Asymptotically Optimal Feedback Planning: FMM Meets Adaptive Mesh Refinement

Dmitry S. Yershov and Emilio Frazzoli

Massachusetts Institute of Technology, Cambridge MA 02139, USA,
[yershov,frazzoli]@mit.edu

Abstract. The current work presents a deterministic asymptotically optimal feedback motion planning algorithm. Our algorithm is based on two well-established numerical practices: 1) the Fast Marching Method (FMM), which is a numerical solver for Hamilton-Jacobi-Bellman optimality equations, and 2) the adaptive mesh refinement algorithm designed to improve the resolution a simplicial mesh and the quality of a numerical solution. In our method, we exploit the dynamic programming principle to focus the refinement in the vicinity of the optimal trajectory. Numerical experiments confirm that this novel goal-oriented refinement algorithm outperforms previous optimal planning algorithms, for example, PRM* or RRT*, in that our method uses fewer discretization points to achieve a similar quality approximate optimal path.

Keywords: optimal planning, feedback planning, shortest path problem, Fast Marching Method, adaptive mesh refinement

1 Introduction

After decades of considerable research effort, the problem of optimal motion planning remains a challenging task in robotics. Even for a simple point robot, which moves freely among polyhedral obstacles in 3D, the optimal planning problem is already PSPACE-hard [8]. There is no doubt that planning an optimal path under general mathematical models, which usually include underactuated systems, kinodynamic constrains, configuration spaces with a nontrivial topology, obstacles that are not polyhedral, or some combination of the above, is even harder. Recently, the development of sampling-based asymptotically optimal planners [16] led to a flurry of related planning algorithms [15, 20, 28, 38]. Based on the Rapidly-exploring Random Graph structure and its variations (RRT* and PRM*), these algorithms are guaranteed to approach an optimal solution in the limit of a dense sample sequence. In this paper, we contribute to this recent interest in asymptotically optimal planning and introduce a novel, but fundamentally different, approach to this problem.

Historically, the attention was given to the shortest path problem in 2D polygonal environments. This problem admits a semianalytical solution, which is computed using a visibility graph of a polygon [29]. The time complexity of Dijkstra's graph search algorithm is $O(N \log(N) + E)$ with respect to the

number of vertices, N, and number of edges E. In the worst case, $E = O(N^2)$ for visibility graphs. Thus, various algorithms have been proposed to reduce this cost to the theoretical lower bound $O(N \log(N))$ [14, 22, 23]. In 3D, however, this method has been used only as an approximate algorithm [10, 25, 29].

Methods based on visibility graphs do not generalize to complex models of motion and nonpolygonal environments. Therefore, many approximate motion planning algorithms use reachability graphs instead. Once again, Dijkstra's algorithm is used commonly to search the shortest path in the reachability graph. Moreover, the A* algorithm employs an *admissible* and *consistent* heuristic for computing a provably minimum number of vertices and finding the optimal path on a graph [13]. Advanced graph search algorithms for efficient planning and replanning in dynamic environments have been proposed in [17, 18, 37].

Unfortunately, no convergence guarantees have been proven for reachability-graph-based discretizations. Motion planning algorithms that employ numerical Hamilton-Jacobi-Bellman (HJB) solvers, on the other hand, converge to the optimal solution as the resolution of discretization points increases [39]. However, a major disadvantage of HJB solvers is that a discretization is given as an input, and the output has a fixed accuracy. Thus, this approach is not *asymptotically optimal.*

In scientific computing, solution improvement techniques using mesh refinement algorithms are well-established. The early mention of mesh refinement strategy is found in [2], in which a discrete Finite Element solution is improved by increasing the resolution of a 2D Cartesian grid. In later years, mesh refinement algorithms have been extended towards non-Cartesian two- and three-dimensional meshes [6] and, eventually, d-dimensional simplicial complexes [21]. Recent developments in mesh refinement are focused on computing high-quality meshes [1, 4, 21, 27, 33], as degenerate simplices tend to result in ill-conditioned numerical discretizations.

In this paper, we present a novel asymptotically optimal feedback motion planning algorithm based on a combination of a HJB solver and a mesh refinement algorithm. Our algorithm is built on fundamentally different principles than previous asymptotically optimal planning algorithms. While the RRT* computes the shortest path only, our algorithm uses a simplicial discretization for computing an approximation of the optimal feedback plan in the free space. Global numerical error bound that depends on the mesh resolution has been proven for HJB solvers [39]; whereas, the RRT* gives no such guarantees. Both algorithms, however, approach the optimal solution in the limit as the resolution of discrete points improves. Our algorithm requires the initial coarse mesh that captures the topology of the free space. It may be difficult to compute such mesh in general; however, cell decomposition algorithms are used commonly for this purpose. In this aspect, our method is disadvantageous to the RRT* that finds an optimal solution without any prior information regarding the free space. Nevertheless, numerical experiments show that, compared with RRT*, the HJB solver computes an orders of magnitude better quality solution using the same number of discretization points, which justifies expensive initial mesh computations. Fi-

nally, our algorithm is different from the previous attempt to combine a Fast Marching HJB solver with RRT* in [15]. Previously proposed Fast Marching Trees (FMT*) algorithm solves one-dimensional local minimization problem instead of considering a provably accurate d-dimensional interpolation. In contrast to our method, FMT* is ideologically closer to Dijkstra's algorithm rather than the FMM.

2 Preliminaries

In this section we formalize the shortest path planning problem. In addition, we present an equivalent optimal control problem and derive the governing Hamilton-Jacobi-Bellman (HJB) optimality equation for a feedback control model. At the end of this section, we present the Fast Marching Method, a fast numerical HJB solver, and discuss its accuracy.

2.1 The Shortest Path Problem: Optimal Control Formulation

Let a d-dimensional Riemannian manifold X be the configuration space of a robot. A configuration $x \in X$ corresponds to a set $A(x)$ occupied by the robot in the *world* W (usually 2D or 3D). Static obstacles are present in W, and they occupy the *obstacle set* O. Robot collisions with obstacles correspond to a set of inadmissible configurations. This set we call the *configuration space obstacles*, $X_{\text{obs}} = \{x \in X \mid A(x) \cap O \neq \emptyset\}$. Configurations that do not result in a collision are denoted $X_{\text{free}} = X \setminus X_{\text{obs}}$ and called the *free space*.

The initial configuration of the robot is given as a point x_{init} in X_{free}. Let the goal set X_{goal} be a subset of X_{free}, which consists of all desired final configurations of the robot. The robot moves freely in the world, and this motion correspond to a continuous trajectory in X. The shortest path problem is to find a rectifiable trajectory between x_{init} and X_{goal}, whose graph is entirely in X_{free}, such that it is the shortest among all such trajectories.

The problem of finding the shortest path is equivalent to a minimum time optimal control problem, which is formulated as follows.

First, let $U(x)$ be the set of all unit vectors from T_x (T_x is the tangent space of X at x), and it is called the *local input set* at a point x. Let the *global input set* be defined as $U = \bigcup_{x \in X} U(x) \subset TX$. Here, TX is a tangent bundle of X.

Second, we define robot motion using an Ordinary Differential Equation (ODE) with control:

$$\dot{\tilde{x}}(t) = \tilde{u}(t) \text{ for all } t > 0, \text{ and } \tilde{x}(0) = x_{\text{init}}\,, \tag{1}$$

in which $\tilde{x} : [0, +\infty) \to X$ is a *trajectory* of a robot in the configuration space, and $\tilde{u} : [0, +\infty) \to U$ such that $\tilde{u}(t) \in U(\tilde{x}(t))$ for all $t \geq 0$ is an *input signal*. For all $\tilde{u}$ and $x_{\text{init}} \in X_{\text{free}}$, let $\tilde{x}(x_{\text{init}}, \tilde{u})$ be a Filippov solution [12] of (1).

Finally, let $t^* = \inf\{t \geq 0 \mid \tilde{x}(t) \in X_{\text{goal}}\}$. We introduce the *cost functional*

$$J(\tilde{x}) = \begin{cases} t^* & \text{if } \forall t \leq t^* \;\; x(t) \in X_{\text{free}} \\ +\infty & \text{otherwise} \end{cases}. \tag{2}$$

In this setting, the optimal control problem is to find the optimal input signal $\tilde{u}^*$ for a given initial position x_{init} such that the cost functional is minimized on the corresponding trajectory. Formally,

$$\tilde{u}^* = \arg\min_{\tilde{u}} J(\tilde{x}(x_{\text{init}}, \tilde{u}))\,. \tag{3}$$

2.2 Feedback Control Model and Feedback Planning

We compute optimal control $\tilde{u}^*$ using a feedback control model. In this model, a feasible *feedback function* $\pi : X \to U$ satisfies $\pi(x) \in U(x)$ and defines the control signal:

$$\tilde{u}(t) = \pi(\tilde{x}(t))\,. \tag{4}$$

By substituting (4) into (1), we find that, for all $x_{\text{init}} \in X_{\text{free}}$ and feasible π, the corresponding trajectory $\tilde{x}(x_{\text{init}}, \pi)$ is a solution of a regular ODE:

$$\dot{\tilde{x}}(t) = \pi(\tilde{x}(t)) \text{ for all } t > 0, \text{ and } \tilde{x}(0) = x_{\text{init}}\,. \tag{5}$$

In order to find the optimal feedback control, we introduce the optimal *cost-to-go* function $V : X_{\text{free}} \to [0, +\infty)$, which is equal to the minimum time to reach the goal when traveling in X_{free} starting from x. Formally, it is defined as follows:

$$V(x) = \min_{\pi} J(\tilde{x}(x, \pi))\,. \tag{6}$$

This cost-to-go function satisfies the HJB partial differential equation (HJB PDE), which is derived from Bellman's dynamic programming principle. It reads

$$\inf_{u \in U(x)} \langle \nabla V(x)\,, u \rangle_x + 1 = 0\,. \tag{7}$$

Here, $\langle \cdot\,, \cdot \rangle_x$ is a bilinear form between T_x and its dual, a cotangent space at $x \in X$.

Once the optimal cost-to-go function is computed, the optimal feedback function is given as the direction of the *steepest descent* of V:

$$\pi^*(x) = \arg\min_{u \in U(x)} \langle \nabla V(x)\,, u \rangle_x\,. \tag{8}$$

In the terminology of partial differential equations, this direction is called the *local characteristic* of (7). Coincidentally, the optimal trajectory is the *characteristic curve* of (7).

In Euclidean space, the HJB equation is equivalent to the Eikonal equation

$$\|\nabla V(x)\| = 1\,, \tag{9}$$

and the feedback function is simply

$$\pi^*(x) = -\nabla V(x)\,. \tag{10}$$

In this paper, however, we use (7) and (8) for a greater generality.

In the next section, we discuss a numerical discretization of the HJB PDE using a simplicial approximation of X_{free} and present a fast numerical algorithm that computes an approximation of the optimal cost-to-go function.

2.3 Fast Marching Method

The exact analytic solution of the HJB equation is rarely available. Thus, we must resort to numerical algorithms that discretize (7) and compute an approximation of the optimal cost-to-go function. Among many numerical algorithms, we now consider the Fast Marching Method (FMM) on simplicial grids, which runs in $O(N \log(N))$ time [36]. Here, N is the number of discretization points of X_{free}.

First, we begin the description of the FMM with a definition of simplicial discretization of X_{free}. Let $X_{\text{d}} = \{x_i\}_{i=1}^{N}$ be a discrete set of points in X_{free}, which we call *vertices*. An abstract simplex τ is an index set of vertices that define a *geometric simplex*, $X_\tau = \text{conv}(\{x_i\}_{i\in\tau}) \subset X_{\text{free}}$. Here, conv denotes a convex hull of a set of points. A tuple of X_{d} and $\mathcal{T}$ is called a simplicial complex if 1) for all $\tau \in \mathcal{T}$ and $\tau' \subseteq \tau$, τ' is also in $\mathcal{T}$, 2) for all $\tau, \tau' \in \mathcal{T}$ it follows $\tau \bigcap \tau' \in \mathcal{T}$, and 3) for all $\tau, \tau' \in \mathcal{T}$ it follows $X_\tau \bigcap X_{\tau'} = X_{\tau \bigcap \tau'}$.

Second, using a simplicial complex, we define a piecewise linear interpolation $\hat{V}$ of the cost-to-go function:

$$\hat{V}(x) = \sum_{i\in\tau} \hat{V}_i \alpha_i(x)\,. \tag{11}$$

In the above, τ is such that $x \in X_\tau$, and $\{\alpha_i(x)\}_{i\in\tau}$ is a unique set of positive coefficients such that $x = \sum_{i\in\tau} x_i \alpha_i(x)$ and $\sum_{i\in\tau} \alpha_i(x) = 1$. Coefficients $\alpha_i(x)$ are called *barycentric coordinates* of x in X_τ. Note that $\hat{V}(x_i) = \hat{V}_i$ for all i.

In general, a piecewise linear function does not satisfy (7) at all points of X_{free}. Therefore, we enforce this relation at vertices only:

$$\min_{\tau\in\text{St}(i)} \inf_{u\in U_{i,\tau}} \langle \nabla_\tau \hat{V}\,, u\rangle_x + 1 = 0\,. \tag{12}$$

In the above, $\text{St}(i)$ (which is called a *star* of i) is a set of abstract simplices that contain index i, ∇_τ is the gradient operator constrained to the geometric representation X_τ, and $U_{i,\tau}$ is a subset of U that consists of all unit vectors with the origin at x_i pointing inside X_τ. In general, this discretization has *positive* coefficients, and it is *monotone* if simplicial discretization is acute [5]. It has been shown that monotone and consistent discretization of HJB converges to the *viscosity* solution [11].

Considered at all vertices, (12) defines a system of nonlinear equations with respect to unknown values $\hat{V}_i$ for $1 \le i \le N$. In contrast to traditional numerical methods that solve such systems iteratively, the FMM exploits the monotonicity of the discrete Hamiltonian and computes $\hat{V} = \{\hat{V}_i\}_{i=1}^{N}$ in optimal $O(N \log(N))$ time [35]. The FMM is outlined in Algorithm 1.

Algorithm 1 parallels Dijkstra's algorithm in that vertices are computed in the increasing order of the respective cost-to-go values. The difference between Dijkstra's algorithm and the FMM is in line 8. In the former, **minloc** is equal to the value of the neighbor vertex plus the corresponding edge wight. In the latter, this function is defined as a solution of the following constrained minimization

Algorithm 1 Fast Marching Method

Input: A simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$ and a goal set X_{goal}
Output: An approximation $\hat{V}$ of the cost-to-go function at all vertices x_i

Initialize a priority queue PQ of indices i for which $x_i \in X_{\mathrm{goal}}$
Set the priority key $\hat{K}_i \leftarrow 0$ if $x_i \in X_{\mathrm{goal}}$, and $\hat{K}_i \leftarrow \infty$ otherwise
while PQ is not empty **do**
  Pop j with the least key $\hat{K}_j$ from PQ
  Set $\hat{V}_j \leftarrow \hat{K}_j$
  for all $\tau \in \mathrm{St}(j)$ **do**
    for all $i \in \tau \setminus \{j\}$ **do**
      $\hat{V}^* \leftarrow \mathbf{minloc}(i, \tau)$
      if $\hat{V}^* < \hat{K}_i$ **then**
        Update the key of i to $\hat{V}^*$ in PQ
        Push i into PQ if $i \notin PQ$
return $\hat{V} = \{\hat{V}_i\}_{i=1}^N$

problem: minimize

$$\hat{V}_i = \inf_{\{\alpha_j : j \in \tau \setminus \{i\}\}} \Big\{ \sum_j \alpha_j \hat{V}_j + \mathrm{dist}\left(x_i, \sum_j \alpha_j x_j\right) \Big\} \tag{13}$$

such that

$$\alpha_j \geq 0 \text{ for all } j \in \tau \setminus \{i\} \text{ and } \sum_j \alpha_j = 1\,. \tag{14}$$

Here, dist is a distance function on X. If in (13) the minimizing argument $\alpha_j \neq 0$, then we say that $\hat{V}_i$ depends on $\hat{V}_j$. Unlike in Dijkstra's algorithm, $\hat{V}_i$ may depend on multiple $\hat{V}_j$s in FMM.

2.4 Numerical Error

As with most numerical discretizations, (12) introduces the numerical error, which is defined as follows:

$$E = \max_{x \in X_{\mathrm{free}}} |V(x) - \hat{V}(x)|\,. \tag{15}$$

The following theorem establishes that E is proportional to the simplicial mesh *resolution*, $h = \max_{\tau \in \mathcal{T}} \max_{x, x' \in X_\tau} \mathrm{dist}(x, x')$, which means that the approximation (12) is of the *first order*.

Theorem 1 (Global Error Bound). *For an approximate solution computed using* (12)*,*

$$E \leq Ch \tag{16}$$

for some $C > 0$, independent of h.

Proof. See the proof of Theorem 4 in [39].

3 Accuracy Improving Adaptive Mesh Refinement

In the previous section, we established the relation between the mesh resolution and the numerical error. In this section, we introduce the concept of mesh refinement, which, when combined with the FMM, results in an asymptotically optimal path planning algorithm.

3.1 Overview of Mesh Refinement

In two-dimensions, the *uniform* mesh refinement generates a sequence of two-dimensional meshes by splitting all triangles into four congruent triangles using their middle lines. The resolution of the generated meshes doubles at each refinement step. According to Theorem 1, the numerical error of the solution sequence computed by FMM on the generated meshes converges to zero.

The uniform refinement is intuitive in 2D; however, it is not trivial to extend it to higher dimensions. In [6], octasection based subdivision has been proposed for tetrahedral meshes (simplicial complexes in 3D), but d-dimensional implementations are not known to this day.

Even if d-dimensional uniform refinement can be computed, a major disadvantage of this approach is that the rate of computational demand dwarfs the convergence rate for uniformly refined meshes. Indeed, assuming that all edges of the simplicial complex are split into two edges, the number of vertices increases at the rate 2^d; whereas, the resolution decreases at the rate $1/2$. Thus, the amount of computations per digit of accuracy increases exponentially.

This problem has been realized in scientific computing, and *adaptive* mesh refinement algorithms have been proposed. The idea is to automatically improve the mesh resolution in the region where solution has a high variance and keep the resolution low elsewhere in order to reduce the computational cost.

In 2D, an intuitive adaptive refinement algorithm is to split the *goal* triangles into four congruent triangles and bisect the adjacent triangles [3]. The bisection of adjacent triangles is called the *closure* refinement, and it is necessary to ensure that the output of the algorithm is a simplicial complex.

As it was discussed earlier, the proposed splitting of goal triangles cannot be extended to higher dimensions. Thus, a simpler *bisection* refinement has been introduced in 2D, 3D, and d-dimensional cases; see [34], [4, 33], and [21] respectively. The bisection refinement algorithm consists of two stages: 1) refine goal simplices by splitting selected edges into two and 2) apply the closure refinement.

Various edge-selection criteria for bisection refinement have been developed in the past. For example, the method of Maubach [21] selects previously unrefined edges in the lexicographic order of their combinatorial representation. It has been shown that, if the initial complex is a tessellation of the hypercube, then the number of similarity classes among refined simplices is uniformly bounded. This property is important to guarantee that simplices do not degenerate and the discrete problem is well-conditioned. However, this property fails to hold for general simplicial meshes [30].

Algorithm 2 Skeleton-Based Bisection Refinement

Input: A simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$ and a set of marked edges $(\mathcal{E}, \{\kappa_\varepsilon\}_{\varepsilon\in\mathcal{E}})$
Output: A refined simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$

for all ε in $\mathcal{E}$ **do**
  Let $\varepsilon = (i, j)$
  Add vertex $x_k = \kappa_\varepsilon x_i + (1 - \kappa_\varepsilon)x_j$ to X_{d}
  for all τ in $\mathrm{St}(i) \bigcap \mathrm{St}(j)$ **do**
    Split τ into τ' and τ'' at vertex x_k, and replace τ with τ' and τ'' in $\mathcal{T}$
return $(X_{\mathrm{d}}, \mathcal{T})$

In 2D, Mitchell proposed selecting the edge opposite the newly introduced vertex for the next refinement step [24]. This algorithm also guarantees a bound on the number of similarity classes of refined triangles. Unfortunately, this *newest vertex* edge-selection strategy cannot be extended to high-dimensional cases. However, this method is equivalent to Rivara's four-triangle longest-edge splitting [32], which, in turn, extends to higher dimensions.

In [33], Rivara and Levin presented the longest-edge refinement algorithm for three-dimensional simplicial meshes. Although no provable guarantees on mesh quality are given, empirical evidence suggest that the refined tetrahedra do not degenerate in the limit of an infinite refinement.

The edge-selection method proposed in [1] avoids costly edge-length computations and relies on a *marked simplex* combinatorial data structure instead. Moreover, it has been proven that the number of similarity classes of simplices is bounded by $dd!2^{d-2}$, which in 3D is equal to 36. However, in high-dimensional cases this bounds is extremely large, and can be considered infinite for practical purposes.

Considering a dynamical system of sequentially refined triangles in a hyperbolic geometry, it was shown in [27] that trisection guarantees high-quality refined meshes; whereas the n-section method for $n \geq 4$ can produce degenerate triangles [26]. More advanced refinement strategies are discussed in [7, 9, 19].

In this paper, we use a skeleton-based edge-selection and bisection refinement algorithms, as this combination demonstrates a good balance between the computational complexity of the refinement algorithm [31] and empirical mesh quality guarantees [30]. In addition, this algorithm is easily generalized to d-dimensional simplicial meshes.

3.2 Skeleton-Based Mesh Refinement

In Algorithm 2, we outline the skeleton-based bisection refinement algorithm. This algorithm takes a list of selected edges and refines adjacent simplices according to predefined weights.

Note that different order of skeleton edges results in different refined meshes. Thus, edge selection algorithm plays an important role in the refinement process. In the next section, we present a characteristic-driven edge selection algorithm.

3.3 Selecting Refinement Edges

Ideally, we would like to refine only those simplices, geometric representations of which contain the shortest path. However, it is not possible for two reasons: 1) the exact shortest path is unknown and 2) the interpolation requires accurate computations outside of immediate vicinity of the shortest path. In Fig. 1, we illustrate conditions, under which the dependency of the cost-to-go values and the shortest path are well-separated in X_{free}. In this example, a mesh refinement must be carried in the volume of space rather than along the shortest path. However, the dynamic programming principle dictates that values of the optimal cost-to-go function depend only on its values along the characteristics curve (the optimal trajectory). Motivated by this principle, we would like to reduce the dependency region in the refined meshes.

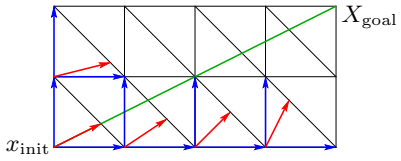
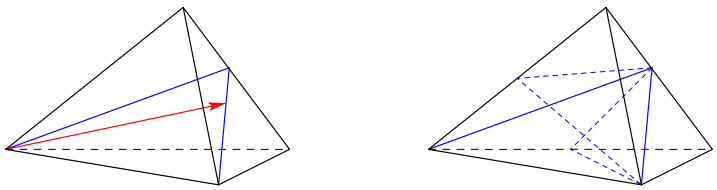


Fig. 1: Local characteristic (red arrows), approximate cost-to-go function dependencies (blue arrows), and the shortest path (green line).

To this end, we introduce characteristic-driven edge selection algorithm (see Algorithm 3), which computes a skeleton based on local characteristic direction and the dependency tree of cost-to-go value at the initial robot position. The selection process consists of two steps.

First, we find the dependency tree of the cost-to-go function at x_{init}. This tree is an overestimation of the set of simplices which contain the shortest path. For all simplices in the dependency tree, we compute the local characteristic ($\arg\min_{u\in U} \langle \nabla_\tau \hat{V}, u \rangle_x$) and select the edge that is the closest to it. The splitting weight is such that the newly introduced point is in the hyperplane spanned by the local characteristic and the remaining vertices of the simplex that are not on the selected edge. See Fig. 2(a) and lines 3–11 in Algorithm 3 for details.



(a) Splitting along the local characteristic (b) Successive bisection splitting

Fig. 2: Characteristic-driven mesh refinement algorithm.

Algorithm 3 Characteristic-Driven Edge Selection

Input: A simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$ and computed $\hat{V}$, $1/2 \leq \beta_1, \beta_2 \leq 1$
Output: A set of edges $\mathcal{E}$ and weights $\{\kappa_\varepsilon\}_{\varepsilon\in\mathcal{E}}$ marked for refinement

1: Initialize $\mathcal{E} = \emptyset$
2: Initialize vertex queue $Q_{\mathrm{vertex}} \leftarrow \tau_{\mathrm{init}}$ such that $x_{\mathrm{init}} \in X_{\tau_{\mathrm{init}}}$
3: **while** Q_{vertex} is not empty **do**
4: Pop i from Q_{vertex}
5: Find J and $\{\alpha_j\}_{j\in J}$ such that $\hat{V}_i = \sum_{j\in J} \alpha_j \hat{V}_j + \mathrm{dist}\left(x_i, \sum_{j\in J} \alpha_j x_j\right)$
6: Let j^* and j^{**} be such that $\alpha_{j^*} \geq \alpha_{j^{**}} \geq \alpha_j$ for all $j \in J \setminus \{j^*, j^{**}\}$
7: **if** $(1-\beta_1) \leq \alpha_{j^*}/(\alpha_{j^*} + \alpha_{j^{**}}) \leq \beta_1$ **then**
8: $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{(j^*, j^{**})\}$ and $\kappa_{(j^*,j^{**})} \leftarrow \alpha_{j^*}/(\alpha_{j^*} + \alpha_{j^{**}})$
9: **for all** $j \in J$ **do**
10: **if** $\alpha_j \geq (1-\beta_2)$ **then**
11: Push j to Q_{vertex}
12: Initialize simplex queue $Q_{\mathrm{simplex}} \leftarrow \{\tau \in \mathcal{T} \mid \exists \varepsilon \in \mathcal{E}$ such that $\varepsilon \subset \tau\}$
13: **while** Q_{simplex} is not empty **do**
14: Pop τ from Q_{simplex}
15: Find ε, the longest edge in τ
16: Let $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{\tau\}$ and $\kappa_\varepsilon \leftarrow 1/2$
17: **for all** τ' that contain ε as edge, and $\tau' \neq \tau$ **do**
18: **if** ε is **not** the longest edge in τ' **then**
19: Push τ' to Q_{simplex}
20: Sort all newly introduced edges of $\mathcal{E}$ in the decreasing order of their lengths
21: **return** $\mathcal{E}$ and $\{\kappa_\varepsilon\}_{\varepsilon\in\mathcal{E}}$

Second, for all simplices involved in the refinement, select the longest edge for further refinement. This longest-edge selection is done recursively until no new simplex is involved (lines 13–19 of Algorithm 3). The longest-edge selection improves the quality of the refined mesh Fig. 2(b).

Two parameters of the algorithm, β_1 and β_2, are introduced. We call them *badness* parameters. The first parameter, β_1, controls the maximum aspect ratio between the longest and the shortest edges of the refined simplices. If $\beta_1 = 1/2$, then our algorithm is equivalent to the longest-edge bisection refinement. The second parameter, β_2, controls the dependency tree width. All dependencies are included if $\beta_2 = 1$.

A rather straightforward combination of the FMM and the characteristic-driven skeleton-based refinement strategy results in the asymptotically optimal planning algorithm (Algorithm 4).

4 Experimental Results

To illustrate the performance of the proposed asymptotically optimal planning algorithm, we consider several test cases similar to those in [16]. In many cases, a direct comparison between our planning algorithm and the previous planners, for example, the PRM* or RRT*, was virtually impossible due to significant scale

Algorithm 4 Planning Algorithm

Input: The initial (coarse) simplicial complex $(X_{\mathrm{d}}, \mathcal{T})$ and the goal set X_{goal}

while true do
  $\hat{V} \leftarrow$ Fast Marching Method$((X_{\mathrm{d}}, \mathcal{T}), X_{\mathrm{goal}})$
  $(\mathcal{E}, \{\kappa_\varepsilon\}_{\varepsilon\in\mathcal{E}}) \leftarrow$ Characteristic-Driven Edge Selection$((X_{\mathrm{d}}, \mathcal{T}), \hat{V})$
  $(X_{\mathrm{d}}, \mathcal{T}) \leftarrow$ Execute Skeleton-Based Bisection Refinement$((X_{\mathrm{d}}, \mathcal{T}), (\mathcal{E}, \{\kappa_\varepsilon\}_{\varepsilon\in\mathcal{E}}))$
  yield return $\hat{V}$ and the corresponding feedback function to a controller

difference in the numerical error, which was consistently lower for the FMM. Motivated by nondegeneracy of trisection refinement algorithm [27], we chose $\beta_1 = 2/3$. Also, $\beta_2 = 0.9$ in all test cases.

In the first four test cases, a robot is located at the vertex of a d-dimensional hypercube for d between 2 and 5. The goal is at the vertex opposite the initial position. Since the length of the shortest path is known, we compute the numerical error with respect to the vertex number (Fig. 3(a)). Note that E is proportional to $\sqrt{N}$ in 2D, and the convergence rate decreases in higher dimensions.

For the next three test cases, we add a hypercube obstacle in the middle of the previously considered environment. The convergence of the shortest path length is shown in Fig. 3(b), (c), and (d). In this case, however, we plot the actual cost because the optimal solution is unknown.

Next, we compute the shortest path in a 2D environment, which is similar to that in [16], with and without obstacles. The initial position is at the center of the environment; the goal set is the square region in the upper-right corner. For the environment without obstacles, the initial coarse simplicial discretization is presented in Fig. 4(a). In Fig. 4(b), we show the mesh after five refinement steps using uniform refinement algorithm. In Figs. 4(c) and (d), the result of 15 steps of the longest-edge bisection and characteristic-driven refinements are illustrated. In all cases, only simplices for which the approximate cost-to-go function is computed are visible. Note that adaptive refinement algorithms increase the mesh resolution in the vicinity of the optimal path. Moreover, characteristic-driven mesh refinement focuses numerical computations more tightly. The convergence of the numerical shortest path is shown in Fig. 5, for the environment without obstacles, and in Fig. 7, for the environment with obstacles. Compared with Figs. 13 and 16 in [16], the solution computed by our algorithm on the initial mesh with fewer than a hundred vertices is of the same quality as that computed by RRT* using more than two thousand vertices, in the case without obstacles, and six thousand vertices, in the case with obstacles. We attribute a faster convergence of the FMM to the use of the interpolation, which allows to chose the shortest path in the continuum of directions between discretization points; whereas, RRT* is constrained to the shortest path on the graph.

Note that, in the environment with obstacles, the numerical error convergence of characteristic-driven refinement is close to that of the longest-edge bisection refinement. This behavior is due to moderately cluttered environment, in which obstacles constrain the refinement process; see Fig. 6. In the case of no ob-

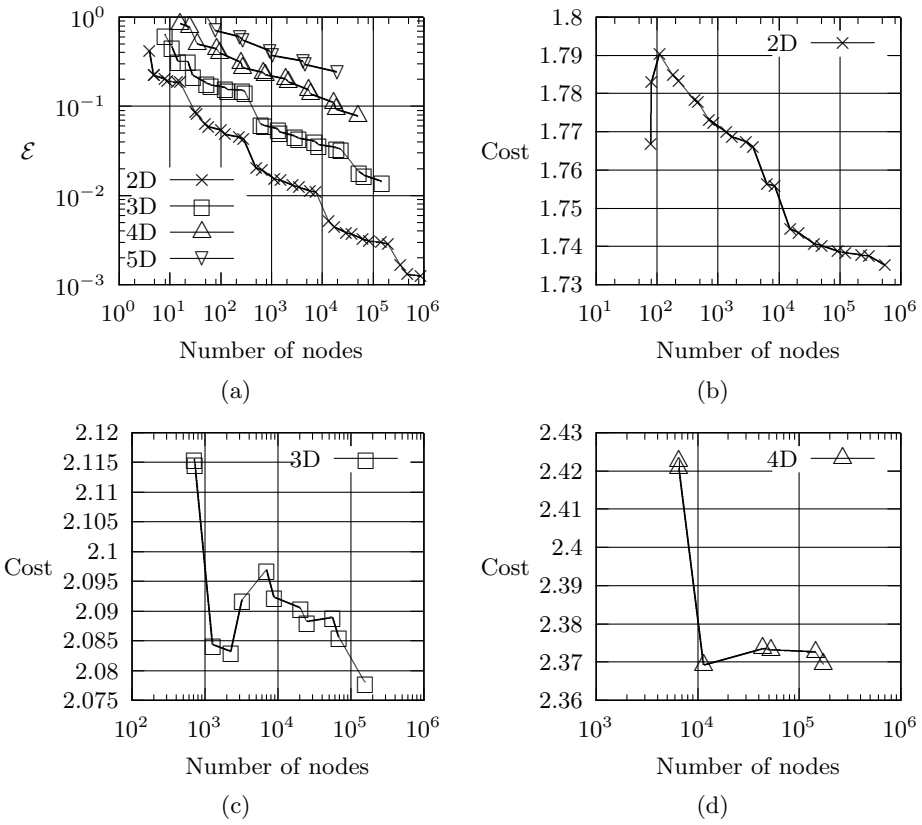


Fig. 3: Shortest path convergence in d-dimensional hypercube without obstacles (a), and 2D, 3D, and 4D with a hypercube obstacle at the center of volume 0.5 (b), (c), and (d).

stacles, however, characteristic-driven edge selection algorithm outperforms the traditionally used longest-edge selection algorithm.

5 Summary

Considered is the novel asymptotically optimal algorithm for the shortest path problem among obstacles. Compared with the previous algorithms, our method is build on fundamentally different principles: the Fast Marching HJB solver and adaptive mesh refinement. Benefits of using our approach include provable error bounds with respect to the mesh resolution, feedback control computations for stable optimal trajectory executions, improved solution accuracy at a similar resolution of discretization points. Numerical experiments show that, on the test cases considered in [16], the numerical error is consistently lower for our

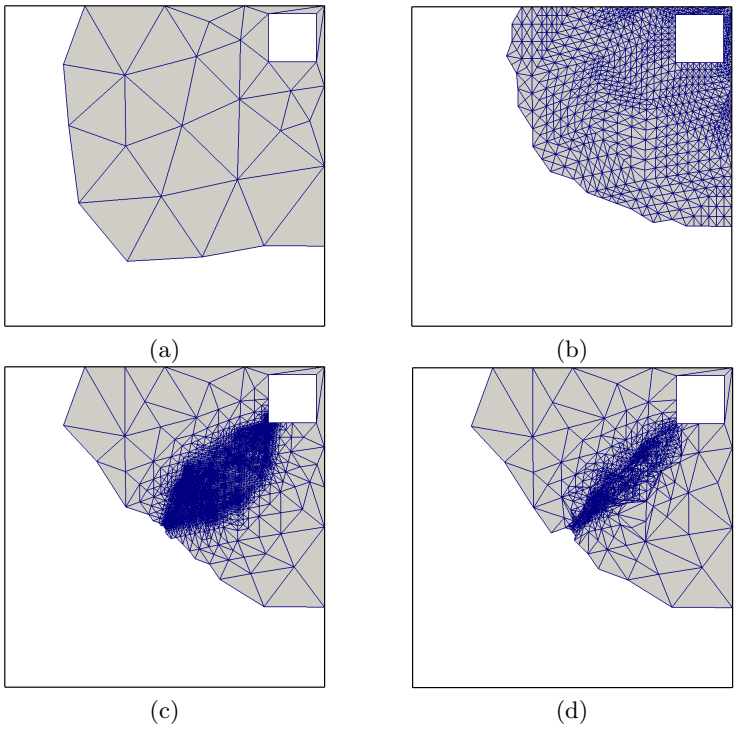


Fig. 4: Discretization mesh of the 2D environment without obstacles: (a) the initial coarse mesh, (b) uniform refinement—5 steps, (c) longest-edge bisection—15 steps, (d) characteristic-driven—15 steps.

algorithm than that for the RRT*. We hope that our findings will fuel the interest in optimal motion planning further.

References

1. D. Arnold, A. Mukherjee, and L. Pouly, "Locally adapted tetrahedral meshes using bisection," *SIAM Journal on Scientific Computing*, vol. 22, no. 2, pp. 431–448, 2000.
2. I. Babuška and W. Rheinboldt, "Error estimates for adaptive finite element computations," *SIAM Journal on Numerical Analysis*, vol. 15, no. 4, pp. 736–754, 1978.
3. R. E. Bank and A. H. Sherman, "The use of adaptive grid refinement for badly behaved elliptic partial differential equations," *Mathematics and Computers in Simulation*, vol. 22, no. 1, pp. 18–24, Mar. 1980.
4. E. Bänsch, "Local mesh refinement in 2 and 3 dimensions," *IMPACT of Computing in Science and Engineering*, vol. 3, no. 3, pp. 181–191, Sep. 1991.
5. T. J. Barth and J. A. Sethian, "Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains," *Journal of Computational Physics*, vol. 145, no. 1, pp. 1–40, Sep. 1998.

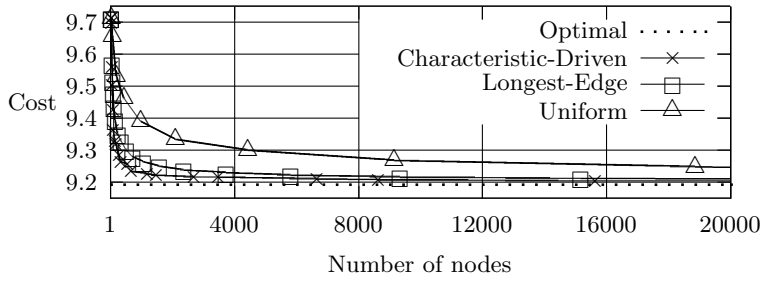


Fig. 5: Convergence of the numerical shortest path in the 2D environment without obstacles.

6. J. Bey, "Tetrahedral grid refinement," *Computing*, vol. 55, no. 4, pp. 355–378, 1995.
7. J. Brandts, S. Korotov, M. Křížek, and J. Šolc, "On nonobtuse simplicial partitions," *SIAM Review*, vol. 51, no. 2, pp. 317–335, 2009.
8. J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Proceedings of 28th Annual Symposium on Foundations of Computer Science*, Oct. 1987, pp. 49–60.
9. J.-H. Choi, K.-R. Byun, and H.-J. Hwang, "Quality-improved local refinement of tetrahedral mesh based on element-wise refinement switching," *Journal of Computational Physics*, vol. 192, no. 1, pp. 312–324, Nov. 2003.
10. J. Choi, J. Sellen, and C. K. Yap, "Approximate euclidean shortest path in 3-space," in *Proceedings of the tenth annual symposium on Computational geometry*, ser. SCG '94. New York, NY, USA: ACM, 1994, pp. 41–48.
11. M. G. Crandall and P. L. Lions, "Two approximations of solutions of Hamilton-Jacobi equations," *Mathematics of Computation*, vol. 43, no. 167, pp. 1–19, 1984.
12. A. F. Filippov, *Differential Equations with Discontinuous Righthand Sides: Control Systems*, 1st ed. Springer, Sep. 1988.
13. P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Feb. 1968.
14. J. Hershberger and S. Suri, "An optimal algorithm for euclidean shortest paths in the plane," *SIAM J. Comput.*, vol. 28, pp. 2215–2256, Aug. 1999.
15. L. Janson and M. Pavone, "Fast marching trees: a fast marching sampling-based method for optimal motion planning in many dimensions – extended version," Jul. 2013. [Online]. Available: http://arxiv.org/abs/1306.3532
16. S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
17. S. Koenig and M. Likhachev, "D* lite," in *AAAI Conference of Artificial Intelligence*, 2002.
18. S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1, pp. 93–146, 2004.
19. S. Korotov and M. Křížek, "Acute type refinements of tetrahedral partitions of polyhedral domains," *SIAM J. Numer. Anal.*, vol. 39, no. 2, pp. 724–733, Feb. 2001.

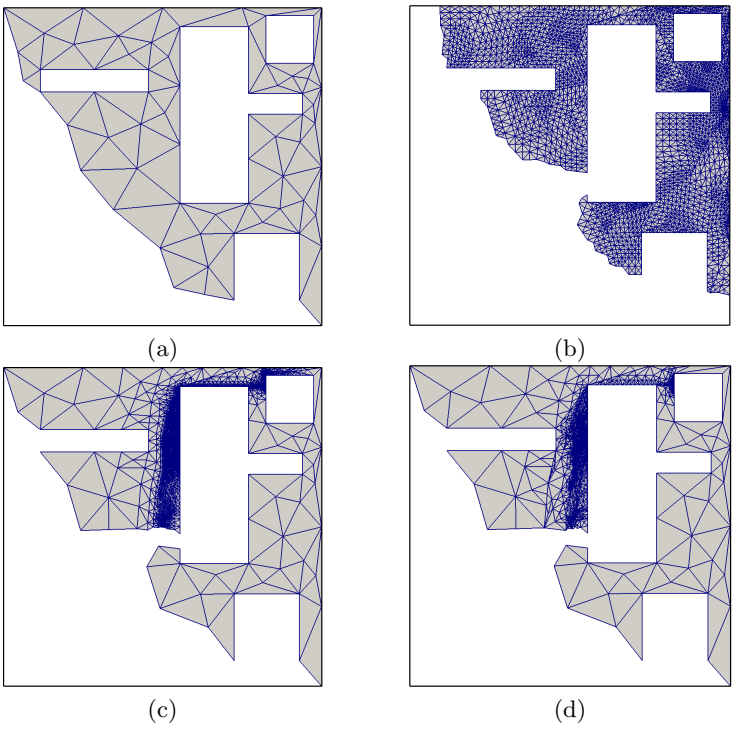


Fig. 6: Discretization mesh of the 2D environment with obstacles: (a) the initial coarse mesh, (b) uniform refinement—5 steps, (c) longest-edge bisection—15 steps, (d) characteristic-driven—15 steps.

20. J. Marble and K. E. Bekris, "Asymptotically near-optimal planning with probabilistic roadmap spanners," *IEEE Transactions on Robotics*, vol. 29, pp. 432–444, 2013.
21. J. Maubach, "Local bisection refinement for n-simplicial grids generated by reflection," *SIAM Journal on Scientific Computing*, vol. 16, no. 1, pp. 210–227, 1995.
22. J. S. B. Mitchell, "Shortest paths among obstacles in the plane," *Int. J. Comput. Geometry Appl.*, vol. 6, no. 3, pp. 309–332, 1996.
23. J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM Journal on Computing*, vol. 16, no. 4, pp. 647–668, 1987.
24. W. Mitchell, "Optimal multilevel iterative methods for adaptive grids," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 1, pp. 146–167, 1992.
25. C. H. Papadimitriou, "An algorithm for shortest-path motion in three dimensions," *Information Processing Letters*, vol. 20, no. 5, pp. 259–263, Jun. 1985.
26. F. Perdomo and A. Plaza, "A new proof of the degeneracy property of the longest-edge n-section refinement scheme for triangular meshes," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 2342–2344, Nov. 2012.
27. ——, "Proving the non-degeneracy of the longest-edge trisection by a space of triangular shapes with hyperbolic metric," *Applied Mathematics and Computation*, vol. 221, pp. 424–432, Sep. 2013.

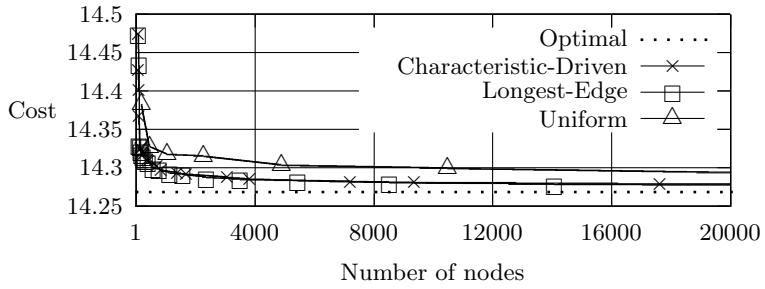


Fig. 7: Convergence of the numerical shortest path in the 2D environment with obstacles.

28. A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, May 2012, pp. 2537–2542.
29. T. L. Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.
30. A. Plaza and G. F. Carey, "Local refinement of simplicial grids based on the skeleton," *Applied Numerical Mathematics*, vol. 32, no. 2, pp. 195–218, Feb. 2000.
31. A. Plaza and M.-C. Rivara, "On the adjacencies of triangular meshes based on skeleton-regular partitions," *Journal of Computational and Applied Mathematics*, vol. 140, no. 1-2, pp. 673–693, Mar. 2002.
32. M.-C. Rivara, "A grid generator based on 4-triangles conforming mesh-refinement algorithms," *Int. J. Numer. Meth. Engng.*, vol. 24, no. 7, pp. 1343–1354, Jul. 1987.
33. M.-C. Rivara and C. Levin, "A 3-D refinement algorithm suitable for adaptive and multi-grid techniques," *Commun. appl. numer. methods*, vol. 8, no. 5, pp. 281–290, May 1992.
34. I. G. Rosenberg and F. Stenger, "A lower bound on the angles of triangles constructed by bisecting the longest side," *Mathematics of Computation*, vol. 29, no. 130, pp. 390–395, 1975.
35. J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, Feb. 1996.
36. J. A. Sethian and A. Vladimirsky, "Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes," *Proceedings of the National Academy of Sciences*, vol. 97, no. 11, pp. 5699–5703, May 2000.
37. A. Stentz, "The focussed D* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, Aug. 1995.
38. D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, May 2013, pp. 5054–5061.
39. D. S. Yershov and S. M. LaValle, "Simplicial Dijkstra and A* algorithms: From graphs to continuous spaces," *Advanced Robotics*, vol. 26, no. 17, pp. 2065–2085, Oct. 2012.