Planning Optimal Collision-Free Trajectories with Non-Convex Cost Functions Using Graphs of Convex Sets

Charles L. Clark and Biyun Xie

I. Introduction

Recently, an effective motion planning framework called motion planning in graphs of convex sets (GCS) was developed in [1] to plan collision-free trajectories while optimizing the time, length, and energy. To guarantee collision avoidance, obstacles in the task space are transferred to the joint space using nonlinear optimization to grow collision-free polytopes in the joint space. To accomplish this, a two step optimization approach is repeated until it converges. The first step is to add hyperplanes to a polytope to refine its shape based on nearby obstacles in the task space. The second step involves maximizing the volume of an ellipse inside of the collision-free polytope from the previous step [2]. This process is repeated to produce a set of collision-free polytopes that cover large portions of the joint space. To plan optimal paths within these collision-free regions, the GCS first builds a graph to represent the possible collision-free paths in the joint space. Unlike traditional graph-based motion planning, the vertices in the GCS framework are dynamic points inside the intersection of the collision-free regions. The edges in the graph connect vertices that share a collision-free region, and thus represent all of the collision-free paths from a starting point to a target point. After building the graph, the GCS framework minimizes the cost of each edge by adjusting their vertices while also solving for the probability that a given edge belongs to the optimal path. These probabilities and the costs associated with each edge are then used to determine the optimal path.

A fundamental drawback of the GCS planner is its inability to handle non-convex cost functions. Additionally, if the graph structure contains a cycles or large numbers of edges, the quality of the paths produced can severely decline. Therefore, this work is focused on extending the GCS framework to include non-convex cost functions as well as establishing methods that can improve the underlying graph structure of the GCS planner. The main contributions of this work are as follows: (1) Developing a method based on ReLU neural networks to produce sets of locally linear cost functions to approximate a given non-convex cost function, (2) Removing cycles and high cost paths from the underlying graph structure of the GCS planner by using a graph processing technique.

This work was supported by the NSF Foundation under Grant #2205292.

Charles L. Clark and Biyun Xie are with the Electrical and Computer Engineering Department, University of Kentucky, Lexington, KY 40506, USA landon.clark@uky.edu, biyun.xie@uky.edu

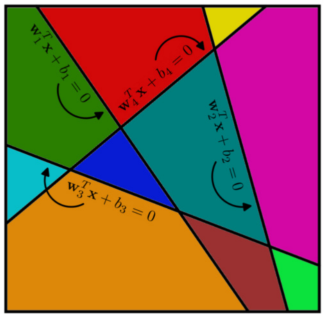
$\mathbf{w}_1^T\mathbf{x}+b_1=0$
$\mathbf{w}_4^T\mathbf{x}+b_4=0$
$\mathbf{w}_2^T\mathbf{x}+b_2=0$
$\mathbf{w}_3^T\mathbf{x}+b_3=0$

Fig. 1. An example set of convex decompositions of a non-convex cost function is shown.

II. Methods

A. Approximating Non-Convex Functions as Disjoint Linear Functions

Convexity is one of the most important properties in optimization because problems with convex costs and constraints can be solved consistently and efficiently to global optimality [3]. To optimize non-convex cost functions alongside the GCS planner, the transformation of a non-convex function into many locally convex functions is proposed. This can be realized by using a feedforward neural network with the ReLU activation function to learn a piecewise linear approximation of a given function. The neural network can be decomposed into disjoint convex regions, inside of which the approximated function is linear [4]. An example set of convex decompositions of a non-convex cost function is given in Fig. 1, where each colored region represents a locally linear cost function. Collision-free regions with linear costs are obtained by intersecting the neural network's convex regions with the collision free polytopes from the original GCS formulation. The graph of convex sets used for the motion planning problem is constructed such that its vertices represent a point on the boundary of two adjacent collision-free linear regions, and edges are made between vertices exist on the boundary of the same region.

While the cost function inside of each linear region is a convex function, the actual cost of the trajectory travelling through these regions is not necessarily convex. This is because we must consider the time spent in each region, where this time is not a constant and is associated with decision variables of the optimization problem, such as speed. When time is made to scale with the distance of the path inside of each region with a constant speed, this resulting cost function is bilinear, which means it is very unlikely to be convex. To

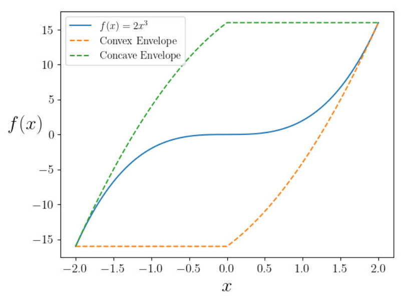

Fig. 2. An example of the McCormick relaxation applied to the non-convex function $f(x) = 2x^3$ is shown.

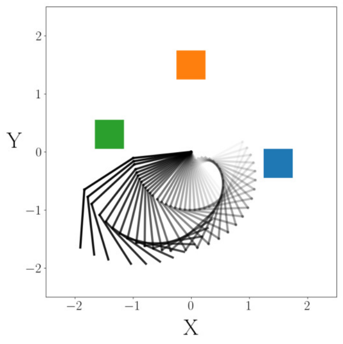

Fig. 3. An equal link length 3R robot following a collision-free trajectory generated by the proposed method is shown.

convert the cost of the entire trajectory to be convex, the McCormick relaxation is applied. This relaxation converts a bilinear function into a linear function that is constrained within the convex and concave envelopes of the original bilinear term. An example of the McCormick relaxation is shown in Fig. 2, where the non-convex function $f(x) = 2x^3$ is bounded by its convex and concave envelopes, the orange and grenn dotted curves respectively. Once the non-convex cost function is transformed into a set of disjoint linear functions and the cost of the trajectory is also convexified, the GCS planner can be used to solve for optimal cost collision-free paths using convex optimization.

B. Removing Cycles and High Cost Paths

Because this initial graph structure can contain cycles and many high cost paths that negatively effect the performance of the GCS planner, it is essential that some form of graph pre-processing be done to refine the underlying graph structure. Both traditional and optimization-based methods can be used, each with important benefits and drawbacks [5]. This work utilizes traditional methods of graph processing to solve for the k-shortest paths, which will return k low cost cycle-less paths from the original graph. Yen's algorithm is used to solve this problem as it provides a good balance between computational efficiency and memory overhead.

III. Results

The above proposed motion planning algorithm is evaluated on a planar 3R robot with equal link lengths. The starting point and target point are randomly selected, and there exists three obstacles in the task space, as shown in Fig. 3. The algorithm is applied to compute an optimal trajectory to avoid the three obstacles and while maximizing the worst-case failure tolerance cost function along the trajectory. The worst-case failure tolerance is defined as the minimum of all the possible minimum singular values of the Jacobian after a locked joint failure, which is a non-convex function and represents the worst-case motion ability of the robot after an arbitrary locked joint failure. Fig. 4(a) and Fig. 4(b) show the computed joint trajectory projected onto the θ_1-θ_2 plane and θ_2-θ_3 plane, respectively, and Fig. 3 shows the corresponding end-effector trajectory in the task space. The total cost function value of the computed trajectory is 5.332. The k-shortest path finding took approximately 6 seconds, and the convex optimization used to compute the optimal trajectory completed in 0.007 seconds.

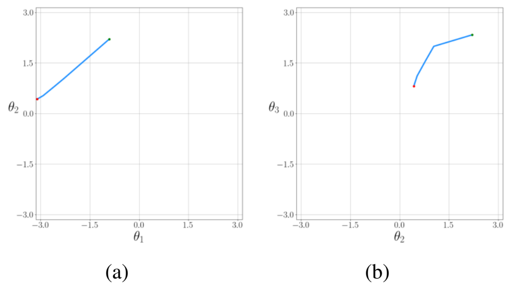

Fig. 4. A trajectory generated by the proposed method is shown. The trajectory is projected onto the θ_1-θ_2 plane in (a) and θ_2-θ_3 plane in (b).

IV. Conclusion and Future Work

A novel motion planning algorithm is developed to compute a collision-free trajectory while optimizing non-convex cost functions based on graph of convex sets. This motion planning algorithm is evaluated on a planar 3R robot with equal link lengths. The results show that the proposed method is capable of producing high quality paths in the presence of obstacles. In future work, this algorithm will be applied to higher degrees of freedom robots, such as spatial 4R and spatial 7R robots, to further prove the effectiveness of this method. In addition, this algorithms will be compared with other motion planning algorithms with optimization abilities, such as PRM* and RRT*, in terms of computational efficiency and optimality of the results.

References

[1] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *arXiv preprint arXiv:2205.04422*, 2022.

[2] M. Petersen and R. Tedrake, "Growing convex collision-free regions in configuration space using nonlinear programming," *arXiv preprint arXiv:2303.14737*, 2023.

[3] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[4] Q. Tao, L. Li, X. Huang, X. Xi, S. Wang, and J. A. Suykens, "Piecewise linear neural networks and deep learning," *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 42, 2022.

[5] L. Taccari, "Integer programming formulations for the elementary shortest path problem," *European Journal of Operational Research*, vol. 252, no. 1, pp. 122–130, 2016.