Asymptotically optimal planning under piecewise-analytic constraints

William Vega-Brown and Nicholas Roy

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139
{wrvb,nickroy}@csail.mit.edu

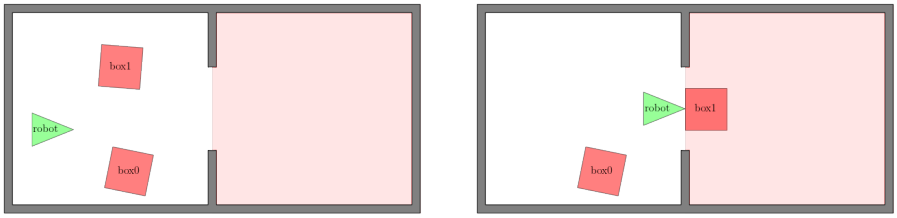
Abstract. We present the first asymptotically optimal algorithm for motion planning problems with piecewise-analytic differential constraints, like manipulation or rearrangement planning. This class of problems is characterized by the presence of differential constraints that are local in nature: a robot can only move an object once the object has been grasped. These constraints are not analytic and thus cannot be addressed by standard differentially constrained planning algorithms. We demonstrate that, given the ability to sample from the locally reachable subset of the configuration space with positive probability, we can construct random geometric graphs that contain optimal plans with probability one in the limit of infinite samples. This approach does not require a hand-coded symbolic abstraction. We demonstrate our approach in simulation on a simple manipulation planning problem, and show it generates lower-cost plans than a sequential task and motion planner.

1 Introduction

Consider a robot tasked with building a structure; a general-purpose planning algorithm for such a task must infer plans that respect both kinematic constraints, such as joint limits and collision between moving objects, as well as differential constraints, such as the limitation that the robot must be in contact with an object in order to affect its configuration. There exist provably asymptotically optimal algorithms for motion planning under analytic differential constraints. However, problems that involve discrete decisions, such as whether to grasp or release an object, cannot be described by analytic differential constraints.

This class of problems includes all contact-based manipulation. In the assembly task, for instance, the robot must decide where to place any object it sets down. This decision will affect all future decisions, as once the object has been set down, the robot cannot affect its position without grasping it again. While standard algorithms can decide *how* to place an object, they cannot make optimal decisions about *where* to place the object. See fig. 1 for a simple illustration of this problem.

There are two families of approaches to address this problem. One option is to break up the planning problem into a sequence of simpler planning problems in which the constraints are analytic. For example, the robot may first use a task



(a) (b)

Fig. 1: a: In this block pushing problem, the triangular robot seeks to move both the red blocks into the shaded room. b: A locally-optimal decision early on in the plan can have dramatic consequences later. Because it placed the first block just over the threshold of the door to the target room, the robot cannot move the second block without grasping the first block again.

planner to choose a sequence of high-level actions, such as grasping or placing a component. A motion planner can then generate a detailed plan corresponding to each action.

Substantial research effort has gone into accounting for the interplay between the high-level task planner and the low-level geometric motion planner.For example, Kaelbling and Lozano-Pérez [10] use a hierarchy to guide high-level decision making, resolving low-level decisions arbitrarily and trusting in the reversibility of the system to ensure hierarchical completeness. Wolfe et al. [17] ensure hierarchical optimality by expanding high-level plans in a best-first way, using domain specific subtask irrelevance detection to increase efficiency. Krontiris and Bekris [13] use a backtracking search as a low-level planner in the context of a high-level task planning problem. While these approaches are often computationally efficient, the guarantees they provide are *hierarchical*: they will return the best feasible plan that can be expressed within the task hierarchy they search. These hierarchies are typically hand-coded and domain-specific; the quality of the solution and the reliability of the planner hinges on the existence of a concise symbolic description of the planning domain, which may be expensive or impossible to find.

An alternative strategy is to solve the problem in the joint configuration space of the robot and objects, modifying differentially constrained planning algorithms like RRT* [11] to account for non-analytic differential constraints. This approach avoids the need for hand-coded motion primitives or a symbolic representation of the domain and allows for guarantees of probabilistic completeness under various assumptions. For example, Van Den Berg et al. [16] present a probabilistically complete algorithm for planning in domains including moveable objects. Hauser and Latombe [5] and Berenson et al. [2] show how to plan when the differential constraints create a fixed, finite set of manifolds on which the dynamics are unconstrained, as when a single object may be lifted or dragged. The core idea of both algorithms involves building many separate planning graphs on subsets of the configuration space and connecting them to formulate a global planner. Others have expanded on this idea to choose which local planning graphs to construct in an automated fashion. For example, Jail-

let and Porta [8] construct an atlas of local planning graphs and use that atlas to search for paths on manifolds. Their approach allows for efficient planning in situations where working in the ambient space would be prohibitively expensive. While their algorithm is asymptotically optimal, it cannot be directly applied to problems with non-analytic constraints. Algorithms that do apply to non-analytically constrained domains, such as Hauser and Ng-Thow-Hing's extension of their earlier work [6], are probabilistically complete but are not known to be asymptotically optimal.

In this work, we present two algorithms that are provably asymptotically optimal for problems involving differential constraints that are piecewise-analytic, a broad class that includes manipulation planning problems. Our first algorithm extends the result of Hauser and Ng-Thow-Hing to ensure asymptotic optimality, much as the PRM* algorithm [11] extends the PRM algorithm [7]. This extension leads to a provably optimal but prohibitively computationally expensive algorithm; our second algorithm mitigates this complexity by factoring the configuration space. Our key analytical result is that we can ensure asymptotic optimality by considering only a finite collection of subsets of the configuration space, each of which is subject only to analytic constraints. By building a graph on each of these subsets and connecting the resulting collection of graphs, we can construct a random graph that spans the configuration space; as the collection grows sufficiently large, it will contain a near-optimal plan with probability one and an optimal plan in the limit of infinite samples. We show experimentally that these algorithms obtain plans with lower cost than conventional sequential task and motion planning approaches.

2 Background and notation

Formally, we can define a differentially-constrained planning domain by a tuple $(f, \mathcal{X})$, where $\mathcal{X}$ is an N-dimensional Riemannian manifold with tangent bundle $T\mathcal{X}$ defining the configuration space, and $f : \mathcal{X} \times T\mathcal{X} \to \mathbb{R}^k$ is a set of k constraints on the allowable motions, $K < N$. We assume this manifold is specified by an embedding into a higher-dimensional Euclidean space, such that it can be represented as the zero level set of some function $g : \mathcal{X} \to \mathbb{R}^M$, $N < M$. Then a continuous function $\sigma : [0, T] \to \mathcal{X}$ is a *feasible path* if $f(\sigma(t), \dot{\sigma}(t)) = 0 \, \forall t \in [0, T]$. We denote the set of feasible paths in a given domain by $\Sigma_{\mathcal{X}}$. Note that this formulation can model kinematic constraints, such as collisions between objects or joint limits, and dynamic constraints.

We define a planning problem as a tuple $(x_0, \mathcal{X}_G, c)$, where $x_0 \in \mathcal{X}$ is an initial configuration, $\mathcal{X}_G$ is a set of goal configurations, and $c : \Sigma \to \mathbb{R}^+$ is a piecewise Lipschitz continuous cost function. We restrict our attention to problems where the cost of a path is independent of the time taken to traverse the path; accordingly, the cost of a plan is the line integral of the cost function over the path.

Definition 1. *A solution to a planning problem* $(x_0, \mathcal{X}_G, c)$ *in a domain* $(f, \mathcal{X})$ *is a continuously differentiable path* $\sigma^* : [0, T] \to \mathcal{X}$*, where*

$$\sigma^* = \arg\min_{\sigma \in \Sigma_{\mathcal{X}}} \int_0^T c(\sigma(t)) \|\dot{\sigma}(t)\| \, \mathrm{d}t \tag{1}$$

$$\text{s.t.} \quad f(\sigma(\tau), \dot{\sigma}(\tau)) \geq 0 \qquad \forall \tau \in [0, T] \qquad \text{(feasibility)}$$

$$\sigma(0) = x_s, \quad \sigma(1) \in X_g$$

We can solve problems of this form under a variety of conditions, by constructing and searching a *random geometric graph*. A random geometric graph $G_n^{\mathrm{RGG}} = (\mathcal{V}, \mathcal{E})$ is a graph whose vertices are a randomly chosen finite set of configurations $\mathcal{V} \subset \mathcal{X}$, and whose edges are paths between nearby configurations, with adjacency determined using simple geometric rules. There are many motion planning algorithms that exploit this underlying concept; see LaValle [14] for a survey.

Planning algorithms based on random geometric graphs are provably optimal under a variety of conditions. For holonomic systems, Karaman and Frazzoli [11] demonstrated that a random geometric graph with n vertices drawn uniformly at random from the configuration space $\mathcal{X}$ will contain an optimal path with probability one in the limit as $n \to \infty$. This result was proven for a graph with an edge between any pair of samples with geodesic distance less than $\gamma_{\mathcal{X}}(\frac{\log n}{n})^{\frac{1}{k}}$, provided the straight-line path between those vertices is collision free; $\gamma_{\mathcal{X}}$ is a constant depending only on the manifold $\mathcal{X}$ and k is the dimensionality of the configuration space $\mathcal{X}$. The same authors extended this result [12] to the broad class of non-holonomic real analytic dynamical systems. Note these approaches can be applied to planning on manifolds, assuming we know how to sample random configurations from the manifold and how to connect nearby configurations.

The class of systems addressed by these algorithms does not include many systems of practical interest in robotics. Consider a simple model of object manipulation. A holonomic robot is tasked with moving K boxes. When in contact with a box, the robot can rigidly grasp the box, so that the robot-box pair behave as a single rigid body as long as the box is grasped. Any box not grasped by the robot does not move. The robot can release a grasped box at any time. This model avoids much of the complexity of real contact dynamics; there is no consideration of momentum, or even of form or force closure. These constraints can be written explicitly; we define the indicator function $f_{\mathrm{contact}} : \mathrm{SE}(2) \times \mathrm{SE}(2) \to \{0, 1\}$ such that for any object o, $f_{\mathrm{contact}}(x_r, x_o) = 0$ if the robot can grasp the object when the robot pose is x_r and the object pose is x_o, and $f_{\mathrm{contact}}(x_r, x_o) = 1$ otherwise. Then the constraints on permissible motions can be expressed as

$$f_{\mathrm{contact}}(x_r, x_o)\dot{x}_o = 0 \quad \forall o. \tag{2}$$

Because the support of the function $f_{\mathrm{contact}}(x_r, x_o)$ is compact, the function is not analytic. Note that although f_{contact} is not smooth, a similar result holds for smooth constraints, provided they have compact support.

Non-analytic constraints cause motion planning algorithms to fail for a simple reason: arbitrary pairs of configurations can be quite close in geodesic distance but require very long trajectories to connect. Consider two world configurations where the position of each object is perturbed slightly from one configuration to the other; to move the world from the first configuration to the second, the robot must travel to each object in turn. Typical PRM* implementations use either straight-line connections or two-point boundary value solvers as local planners; in order to apply an algorithm like PRM* to this domain, we would need a more powerful local planner capable of computing the very long path between the original and perturbed configurations. Similarly, the rapid exploration of algorithms like the RRT* is dependent on a steer function that finds a plan that moves toward an arbitrary configuration; in manipulation planning problems, simple implementations of the steer function may not bring the world configuration closer to a perturbed configuration, leading to slow exploration. Note this does not constitute a proof of incompleteness of RRT* or PRM* in problems with non-analytic constraints; little is known about the completeness or optimality of sampling-based motion planning in settings where the geodesic distance does not accurately capture the complexity of moving between configurations.

3 Modes and orbits

Although the conditional constraints that are a fundamental characteristic of manipulation planning domains cannot be represented as analytic differential constraints, they can be represented using the more general class of *piecewise-analytic* constraints. We define a function f as piecewise-analytic if there exists a finite set $\mathcal{M}$ of $N_{\mathcal{X}}$ connected Riemannian manifolds $\{\mathcal{X}_i\}_{i=1}^{N_{\mathcal{X}}}$, each a subset of the configuration space $\mathcal{X}$, such that $\mathcal{X} = \bigcup_{i=1}^{N_{\mathcal{X}}} \mathcal{X}_i$ and such that the restriction of f to the interior of $\mathcal{X}_i$ is analytic for each i. Following Alami et al. [1] and Hauser and Latombe [5], we refer to the manifolds $\mathcal{X}_i$ as *modes*. In the simplified block-pushing problem with k blocks, there are $k+1$ modes: one describing the motion of the robot when not in contact with any blocks, and one for each block describing the evolution of the system when the robot grasps that block.

Recall that in our block-pushing problem, a block moves only if grasped by the robot. Consequently, it is impossible to move between arbitrary configurations on a mode without grasping and releasing blocks—that is, without switching modes. Consider the mode defined by which block the robot has grasped; the reachable configurations *within* this mode are defined by the locations of all the other blocks. We will refer to the collection of configurations reachable from an arbitrary configuration as an *orbit*.

Formally, an orbit $\mathbf{O}_{\mathcal{X}_i}(x)$ of a mode $\mathcal{X}_i$ through a configuration $x \in \mathcal{X}_i$ is the subset of $\mathcal{X}_i$ connected to x by a feasible path that lies wholly on the manifold $\mathcal{X}_i$. Because the constraints are analytic on each mode, the orbits are disjoint submanifolds of the same dimensionality [15]. This means that a planning problem where the start and goal states both lie on the same orbit is straightforward to solve using standard sampling-based planning techniques. The key to our approach is recognizing that we can solve arbitrary planning problems in a given domain by choosing a finite set of orbits, building a random geometric graph

on each orbit, and connecting the resulting set of graphs. In the sections that follow, we will describe in detail how to construct and link these graphs in a way that preserves the optimality guarantees of the PRM* while also remaining computationally tractable.

Note that as with most sampling-based motion planning algorithms, we do not require an explicit geometric representation of the modes or orbits in order to plan; we require only the ability to sample from them. This may be non-trivial, as standard rejection-sampling approaches will fail if the modes or orbits have dimension less than that of the configuration space. However, for many problems of interest, it is straightforward to write subroutines that uniformly sample configurations from a given orbit. These subroutines are also a prerequisite for many other manipulation planning algorithms.

For the purposes of this work, we assume that the modes and orbits are exogenously given in a form that permits sampling. That is, we assume that given a configuration x, there exists a subroutine `modes`(x) that returns a list of identifiers of the modes that contain x, a subroutine `sample`(x) that generates a configuration chosen uniformly at random from orbits containing x, and a subroutine `sample_boundary`(x) that generates a configuration uniformly at random from the boundary of one of the orbits containing x, such that the probability of the generated sample belonging to any boundary manifold is greater than zero.

4 Algorithms

We first describe the implementation of our algorithms on the block-pushing domain, then describe the general formulation. The complexity of the algorithms involved leads to unwieldy pseudocode; instead, we provide an open source Python implementation[1] of the algorithms described here. In addition to the subroutines `sample` and `sample_boundary` mentioned in section 3, we assume we have a local planner available, which must satisfy several regularity conditions described in section 5.

Our algorithms both follow the same basic procedure: we construct an implicit random geometric graph by choosing a set of configurations (the *vertex set*) and specifying a subroutine to generate the neighboring configurations for a given vertex. In addition to a problem specification (x_0, X_G, c), we take as input an integer parameter n. As n increases, the size of the graph increases, which increases the computational resources required to search for a trajectory but also improves the quality of the path returned.

Two vertices x and x' are connected by an edge in our graph if the geodesic distance on some orbit including x and x' is less than a critical threshold value depending on n. We use the local planner to determine the cost of an edge; if the local planner cannot find a feasible path, perhaps due to the presence of an obstacle, the edge is assigned a cost of ∞. Together, the vertex set and the neighbor function specify an implicit random geometric graph. An eager PRM* implementation would explicitly evaluate the cost of every edge; instead, we lazily evaluate only those edges that may be part of an optimal path.

[1] `https://github.com/robustrobotics/forgg`

We then search the implicit random geometric graph using the A* algorithm of Hart et al. [4]. Only when a vertex is expanded do we actually invoke the local planner to determine whether it can generate a collision-free feasible trajectory to any of the neighbors of the expanded vertex; if it can, the neighboring vertex is added to a priority queue with priority equal to the minimal cost to reach that vertex. Note that this idea of lazily searching a random geometric graph is not novel; the core idea was described by Bohlin and Kavraki [3], and recent work from Janson et al. [9], among others, suggest this approach can be significantly more efficient than RRT* or searching an explicit graph constructed with the PRM* algorithm. The innovation in our approach is the way in which we construct the graph, which ensures that the graphs constructed on each orbit are constructed in a way that ensures asymptotic optimality.

4.1 Orbital Bellman trees

To extend the asymptotic optimality of PRM* to problems with piecewise-analytic constraints, we must ensure that as the graph size n tends to infinity, the number of orbits on the connected component containing x_s tends to infinity and that the number of samples on each of those orbits tends to infinity. Building on the Random-MMP algorithm described by Hauser and Ng-Thow-Hing [6], we provide a graph construction and search algorithm that incrementally builds the graph such that as n increases, the graph contains an increasing number of samples from an increasing number of orbits yet remains connected with probability one as $n \to \infty$.

The algorithm has two free parameters, $\nu \in (0, 1)$ and $\eta \in (0, \infty)$. Increasing η increases the number of edges in the graph by inflating the radius defining whether two vertices are connected. ν represents a trade-off between exploring the interior of each orbit and exploring the relations between different orbits.

We initialize the vertex set to include the initial configuration x_0 and place the initial configuration in a priority queue with priority zero. We then repeatedly remove the lowest-cost vertex from the queue and perform two algorithmic operations. First, we check whether the removed vertex belongs to any orbits that do not contain samples and add samples from those orbits if so. Concretely, if we remove the vertex x from the priority queue and for some mode $\mathcal{M}$ containing x the orbit $\mathbf{O}_{\mathcal{M}}(x)$ does not contain any samples, we call the subroutine `sample` n times to sample n configurations from $\mathbf{O}_{\mathcal{M}}(x)$. Then for each mode $\mathcal{M}'$ adjacent to $\mathcal{M}$, we call the `sample_boundary` subroutine νn times to generate νn samples from the intersection of $\mathbf{O}_{\mathcal{M}}(x)$ and $\mathcal{M}'$. Finally, we build a search index such as a k-d tree or a cover tree from the n samples, to enable efficient lookup of the neighboring vertices.

Second, we perform a standard iteration of A*, considering the neighbors of x on each orbit to which it belongs. For each neighbor x', we evaluate the cost of the path returned by the local planner from x to x'; if that cost is finite, we add the neighbor x' to the priority queue as usual. The algorithm terminates when a vertex is expanded that lies in X_g, when the queue is empty, or when a predefined maximum number of samples have been removed from the priority queue.

As the search proceeds, this process grows a graph of interconnected orbits. We refer to this graph construction as a random orbital geometric graph (ORGG), as it is a random geometric graph that respects the structure of the orbits created by the constraints. As with any best-first search algorithm, the search produces a tree in which each vertex v is labelled with the minimal cost of a path through the ORGG from the start vertex to v, the parent of each vertex except the start vertex lies along that minimum cost path. We refer to this tree as an orbital Bellman tree, as the cost to reach each vertex satisfies Bellman's equations; we refer to the search algorithm as the orbital Bellman tree (OBT) algorithm.

The OBT algorithm is asymptotically optimal for any ν, η if the set of neighbors of a given vertex includes all configurations within a distance $r_{\mathbf{O}}(n)$, for each orbit $\mathbf{O}$ containing the configuration. The function $r_{\mathbf{O}}(n)$ is determined by ν, η, the dimensionality $d_{\mathbf{O}}$, and the Lebesgue measure $\mathrm{vol}(\mathbf{O})$ of the orbit, where the measure is induced by the volume form of the manifold $\mathbf{O}$. As with other sampling-based motion planners, the Lebesgue measure of the orbit can be approximated using rejection sampling.

$$r_{\mathbf{O}}(n) = (1+\eta)\gamma_{\mathbf{O}} \left(\frac{\log n}{n}\right)^{1/d_{\mathbf{O}}} \tag{3}$$

$$\gamma_{\mathbf{O}} = 4\left((1+\frac{1}{d_{\mathbf{O}}})\frac{\mathrm{vol}(\mathbf{O})}{\zeta_{d_{\mathbf{O}}}}\right)^{\frac{1}{d_{\mathbf{O}}}} \tag{4}$$

$$\zeta_{d_{\mathbf{O}}} = \frac{\pi^{d_{\mathbf{O}}}}{\Gamma(\frac{d_{\mathbf{O}}}{2}+1)} \qquad \text{(volume of a } d_{\mathbf{O}}\text{-sphere)}$$

Note this connectivity radius is nearly identical to that presented by Karaman and Frazzoli; the only change is the factor of 4 in γ, which is needed to ensure optimality when the path obtained is on a different orbit from the optimal path.

OBT is different from Random-MMP in two important ways. First, OBT obtains provably asymptotically optimal paths across each orbit by constructing an optimal random geometric graph on each orbit. Random-MMP instead adds a single path across an orbit to a configuration on another mode; finding this path is sufficient to guarantee completeness but not optimality. Second, by choosing a fixed fraction of samples from each orbit to be from the intersection of the orbit and the adjacent modes, OBT ensures that enough orbits are considered to guarantee optimality.

4.2 Factored orbital Bellman trees

The OBT algorithm is extremely computationally intensive. Each new orbit considered requires sampling n new configurations and building a new search index; this takes $\mathcal{O}(n \log n)$ time per orbit. Consider the block-pushing problem; every grasp configuration is a part of two orbits, as the robot can either maintain the grasp and move with the object or immediately drop the object at that location and move only itself. This means that we must consider n new orbits whenever the robot grasps an object. Consequently, the number of samples we

must store grows exponentially with the number of objects the shortest plan must grasp. There are K objects to grasp whenever the robot is not holding an object. Each time we consider a possible set of grasps and releases, we generate $\mathcal{O}(Kn)$ samples and take $\mathcal{O}(Kn \log n)$ time, and the OBT algorithm does so for each combination of grasps and releases, leading to $\mathcal{O}((Kn \log n)^d)$ time and $\mathcal{O}(K^d n^d)$ space to search through all possible combinations of objects to grasp and locations to release in a sequence of d grasps.

One avenue toward reducing this computational burden is to take advantage of the structure of the problem domain to reduce the number of samples and search indices we must generate. The block-pushing domain has a key feature that makes this possible: the constraints that define the modes and orbits *factor*, allowing us to consider different parts of the configuration space independently. The constraint that an object cannot move unless grasped is unary: it does not affect the permissible locations of the other objects. The constraint that the robot is grasping an object is binary: it affects the robot and the object, but has no effect on the configurations of the other objects.

We can encode this structure in a factor graph representing the uniform distribution over a mode. The vertices of this factor graph represent the configuration of each object, and the unnormalized factors are the constraints defining the orbits. We can generate samples from this uniform distribution over a mode by sampling from each connected component of this factor graph independently and then taking the Cartesian product of the resulting sample sets. We can exploit this factorization to generate samples from the full configuration space efficiently.

In the block-pushing domain, this factored sampling amounts to sampling a set of poses of each object and a set of grasping poses of the robot for each object. A graph whose vertices are the union of the products of these sets of poses is an orbital random geometric graph. Because the vertices of such a graph are generated by factoring the uniform distribution over a mode, we refer to this graph construction as a factored orbital random geometric graph (FORGG). This factorized sampling strategy can be generalized to arbitrary configuration spaces and sets of constraints, and will be beneficial if the factor graph encoding a uniform distribution over a given orbit has multiple connected components. Arbitrarily complex models of contact dynamics satisfy this requirement due to the local nature of contact dynamics.

The algorithm that constructs and searches a factored orbital random geometric graph closely resembles the OBT search algorithm, with the key distinction that the samples are generated from independent factors. Accordingly, we refer to this search algorithm as the factored orbital Bellman tree (FOBT). On the block pushing problem, this reduces the amount of space needed to store the orbital geometric graph from $\mathcal{O}((nk)^D)$ to $\mathcal{O}(nk)$ and the amount of time required from $\mathcal{O}((kn \log n)^D)$ to $\mathcal{O}(kn \log n)$, with no dependence on D, the number of grasps in the shortest solution. Note that we have not avoided the exponential cost of graph search; the search itself still takes $\mathcal{O}(b^d)$ time and space, where b is the graph branching factor and d is the depth of the shortest solution. Although this is only a constant factor improvement, it represents a significant practical advance.

5 Analysis

We now prove the asymptotic optimality of OBT and FOBT, subject to a regularity condition on the local planner used. For brevity, we present several propositions without proof; proofs of these propositions are available in our supplementary material. In addition to the piecewise-analyticity of the constraints, we require an additional technical condition on the local planner π used to connect nearby states. First, there must exist a radius $r_0 > 0$ such that the planner will return a feasible path if invoked to connect two configurations that lie inside an open geodesic ball of free space with radius less than r_0. Second, for any $\epsilon > 0$ there must exist $r_\epsilon > 0$ such that for all $x, x' \in \mathcal{M} : d_{\mathcal{M}}(x, x') < r_\epsilon, L_{\mathcal{M}}(\pi(x, x')) < (1 + \epsilon) d_{\mathcal{M}}(x, x')$. If a local planner has these two properties, we say it is locally complete. Note that a local planner that connects trajectories with geodesic curves is locally complete.

Theorem 1 (Optimality of OBT). *Given a planning problem (x_s, X_g, c) in a domain (X, f), let c_n be the shortest path between x_s and X_g on an orbital random geometric graph G_n with n vertices, built using a locally complete local planner. Then* $\mathbb{P}(\{\limsup_{n\to\infty} c_n = c^*\}) = 1$.

5.1 Construction of a sequence of paths

Let $\bar{\sigma}^*$ be an optimal solution to the planning problem. Decompose $\bar{\sigma}^*$ into a sequence of M paths $\{\sigma^*_m\}_{m\in[1,M]}$, each lying on a single manifold.

$$\bar{\sigma}^* = \bigoplus_{m=1}^{M} \sigma^*_m \tag{5}$$

Define $\varphi_m \in V_M$ as the mode on which the path σ^*_m lies. Let d_m be the dimensionality of the orbit containing σ^*_m. Let $\bar{d}$ be the maximum dimension of any intersection orbit expressed in the path: $\dim(\varphi_m \cap \varphi_{m+1}) \leq \bar{d} \forall m$.

Because the modes are analytic manifolds, there exists $\delta > 0$ such that each path σ^*_m is homotopy-equivalent to a path that lies in the union of the δ-interior of the mode φ_m, an open ball of radius δ whose closure contains $\sigma^*_m(0)$, and an open ball of radius δ whose closure contains $\sigma^*_m(1)$. Define the weakly monotonically decreasing sequence $\{\delta_n\}_{n\in\mathbb{Z}}$.

$$\delta_n = \min\left(\delta, n^{-\frac{1}{2\bar{d}}}\right) \tag{6}$$

Clearly, this sequence satisfies $0 < \delta_n \leq \delta$ and $\lim_{n\to\infty} \delta_n = 0$; let $n_0 = \min\{n \in \mathbb{Z} : \delta_n < \delta\}$. Because the problem is δ-robust, there exists a sequence $\{\bar{\sigma}_n\}_{n\in\mathbb{N}}$ such that $\bar{\sigma}_n$ has δ_n-clearance. Decompose each path $\bar{\sigma}_n$ into a sequence of M paths $\sigma_{n,m}$, just as with $\bar{\sigma}^*$.

5.2 Construction of balls on the intersections between modes

Define $r_{\cap,n,m} = a_m n^{-\frac{1}{2d}}$, where a_m is recursively defined to ensure that if a leaf intersects $r_{\cap,n,m}$, it also intersects $r_{\cap,n,m+1}$.

$$a_M = \delta \tag{7}$$

$$a_m = \sup\{a > 0 : \forall y \in B(\sigma_{n,m}, a) \sup_{t\in(0,1)} \inf_{y'\in\mathcal{O}_m(y)} d(\sigma_{n,m}(t), y') < a_{m+1}\} \tag{8}$$

Note that $r_{\cap,n,m} \leq \delta_n \, \forall m$ for large n. For each path $\sigma_{n,m}$, define the region $B_{\cap,n,m}$ as the geodesic ball centered at $\sigma_{n,m}(0)$ on the manifold $\varphi_{m-1} \cap \varphi_m$ with radius $r_{\cap,m,n}$, Let $E_{\cap,n,m}$ be the event that the ball $B_{\cap,n,m}$ contains a sample: that is, $E_{\cap,n,m}$ occurs when the intersection of the vertex set $\mathcal{V}_n$ and the ball $B_{\cap,n,m}$ is nonempty. Let $A_{\cap,n} = \bigcap_m E_{\cap,n,m}$ be the event that each ball $B_{\cap,n,m}$ contains a sample.

5.3 Construction of balls on an arbitrary orbit

Fix $\theta \in (0,1)$ and $r > 0$; let $\sigma : [0,1] \to \mathcal{M}$ be a feasible path on a mode $\mathcal{M}$ such that there exist real numbers $t_-, t_+, 0 \leq t_- < t_+ \leq 1$ and the following conditons hold: for all $t \in (t_-, t_+)$, $\sigma(t) \in \mathrm{Int}_r(\mathcal{M})$; for all $t \in (0, t_-]$, $\sigma(t) \in B_r(\sigma(t_-))$; and for all $t \in [t_+, 1)$, $\sigma(t) \in B_r(\sigma(t_+))$. Then there exists a finite collection of configurations $Y(\sigma, y_0, r) = \{y_k\}$ drawn from the orbit containing y_0 with the property that if $z_k \in B(y_k, \frac{r}{4+\theta}), z_{k+1} \in B(y_{k+1}, \frac{r}{4+\theta})$ are two vertices in an orbital random geometric graph, the OBT algorithm will call the local planner for the pair (z_k, z_{k+1}), and the local planner will succeed. We provide a construction of Y in two steps. First, we consider the part of the path that lies in the r-interior of the manifold. Define a strictly monotonically increasing sequence (t_k) as follows.

$$\tau_0 = t_- \tag{9}$$

$$\tau_{k+1} = \sup_{\tau\in(\tau_k, t_+)} \{d_{\mathcal{M}}(\sigma(\tau_k), \sigma(\tau)) < \frac{\theta r}{4+\theta})\} \tag{10}$$

Let K be the smallest integer k such that $\tau_k = t_+$. Define $(x_k)_{k\in[K]}$ so that $x_k = \sigma(\tau_k)$. Define $(y_k)_{k\in[K]}$ so that $d(y_k, x_k) < \frac{r}{4+\theta}$; by the assumptions on the leaf, such a sequence must exist. Define the set of balls $B_{k\,k\in[K]}$, where $B_k = B(y_k, \frac{r}{4+\theta})$.

Let z_k be an arbitrary configuration in B_k.

$$d(z_k, x_k) \leq d(z_k, y_k) + d(y_k, x_k) \leq \frac{r}{4+\theta} + \frac{r}{4+\theta} \leq r \tag{11}$$

$$\begin{aligned} d(z_{k+1}, x_k) &\leq d(z_{k+1}, y_{k+1}) + d(y_{k+1}, x_{k+1}) + d(x_{k+1}, x_k) \\ &\leq \frac{\theta r}{4+\theta} + \frac{r}{4+\theta} + \frac{r}{4+\theta} \leq \frac{2+\theta}{4+\theta} r \leq r \end{aligned} \tag{12}$$

$$d(z_k, z_{k+1}) \leq d(z_k, x_k) + d(x_k, z_{k+1}) \leq \frac{2}{4+\theta} r + \frac{2+\theta}{4+\theta} r \leq r \tag{13}$$

From eq. (13), if the set of vertices includes a configuration in each of the pair of balls B_k and B_{k+1}, the local planner will be invoked for the pair; from eq. (11) and eq. (12), both samples lie inside the ball $B(x_k, r)$, and therefore by the assumptions of the theorem the local planner will succeed if called. Note that with the exception of τ_K, sequential centers $\sigma(\tau_k)$ and $\sigma(\tau_{k+1})$ are separated by $\frac{\theta r}{4+\theta}$; if $L(\sigma)$ is the length of the path, it follows that

$$K \leq \left\lceil \frac{4+\theta}{\theta r} L(\sigma) \right\rceil + 1. \tag{14}$$

Next, we consider the part of the path that lies near the boundary. We will prove the result for $t \in (0, t_-)$, assuming $t_- \neq 0$; the proof for $t \in (t_+, 1)$ is similar. Fix an arbitrary $y_0 \in \partial\mathcal{M} \cap L$ such that $d(\sigma(0), y_0) \leq \frac{r}{4+\theta}$. Define a chart $\phi_{y_0} : U \subset \mathcal{M} \to S \times V \times W$ in *collar coordinates*, such that $S \subseteq \mathbb{R}_{\geq 0}, V \subseteq \mathbb{R}^{k-1}, W \subseteq \mathbb{R}^{n-k}$. Note that the coordinate s is equal to the minimum distance of a configuration to the boundary of the manifold. Note also that $S \times V$ is diffeomorphic to a subset of the leaf $\mathbf{O}$; any curve with for which the coordinates in W are constant will be a feasible path. For sufficiently small r, such a chart must exist [15]. Without loss of generality assume $\phi_{y_0}(y_0) = (0, 0, 0)$.

Assume r is small enough that $B_r(\sigma(t_-)) \subset U$. Choose $y_- \in \mathbf{O}$ such that $d(y_-, \sigma(t_-)) \leq \frac{r}{4+\theta}$, and define $\phi_{y_0}(y_-) = (s_-, v_-, 0)$. Let $y_k = \phi_{y_0}^{-1}((1 - \alpha_k)\frac{3r}{4+\theta} + \alpha_k s_-, \alpha_k v_-, 0)$, where $\alpha_1 = 0$ and the sequence (α_k) is defined recursively as follows.

$$\alpha_{k+1} = \sup_{\alpha \in (\alpha_k, 1)} \{ d(y_k, y_{k+1}) < \frac{2r}{4+\theta} \} \tag{15}$$

Note that the total distance from y_0 to y_- is upper-bounded.

$$d(y_0, y_-) \leq d(y_0, \sigma(0)) + d(\sigma(0), \sigma(t_-)) + d(\sigma(t_-), y_-) \tag{16}$$

$$\leq \frac{r}{4+\theta} + 2r + \frac{r}{4+\theta} = \frac{10+2\theta}{4+\theta} \tag{17}$$

Since with the exception of the first and last centers the distance between successive centers (y_k) is at least $\frac{2r}{4+\theta}$, this part of the construction adds at most $2 + (\frac{10+2\theta}{4+\theta} r)/(\frac{2}{4+\theta} r) = 7 + \theta \leq 8$ configurations to the set.

The following claim is proven as Proposition 1 in our supplementary material; we omit the proof here. If the set of vertices includes configurations $z_k \in B(y_k, \frac{r}{4+\theta})$ and $z_{k+1} \in B(y_{k+1}, \frac{r}{4+\theta})$, these configurations will be connected by an edge. Similarly, if the set of vertices includes y_0 and a configuration $z_k \in B(y_k, \frac{r}{4+\theta})$, those vertices will be connected by an edge.

If $t_+ \neq 1$, we can apply a similar construction at the other end of the path. In total, we have constructed a set of at most

$$K_\sigma = \left\lceil \frac{4+\theta}{\theta r} L(\sigma) \right\rceil + 17 \tag{18}$$

balls, such that if each ball contains a sample, the resulting graph will contain a path with the desired properties. Then the cardinality of the set $Y(\sigma, y_0, r)$ is upper-bounded by $\left\lceil \frac{4+\theta}{\theta r} L(\sigma) \right\rceil + 17$.

We now apply the construction $Y(\sigma, y_0, r)$ to each mode. If the event $E_{\cap,n,m}$ occurs, there exists some $y_{n,m,0} \in B_{\cap,n,m} \cap \mathcal{V}_n$. Let $r_{n,m} = \gamma_m (\frac{\log n}{n})^{\frac{1}{d_m}}$. Let $Y_{n,m} = Y(\sigma_{n,m}, y_{n,m,0}, r_{n,m})$. Let $K_{n,m} = \text{card}(Y_{n,m})$. Let $B_{n,m,k}$ be the geodesic ball centered at $y_{n,m,k}$ of radius $\frac{1}{4+\theta} r_{n,m}$. Let $E_{n,m,k}$ be the event that the intersection of the vertex set $\mathcal{V}_n$ and the ball $B_{n,m,k}$ is nonempty. Let A_n be the event that all balls on each mode contains a sample. Note that A_n occurs only if $A_{\cap,n}$ occurs, as A_n is meaningful only if there exists an $y_{n,m,0} \in B_{\cap,n,m}$ to define the orbit on which $Y_{n,m}$ is defined. By construction, if A_n occurs, then algorithm OBT will return a solution with finite cost.

5.4 Bounding the cost of the path returned

Fix an arbitrary $\beta \in (0,1)$, and assume there exists $x_m \in B_{\cap,n,m}\, \forall m$. For each $y_{n,m,k}$, define a smaller ball $\tilde{B}_{n,m,k}$ with the same center and radius $\frac{\beta r_{n,m}}{4+\theta}$. Let $I_{n,m,k}$ be the indicator for the event that the intersection of the vertex set $\mathcal{V}_n$ and the ball $\tilde{B}_{n,m,k}$ is nonempty.

$$I_{n,m,k} = \begin{cases} 1 & \text{card}(\tilde{B}_{n,m,k} \cap \mathcal{V}_n) > 0 \\ 0 & \text{else} \end{cases} \tag{19}$$

Let $S_{n,m,k} = \sum_{m=1}^{M} \sum_{k=1}^{K_{n,m}} I_{n,m,k}$ be the number of smaller balls $\{\tilde{B}_{n,m,k}\}$ containing a configuration, and let $K_n = \sum_{m=1}^{M} K_{n,m}$ be the total number of smaller balls. If the cost function c is Lipschitz continuous, then for any $\epsilon > 0, \theta > 0$ there exists $\alpha > 0, \beta > 0, n_0 > 0$ such that if $S_n \geq \alpha K_n$, then for all $n > n_0$, $c(\sigma_n) \leq (1+\epsilon) c^*$ (**??** in the supplement). Let $A_{n,\alpha,\beta}$ be the event that $S_n \geq \alpha K_n$.

We can then upper-bound the probability that the path returned by OBT has cost more than $1+\epsilon$ times the optimal cost in terms of the probabilities $\mathbb{P}(\overline{A_{\cap,n}})$, $\mathbb{P}(\overline{A_n}|A_{\cap,n})$, and $\mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n})$ (**??** in the supplement).

$$\mathbb{P}(c_n \geq (1+\epsilon)c^*) \leq \mathbb{P}(\overline{A_{\cap,n}}) + \mathbb{P}(\overline{A_n}|A_{\cap,n}) + \mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n}) \tag{20}$$

Since each of the terms on the right side of eq. (20) is summable (**??**, **??**, and **??** in the supplement) the term on the left side of eq. (20) is summable. Consequently, the term on the left side of eq. (20) is summable.

$$\sum_{n=1}^{\infty} \mathbb{P}(\overline{A_{\cap,n}}) < \infty, \qquad \sum_{n=1}^{\infty} \mathbb{P}(\overline{A_n}|A_{\cap,n}) < \infty, \qquad \sum_{n=1}^{\infty} \mathbb{P}(\overline{A_{n,\alpha,\beta}}|A_{\cap,n}) < \infty \tag{21}$$

$$\therefore \sum_{n=1}^{\infty} \mathbb{P}(c_n \geq (1+\epsilon)c^*) < \infty \tag{22}$$

Therefore, by the Borel-Cantelli lemma, the event that the algorithm returns a feasible path with cost less than $(1+\epsilon)c^*$ occurs infinitely often as $n \to \infty$. The sequence c_n then converges almost surely to c^*. □

The proof of asymptotic optimality of FOBT employs the same geometric construction as for OBT. We need only modify the proof that the terms on the right side of eq. (20) are summable. The first two terms can be shown to be summable using identical logic to OBT, by noting that a ball in a product space contains a product of smaller balls in each component space and applying the union bound. The third term requires more effort to adapt, as the proof relies on the independence of the small balls. However, we can define a looser bound that does not require independence (**??** in the supplment). Due to space constraints, we omit this construction here.

6 Computational experiments

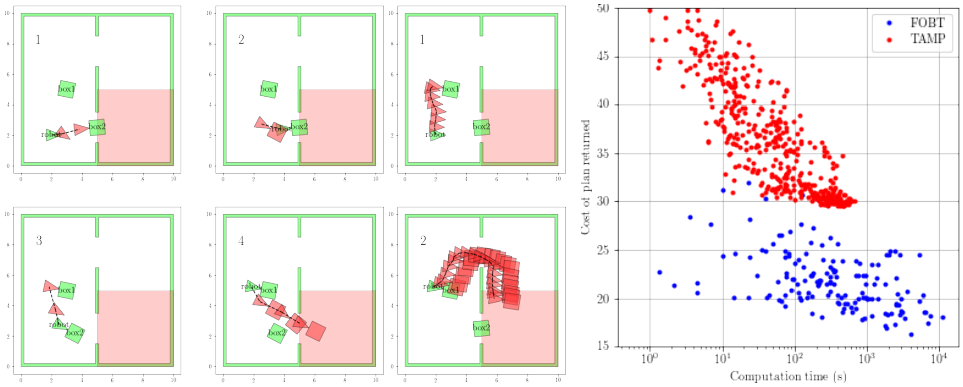
We implemented FOBT for the simplified block-pushing problem (fig. 2a). The goal is to move the block labelled 'box1' into the region shaded red, past a moveable obstacle. The planner must either decide to go around or must choose where to place the moveable object to get it out of the way. For comparison, we considered a simple task and motion planning algorithm approach, labelled TAMP. The TAMP planner can invoke a motion planner as a subroutine to accomplish a set of tasks, such as grasping an object or moving a grasped object to one of a fixed, hand-coded set of regions. In practice, this amounts to evaluating both sensible plans and choosing the one with the lower-cost solution.

As expected, we find that while TAMP can often quickly find a solution, more computational time does not allow that solution to be improved. In contrast, FOBT continues to perform better as the available computational time increases. Note that TAMP is suboptimal because it can only consider a finite set of goal locations; this set does not grow as n increases. By injecting domain knowledge in the form of a better task hierarchy, it is likely the TAMP planner could find plans as good as FOBT; FOBT finds these plans without such domain knowledge.

The quantitative comparison in fig. 2c highlights the main deficiency of FOBT: it is computationally demanding. As the parameter n increased above 1000, the implementation exhausted available memory and the algorithm failed due to space constraints. Improving the computational efficiency of FOBT is an important avenue for future work. Augmenting our algorithm with intelligent heuristics or nonuniform sampling strategies derived from domain knowledge could greatly increase computational efficiency.

7 Conclusion

To our knowledge, these are the first algorithms for asymptotically optimal motion planning that are applicable to piecewise-analytically constrained problem domains like manipulation planning. We note that the ideas in these algorithms can likely be combined with the ideas in many other sampling-based motion planning and graph search algorithms. This would improve performance and extend our results to domains such as kinodynamic planning, planning under uncertainty, and adversarial planning. In particular, if we combine symbolic task hierarchies with asymptotically optimal motion planning algorithms like the ones



(a) (b) (c)

Fig. 2: Plans obtained with FOBT and TAMP for a simple block pushing problem. The goal is to move the block labeled 'box1' to the region shaded in red, but the most direct path is blocked by another box. (a): The four-step plan obtained by FOBT. (b) A simpler, but more expensive, plan returned by a sequential task and motion planning algorithm. (c) Quantitative comparison of the cost of the plan returned and the computational time used for various graph sizes with both methods.

presented here, we can perhaps create task and motion planning algorithms with strong asymptotic performance guarantees.

In addition, our analytical results provide a foundation for a rigorous evaluation of the performance gap between hierarchically optimal planners and asymptotically optimal planners like those presented here. In principle, we could utilize such a bound to learn better symbolic representations, rather than requiring they be hand-coded by domain experts. This presents a promising route toward linking recent advances in planning and unsupervised learning.

Bibliography

[1] Rachid Alami, Jean-Paul Laumond, and Thierry Siméon. Two manipulation planning algorithms. In *Proceedings of the workshop on Algorithmic foundations of robotics*, pages 109–125. AK Peters, Ltd., 1995.

[2] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, and James J Kuffner. Manipulation planning on constraint manifolds. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 625–632. IEEE, 2009.

[3] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 521–528. IEEE, 2000.

[4] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[5] Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 2009.

[6] Kris Hauser and Victor Ng-Thow-Hing. Randomized multi-modal motion planning for a humanoid robot manipulation task. *The International Journal of Robotics Research*, 30(6):678–698, 2011.

[7] David Hsu, Jean-Claude Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 3, pages 2719–2726. IEEE, 1997.

[8] Léonard Jaillet and Josep Porta. Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117, 2013.

[9] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 2015.

[10] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proceedings of the International Conference on Robotics and Automation*, pages 1470–1477. IEEE, 2011.

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

[12] Sertac Karaman and Emilio Frazzoli. Sampling-based optimal motion planning for non-holonomic dynamical systems. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

[13] Athanasios Krontiris and Kostas E Bekris. Dealing with difficult instances of object rearrangement. In *Robotics: Science and Systems (RSS)*, 2015.

[14] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[15] Peter W Michor. *Topics in differential geometry*, volume 93. American Mathematical Soc., 2008.

[16] Jur Van Den Berg, Mike Stilman, James Kuffner, Ming Lin, and Dinesh Manocha. Path planning among movable obstacles: a probabilistically complete approach. In *Algorithmic Foundation of Robotics VIII*, pages 599–614. Springer, 2009.

[17] Jason Wolfe, Bhaskara Marthi, and Stuart J Russell. Combined task and motion planning for mobile manipulation. In *the proceedings of the International Conference on Automated Planning and Scheduling*, pages 254–258, 2010.