Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Continuous Path Planning with Multiple Constraints*

Ian M. Mitchell† and Shankar Sastry
Department of Electrical Engineering & Computer Science
University of California, Berkeley
{imitchel,sastry}@eecs.berkeley.edu

Abstract

We examine the problem of planning a path through a low dimensional continuous state space subject to upper bounds on several additive cost metrics. For the single cost case, previously published research has proposed constructing the paths by gradient descent on a local minima free value function. This value function is the solution of the Eikonal partial differential equation, and efficient algorithms have been designed to compute it. In this paper we propose an auxiliary partial differential equation with which we can evaluate multiple additive cost metrics for paths which are generated by value functions; solving this auxiliary equation adds little more work to the value function computation. We then propose an algorithm which generates paths whose costs lie on the Pareto optimal surface for each possible destination location, and we can choose from these paths those which satisfy the constraints. The procedure is practical when the sum of the state space dimension and number of cost metrics is roughly six or below.

1 Introduction

Few problems are as well studied as the path planning or routing problem; it appears in engineering disciplines that vary from robotics to wireless communication to matrix factorization. A major challenge in developing solutions to the problem are the many, sometime subtle, variations it can adopt: the topology of the state space and cost metrics, the types of acceptable paths, the number of sources and destinations, the acceptable degree of optimality, etc. While every variant has at least one solution method—enumerate all feasible paths until an acceptably optimal one is found—the key to developing efficient solution algorithms is to take advantage of the particular properties of the variant of interest.

In this paper we examine the path planning problem in a continuous state space subject to constraints on additive path integral cost metrics. The original motivation for this work was the planning of fuel constrained flight paths for unmanned aerial vehicles through enviroments with varying levels of threat. Paths are generated by gradient descent on a value function (with no local minima), which is the solution of an Eikonal partial differential equation (PDE).* Path integral costs are evaluated by solving an auxiliary PDE. Both PDEs can be solved quickly for low dimensional systems, thus yielding a practical algorithm for path planning. Because both PDEs are solved over the entire state space, paths to any possible destination can be rapidly evaluated.

To handle constraints, we sample the Pareto optimal surface looking for paths with feasible combinations of costs. The sampling method only reaches the convex hull of the Pareto surface, so for nonconvex problems it may not always find the optimal feasible path; however, in our experience the degree of nonconvexity has not been enough to cause significant problems.

The asymptotic cost of the algorithm is $\mathcal{O}(MdN^d \log N)$, where M is the number of sampled points on the Pareto optimal surface, d is the state space dimension, and N is the number of grid points in each state space dimension. To adequately sample the Pareto surface, M will typically be exponential in the number of separate cost functions k. While these two exponentials are daunting, in practice the algorithms described below are quite practical on the desktop when the sum $k + d$ is less than around five or six; for example, section 3 includes a problem in two dimensions with three cost functions that can be solved to reasonable accuracy in about one minute on the authors' laptop computer.

Gradient descent on a value function solution of the Eikonal equation has been used previously for unconstrained, single cost path planning problems. The innovative contribution of this paper is the application

*Research supported by ONR MURI N00014-02-1-0720.
†Corresponding author.

*Classical applications of the Eikonal PDE are in the fields of optics and seismology. Its solution can be interpreted as a first arrival time or a cost to go, depending on whether the boundary conditions represent sources or sinks.

0-7803-7924-1/03/$17.00 ©2003 IEEE

of auxiliary PDEs to calculate multiple path integral costs, and the use of those costs to find constrained optimal paths.

In the remainder of this section we formally outline our path planning problem and examine related work. Subsequent sections describe the algorithm, provide an example, and discuss extensions to more general problems.

We work in a state space $\mathbb{R}^d$. Unless otherwise specified, norms are Euclidean $\|\cdot\| = \|\cdot\|_2$. Let $\mathbb{R}^+ = (0, \infty)$ be the set of strictly positive real numbers, and $\mathbb{R}^\oplus = [0, \infty)$ be the set of non-negative real numbers.

1.1 Problem Definition

A *path* $p : \mathbb{R}^\oplus \times \mathbb{R}^d \to \mathbb{R}^d$ is parameterized by an arclength $s \in \mathbb{R}^\oplus$ and a destination location $x \in \mathbb{R}^d$. Assume that all paths have a single *source location* $x_s \in \mathbb{R}^d$ (we will relax this assumption later). The *path cost functions* $\{c_i(x)\}_{i=1}^k$, where $c_i : \mathbb{R}^d \to \mathbb{R}^+$, are continuous, bounded and strictly positive. The cost along a path is additive, so the total cost of a path can be evaluated by a *path integral*

$$P_i(x) = \int_0^T c_i(p(s,x))ds, \text{ with } \begin{cases} p(0,x) & = x_s, \\ p(T,x) & = x. \end{cases} \quad (1)$$

In words, $P_i : \mathbb{R}^d \to \mathbb{R}$ is the total cost, according to path cost function $c_i(\cdot)$, of following the path $p(\cdot, x)$ from the source location x_s to the point x.

As an example, consider planning the flight path of an aircraft from its base at x_s to various destinations. The most obvious path cost function is fuel, which we approximate as a constant $c_{\text{fuel}}(x) = c_{\text{fuel}}$. A second path cost function might be the threat of inclement weather $c_{\text{wthr}}(x)$. A third might be uncertainty about the enviroment, encoded as $c_{\text{uncr}}(x)$. The latter two costs are inhomogenous, meaning that their value depends on x. Examples of cost functions are shown in figures 2 and 3.

There are two related problems that we might wish to solve starting from the parameters x_s and $\{c_i(x)\}_{i=1}^k$ described above. Given some set of *cost constraints* $\{C_i\}_{i=1}^k$, where $C_i \in \mathbb{R}^+$, we might want to find feasible paths such that $P_i(x) \leq C_i$ for all $i = 1 \ldots k$. Alternatively, we might try to minimize $P_1(x)$ subject to constraints on the remaining costs $P_i(x) \leq C_i$ for all $i = 2 \ldots k$. In either case, we will usually be interested in quantitative measures of the tradeoffs between the various path cost functions.

1.2 Related Work

The significance of the most closely related algorithmic work [1, 2, 3] is discussed in section 2.4. However, similar problems have been investigated in several other fields.

Path planning is a central endeavor in robotics research [4], so we mention only the most closely related work. The algorithm discussed in this paper could be categorized as a potential field approach [5], in the sense that the paths are determined by gradient descent on a scalar function defined over the state space. In particular, the value function constructed in section 2 is an example of a navigation function [6]—a potential field free of the local minima that hinder most potential field methods (although in general it will contain saddle points). The specific use of the Eikonal equation for robot path planning in the single cost case was examined in [7], and is equivalent to the approach used in [8].

Independently, the networking community has been solving constrained shortest path planning on discrete graphs [9, 10, 11], primarily for the purpose of network routing. While this research involves problems with multiple costs, it makes some discreteness assumptions that do not apply in this setting. It should be noted, however, that our method for exploring the Pareto optimal surface of possible path costs by sampling values of λ (see section 2.3) is equivalent to the fastest algorithm proposed for finding constrained shortest paths in [11].

The related work that is closest mathematically is a tomographic application [12], which uses the Eikonal equation (2) to calculate travel time and a version of the path integral PDE (4) to determine perturbations of a linearized form of the Eikonal equation. To our knowledge, the use of (4) for evaluating path integral costs is original.

2 Value Function Solution

We discuss the value function method for finding the shortest path in the single cost case, and then how to compute path integrals along value function generated paths. With these tools we can explore the range of paths that might meet the constraints when multiple cost functions are involved. This section concludes with a discussion of an efficient algorithm for solving the required differential equations.

2.1 Single Objective Shortest Path

Consider the simplest case $k = 1$ with a single path cost function $c(x) = c_1(x)$ (because it will be used to generate a value function, we call this cost $c(x)$ the *value cost function*). It can be shown that the minimum cost to go from the source x_s to any point x in the state space is the solution of the inhomogenous Eikonal equation

$$\|\nabla V(x)\| = c(x) \quad \text{for } x \in \mathbb{R}^d, \quad \text{with boundary condition } V(x_s) = 0. \quad (2)$$

The solution $V : \mathbb{R}^d \to \mathbb{R}^{\oplus}$ of this PDE is called the *value function*. In practice V is rarely differentiable and therefore (2) does not have a solution in the classical sense. The viscosity solution [13] is the appropriate weak solution for the shortest path problem.

Given the viscosity solution V, the optimal path $p^*(\cdot, x)$ can be determined by gradient descent of V from a fixed target location x. In practical terms, let $\hat{p}(s, x)$ be a path that starts at a particular x and terminates at x_s. Then $\hat{p}$ is the solution to the ordinary differential equation (ODE)

$$\begin{aligned} \frac{d\hat{p}(s,x)}{ds} &= \frac{\nabla V(\hat{p}(s,x))}{\|\nabla V(\hat{p}(s,x))\|} \\ &\text{for } s \in \mathbb{R}^{\oplus} \text{ and fixed } x \in \mathbb{R}^d, \\ &\text{with initial condition } \hat{p}(0,x) = x. \end{aligned} \tag{3}$$

We stop extending the solution at some $\hat{s}$ such that $\hat{p}(\hat{s}, x) = x_s$. Then $\hat{s} = T$ is the arclength of the shortest path from x_s to x, and that path is given by $p^*(s, x) = \hat{p}(T - s, x)$. Because $V(x)$ is the cost to get to x from x_s along path p^*, the path integral for this path is $P^*(x) = V(x)$. The gradient descent (3) cannot get stuck in local minima because V has none.[†] In theory, (3) can get stuck at saddle points of V, but the stable manifolds of such points are of measure zero in the state space, and are thus unlikely to be a problem in practical implementations subject to floating point roundoff noise.

2.2 Computing Path Integrals

Throughout the remainder of this paper, we consider only paths generated by (3) for some value function V. In this section we examine how to compute the path integral when the value cost function is not the same as the path cost function. To differentiate the two cost functions, we denote the value cost function in (2) by $c(x)$ and the path cost function in (1) by $c_i(x)$. Both must use the same source location x_s.

Starting from the differential form of (1), we formally derive a PDE for the path integral $P_i(x)$

$$\begin{aligned} \frac{dP_i(p(s,x))}{ds} &= c_i(p(s,x)), \\ \frac{\partial P_i(p(s,x))}{\partial p(s,x)} \cdot \frac{dp(s,x)}{ds} &= c_i(p(s,x)), \\ \nabla P_i(p(s,x)) \cdot \frac{\nabla V(p(s,x))}{\|\nabla V(p(s,x))\|} &= c_i(p(s,x)), \\ \nabla P_i(p(s,x)) \cdot \nabla V(p(s,x)) &= c_i(p(s,x))c(p(s,x)), \end{aligned}$$

where (3) is used in the second step and (2) is used in the third. Consequently, for all reachable points in the state space,

$$\begin{aligned} \nabla P_i(x) \cdot \nabla V(x) = c_i(x)c(x) \quad &\text{for } x \in \mathbb{R}^d, \\ &\text{with boundary condition } P_i(x_s) = 0. \end{aligned} \tag{4}$$

Because the cost structure is isotropic (independent of path direction) the system is small time controllable and for our single source version all states will be reachable. The derivation above assumes that all the functions involved are differentiable, but as was stated earlier this assumption will fail for $V(x)$ and therefore likely also for $P_i(x)$. We are in the process of developing a robust proof that the viscosity solution of (4) is the path cost integral we seek.

[†] Easily seen if V is differentiable, since a local minimum would require $\nabla V(x) = 0$, but $c(x) > 0$. A more rigorous argument based on the positivity of c can be constructed when V is a viscosity solution.

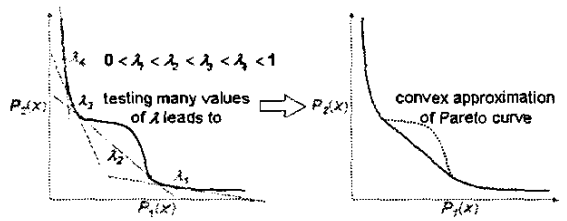

Figure 1: Pareto optimal curve for a particular destination state x. *Left: each value of λ samples* a point on the curve. Right: testing all values of λ would yield a convex approximation of the Pareto curve.

2.3 Exploring Potential Paths

As discussed in section 1.1, one of our goals was an algorithm to generate feasible paths subject to a collection of cost constraints. In the previous two sections we described PDEs whose solutions were a path generating value function V in (2) and the path integrals P_i for those paths in (4). The remaining missing ingredient is the value cost function $c(x)$ in (2). In this section we discuss the results of using convex combinations of the path cost functions as the value cost function.

We start with the simplest multiobjective case, $k = 2$. Let

$$c^\lambda(x) = \lambda c_1(x) + (1 - \lambda)c_2(x) \quad \text{for some } \lambda \in [0, 1].$$

Then evaluate (2) and (4) for $V^\lambda(x)$, $P_1^\lambda(x)$ and $P_2^\lambda(x)$. The first thing to notice is that $\lambda = 1$ calculates paths optimal in c_1 and $\lambda = 0$ paths optimal in c_2. Therefore, if $P_1^{\lambda=1}(x) > C_1$ or $P_2^{\lambda=0}(x) > C_2$ for some point x, there cannot be any feasible paths from x_s to x. Intermediate values of λ will generate paths lying somewhere between these two extremes.

Testing all possible values of λ would effectively construct the convex hull of the Pareto optimal tradeoff curve between the two cost functions. Figure 1 shows a possible Pareto curve for a single point x, the points

on that curve determined by several values of λ, and the convex hull of that curve. A point on the curve is a pair $(P_1^\lambda(x), P_2^\lambda(x))$ and lies where a line of slope $\frac{\lambda}{(\lambda-1)}$ is tangent to the Pareto curve. Therefore, λ is a quantitative measure of the tradeoff between the two cost functions.

In general the Pareto curve is not convex, so this method may fail to detect a feasible path even if one exists. Nonconvexity in the Pareto curve will manifest itself by jumps in the values of the path integrals $P_1^\lambda(x)$ and $P_2^\lambda(x)$ for fixed x as λ is varied continuously; for example, consider the jump in path integrals as λ is varied in the range $[\lambda_2 - \epsilon, \lambda_2 + \epsilon]$ for some small $\epsilon > 0$ in figure 1. However, neighboring values of λ can be used to bound the error in the convex approximation and nonconvexity has not been a problem in our experience. It should be pointed out that the Pareto curve characterised above is for a single point x in the state space. Because V^λ and P_i^λ are calculated over the entire state space, the technique actually approximates a separate Pareto curve for all points x.

To handle the case $k > 2$, we simply choose a set $\{\lambda_j\}_{j=1}^k$ such that $\lambda_j \in [0,1]$ for all j and $\sum_{j=1}^k \lambda_j = 1$. Then $c^{\{\lambda_j\}}(x) = \sum_{j=1}^k \lambda_j c_j(x)$, and we can solve for the corresponding value function and path cost integrals. In this case it is the convex hull of the Pareto optimal surface that is explored as the set $\{\lambda_j\}$ is varied.

2.4 Numerical Algorithms

The discussion above would be nothing more than a mathematical diversion if it were not possible to solve (2), (3) and (4) numerically for some practical problems. In this section we provide some pointers to the algorithms that we have used. For more details, see [14].

To treat (3), we assume that (2) and (4) can be computed for a variety of λ values to generate $V^\lambda(x)$ and $\{P_i^\lambda(x)\}_{i=1}^k$. Then a particular $\hat{\lambda}$ is chosen such that any path integral constraints are satisfied ($P_i^{\hat{\lambda}}(x) \leq C_i$). A path is determined by solving (3) for value function $V^{\hat{\lambda}}(x)$ with a standard ODE integration method, such as Runge-Kutta.

Solving (2) efficiently relies on an algorithm first described in [1], although our implementation is based on an equivalent version [2] commonly known as the *Fast Marching Method* (FMM). This algorithm is basically the Dijkstra algorithm for computing shortest paths in a discrete graph [15], suitably modified to deal with a continuous state space. For readers interested in alternatives, there are other algorithms for solving (2); for example, [16, 17].

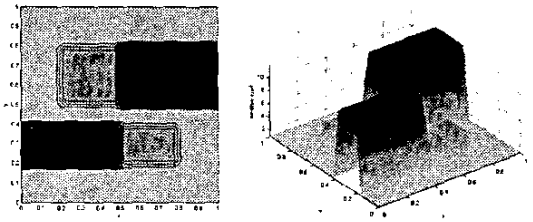

Figure 2: Weather threat cost function $c_{\text{wthr}}(x)$

The value function $V(x)$ is approximated on a Cartesian grid over the state space with N nodes in each dimension, for a total of N^d nodes. Direct application of Dijkstra's algorithm on this discrete Cartesian graph remains a popular approximation method for this problem; however, the paths generated by such an approximation measure their cost metrics in a coordinate dependent manner,[‡] and are visibly segmented at the grid's resolution. In contrast, FMM approximations can generate paths with subgrid resolution (see section 3); paths that are reasonably smooth for practically sized grids. Furthermore, these approximations are theoretically convergent, meaning that the approximation approaches the true value function solution of (2) as $N \to \infty$ on all of the state space except a subset of measure zero. The cost of this algorithm is $\mathcal{O}(dN^d \log N)$.

To solve (4), we use an approximation scheme outlined in [3]. The "extension velocity" $F_{\text{ext}}(x)$ described in that paper is computed by solving

$$\nabla F_{\text{ext}}(x) \cdot \nabla V(x) = 0,$$

which is just (4) with a zero right hand side. In practice, we integrate the computation of $P_i(x)$ into the FMM computation of $V(x)$, and it requires little additional work.

3 An Example

For our example we consider planning a path for an aircraft flying across the idealized unit square country from lower left to upper right. We focus on a two dimensional example primarily because three dimensional paths are very challenging to visualize on paper. The first cost function will be fuel, which we assume is a constant $c_{\text{fuel}}(x) = c_{\text{fuel}} = 1$.

[‡]For example, Dijkstra on a square Cartesian grid measures distance with the Manhattan or 1-norm; in this norm the distance between two points depends on the alignment of the coordinate axes. While this axis alignment bias can be reduced by adding more edges to the graph, it will persist unless every possible path is enumerated by making the graph completely dense. The solution of the Eikonal equation (2) measures distance in the Euclidean or 2-norm, which is independent of axis alignment.

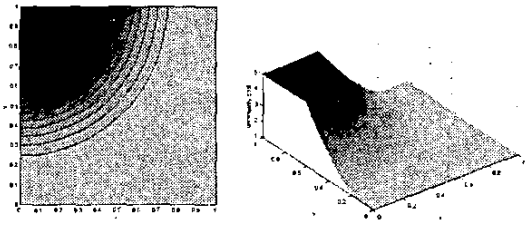

Figure 3: Uncertainty cost function $c_{\text{uncr}}(x)$

The second cost function will represent the threat of weather related problems $c_{\text{wthr}}(x)$. Note that the intuitive quantification of weather threat would be the probability of encountering a storm along the flight path. This quantification cannot be used as a cost because probabilities are not additive; however, under an independence assumption they can be transformed into an additive cost by a logarithmic transformation. The figures and tables below assume that this transformation has been performed in generating $c_{\text{wthr}}(x)$ from meteorologically determined storm probabilities. Figure 2 shows a simple weather threat cost map $c_{\text{wthr}}(x)$. Notice that the lower high threat bar extending from the left is slightly thinner than the upper high threat bar extending from the right.

Ideally, this weather forecast would be an accurate short term estimate of weather threat. Unfortunately, only part of our fictional country (the right and bottom sides) is well monitored and can thus generate accurate short term estimates. The remaining part of the country (the upper left) is poorly monitored and in this region we are forced to resort to long term climatological estimates. Because these long term estimates are less accurate, we introduce a third cost function $c_{\text{uncr}}(x)$ which will penalize paths through the poorly instrumented region of the country, and which is shown in figure 3.

To compute approximations to (2) and (4), we have implemented a version of the FMM described in section 2.4 in C++ for Cartesian grids. While the code itself can handle any dimension, in practice the physical memory limits of desktop machines restrict the dimension to at most five even with very coarse grids. Using a MEX interface, these PDE solving routines can be called directly from Matlab. We use Matlab scripts to sample in λ space, and Matlab's `ode23t` to solve (3).

For this example, we plan paths from the source to a single destination; however, once the $V^{\lambda}(x)$ and $P_i^{\lambda}(x)$ are built, the cost of a path to any destination can be found by simply evaluating these functions at that destination point.

A variety of optimal paths from the source $x_s = \begin{bmatrix}0.1 & 0.1\end{bmatrix}^T$ (marked by a ○ symbol) to the destination $x_d = \begin{bmatrix}0.9 & 0.9\end{bmatrix}^T$ (marked by a □ symbol) are shown in figure 4 and described in table 1. The first three (marked by dotted lines) are each optimal in one of the three cost metrics, ignoring the others. They can be distinguished by their lengths: the shortest is the fuel optimal cost, the slightly longer one is the uncertainty optimal cost, and the longest is the weather optimal cost. The remaining paths are constrained in some manner. Notice in particular the difference between the dashed and dash dot paths. The latter satisfies the same fuel constraint, but trades a lower uncertainty cost (taking a route in the lower right) for a higher weather cost (it crosses the thicker high weather threat bar at the top rather than the thin one at the bottom).

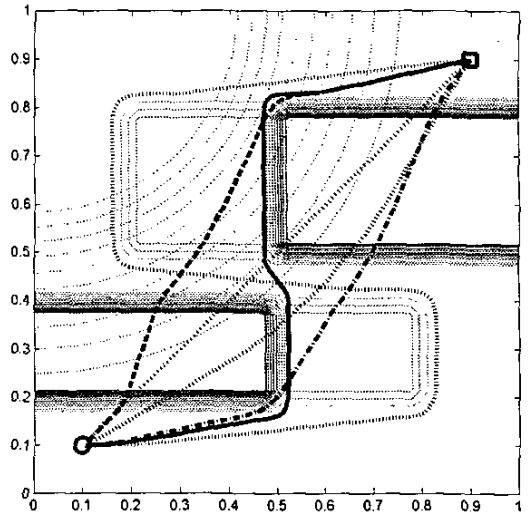

Figure 4: Some constrained and unconstrained optimal paths. The properties of each path are explained in table 1.

line type	minimize what?	constraints fuel	constraints wthr	costs fuel	costs wthr	costs uncr
dotted	fuel	none	none	1.14	8.81	1.50
dotted	uncr	none	none	1.17	8.41	1.17
dotted	wthr	none	none	2.69	2.71	5.83
solid	wthr	1.6	none	1.60	3.02	2.84
dashed	wthr	1.3	none	1.30	4.42	2.58
dash dot	uncr	1.3	6.0	1.23	5.84	1.23

Table 1: Properties of paths in figure 4.

This example in $d = 2$ dimensions and with $k = 3$ cost metrics was run on an $N = 201$ grid, sampling the Pareto surface uniformly at $\Delta\lambda = 0.01$ intervals (so $M = 101^2$). It took 13.5 minutes to run, of which all but 10 seconds were spent solving instances of (2) and (4). Results of nearly the same quality can be achieved in just one minute by halving N and doubling

$\Delta\lambda$. For a three dimensional example and more details on the implementation, timing and the effects of grid refinement, see [14].

4 Discussion

We have demonstrated an algorithm for constrained path planning in continuous state spaces for additive cost metrics and isotropic but inhomogenous and non-convex cost functions. In those cases with multiple cost functions, a convex approximation of the Pareto optimal surface is explored; consequently, the algorithm may not find all feasible paths although in practice this has rarely been a problem. While the asymptotic cost of the algorithm is exponential in the dimension and in the number of cost functions (assuming uniform sampling of the Pareto optimal surface), it can be run at interactive rates on the desktop if their sum is five or less, and overnight if their sum is six.

There are several straightforward extensions of this work to more general path planning problems. We can immediately incorporate multiple source locations, by making each source a boundary condition with value zero of the PDEs (2) and (4). The resulting value function will generate paths from the nearest source to each destination state. Hard obstacles in the state space can be treated by either making the cost function very large in their interior or by making the obstacle's boundary a part of the PDEs' boundaries with very large value. Creating boundary nodes with intermediate values (neither zero nor very large) can be interpreted as penalizing those nodes as possible source locations. We can also swap the meaning of source and destination, in which case the value function can be used to generate a feedback control.

The basic FMM algorithm described in section 2.4 has been extended to unstructured meshes, and a more accurate second order approximation scheme has been developed. For more details on FMM and its extensions, we refer the reader to [18]. We are in the process of developing a version of FMM that runs on an adaptively refined Cartesian grid, so as to better represent problems with hard obstacles. We are also investigating how other path cost models might be incorporated into this framework, including anisotropic cost metrics (the cost depends on state and direction of travel) and ways of evaluating maximum cost along a path, rather than integral cost.

Acknowledgements: We would like to thank Aniruddha Pant for suggesting the λ sweeping procedure, and Professor Pravin Varaiya for the interpretation of this procedure as a convex approximation of the Pareto optimal surface and for several very useful discussions of value function properties. Thanks are also due to the Berkeley MICA team for providing the examples which originally motivated this work.

References

[1] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. AC-40, no. 9, pp. 1528–1538, 1995.

[2] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences, USA*, vol. 93, no. 4, pp. 1591–1595, 1996.

[3] D. Adalsteinsson and J. A. Sethian, "The fast construction of extension velocities in level set methods," *Journal of Computational Physics*, vol. 148, pp. 2–22, 1999.

[4] J.-C. Latombe, *Robot Motion Planning*. Boston: Kluwer Academic Publishers, 1991.

[5] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[6] J. Barraquand, B. Langlois, and J. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[7] R. Kimmel and J. A. Sethian, "Optimal algorithm for shape from shading and path planning," *Journal of Mathematical Imaging and Vision*, vol. 14, no. 3, pp. 237–244, 2001.

[8] K. Konolige, "A gradient method for realtime robot control," in *International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, (Takamatsu, Japan), pp. 639–646, 2000.

[9] A. Orda, "Routing with end-to-end QoS guarantees in broadband networks," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 365–374, June 1999.

[10] G. Liu and K. G. Ramakrishnan, "A*prune: An algorithm for finding k shortest paths subject to multiple constraints," in *INFOCOM 2001*, vol. 2, pp. 743–749, 2001.

[11] A. Puri and S. Tripakis, "Algorithms for routing with multiple constraints," in *AIPS 2002 Workshop on Planning and Scheduling using Multiple Criteria*, (Toulouse, France), pp. 7–14, April 2002.

[12] A. Sei and W. W. Symes, "Convergent finite-difference traveltime gradient for tomography," in *Proceedings of 65^{th} Society of Exploration Geophysicists Annual Meeting*, (Houston, TX), pp. 1258–1261, 1995.

[13] M. G. Crandall, L. C. Evans, and P.-L. Lions, "Some properties of viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 282, no. 2, pp. 487–502, 1984.

[14] I. Mitchell and S. Sastry, "Continuous path planning with multiple constraints," Tech. Rep. UCB/ERL M03/34, Department of Electrical Engineering and Computer Science, University of California, Berkeley, August 2003.

[15] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik 1*, pp. 269–271, 1959.

[16] M. Falcone, "Numerical solution of dynamic programming equations," in *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*, Birkhäuser, 1997. Appendix A of [19].

[17] H.-K. Zhao, "Fast sweeping method for Eikonal equations I: Distance function," tech. rep., UCI, Department of Mathematics, University of California, Irvine, CA, 92697-3875, 2002. Under review, SINUM.

[18] J. A. Sethian, *Level Set Methods and Fast Marching Methods*. New York: Cambridge University Press, 1999.

[19] M. Bardi and I. Capuzzo-Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations*. Boston: Birkhäuser, 1997.