Complexity of Planning

Kiril Solovey

Synonyms

Computational complexity, time complexity.

Definitions

Complexity analysis of complete algorithms for robot motion planning.

Overview

This chapter is devoted to the study of complexity of complete (or exact) algorithms for robot motion planning. The term "complete" indicates that an approach is guaranteed to find the correct solution (a motion path or trajectory in our setting), or to report that none exists otherwise (in case that for instance, no feasible path exists). Complexity theory is a fundamental tool in computer science for analyzing the performance of algorithms, in terms of the amount of resources they require. (While complexity can express different quantities such as space and communication effort, our focus in this chapter is on time complexity.) Moreover, complexity theory helps to identify "hard" problems which require excessive amount of computation time to solve. In the context of motion planning, complexity theory can come in handy in various ways, some of which are illustrated here.

––––––––––––––––––––

Kiril Solovey (kirilsol.github.io)

Department of Aeronautics and Astronautics, Stanford University, CA, USA, e-mail: `kirilsol@stanford.edu`

First, when designing a motion planner, complexity analysis can predict the execution time of the algorithm as a function of the problem's input size. Importantly, this analysis is performed prior to the deployment of the algorithm in a real-world setting with physical robots. Additionally, it can be used to identify the most time-consuming components in a proposed method, which can then be replaced with more efficient subroutines, if available. Furthermore, complexity analysis can be used for qualitative comparison between a number of proposed methods, thus choosing the one with the lowest execution time.

Secondly, from a more theoretical perspective, complexity theory can assist in identifying particularly challenging problems in motion planning. That is, for those "hard" problems (to be defined more precisely below), all algorithms are doomed to run for a tremendously long time for certain instances of the problem, even for seemingly easy scenarios, e.g., those involving a few obstacles and a simple robot representation. One such problem is finding the shortest collision-free path for a rigid-body translating robot in a three-dimensional space amid static obstacles. In fact, many variants of motion planning are known to be hard.

From a practical standpoint, the fact that a given problem is computationally hard can obviously be seen as bad news to the practitioner tackling it. However, this knowledge can suggest a few alternative lines of attack. First, there may be a "relaxed" variant of the problem that can solved efficiently, which meets practitioner's requirements. For instance, it may be the case that finding a short solution path, i.e., not necessarily the shortest possible, requires more modest amount of resources. Secondly, it may be useful to relax the completeness requirements of the algorithm and consider approaches that have milder theoretical guarantees. For instance, search-based methods (Cohen et al 2014) typically discretize the motion-planning problem into a grid according to a certain resolution. While such techniques by definition cannot be complete, they are widely used in practice due to their simplicity. Another popular method is sampling-based planners, which approximate the structure of the problem via random sampling. While the latter approach is incomplete as well, it often come with the guarantee that a solution will be found eventually, i.e., when the number of samples is large enough. See more information on sampling-based planners in (Halperin et al 2016a), and the chapters "Sampling-Based Roadmap Planners" and "Sampling-Based Tree Planners".

Lastly we wish to clarify that the hardness results do not necessarily imply that all the inputs to a given problem are hard, only some of them. Consider for instance the problem of integer linear programming that is widely used in the world of engineering. It is known to be computationally hard, but in practice many instances of the problem, including those involving tens of thousands of variables, can be solved rather quickly.

In this chapter we will cover results both for the single-robot and the multi-robot settings. We will put special emphasis on the latter due to the fact that it has been more actively studied in recent years, and since additional complexity results for the single-robot setting are covered in the chapter on "Roadmaps".

Key Research Findings

This section is dedicated to basic aspects of complexity theory.

Elements of Computational Complexity

The time complexity of an algorithm measures the amount of time required to achieve a solution for a given problem. As the notion of time highly depends on the system on which the algorithm is run, it is often more convenient and informative to measure the execution time by the amount of elementary operations (or basic steps) performed by the algorithm. Generally, it is desirable to understand how the size of the input for the problem affects the running time. The input size expresses the amount of information necessary to represent the input. For instance, for the problem of sorting an array of values, the size of input $n \in \mathbb{N}$ would often represent the size of list. In the motion-planning problem depicted in Figure 1, the input size consists of two variables, which represent the complexity of the robot and workspace, respectively.
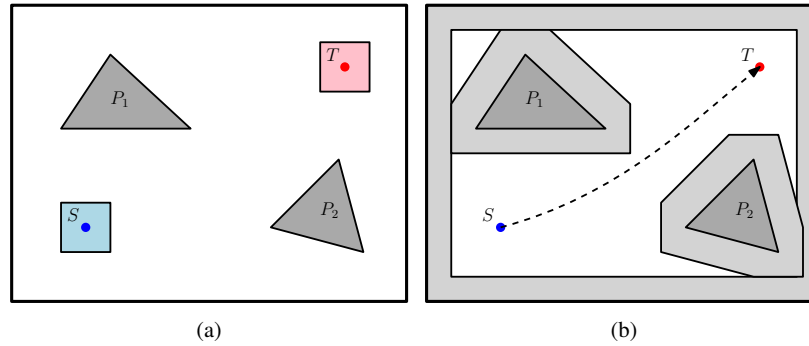

(a)   (b)

Fig. 1: Motion planning for a square robot translating in the plane. In (a) the start and target configurations S, T, are denoted by the blue and pink squares, respectively. The robot is confined to a room (black border) which contains two triangular obstacles (gray). The input size in this example consists of two variables, where $m = 4$ denotes the complexity of the robot, and $n = 3 + 3 + 4 = 10$ denotes the complexity (or the number of corners) of the obstacles. In (b), the forbidden regions of the configurations space are drawn in gray, whereas the white region is the free space. Observe that the depicted path is entirely collision free.

It is often difficult to nail down the precise running time of the algorithm with respect to the input size. Instead, an asymptotic expression capturing the performance for a sufficiently large input is typically derived. In particular, big O notation (as

well as related notations Θ, Ω) is used to characterize the running time according to its growth rate with respect to the input. The expression $O(f(n))$, where $f(n)$ is a function that depends on the size of the input n, indicates that there exists a constant c such that for n large enough the running time of the algorithm is at most $c \cdot f(n)$. For instance, in the context of sorting an array of values, the bubble sort algorithm runs in time $O(n^2)$, whereas merge sort runs in time $O(n \log n)$. Thus, the latter algorithm is considered to be more efficient than the former. We mention that a complexity analysis of a given algorithm typically entails a meticulous study of its different ingredients. See Cormen et al (2009) for more details.

Hardness of computation

As hinted earlier, the majority of motion-planning problems are in fact computationally hard. Generally speaking, problems whose solution requires super-polynomial number of steps, e.g., an exponent whose mantissa is the input size, are considered to be hard. Thus, it is important to establish whether a given problem admits a polynomial-time solution or not. For the brevity of exposition, we chose to omit most details concerning the classification of problems into easy and hard problems. We do mention that there exists a whole hierarchy between hard problems, and that the study of computational complexity is related to the famous problem of P vs NP (see Wikipedia (2019)). In the remainder of this chapter we will only distinguish between problems that are in P (of polynomial complexity) and two categories devoted to hard problems, namely NP-hard (or NP-complete) and PSPACE-hard (or PSPACE-complete), where the latter is considered to be more computationally demanding. The precise description of NP- and PSPACE-hardness requires several more definitions, and is beyond the scope of this chapter. For our purposes it would be sufficient to interpret problems in those two categories as requiring exponential running time (unless a widely accepted conjecture is false). For a thorough treatment of this subject see (Arora and Barak 2009).

Hardness proofs are typically established via a technique of polynomial-time reduction. To prove that a given problem A is computationally hard we rely on another problem B that is already known to be hard. The approach requires to devise a reduction that transforms in polynomial time any input of B to an input to A, such that the input to B is feasible (i.e., has a solution) if and only if the reduced input to A is feasible. A polynomial-time reduction proves that if no efficient feasibility algorithm for B exists, then none exists for A, as otherwise it would be possible to test the feasibility of B by reducing it into A and test the feasibility of the latter.

There is a great variety of problems that are already known to be hard, and which can be used as our "B" problem for reduction purposes. A canonical example for a NP-hard problem is the 3SAT problem, which is concerned with finding a satisfying assignment to a Boolean expression consisting of binary variables $x_1, \ldots, x_n$, for some $n \in \mathbb{N}_+$. The expression is of the form $C_1 \wedge C_2 \wedge \ldots \wedge C_k$, where $\wedge$ is an "and" operator, and every C_i represents a clause that consists of an "or" operator $\vee$ between

three of the above variables or their negation. Consider for instance the following 3SAT expression with two clauses:

$$\phi := (x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3).$$

While it is easy to find a satisfying assignment for ϕ, e.g., $x_1 := \text{true}, x_2 := \text{true}, x_3 := \text{true}$, in general, it is NP-hard to determine if a satisfying assignments exists when n, the number of variables, is no longer a constant.

Many hardness proofs use reductions from 3SAT, including the hardness of shortest-path motion planning in 3D, that we mention below. While most such proofs are beyond the scope of this chapter, we provide a simple proof for the NP-hardness of integer linear programming (ILP) to illustrate the reduction technique. Recall that ILP consists of finding an integer assignment to a set of variables that satisfies a set of linear inequalities. Given an instance of a 3SAT problem, we transform it into ILP in the following manner. For every Boolean variable x_i we assign an integer variable z_i that can take values from $\{0, 1\}$. We represent a clause C_i as a linear inequality constraints which requires that the sum between the corresponding variables, or their negation, will be at least 1. In particular, if a variable x_j appears in its original form in the clause, we will include the variable z_j in the sum, and otherwise we will include $(1 - z_j)$. For example, the above 3SAT expression is reduced into the following two constraints:

$$z_1 + z_2 + z_3 \geq 1, \quad (1 - z_1) + (1 - z_2) + z_3 \geq 1.$$

It is straight forward to verify that this reduction applies to any 3SAT expression. In particular, a 3SAT expression has a satisfying assignment if and only if the reduced ILP problem has a solution. This proves that ILP is NP-hard as well.

Examples of Applications

In this section we provide a more detailed account of complete approaches for motion planning of a single robot as well as for systems involving multiple robots.

Basic Motion Planning

In its most basic form, motion planning consists of finding collision-free paths for a robot in a (two or three-dimensional) workspace cluttered with static obstacles. The spatial pose of the robot, or its configuration, is uniquely defined by its degrees of freedom (DoF). The set of all configurations $\mathcal{C}$ is termed the configuration space of the robot, and decomposes into the disjoint sets of free and forbidden configurations, namely $\mathcal{F} \subseteq \mathcal{C}$ and $\mathcal{C} \setminus \mathcal{F}$, respectively. Thus, given start and target configurations,

the problem can be restated as the task of finding a continuous curve in $\mathcal{F}$ connecting the two configurations. See Figure 1.

The robot's number of DoFs, denoted by $d \in \mathbb{N}_+$, is possibly the most crucial parameter in determining the complexity of the problem. Most complete algorithms for motion planning explicitly construct and maintain the robot's free space $\mathcal{F}$, and the dimension of this space is directly determined by d. Furthermore, the complexity incurred in representing $\mathcal{F}$ is typically exponential in d. For instance, the complexity of representing a single connected component of $\mathcal{F}$ can be as high as roughly $O\left(n^d\right)$ (Basu 2003). It is therefore not surprising that best known algorithm for motion planning with general d is exponential in d, i.e., $O\left(n^{2d-1} \log n\right)$, where n is the complexity of workspace (Chazelle et al 1991). (See also a more detailed account of the Silhouette method (Canny 1993) in the chapter "Roadmaps".) It is also known that motion planning in general is PSPACE-hard when d is part of the input (Reif 1979; Latombe 1991).

We proceed to consider specific cases of the problem with $d = 2$, which admit an efficient solution. We start with the setting of a translating polygonal robot amid polygonal obstacles in the plane, as depicted in Figure 1. In case the robot is convex, and the only obstacle in the environment is a convex room, then the problem can be solved in time $O(m+n)$, where m is the complexity of the robot and n is the complexity of the workspace (Kedem et al 1986). In the more general case of non-convex robot and obstacles the problem can be solved in roughly $O(m^2 n^2)$, where the complexity follows from computing $\mathcal{F}$ via Minkowski sums (Agarwal et al 2002). We do mention that more refined results exist in the literature (see, e.g., Halperin et al (2016b)).

If we allow the polygon to translate and rotate, then the best known algorithm for this case runs in time $O\left((mn)^{2+\varepsilon}\right)$, for any $\varepsilon > 0$ (Halperin and Sharir 1996). However, restricting the robot's form to a rod or an L-shape, lowers the complexity to $O(n^2)$ (Vegter 1990) or $O(n^2 \log^2 n)$ (Halperin et al 1992), respectively. We conclude this part by mentioning that the motion of a translating polyotope within a 3D polyhedral environment can be computed in $O\left((mn)^{2+\varepsilon}\right)$ time, for any $\varepsilon > 0$ (see Aronov and Sharir (1994)).

Optimal planning and kinodynamic constraints

It is often desirable to obtain a high-quality solution path, or the best, i.e., optimal, solution possible. The quality can be associated with the length of the path traversed by the robot, time or amount of energy required to execute it, or safety that corresponds to the distance maintained between the robot and objects in its surrounding (e.g., obstacles or humans), to name just a few examples. Typically, optimality constraints significantly increase the complexity of the problem, and in most cases these problems are computationally hard. For instance, for the 2D case, the shortest path for a point robot amid polygonal obstacles can be computed in $O(n \log n)$ time (this can be extended to a polygonal robot by first computing the free space). However,

the three-dimensional extension is already NP-hard (Canny 1988), via a reduction from the 3SAT problem. (Also see additional information in chapter "Roadmaps".) As noted earlier, compromising for a near-optimal solution, e.g., returning a path that is at most $(1+\varepsilon)$-times longer the than the optimum, for some $\varepsilon > 0$, can lead to lower complexity bounds (see, e.g., Aleksandrov et al (2010)).

So far our discussion has been restricted to simple geometric robotic systems, in which the robot is assumed to be a rigid body free to translate or rotate as it pleases. However, in most real-life robotic systems the robot's motion is subject to kinodynamic or differential constraints, which must be satisfied in addition to collision avoidance. Addressing such constraints within a complete algorithmic framework turns out to be a challenging task. Consider for instance the seemingly simple case of a curvature-constrained planar model, also known as a Dubins path, which corresponds to a wheeled robot with a limited turning rate. The problem of finding the shortest such path turns out to be NP-hard (Kirkpatrick et al 2011), via a reduction from a generalized version of 3SAT. An approximation algorithm to this problem, which computes a solution whose length is at most $(1+\varepsilon)$ times the length of an optimal path, in $O\left(\frac{n^2}{\varepsilon^4}\log n\right)$ time, is given by Agarwal and Wang (2000). Refer to chapter "Kinodynamic Planning" for more information on planning with constraints.

Multi-robot Planning

In practical settings, such as in factory assembly lines, delivery services, and transportation systems, multiple robots are required to operate in a shared workspace, while avoiding collisions with each other, and at times cooperating to accomplish a mutual task. The entire fleet of robots can be viewed as one large robot having multiple moving parts. Thus, it is clear that the number of DoFs of the whole system increases with the number of robots. However, it should be noted that in multi-robot systems, the individual robots are rarely coupled or attached to each other. This property makes this problem slightly more manageable (as we will see later on), when compared to planning for a general system with the same number of DoFs.

One of the first algorithmic studies of multi-robot motion planning can be found in the seminal series of papers on the Piano Movers' Problem by Schwartz and Sharir. They first considered the problem in a general setting (Schwartz and Sharir (1983a)) and then narrowed it down to the case of disc robots moving amidst polygonal obstacles (Schwartz and Sharir (1983b)). In the latter work an algorithm was presented for the case of two and three robots, with running time of $O(n^3)$ and $O(n^{13})$, respectively, where n is the complexity of the workspace. Later Spirakis and Yap (1984) used the retraction method to develop more efficient algorithms, which run in $O(n^2)$ and $O(n^3)$ time for the case of two and three robots, respectively. Several years afterwards, Sharir and Sifrony (1991) presented a general approach based on cell decomposition, which is capable of dealing with additional types of robots and which has a running time of $O(n^2)$.

When the number of robots is no longer a fixed constant the problem can become computationally intractable. Specifically, Hopcroft et al (1984) showed that the problem is PSPACE-hard for the setting of multiple rectangular robots bound to translate in a rectangular workspace. Spirakis and Yap (1984) showed that the problem is NP-hard for disc robots in a simple-polygon workspace. A recent paper (Solovey and Halperin (2016)) presented a PSPACE-hardness proof for unlableled unit-square robots translating amid polygonal obstacles. In the unlabeled setting the robots are given a set of target positions and the goal is to move the robots in a collision-free manner so that each robot ends up at some target, without specifying exactly which. This is in contrast to the standard labeled formulation in which every robot is assigned to a specific target.

Although MRMP is computationally intractable in general, several recent works have demonstrated that the problem can be solved efficiently, if one makes some simplifying separation assumptions with respect to the start and target configurations, as described below. Turpin et al (2013) describe an efficient algorithm for unlabeled planning for disc robots, which also guarantees finding the optimal solution in terms of the length of the longest path traversed by a robot. Their algorithm makes the assumption that a certain portion of the free space, surrounding each start or target position, is star-shaped. A setting involving milder assumptions was studied by Adler et al (2015): here the requirement is that every pair of start positions, every pair of target positions, and every pair of start-target positions, each have a distance of at least 4 units between the centers of the unit-disc robots, and requiring the workspace to be a simple polygon, i.e., a polygon that contains no holes. That paper presented an efficient algorithm whose running time is $O(m^2 + mn)$, where m is the number of robots and n is the complexity of the workspace. A similar setting, with the additional requirement that each start or goal position has a (Euclidean) distance of at least $\sqrt{5}$ to any obstacle in the environment (see illustration in Figure 2) was studied by Solovey et al (2015). Here, the authors lift the assumption of the previous paper that the workspace consists of a single polygon. The algorithm has a running time of roughly $O\left(m^4 + m^2 n^2\right)$. Furthermore, the total length of the returned solution, i.e., the sum of lengths of the individual paths, is at most $\text{OPT} + 4m$, where OPT is the optimal-solution cost.

Future Direction for Research

The practical applicability of most complete motion planners tailored for the single-robot case remains limited, even after 40 years of intensive research. For a few simple cases, polynomial-time algorithms exists, although it should be noted that most of them are practical only for small dimensions, e.g., $d \in \{2, 3\}$, due to exponential dependence on d. Moreover, most problems are either NP-hard or PSPACE-hard, at the very least. One possible reason for existing approaches being so computationally expensive is their insistence on solving any feasible problem instance—including those that require the robot to get arbitrarily close to obstacles in order to reach
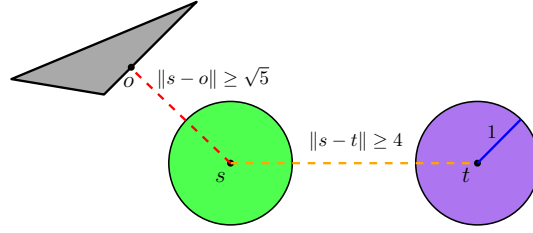
Fig. 2: Illustration of the separation conditions assumed in (Solovey et al 2015). The green and purple discs represent two unit-disc robots placed in start or target configurations. The blue line represents the unit radius of the robot (for scale). The distance between s and t is at least 4 units (see dashed orange line). The gray triangle represents an obstacle, and the point o represents the closest obstacle point to s. Notice that the distance between o and s is at least $\sqrt{5}$ units (see dashed red line).

its target. It has been already observed in sampling-based planning (see, e.g., Tsao et al (2019)) that the difficulty of solving a motion-planning problem increases as its clearance decreases. The clearance of a problem $\delta \geq 0$ denotes the minimal distance of the robot from the obstacles that is necessary to be achieved in order to find a solution. In this context, most complete algorithms are designed to cope with an arbitrary value of δ, even when it is equal to 0. Thus, it would be interesting to study how the clearance of the problem affects its complexity. Alternatively, it may be worth to consider a relaxed family of algorithms that given a parameter $\delta > 0$ are required to find a solution only if one exists with clearance at least δ.

In this respect, we mentioned three algorithms (Adler et al 2015; Solovey et al 2015; Turpin et al 2013) for multi-robot planning that have polynomial running time. Those approaches require some clearance around the start and target positions in order to guarantee their correctness. Perhaps the most limiting and unnatural aspect of such assumptions is the requirement to separate start positions from target positions, even though they need not be occupied simultaneously by the robots. Such assumptions limit the applicability of those techniques to scenarios that have plenty of free space for the robots to make their maneuvers. Thus, an immediate question is whether such assumptions can be relaxed to allow a broader range of problems to be solved efficiently. Additionally, the aforementioned techniques deal exclusively with the unlabeled setting. Would it be possible to come up with similar techniques for the labeled case?

From a theoretical side, a fundamental base case of multi-robot planning has not been sufficiently researched and a deeper understanding of it may lead to insights for the more general case. Suppose that our problem consists of only two moving robots, and moreover, let us assume that they are equal-radius discs operating in a workspace with polygonal obstacles. The best known complete algorithm for this problem, which was developed by Sharir and Sifrony (1991), runs in time $O(n^2)$. It should be noted that the algorithm is rather straightforward, in terms of the algo-

rithmic tools that are employed. This begs the question of whether a more efficient algorithm can be developed, which would possibly exploit the algorithmic progress that has been made in the last 25 years since the publication of the aforementioned paper. Alternatively, would it be possible to come up with a lower bound affirming that the above result is tight? Here recent progress on conditional lower bounds (see, e.g., Bringmann (2014)) may come in handy.

Another interesting and underresearched aspect of this problem is optimal motion, e.g., in terms of the total length of the two robot paths. Is it possible to come up with a polynomial-time algorithm that returns an optimal or near-optimal solution for two discs amid polygonal obstacles? We note that there is some evidence for this problem being computationally hard. First, it requires reasoning about the optimal solution in a four-dimensional space, whereas finding the shortest path for a point robot in a three-dimensional Euclidean space is NP-hard (Canny and Reif (1987)). Secondly, it was recently shown that optimal solutions for the two-disc problem, induce paths that have bounded curvature (Kirkpatrick and Liu (2016)), and recall that finding bounded-curvature paths for a single point robot in the plane amid obstacles is NP-hard (Kirkpatrick et al (2011)) as well.

Cross-References

Roadmaps; Sampling-Based Roadmap Planners (PRM and variations); Sampling-Based Tree Planners (RRT, EST and variations); Kinodynamic Planning.

References

Adler A, de Berg M, Halperin D, Solovey K (2015) Efficient multi-robot motion planning for unlabeled discs in simple polygons. IEEE T Automation Science and Engineering 12(4):1309–1317

Agarwal PK, Wang H (2000) Approximation algorithms for curvature-constrained shortest paths. SIAM J Comput 30(6):1739–1772

Agarwal PK, Flato E, Halperin D (2002) Polygon decomposition for efficient construction of Minkowski sums. Comput Geom 21(1-2):39–61

Aleksandrov L, Djidjev H, Guo H, Maheshwari A, Nussbaum D, Sack J (2010) Algorithms for approximate shortest path queries on weighted polyhedral surfaces. Discrete & Computational Geometry 44(4):762–801

Aronov B, Sharir M (1994) Castles in the air revisited. Discrete & Computational Geometry 12:119–150

Arora S, Barak B (2009) Computational Complexity - A Modern Approach. Cambridge University Press

Basu S (2003) The combinatorial and topological complexity of a single cell. Discrete & Computational Geometry 29(1):41–59

Bringmann K (2014) Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In: IEEE Annual Symposium on Foundations of Computer Science, pp 661–670

Canny J (1988) The complexity of robot motion planning. MIT press

Canny J (1993) Computing Roadmaps of General Semi-Algebraic Sets. The Computer Journal 36(5):504–514

Canny J, Reif JH (1987) New lower bound techniques for robot motion planning problems. In: Foundations of Computer Science, pp 49–60

Chazelle B, Edelsbrunner H, Guibas LJ, Sharir M (1991) A singly exponential stratification scheme for real semi-algebraic varieties and its applications. Theoretical Computer Science 84(1):77 – 105

Cohen BJ, Chitta S, Likhachev M (2014) Single- and dual-arm motion planning with heuristic search. I J Robotics Res 33(2):305–320

Cormen TH, Leiserson CE, Rivest RL, Stein C (2009) Introduction to Algorithms, 3rd Edition. MIT Press

Halperin D, Sharir M (1996) A near-quadratic algorithm for planning the motion of a polygon in a polygonal environment. Discrete & Computational Geometry 16(2):121–134

Halperin D, Overmars MH, Sharir M (1992) Efficient motion planning for an L-shaped object. SIAM J Comput 21(1):1–23

Halperin D, Kavraki L, Solovey K (2016a) Robotics. In: Goodman JE, O'Rourke J, Toth CD (eds) Handbook of Discrete and Computational Geometry, 3rd edn, CRC Press LLC, chap 51, pp 1343–1376, URL http://www.csun.edu/~ctoth/Handbook/HDCG3.html

Halperin D, Salzman O, Sharir M (2016b) Algorithmic motion planning. In: Goodman JE, O'Rourke J, Toth CD (eds) Handbook of Discrete and Computational Geometry, 3rd edn, CRC Press LLC, chap 50, pp 1311–1342, URL http://www.csun.edu/~ctoth/Handbook/HDCG3.html

Hopcroft JE, Schwartz JT, Sharir M (1984) On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "Warehouseman's problem". International Journal of Robotics Research 3(4):76–88

Kedem K, Livne R, Pach J, Sharir M (1986) On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. Discrete and Computational Geometry 1:59–70

Kirkpatrick DG, Liu P (2016) Characterizing minimum-length coordinated motions for two discs. In: Canadian Conference on Computational Geometry, pp 252–259

Kirkpatrick DG, Kostitsyna I, Polishchuk V (2011) Hardness results for two-dimensional curvature-constrained motion planning. In: Proceedings of the 23rd Annual Canadian Conference on Computational Geometry, Toronto, Ontario, Canada, August 10-12, 2011

Latombe JC (1991) Robot Motion Planning (The Springer International Series in Engineering and Computer Science). Kluwer Academic

Reif JH (1979) Complexity of the mover's problem and generalizations. In: Symposium on Foundations of Computer Science, pp 421–427

Schwartz JT, Sharir M (1983a) On the piano movers problem: II. General techniques for computing topological properties of real algebraic manifolds. Advances in applied Mathematics 4(3):298–351

Schwartz JT, Sharir M (1983b) On the piano movers problem: III. Coordinating the motion of several independent bodies. International Journal of Robotics Research 2(3):46–75

Sharir M, Sifrony S (1991) Coordinated motion planning for two independent robots. Annals of Mathematics and Artificial Intelligence 3(1):107–130

Solovey K, Halperin D (2016) On the hardness of unlabeled multi-robot motion planning. International Journal of Robotic Research 35(14):1750–1759

Solovey K, Yu J, Zamir O, Halperin D (2015) Motion planning for unlabeled discs with optimality guarantees. In: Robotics: Science and Systems

Spirakis PG, Yap CK (1984) Strong NP-hardness of moving many discs. Information Processing Letters 19(1):55–59

Tsao M, Solovey K, Pavone M (2019) Sample complexity of probabilistic roadmaps via epsilon-nets. CoRR abs/1909.06363, URL http://arxiv.org/abs/1909.06363

Turpin M, Michael N, Kumar V (2013) Concurrent assignment and planning of trajectories for large teams of interchangeable robots. In: IEEE International Conference on Robotics and Automation, pp 842–848

Vegter G (1990) The visibility diagram: a data structure for visibility problems and motion planning. In: Scandinavian Workshop on Algorithm Theory, Springer, pp 97–110

Wikipedia (2019) P versus NP problem. `https://en.wikipedia.org/wiki/P_versus_NP_problem`, [Online; accessed 16-October-2019]