Optimal Path Planning on a Semi-Dynamic Subdivision Graph*

Yi-Jen Chiang†, David Kirkpatrick§, Chee Yap‡, and Zhaoqi Zhang‡

†Department of Computer Science & Engineering, Tandon, NYU, New York, NY, USA
§Department of Computer Science, University of British Columbia, Vancouver, Canada
‡Department of Computer Science, Courant, NYU, New York, NY, USA

Abstract

Soft Subdivision Search (SSS) is a framework for implementing path planning algorithms in robotics. It has a theoretically rigorous foundation and yet has proven to be practical and efficient. Until now, there is no optimality guarantee on the returned path. Standard algorithms to compute optimal (shortest) paths in graphs are based on Dijkstra's or A-star algorithms. But the graph produced by SSS is semi-dynamic in the sense that it evolves by adding new vertices and new edges. Adapting Dijkstra or A-star to this setting is novel and challenging. We introduce an SSS-based algorithm for the case where the robot is a disc, and discuss the prospects for generalization.

1 Introduction

Beginning in the 1980s, algorithmic path planning has a rigorous foundation using algebraic algorithms [9, 6, 3, 1]. In computational geometry, exact planners were designed for various robots (mostly planar robots) such as a disc, rods, robot arms, multiple discs, etc. However, the implementations of such algorithms are rarely exact except for those implemented using an "exact library" such as LEDA, CGAL or Core [2, 10]. But the use of "exact libraries" is too expensive for most applications. Instead, most roboticists prefer to implement their exact algorithms using machine precision, which immediately loses their a priori guarantees of correctness. To overcome this limitation of exact algorithms, it became popular to replace exact algorithms by randomized sampling method such as PRM or RRT [5, 7]. However, the guarantees of such algorithms are provided by "convergence theorems" whose conditions are often unverifiable.

Starting in [11], we introduced a rigorous "soft foundation" for path planning based on the Subdivision Paradigm. The novelty consists in our definition of **resolution-exactness** as a new correctness criteria for path planning, and our introduction of soft predicates for achieving resolution-exactness. We call our framework the **Soft Subdivision Search** or SSS. Moreover, as shown by a series of papers [11, 8, 12, 13, 4], we were able to implement our algorithms for a variety of robots and exceed the performance of the state-of-the-art sampling algorithms. In [4], our method provided the first rigorously implemented algorithm for 5-DOF (5 degrees of freedom) spatial robots (rod robot and ring robot in 3D).

Previous SSS algorithms were contented to just find *any* path. In the present paper, we address the problem of finding the *shortest* path in the SSS framework. In the exact setting, this is essentially

*Supported in part by NSF grant CCF-2008768. Author Email: chiang@nyu.edu; kirk@cs.ubc.ca; yap@cs.nyu.edu; zz1918@nyu.edu.

a form of Dijkstra's algorithm. But as we shall see, this is considerably more subtle in the soft setting of resolution-exactness.

1.1 The Problem of Semi-Dynamic Shortest Path for a Disc

Consider a disc robot with radius $r_0 > 0$. The configuration space of this robot is $\mathcal{C}space = \mathbb{R}^2$. The input to our path planning problem is a 5-tuple

$$(B_0, \Omega, s, t, \varepsilon) \tag{1}$$

where $B_0 \subseteq \mathcal{C}space$ is an axis-aligned box called the region-of-interest (ROI), $\Omega \subseteq \mathbb{R}^2$ is a polygonal obstacle set, $s, t \in \mathcal{C}space$ are the start and target configurations, and $\varepsilon > 0$. The **free space** $\mathcal{C}_{free} = \mathcal{C}_{free}(\Omega)$ is the set $\{\gamma \in \mathcal{C}space : \Delta(\gamma, r_0) \cap \Omega = \emptyset\}$ where $\Delta(\gamma, r_0)$ is the disc centered at γ of radius r_0. A solution to the input (1) is a path π from s to t restricted to B_0, i.e., $\pi : [0, 1] \to B_0 \cap \mathcal{C}_{free}$ is a continuous function with $\pi(0) = s$ and $\pi(1) = t$.

In this paper, we call π an **ℓ_1-path** if the range of π is a finite union of horizontal and vertical line segments. Let $\Pi_1(s, t, \varepsilon)$ denote the set of all ℓ_1-paths from s to t in which each line segment has length at least ε.

Given a subdivision $\mathcal{S}$, the **skeleton graph** $G_{\mathcal{S}}$ of $\mathcal{S}$ is an undirected graph $G_{\mathcal{S}} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ whose vertices $v \in V_{\mathcal{S}}$ are the corners of boxes in $\mathcal{S}$. We also identify v with a point of $\mathbb{R}^2$. Each edge $(u, v) \in E_{\mathcal{S}}$ corresponds to a horizontal or vertical line segment $[u, v]$ contained in the boundary ∂B of some box $B \in \mathcal{S}$. Moreover, the cost $cost(u, v)$ is just the ℓ_1 distance $\|u - v\|_1$.

Let $C : \mathcal{S} \to \{G, Y, R\}$ be a **coloring** of the boxes in $\mathcal{S}$ into Green/Yellow/Red. We say C is **admissible** if no red box can be adjacent to a green box. This coloring induces a coloring of the vertices and edges of $G_{\mathcal{S}}$ as follows: $C : (V_{\mathcal{S}} \cup E_{\mathcal{S}}) \to \{G, Y, R\}$ where

$$C(v) = \begin{cases} C(B) & \text{if } v \in \partial B \text{ and } C(B) \neq Y, \\ Y & \text{else.} \end{cases} \tag{2}$$

$$C(u, v) = \begin{cases} C(B) & \text{if } [u, v] \subseteq \partial B \text{ and } C(B) \neq Y, \\ Y & \text{else.} \end{cases} \tag{3}$$

See Figure 1.

For simplicity, we assume that s, t are vertices in $V_{\mathcal{S}}$ (it is easy to modify if this assumption fails). We are interested in computing the shortest green path from s to t. Here we define "shortest" to be in the ℓ_1-norm sense. We could use Dijkstra's algorithm or any A-star variant to solve this problem.

What is new is the following twist: the subdivision $\mathcal{S}$ is, in reality, produced by our SSS algorithm. The main issue is how to modify the graph $G_{\mathcal{S}}$ as $\mathcal{S}$ evolves. We call $G_{\mathcal{S}}$ a **semi-dynamic graph** in the sense that we only add new vertices to $V_{\mathcal{S}}$, but never delete vertices. Moreover, what is guaranteed about the "shortest path" produced by such an algorithm?

1.2 Review of Basic Concepts

We briefly review the basic concepts in subdivision path planning (e.g., see [11]). Fix a box $B_0 \subseteq \mathbb{R}^2$. A subdivision tree T rooted in B_0 is a finite tree in which each node of T is a box $B \subseteq B_0$ such that either B is the root or else, B is obtained by splitting its parent B' into four congruent children.

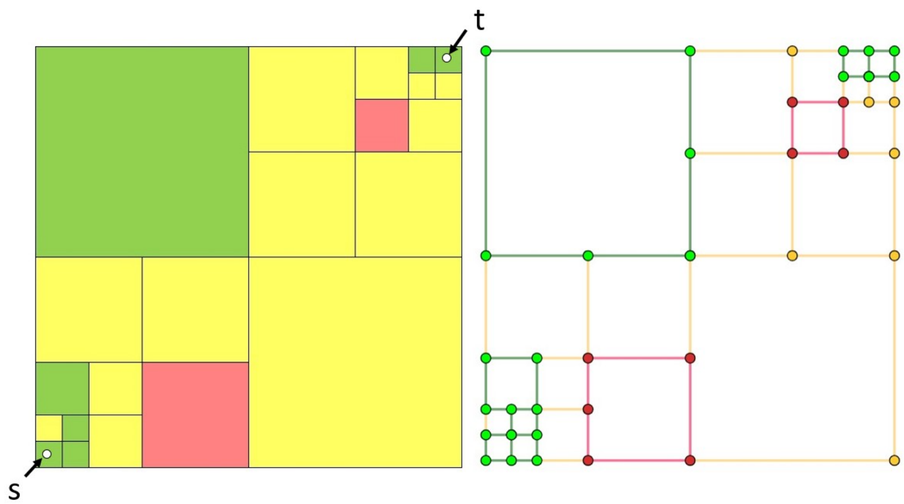

Figure 1: **(Left)** Subdivision $\mathcal{S}$; **(Right)** Skeleton graph $G_{\mathcal{S}}$.

The set $\mathcal{S} = \mathcal{S}(T)$ of leaves of T is called a **subdivision** of B_0. Two boxes $B, B' \in \mathcal{S}$ are **adjacent** if their boundaries intersect in an interval $(\partial B) \cap (\partial B')$ of positive length.

In the context of path planning, B_0 is a set of the configuration space of a planar disc robot. Let $\mathcal{S}$ be a subdivision of B_0. A valid $C : \mathcal{S} \to \{G, Y, R\}$ is one that guarantees that every point in a G-box (resp. R-box) represents a `FREE` (resp., `STUCK`) configuration of a robot. We do not guarantee anything for points in a Y-box. Let s, t be two `FREE` configurations in B_0. Then $Box(s) = Box(s; \mathcal{S})$ is any box in $\mathcal{S}$ that contains s.

2 Approximate Optimal-Path Algorithm

To focus on the main algorithm, we shall assume that the input is a 4-tuple $(\mathcal{S}, s, t, \varepsilon)$ where $\mathcal{S}$ is a subdivision with an admissible coloring in which $Box(s)$ and $Box(t)$ are green, and $\varepsilon > 0$.

The main loop of our algorithm consists of two nested while-loops: the outer while-loop is controlled by a queue Q of *fringe* boxes (which are yellow; to be defined in the algorithm next). While Q is non-empty, we take a fringe box and split it. This produces new vertices that are put into another queue Q'. The inner while-loop is controlled by Q', and it basically executes Dijkstra's algorithm to propagate the d-values of the vertices in Q'.

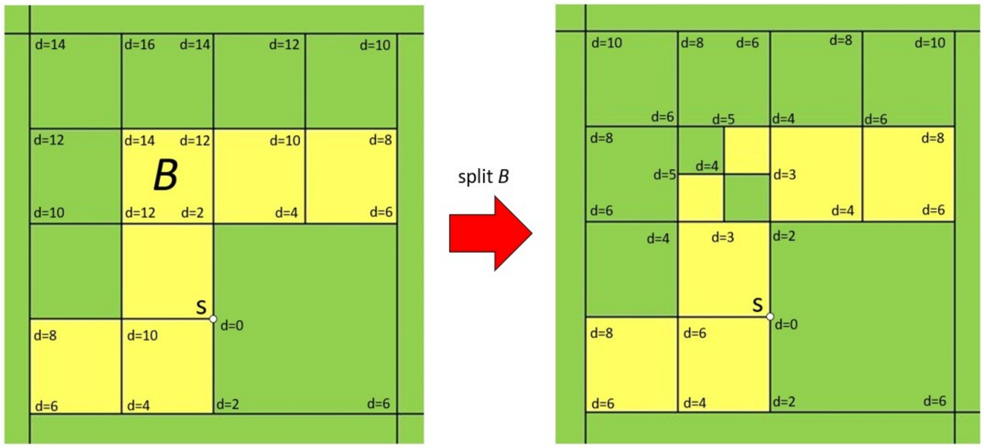

Figure 2: **(Left)** A fringe box B that is going to be split; **(Right)** Updated d-values after the split.

Approximate Optimal-Path Algorithm:
INPUT: $(\mathcal{S}, s, t, \varepsilon)$
OUTPUT: `NO-PATH` or an "approximate" ℓ_1-optimal path between s and t
with path length no larger than the shortest path of clearance $\geq K'\varepsilon$.
▷ *I. Setup Phase*
Initialize the function $d : V_{\mathcal{S}} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ where

$$d(v) = \begin{cases} \|s - v\|_1 & \text{if } v \text{ is a vertex of } Box(s) \\ \infty & \text{else} \end{cases}$$

(Run Dijkstra's algorithm on the graph $(G_{\mathcal{S}})_{\texttt{green}}$ using the d-function.)
▷ *II. Main Loop*
Initialize the queue Q to contain all the *fringe* boxes, where
we define $S_{\mathcal{S}}$, called the *settled set*, to be $\{v :$ there is a path of green edges from s to $v\}$,
and a box $B \in \mathcal{S}$ is defined to be **fringe** if $C(B) = \texttt{yellow}$ and $S_{\mathcal{S}} \cap \partial B \neq \emptyset$.
While $Q \neq \emptyset$
$B \leftarrow Q.getNext()$
$\mathcal{S}.add(split(B))$ and "color" the children of B
Update $G_{\mathcal{S}}$.
▷ *Update the d-function of $G_{\mathcal{S}}$:*
Let $d(v) = \infty$ if v is a new vertex.
Initialize new queue Q' to contain the set $S_{\mathcal{S}} \cap \partial B$.
While $Q' \neq \emptyset$
$v \leftarrow Q'.getMin()$ ◁ *$d(v)$ is minimum*
For each u adjacent to v
If $(d(u) = \infty)$
add to Q any yellow box B with $u \in \partial B$ ◁ *B is a new fringe box*
If $(d(u) > d(v) + cost(v, u))$
$d(u) \leftarrow d(v) + cost(v, u)$ ◁ *Update $d(u)$ in $G_{\mathcal{S}}$*
If (u is not in Q') $Q'.add(u)$ with key $d(u)$
Else $Q'.decrease\_key(u, d(v) + cost(v, u))$ ◁ *Decrease the key of u in Q'*
If $(d(t) = \infty)$ output `NO-PATH`
Else return $d(t)$ and the corresponding path between s and t

CONJECTURE: *If this algorithm outputs a path π, then π satisfies*

$$\ell_1(\pi) \leq \min\left\{\ell_1(\pi') : \pi' \text{ is a path from } s \text{ to } t \text{ with clearance} \geq K'\varepsilon.\right\}$$

Here $K' = O(K)$ with K being the constant associated with the resolution-exact SSS algorithm.

3 Conclusion and Future Work

- This is the first effort to produce an (approximate) optimal path in the soft setting of SSS.
- We can easily turn this Dijkstra-type algorithm into an A-star algorithm by adding a heuristic function $h(v)$ that is a lower bound on the ℓ_1-distance from v to t.
- A trivial lower bound to be used for $h(v)$ is simply $\|v - t\|_1$. But a more sophisticated lower bound can be obtained by the d-function from the vertex t using both green and yellow edges.
- For correctness of the algorithm, the $Q.getNext()$ is unrestricted. However, we plan to implement various heuristics (e.g., breadth first search, random, greedy best first, etc.) to understand the best heuristic.

References

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Boston, 2005.

[2] D. Halperin, E. Fogel, and R. Wein. *CGAL Arrangements and Their Applications.* Springer-Verlag, Berlin and Heidelberg, 2012.

[3] D. Halperin, O. Salzman, and M. Sharir. Algorithmic motion planning. In J. E. Goodman, J. O'Rourke, and C. Toth, editors, *Handbook of Discrete and Computational Geometry*, chapter 50. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017. Expanded from second edition.

[4] C.-H. Hsu, Y.-J. Chiang, and C. Yap. Rods and rings: Soft subdivision planner for **R**ˆ3 x **S**ˆ2. In *Proc. 35th Symp. on Comp. Geometry (SoCG 2019)*, pages 43:1–43:17, 2019.

[5] L. Kavraki, P. Švestka, C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robotics and Automation*, 12(4):566–580, 1996.

[6] S. M. LaValle. *Planning Algorithms.* Cambridge University Press, Cambridge, 2006.

[7] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2002. Original RRT paper.

[8] Z. Luo, Y.-J. Chiang, J.-M. Lien, and C. Yap. Resolution exact algorithms for link robots. In *Proc. 11th Intl. Workshop on Algorithmic Foundations of Robotics (WAFR '14)*, volume 107 of *Springer Tracts in Advanced Robotics (STAR)*, pages 353–370, 2015. Aug. 3-5, 2014, Boğazici University, Istanbul, Turkey.

[9] J. T. Schwartz and M. Sharir. On the piano movers' problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[10] V. Sharma and C. K. Yap. Robust geometric computation. In J. E. Goodman, J. O'Rourke, and C. Tóth, editors, *Handbook of Discrete and Computational Geometry*, chapter 45, pages 1189–1224. Chapman & Hall/CRC, Boca Raton, FL, 3rd edition, 2017.

[11] C. Wang, Y.-J. Chiang, and C. Yap. On soft predicates in subdivision motion planning. *Comput. Geometry: Theory and Appl. (Special Issue for SoCG'13)*, 48(8):589–605, Sept. 2015.

[12] C. Yap, Z. Luo, and C.-H. Hsu. Resolution-exact planner for thick non-crossing 2-link robots. In K. Goldberg, P. Abbeel, K. Bekris, and L. Miller, editors, *Algorithmic Foundations of Robotics XII: Proc. 12th WAFR 2016*, Springer Proceedings in Advanced Robotics, pages 576–591. Springer, 2020. (WAFR 2016: Dec. 13-16, 2016, San Francisco.) Book link: `https://www.springer.com/gp/book/9783030430887`. For proofs and more experimental data, see `arXiv:1704.05123 [cs.CG]`.

[13] B. Zhou, Y.-J. Chiang, and C. Yap. Soft subdivision motion planning for complex planar robots. *Comput. Geometry: Theory and Appl.*, 92, 101683, Jan. 2021. (Conference version appeared in *Proc. 26th European Symp. on Algorithms (ESA 2018)*, pp. 73:1–73:14, 2018).