] P. Oberlin, S. Rathinam, and S. Darbha. Today's traveling salesman problem. *IEEE Robotics Automation Magazine*, 17(4):70–77, Dec 2010.

[14] K. J. Obermeyer. Path planning for a UAV performing reconnaissance of static ground targets in terrain. In *AIAA Conf. on Guidance, Navigation and Control*, Chicago, IL, USA, August 2009.

[15] R. Pěnička, J. Faigl, M. Saska, and P. Váňa. Data collection planning with non-zero sensing distance for a budget and curvature constrained unmanned aerial vehicle. *Autonomous Robots*, 2019.

[16] D. Perez, P. Rohlfshagen, and S. M. Lucas. The physical travelling salesman problem: Wcci 2012 competition. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[17] W. Wang and Y. Li. A multi-RRTs framework for robot path planning in high-dimensional configuration space with narrow passages. In *International Conference on Mechatronics and Automation*, pages 4952–4957, 2009.