Ordered Upwind Methods for Hybrid Control

James A. Sethian[1] and Alexander Vladimirsky[2]

[1] Dept. of Mathematics, University of California, Berkeley, California 94720
sethian@math.berkeley.edu,
http://www.math.berkeley.edu/∼sethian
[2] Dept. of Mathematics, Cornell University, Ithaca, New York
vlad@math.cornell.edu

Abstract. We introduce a family of highly efficient (non-iterative) numerical methods for a wide class of hybrid control systems. The application of Dijkstra's classical method to a discrete optimal trajectory problem on a network obtains the solution in $O(M \log M)$ operations. The key idea behind the method is a careful use of the direction of information propagation, stemming from the optimality principle. In a series of recent papers, we have introduced a number of Ordered Upwind Methods (OUMs) to efficiently solve the fully anisotropic continuous optimal control problems. These techniques rely on using a partial information on the characteristic directions of the Hamilton–Jacobi–Bellman PDE, stemming from the continuous variant of the optimality principle. The resulting non-iterative algorithms have the computational complexity of $O(M \log M)$, where M is the total number of grid points where the solution is computed, regardless of the dimension of the control/state variables. In this paper, we show how Ordered Upwind Methods may be extended to efficiently solve the hybrid (discrete/continuous) control problems. We illustrate our methods by solving a series of hybrid optimal trajectory problems with and without time-dependence of anisotropic speed functions.

1 Introduction

The dynamical programming approach to all (discrete, continuous, or hybrid) control problems relies on some version of the optimality principle [1]. The resulting equations for the value function are quite often non-linear and coupled (in the continuous case, we are referring to the system of difference equations which discretize the appropriate PDE). Solving a system of coupled non-linear equations using iterative numerical methods can be rather expensive. Our goal is to exploit the optimality principle to build fast numerical methods for equations arising in control theory. The basis for our techniques is the notion of *causality* (i.e., the direction of flow of information) corresponding to a particular class of the control problems.

In this paper, we introduce a family of highly efficient (non-iterative) "Ordered Upwind Methods" (OUMs) for a wide class of hybrid control problems. These methods combine the notion of discrete causality (the basis for Dijkstra's

C.J. Tomlin and M.R. Greenstreet (Eds.): HSCC 2002, LNCS 2289, pp. 393–406, 2002.

method [5]) with the causality of continuous anisotropic control problems (stemming from our recent work on Hamilton–Jacobi PDEs [12]). The resulting computational complexity is the order of $O(M \log M)$, where M is the total number of mesh points used to discretize the continuous state space of the system[1].

We apply these methods to a collection of representative problems from hybrid (discrete/continuous) optimal trajectory planning. A traveler wishes to find the minimum time necessary to reach some pre-defined destination in the domain Ω, where the speed may depend on the direction of motion; this might be applicable, for example, to walking in an area of hilly terrain. Here, we imagine the continuous problem in which motion in any direction is allowed. Furthermore, suppose that bus lines are also available from some fixed points to other given points; these represent discrete links/transitions superimposed on the continuous domain. The considered generalizations include dependence of continuous dynamics upon the discrete state (a traveler carries a pair of skates, which can be put on/taken off, thus changing the anisotropic speed function on different slopes) and time-dependent dynamics (a traveler becomes more tired - and slower - as time goes by, and buses follow their prescribed schedules instead of waiting for the traveler at the stop).

The outline of this paper is as follow. In Section 2, we review Dijkstra's method for discrete control, framed as a single-pass algorithm whose efficiency comes from exploiting the causality (i.e., the direction of flow of information). In Section 3, we show how to build Ordered Upwind Methods for continuous optimal control, again exploiting the (obtained and updated during the computation) knowledge of the flow of information. In Section 4, we show how to develop Ordered Upwind Methods for hybrid control systems and illustrate these algorithms by solving several test-problems.

2 Discrete Control: Dijkstra's Method

Consider a discrete optimal trajectory problem on a network. Here, given a network and a time-penalty associated with each node, the global optimal trajectory problem is to determine the quickest path from a starting node to some exit set in the network. Dijkstra's method [5] is a classic algorithm for solving this problem; it is used to compute the minimal time of exiting starting at any node of the network, and the solution is obtained in $O(M \log M)$ operations. We note that the time-penalty can depend not only on the particular node, but also on the particular link chosen in that node. Thus, Dijkstra's method applies to both *isotropic* and *anisotropic* control problems. The distinction is minor for discrete problems, but significant for continuous problems. Dijkstra's method is a "single-pass" algorithm; if r is the maximum incidence of the nodes in the network, each point on the network is "updated" at most r times to produce the

[1] For the sake of notational clarity, we restrict our discussion to hybrid systems with continuous state component in R^2; all results can be restated for R^n and for meshes on manifolds.

solution. This efficiency comes from a careful use of the direction of information propagation and stems from the optimality principle.

We briefly summarize Dijkstra's method, since its overall structure will be important in explaining our Ordered Upwind Methods. For simplicity, imagine a rectangular grid of size h, where the time-penalty $C_{ij} > 0$ is given for passing through each grid point $\boldsymbol{x_{ij}} = (ih, jh)$. Given a starting point, the minimal total time U_{ij} of arriving at the node $\boldsymbol{x_{ij}}$ can be written in terms of the minimal total time of arriving at its neighbors:

$$U_{ij} = \min\left(U_{i-1,j}, U_{i+1,j}, U_{i,j-1}, U_{i,j+1}\right) + C_{ij}. \tag{1}$$

To find the minimal total time, Dijkstra's method works as follows. All the mesh points are divided into three classes: *Far* (no information about the correct value of U is known), *Accepted* (the correct value of U has been computed), and *Considered* (adjacent to *Accepted*).

1. Start with all mesh points in *Far* ($U_{ij} = \infty$).
2. Move the boundary mesh points ($\boldsymbol{x_{ij}} \in \partial\Omega$) to *Accepted* ($U_{ij} = q(\boldsymbol{x_{ij}})$).
3. Move all the mesh points $\boldsymbol{x_{ij}}$ adjacent to the boundary into *Considered* and evaluate the tentative value of U_{ij} using the values at the adjacent *Accepted* mesh points according to formula 1.
4. Find the mesh point $\boldsymbol{x_r}$ with the smallest value of U among all the *Considered*.
5. Move $\boldsymbol{x_r}$ to *Accepted*.
6. Move the *Far* mesh points adjacent to $\boldsymbol{x_r}$ into *Considered*.
7. Re-evaluate the value for all the *Considered* $\boldsymbol{x_{ij}}$ adjacent to $\boldsymbol{x_r}$. If the new computed value is less than the previous tentative value for $\boldsymbol{x_{ij}}$ then update U_{ij}.
8. If *Considered* is not empty then go to 4).

The described algorithm has the computational complexity of $O(M \log(M))$; the factor of $\log(M)$ reflects the necessity of maintaining a sorted list of the *Considered* values U_i to determine the next *Accepted* mesh point. An efficient implementation of the algorithm can be constructed using heap-sort data structures.

3 Continuous Control: Ordered Upwind Methods

Consider now the problem of continuous optimal control; here, the goal is to find the optimal path from a starting position to an exit set. It is well-known that Dijkstra's method does not converge to the continuous solution as the mesh is refined. This can be seen by considering the simple problem of a uniform Cartesian grid with a constant time-penalty $C > 0$ for passing through every node. The minimal time from a starting point to an end point is the Manhattan distance on this grid times C. Thus, Dijkstra's method produces the solution to the partial differential equation

$$\max(|u_x|, |u_y|) = h \cdot C,$$

where h is the grid size (see [10]). As h goes to zero, this does not converge to the solution of the continuous Eikonal problem given by

$$|u_x^2 + u_y^2|^{1/2} = C$$

Thus, Dijkstra's method cannot be used to obtain a solution to the continuous problem.

3.1 Dijkstra-like Solvers for Isotropic Continuous Control Problems

In the isotropic case, when the speed depends only on the position and not on the direction of motion, two recent algorithms, namely Tsitsiklis's Method [15],[16] and Sethian's Fast Marching Method [9] have been introduced to solve the problems with the same computational complexity as Dijkstra's method. Both methods exploit information about the flow of information to obtain this efficiency; the causality allows one to build the solution in increasing order, which yields the Dijkstra-like nature of the solutions.

Both algorithms result from a key feature of Eikonal equations, namely that their characteristic lines coincide with the gradient lines of the viscosity solution $u(\boldsymbol{x})$; this allows the construction of single-pass algorithms. Tsitsiklis' algorithm evolved from studying isotropic min-time optimal trajectory problems, and involves solving a minimization problem to update the solution. Sethian's Fast Marching Method evolved from studying isotropic front propagation problems, and involves an upwind finite difference formulation to update the solution. Each method starts with a particular (and different) coupled discretization and each shows that the resulting system can be decoupled through a causality property. We refer the reader to these papers for details on ordered upwind methods for Eikonal equations.

3.2 Ordered Upwind Methods for General Anisotropic Continuous Control Problems

Consider now the general continuous optimal trajectory problem, in which the speed function depends on both position and direction. In [12], Sethian and Vladimirsky built and developed single-pass "Ordered Upwind Methods" for the anisotropic continuous optimal control problems. They showed how to to produce the solution U_i by recalculating each U_i at most r times, where r depends only on the degree of anisotropy present in the PDE and on the mesh structure, but not upon the number of mesh points. The convergence to the viscosity solution was proven, and numerical schemes were developed for a wide range of applications.

Building efficient single-pass methods for general optimal control problems is considerably more challenging than it is for the Eikonal case, since the characteristics no longer coincide with the gradient lines of the viscosity solution. As a result, the characteristics and gradient lines may in fact lie in different simplexes. Therefore, the approximation U_i to the min-time function $u(\boldsymbol{x_i})$ may well depend upon the approximate min-time value U_j at some adjacent mesh

point $\boldsymbol{x_j}$ even if U_i is smaller than U_j. This is precisely why both Sethian's Fast Marching Method and Tsitsiklis' Algorithm cannot be directly applied in the anisotropic (non-Eikonal) case: it is no longer possible to de-couple the system by computing/accepting the mesh points in the ascending order.

We now explain the idea behind Ordered Upwind Methods for general continuous optimal control. The key idea behind the Ordered Upwind Methods for the non-Eikonal optimal control (introduced in [12],[17]) is to use the measure of the local anisotropy of the speed function to limit the number of *Accepted* points that might contribute to the update of each *Considered* point. Consider the anisotropic min-time optimal trajectory problems, in which the speed of motion depends not only on position but also on direction. The value function u for such problems is the viscosity solution of the static Hamilton-Jacobi-Bellman equation

$$\begin{array}{ll} \max_{\boldsymbol{a}\in S_1} \{(\nabla u(\boldsymbol{x}) \cdot (-\boldsymbol{a}))f(\boldsymbol{x},\boldsymbol{a})\} = 1, & \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) = q(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega. \end{array} \tag{2}$$

In this formulation, $\boldsymbol{a}$ is the unit vector determining the direction of motion, $f(\boldsymbol{x},\boldsymbol{a})$ is the speed of motion in the direction $\boldsymbol{a}$ starting from the point $\boldsymbol{x} \in \Omega$, and $q(\boldsymbol{x})$ is the time-penalty for exiting the domain at the point $\boldsymbol{x} \in \partial\Omega$. The maximizer $\boldsymbol{a}$ corresponds to the characteristic direction for the point $\boldsymbol{x}$. If f does not depend on $\boldsymbol{a}$, Eqn. 2 reduces to the Eikonal equation, see [1]. Furthermore, we assume that

$$0 < F_1 \leq f(\boldsymbol{x},\boldsymbol{a}) \leq F_2 < \infty,$$

and define the anisotropy ratio $\Upsilon = F_2/F_1$.

Technical comment: This is sufficient to show that the value function is continuous in the interior of Ω (and is equal to the viscosity solution of Eqn. 2) even in the presence of continuous-state constraints: the above assumptions can be used to demonstrate both Soner's tangentiality along the boundary of the constraint set (as in [14]) and the local controllability near $\partial\Omega$ (as in [2], for example). For every point $\boldsymbol{x} \in \Omega$, we define the *speed profile* $S_f(\boldsymbol{x}) = \{\boldsymbol{a}f(\boldsymbol{x},\boldsymbol{a}) \mid \boldsymbol{a} \in S_1\}$, and note that, if all speed profiles are convex, then a min-time optimal trajectory exists for all points in Ω; for non-convex speed profiles the value function $u(\boldsymbol{x})$ is still well-defined even if no minimizing trajectory exists for that point (see [17],[13] for the detailed discussion).

In [12,17], the following two lemmas were proven:

- **Lemma 1.** Consider the characteristic passing through a point $\bar{\boldsymbol{x}} \in \Omega$ and a level curve $u(\boldsymbol{x}) = C$, where $q_{max} < C < u(\bar{\boldsymbol{x}})$. The characteristic intersects that level set at some point $\tilde{\boldsymbol{x}}$. If $\bar{\boldsymbol{x}}$ is distance d away from the level set then

$$\|\tilde{\boldsymbol{x}} - \bar{\boldsymbol{x}}\| \leq d\frac{F_2}{F_1}. \tag{3}$$

- **Lemma 2.** Consider an unstructured mesh X of diameter h on Ω. Consider a simple closed curve Γ lying inside Ω with the property that for any point $\boldsymbol{x}$ on Γ, there exists a mesh point $\boldsymbol{y}$ inside Γ such that $\|\boldsymbol{x}-\boldsymbol{y}\| < h$. Suppose the mesh point $\bar{\boldsymbol{x}}_{\boldsymbol{i}}$ has the smallest value $u(\bar{\boldsymbol{x}}_{\boldsymbol{i}})$ of all of the mesh points inside

the curve. If the characteristic passing through $\bar{\boldsymbol{x}}_{\boldsymbol{i}}$ intersects that curve at some point $\tilde{\boldsymbol{x}}_{\boldsymbol{i}}$ then

$$\|\tilde{\boldsymbol{x}}_{\boldsymbol{i}} - \bar{\boldsymbol{x}}_{\boldsymbol{i}}\| \leq h\frac{F_2}{F_1}. \tag{4}$$

This means that one may use the anisotropy ratio to exclude a large fraction of the *Accepted* points in computing the update of any *Considered* points.

Building on these results we construct the following single-pass method. As before, mesh points are divided into three classes (*Far*, *Considered*, and *Accepted*). The *AcceptedFront* is defined as a set of *Accepted* mesh points, which are adjacent to some not-yet-accepted mesh points. Define the set AF of the line segments $\boldsymbol{x}_{\boldsymbol{j}}\boldsymbol{x}_{\boldsymbol{k}}$, where $\boldsymbol{x}_{\boldsymbol{j}}$ and $\boldsymbol{x}_{\boldsymbol{k}}$ are adjacent mesh points on the *AcceptedFront*, such that there exists a *Considered* mesh point $\boldsymbol{x}_{\boldsymbol{i}}$ adjacent to both $\boldsymbol{x}_{\boldsymbol{j}}$ and $\boldsymbol{x}_{\boldsymbol{k}}$. For each *Considered* mesh point $\boldsymbol{x}_{\boldsymbol{i}}$ we define the part of AF "relevant to $\boldsymbol{x}_{\boldsymbol{i}}$":

$$NF(\boldsymbol{x}_{\boldsymbol{i}}) = \left\{ (\boldsymbol{x}_{\boldsymbol{j}}, \boldsymbol{x}_{\boldsymbol{k}}) \in AF \mid \exists \tilde{\boldsymbol{x}} \text{ on } (\boldsymbol{x}_{\boldsymbol{j}}, \boldsymbol{x}_{\boldsymbol{k}}) \text{ s.t. } \|\tilde{\boldsymbol{x}} - \boldsymbol{x}_{\boldsymbol{i}}\| \leq h\frac{F_2}{F_1} \right\}.$$

We will further assume that some consistent upwinding update formula is available: if the characteristic for $\boldsymbol{x}_{\boldsymbol{i}}$ lies in the simplex $\boldsymbol{x}_{\boldsymbol{i}}\boldsymbol{x}_{\boldsymbol{j}}\boldsymbol{x}_{\boldsymbol{k}}$ then $U_i = K(U_j, U_k, \boldsymbol{x}_{\boldsymbol{i}}, \boldsymbol{x}_{\boldsymbol{j}}, \boldsymbol{x}_{\boldsymbol{k}})$. For the sake of notational simplicity we will refer to this value as $K_{j,k}$.

1. Start with all mesh points in *Far* ($U_i = \infty$).
2. Move the boundary mesh points ($\boldsymbol{x}_{\boldsymbol{i}} \in \partial\Omega$) to *Accepted* ($U_i = q(\boldsymbol{x}_{\boldsymbol{i}})$).
3. Move all the mesh points $\boldsymbol{x}_{\boldsymbol{i}}$ adjacent to the boundary into *Considered* and evaluate the tentative value of $U_i = \min_{(\boldsymbol{x}_{\boldsymbol{j}}, \boldsymbol{x}_{\boldsymbol{k}}) NF(\boldsymbol{x}_{\boldsymbol{i}})} K_{j,k}$.
4. Find the mesh point $\boldsymbol{x}_{\boldsymbol{r}}$ with the smallest value of U among all the *Considered*.
5. Move $\boldsymbol{x}_{\boldsymbol{r}}$ to *Accepted* and update the *AcceptedFront*.
6. Move the *Far* mesh points adjacent to $\boldsymbol{x}_{\boldsymbol{r}}$ into *Considered*.
7. Recompute the value for all the *Considered* $\boldsymbol{x}_{\boldsymbol{i}}$ within the distance $h\frac{F_2}{F_1}$ from $\boldsymbol{x}_{\boldsymbol{r}}$. If the new computed value is less than the previous tentative value for $\boldsymbol{x}_{\boldsymbol{i}}$ then update U_i.
8. If *Considered* is not empty then goto 4).

– **Efficiency:** This results in a "single-pass" method since the maximum number of times each mesh point can be re-evaluated is bounded by the number of mesh points in the $h\frac{F_2}{F_1}$ neighborhood of that point. Thus, this method formally has the computational complexity of $O((\frac{F_2}{F_1})^2 M \log(M))$. Moreover, since the *AcceptedFront* is approximating the level set of the viscosity solution u, as the mesh is refined, the complexity will behave as $O(\frac{F_2}{F_1} M \log(M))$. Here, the "efficiency" refers to the complexity of computing an approximate solution on a fixed grid; the above orders of complexity are proven for the implementation using heap-sort data structures [12,17].

- **Convergence:** In [12,17] we prove the convergence (assuming the continuity of f) of the numerical solution to the viscosity solution of the PDE for a particular update formula K_{ij}, related to the iterative schemes described in [7], [8], and [6]. The asymptotic order of convergence of the method generally depends upon the order of the upwinding update formula $U_i = K(U_j, U_k, \boldsymbol{x_i}, \boldsymbol{x_j}, \boldsymbol{x_k})$.
- In [12] we also use the above method with other the finite-difference upwinding update formulas, obtained as the anisotropic generalizations of the discretizations presented in [11]. The notable advantage of this approach is that it can be easily generalized for the higher-order upwinding finite difference approximations.

3.3 Numerical Results

The first example we consider corresponds to finding geodesic distances on a surface. Consider a pedestrian walking with the unit speed on a surface $z = g(x, y)$. The pedestrian is interested in finding the shortest path on the surface. As shown in [12], the problem can be solved in the $x - y$ plane by considering the dynamics of the pedestrian's shadow. The shadow will move from point A to point B in the plane, as the pedestrian walks from $(A, g(A))$ to $(B, g(B))$ on the manifold. The speed of the shadow in the direction $\boldsymbol{a} \in S_1$ is, therefore,

$$f(x, y, \boldsymbol{a}) = \left(1 + (\nabla g(x, y) \cdot \boldsymbol{a})^2\right)^{-\frac{1}{2}}.$$

Solving the equation 2 in the $x-y$ plane with the boundary condition $u(B) = 0$, we obtain the level sets of the min-time to exit function u. The characteristics of the PDE will correspond to the globally optimal trajectories for the pedestrian's shadow (and, hence, to the projections of the pedestrian's optimal walking paths). Figure 1 illustrates the solution of this PDE for the surface $g(x, y) = 45\, sin(\frac{\pi x}{50}) sin(\frac{\pi y}{50})$ over the square $[0, 100] \times [0, 100]$.

We now make the problem harder by accounting for the fact that the (sustained) speed of walking on the surface is generally dependent on the slope of the surface in that direction. Let θ be the angle between the direction of motion on the surface and the positive direction of z-axis:

$$\theta_{\boldsymbol{a}} = \frac{\pi}{2} - \arctan(\nabla g(x, y) \cdot \boldsymbol{a}).$$

If the pedestrian is moving on the surface with the positive speed $\phi(\theta_{\boldsymbol{a}})$, then the shadow moves in the direction $\boldsymbol{a} \in S_1$ with the speed

$$f(x, y, \boldsymbol{a}) = \phi(\theta_{\boldsymbol{a}}) \left(1 + (\nabla g(x, y) \cdot \boldsymbol{a})^2\right)^{-\frac{1}{2}}.$$

We use two different slope-dependencies $\phi_w(\theta) = \sin^6(\theta) + 0.1$ and $\phi_s(\theta) = 2\sin^{40}(\theta) + 0.1$, generating the speed functions f_w and f_s. (In our simplistic model, we consider them as speeds for *walking* and *skating* on the surface, respectively.) The level sets of the corresponding min-time functions are shown in Figure 2.

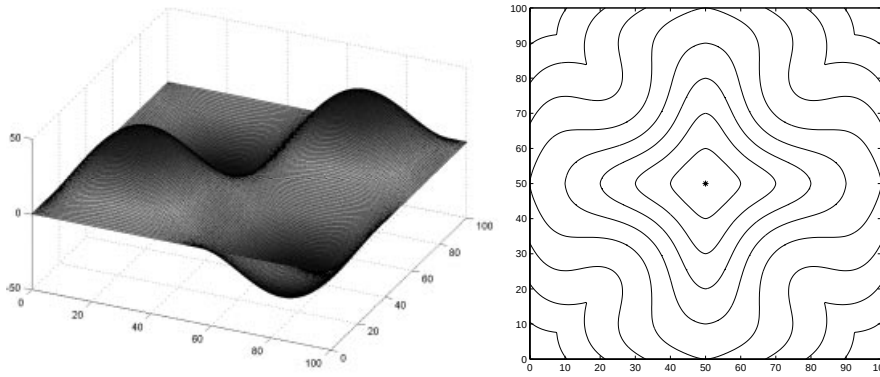

Fig. 1. The test surface and the level sets of the min-time function (for traveling to the origin with unit speed).

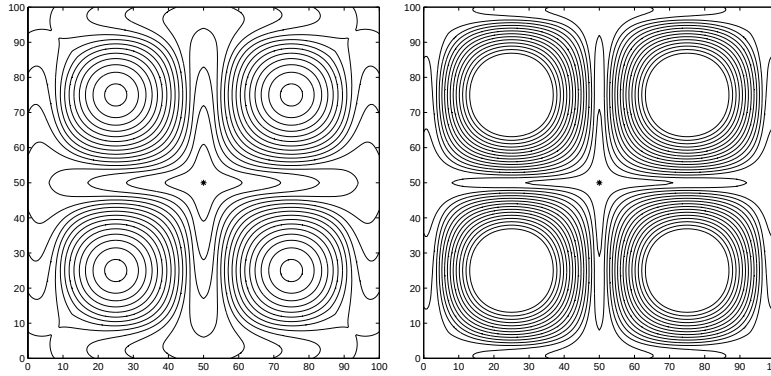

Fig. 2. The min-time function for walking (left) and skating (right) to the origin.

Our third example is the problem of finding an optimal continuous trajectory for traveling on the manifold if the traveler becomes "tired" (i.e., slows down) as time goes by:

$$f_{wt}(\boldsymbol{y}, \boldsymbol{a}, t) = f_w(\boldsymbol{y}, \boldsymbol{a})\psi(t),$$

where ψ is a positive, monotone decreasing function of t. This is a simple example in which the speed function depends on time in the controlled dynamics.

In general, the time-dependence of speed conflicts with the optimality principle: the optimal route from A to B might be irrelevant for computing the optimal route from C to B even if that optimal route passes through the point A. A simple way to deal with the situation is to add the time to the state variables, but the computational cost of such a solution is prohibitive. We use a different approach instead: reversing the direction of information propagation.

If the dynamics were time independent (i.e., $\boldsymbol{y}'(t) = f(\boldsymbol{y}(t), \boldsymbol{a}(t))\boldsymbol{a}(t)$), we would use $u(B) = 0$ as a boundary value for the HJB PDE:

$$\max_{\boldsymbol{a}\in S_1} \{(\nabla u(\boldsymbol{x}) \cdot (-\boldsymbol{a}))f(\boldsymbol{x}, \boldsymbol{a})\} = 1, \boldsymbol{x} \in \Omega. \tag{5}$$

The level sets $u(\boldsymbol{x}) = T$ would include all the points, **from** which one could (optimally) travel to B in time T. In principle, the value function u could be used to compute the optimal route to B **from** any point in the domain.

For the time dependent dynamics (i.e., $\boldsymbol{y}'(t) = f(\boldsymbol{y}(t), \boldsymbol{a}(t), t)\boldsymbol{a}(t)$), we are using the initial condition $u(A) = 0$ and solving the HJB PDE:

$$\max_{\boldsymbol{a}\in S_1} \{(\nabla u(\boldsymbol{x}) \cdot \boldsymbol{a})f(\boldsymbol{x}, \boldsymbol{a}, u(\boldsymbol{x}))\} = 1, \boldsymbol{x} \in \Omega. \tag{6}$$

The level sets of the viscosity solution ($u(\boldsymbol{x}) = T$) include all the points, **to** which one could (optimally) travel from A in time T. In principle, the value function u could be used to compute the optimal route from A **to** any point in the domain[2].

Figure 3 shows the level sets of the viscosity solution $u(x)$ for the speed function $f_{wt}(\boldsymbol{y}, \boldsymbol{a}, t) = f_w(\boldsymbol{y}, \boldsymbol{a})e^{-\lambda t}$.

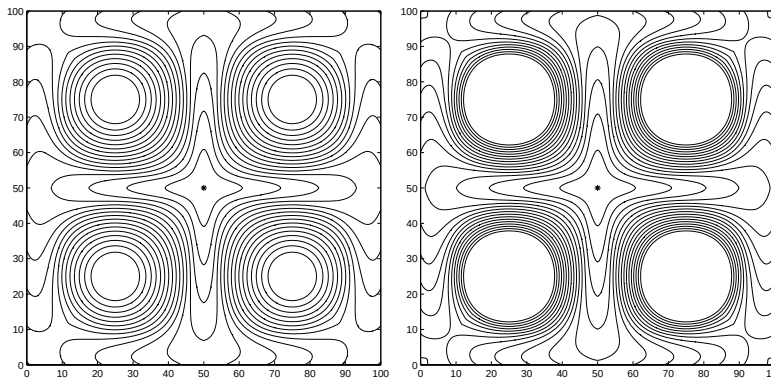

Fig. 3. The min-time function for walking **from** the origin with time-dependent speed: $\lambda = 0.001$ (left) and $\lambda = 0.005$ (right). The level sets are plotted at the same values as in Figure 2.

[2] It is easy to see that, if the speed f is not a continuous function of time, the value function u does not have to be a continuous function of $\boldsymbol{x}$. In such cases, the HJB equation 6 can still be interpreted if $u(\boldsymbol{x})$ is substituted by $\limsup_{\boldsymbol{y}\to\boldsymbol{x}} u(\boldsymbol{y})$.

We also note that a wide class of static convex Hamilton–Jacobi PDEs of the form $\|\nabla u\| F\left(\boldsymbol{x}, \frac{\nabla u}{\|\nabla u\|}, u\right) = 1$ can be interpreted as the HJB equation 6 and, therefore, can be solved by the OUMs; see [13] for details.

4 Ordered Upwind Methods for Hybrid Control Problems

The Fast Marching Method for the continuous isotropic Eikonal equation can be extended to hybrid control, as was done in [3],[4]; this can be applied in cases in which the speed of motion is isotropic (dependent only upon the current state of the system). We now extend the continuous control Ordered Upwind Methods to compute the value function for the hybrid systems with the general (anisotropic) continuous-state dynamics.

4.1 Formulation of Algorithm

Hybrid systems are often modeled as hybrid automata, represented by a directed graph with continuous dynamics at each node. The continuous state of the system evolves according to the chosen continuous control and the differential equations specified for the particular node; when certain conditions are satisfied, the transition to another node may occur, leading to a change in the dynamics of a continuous state of the system.

In order to find an optimal hybrid control numerically, we also need to discretize the continuous state space corresponding to each discrete state of the hybrid system. Let X_i be the discretization of the continuous state space corresponding to the hybrid automata node σ_i. The full computational mesh $X = \bigcup X_i$ will also include the directed links representing transitions between different modes of continuous dynamics. Let $\boldsymbol{x_i} \in X_i$ be a node for which the transition rules are satisfied. Define the sets

$$L_{from}(\boldsymbol{x_i}) = \{\boldsymbol{x_j} \in X_j \mid \text{there is a transition from } \boldsymbol{x_i} \text{ to } \boldsymbol{x_j}\}$$

and

$$L_{to}(\boldsymbol{x_i}) = \{\boldsymbol{x_j} \in X_j \mid \text{there is a transition from } \boldsymbol{x_j} \text{ to } \boldsymbol{x_i}\}$$

(*jump successors* and *jump predecessors* of $\boldsymbol{x_i}$ in the terminology of [4]). These transitions are represented in X by the directed links $(\boldsymbol{x_i}, \boldsymbol{x_j})$ with the associated transition/link costs $C_{ij} \geq 0$.

As before, mesh points are divided into three classes (*Far*, *Considered*, *Accepted*). The *AcceptedFront* is defined as a set of *Accepted* mesh points, which are adjacent to some not-yet-accepted mesh points. Define the set AF of the line segments $\boldsymbol{x_j}\boldsymbol{x_k}$, where $\boldsymbol{x_j}$ and $\boldsymbol{x_k}$ are adjacent mesh points on the *AcceptedFront*, such that there exists a *Considered* mesh point $\boldsymbol{x_i}$ adjacent to both $\boldsymbol{x_j}$ and $\boldsymbol{x_k}$. For each *Considered* mesh point $\boldsymbol{x_i}$ we define the part of AF "relevant to $\boldsymbol{x_i}$":

$$NF(\boldsymbol{x_i}) = \left\{ (\boldsymbol{x_j}, \boldsymbol{x_k}) \in AF \mid \exists \tilde{\boldsymbol{x}} \text{ on } (\boldsymbol{x_j}, \boldsymbol{x_k}) \text{ s.t. } \|\tilde{\boldsymbol{x}} - \boldsymbol{x_i}\| \leq h\frac{F_2}{F_1} \right\}.$$

We also define the set of *Accepted* "neighbors" accessible via discrete transition links $LA_{\text{from}}(\boldsymbol{x_i}) = L_{\text{from}}(\boldsymbol{x_i}) \bigcap Accepted$. We will further assume that some consistent upwinding update formula is available for each "mode" of continuous

dynamics: if the characteristic for $\boldsymbol{x_i}$ lies in the simplex $\boldsymbol{x_i x_j x_k}$ then $U_i = K(U_j, U_k, \boldsymbol{x_i}, \boldsymbol{x_j}, \boldsymbol{x_k})$. For the sake of notational simplicity we will refer to this value as $K_{j,k}$.

1. Start with all mesh points in Far ($U_i = \infty$).
2. Move the boundary mesh points ($\boldsymbol{x_i} \in \partial\Omega$) to *Accepted* ($U_i = q(\boldsymbol{x_i})$).
3. Move all the mesh points $\boldsymbol{x_i}$ adjacent to the boundary (and all the $\boldsymbol{x_i}$ s.t. $LA_{\text{from}}(\boldsymbol{x_i}) \neq \emptyset$) into *Considered* and evaluate the tentative value of $U_i = \min\left\{\min_{(\boldsymbol{x_j},\boldsymbol{x_k})NF(\boldsymbol{x_i})} K_{j,k},\ \min_{\boldsymbol{x_s}\in LA_{\text{from}}(\boldsymbol{x_i})}\{U_s + C_{is}\}\right\}$.
4. Find the mesh point $\boldsymbol{x_r}$ with the smallest value of U among all the *Considered*.
5. Move $\boldsymbol{x_r}$ to *Accepted* and update the *AcceptedFront*.
6. Move the Far mesh points adjacent to $\boldsymbol{x_r}$ into *Considered*.
7. Move the Far mesh points in $L_{\text{to}}(\boldsymbol{x_r})$ into *Considered*.
8. Recompute the value for all the *Considered* $\boldsymbol{x_i}$ such that $\|\boldsymbol{x_r} - \boldsymbol{x_i}\| \leq h\frac{F_2}{F_1}$ **or** $\boldsymbol{x_i} \in L_{\text{to}}(\boldsymbol{x_r})$. If the new computed value is less than the previous tentative value for $\boldsymbol{x_i}$ then update U_i: $U_i = \min\left\{U_i,\ \min_{(\boldsymbol{x_r}\boldsymbol{x_j})\in NF(\boldsymbol{x_i})} K_{r,j},\ U_r + C_{ir}\right\}$.
9. If *Considered* is not empty then goto 4).

The efficiency of the resulting method is $O\left((\frac{F_2}{F_1} + d)MlogM\right)$, where M is the total number of mesh points in X and d is the maximum number of discrete transitions/links from a single mesh point.

4.2 Examples

As the first example, we consider the problem of finding an optimal trajectory on a surface; the continuous problem is augmented by assuming that there are discrete transitions between a finite number of pre-defined points on the mesh. A realistic analogue is the time-optimal path planning for traveling on a varied landscape if there are shuttle buses waiting for travelers at pre-defined locations $\boldsymbol{x_i}$ and carrying them (for a fee - or time penalty - $C_{ij} \geq 0$) to the pre-defined locations $\boldsymbol{x_j}$. Thus, even if the traveler is trying to get from A to B, it might save him time to diverge from the geodesic path to the point C if the shuttle going from C to D is fast and if D is not far from the final destination.

This example is not the most general possible hybrid system, since the directed discrete links (i.e., the transitions) change only the position in the continuous state space but not the underlying dynamics. Thus, we are able to compute the value function using a single discretization X of the continuous state space and superimposing on it the "shuttle lines" (Figure 4). This example can also be interpreted for the buses (unlike shuttles, buses are not waiting for the traveler); if C_{ij} is the average time for traveling on bus from $\boldsymbol{x_i}$ to $\boldsymbol{x_j}$, including the average waiting time at the bus stop. In this case, the computed value function is interpreted as the expected value of the time taken by the optimal route.

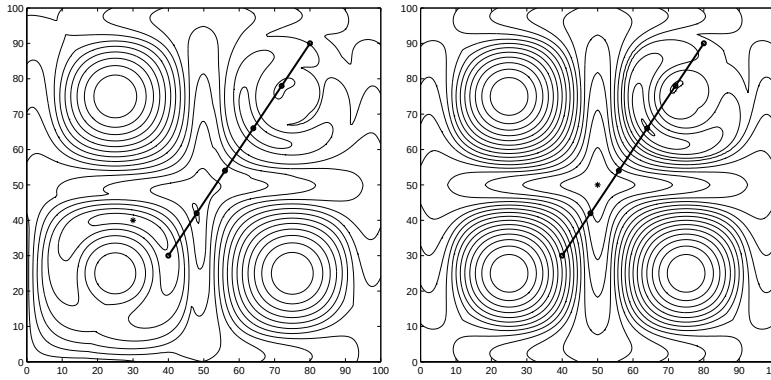

Fig. 4. The min-time function for traveling **to** the point B in the presence of shuttles (left), and the min-time function for traveling **from** the point A in the presence of the buses running on schedule (right).

Our second test problem involves the buses going exactly on schedule[3]. This is an example of the discrete dynamics depending on time: the time-penalty associated with the bus-route transition is assumed to be infinite at all times other than the scheduled departures. As in the fully continuous case, this endangers the optimality principle: the optimal route from A to B might be irrelevant for computing the optimal route from C to B even if that optimal route passes through the point A. (The optimal route from A to B could include catching a bus, which might be gone by the time we get from C to A.)

As before, we deal with this difficulty by setting the boundary condition to be zero at the "source" A rather than the "target" B. As shown in Figure 4, the level sets of the viscosity solution ($u(\boldsymbol{x}) = T$) include all the points, **to** which one could (optimally) travel from A in time T.

For a more complete and realistic example, we now demonstrate the application of OUMs to "real" hybrid control problems - i.e., the problems, in which taking the discrete transition forces a change in the dynamics (rather than just a position of) the continuous state. In most cases, such problems require using multiple discretizations X_i of the continuous state space, corresponding to multiple nodes σ_i of the hybrid automata. Consider a person walking on a varied landscape, but also carrying a pair of inline roller skates. This *walker* has an option of paying the time-penalty (spending time to put on the skates) to become a *skater* and to modify his speed function as a result. Correspondingly, the *skater* can pay a different time-penalty to take of the skates and return to *walking*. We compute the value function u for this problem using two copies (X_s and X_w) of the discretized continuous state space[4]. Figure 5 shows the level sets of $u(\boldsymbol{x})$

[3] Depending on where you live, this may be extremely unrealistic.

[4] Of course, if the time-penalties are zero, the optimal strategy will correspond to the value function obtained by using the speed $f = \max\{f_w, f_s\}$. In this simple case,

computed under the assumption that putting on the skates requires 10 seconds and taking them off - only 5.

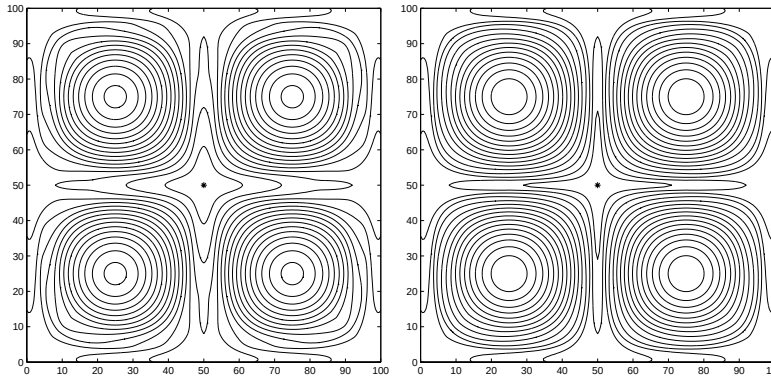

Fig. 5. The min-time function for traveling to the origin using the hybrid (walking/skating) dynamics: for those, who start out walking (left), or skating (right). The level sets are plotted at the same values as in Figure 2.

5 Conclusions

We have demonstrated that single-pass Ordered Upwind Methods can be built for a variety of anisotropic hybrid control problems. The obvious generalizations of the considered optimal hybrid trajectory problems (simultaneously taking into account multiple modes of continuous dynamics, time dependence of the continuous and discrete controls, etc) can be treated similarly. The continuous state constraints can be handled by representing only the constraint set in the meshes X_i; the numerical evidence confirms the convergence to the viscosity solution even for piecewise continuous anisotropic speed functions [12].

We are currently investigating applicability of OUMs to other classes of PDEs, and to other application domains. Work in progress includes devising Ordered Upwind Methods for differential games and extending the capabilities to more complex systems [13].

References

1. Bellman, R.: Introduction to the Mathematical Theory of Control Processes. (1967) Academic Press, New York.

the calculation can be performed using a single discretization X of the continuous state space.

On the other hand, if the speeds f_w and f_s were fully isotropic, this problem would reduce to the second example considered in [4].

2. Bardi, M. & Falcone, M.: An Approximation Scheme for the Minimum Time Function. (1990) SIAM J. Control Optim., 28, 950-965.
3. Branicky, M.S. & Hebbar, R.: Fast marching for hybrid control. (1999) Proceedings, IEEE Intl. Symp. Computer Aided Control System Design, Kohala-Kona, HI, August 22-27.
4. Branicky, M.S., Hebbar, R., Zhang G.: A fast marching algorithm for hybrid systems. (1999) Proceedings, IEEE Conf. on Decision and Control, Phoenix, AZ, December 7-10.
5. Dijkstra, E.W.: A Note on Two Problems in Connection with Graphs. (1959) Numerische Mathematik, 1, 269-271.
6. Gonzales, R. & Rofman, E.: On Deterministic Control Problems: an Approximate Procedure for the Optimal Cost, I, the Stationary Problem. (1985) SIAM J. Control Optim., 23, 2, 242-266.
7. Kushner, H.J.: Probability Methods for Approximations in Stochastic Control and for Elliptic Equations. (1977) Springer-Verlag, New York.
8. Kushner, H.J. & Dupuis, P.G.: *Numerical Methods for Stochastic Control Problems in Continuous Time. (1992) Academic Press, New York.*
9. *Sethian, J.A.: A Fast Marching Level Set Method for Monotonically Advancing Fronts. (1996) Proc. Nat. Acad. Sci., 93, 4, 1591-1595.*
10. *Sethian, J.A.: Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences. (1999) Cambridge University Press).*
11. *Sethian, J.A. & Vladimirsky, A.: Fast Methods for the Eikonal and Related Hamilton–Jacobi Equations on Unstructured Meshes. (2000) Related Hamilton–Jacobi Equations on Unstructured Meshes*, Proc. Nat. Acad. Sci., 97, 11, 5699-5703.
12. Sethian, J.A. & Vladimirsky, A.: Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms. (2001) Center for Pure and Applied Mathematics Technical Report PAM-792 (University of California, Berkeley).
13. Sethian, J.A. & Vladimirsky, A.: Ordered Upwind Methods for a Variety of Control Problems. Generalizations and Optimizations, in progress. (2001).
14. Soner, M.H.: Optimal control problems with state-space constraints. (1986) SIAM J. Control Optim., 24, 552-562 and 1110-1122.
15. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. (1994) Proceedings, IEEE 33rd Conference on Decision and Control, Lake Buena Vista, Florida, December (1994), 1368-1373.
16. Tsitsiklis, J.N.: Efficient algorithms for globally optimal trajectories. (1995) IEEE Tran. Automatic Control, 40, 1528-1538.
17. Vladimirsky, A. : Fast Methods for Static Hamilton-Jacobi Partial Differential Equations. (2001) Ph.D. Dissertation, Dept. of Mathematics, Univ. of California, Berkeley.