AN EFFICIENT METHOD FOR MULTIOBJECTIVE OPTIMAL CONTROL AND OPTIMAL CONTROL SUBJECT TO INTEGRAL CONSTRAINTS*

Ajeet Kumar

Department of Theoretical & Applied Mechanics, Cornell University, Ithaca, NY 14853, USA
Email: ak428@cornell.edu

Alexander Vladimirsky

Department of Mathematics, Cornell University, Ithaca, NY 14853, USA
Email: vlad@math.cornell.edu

Abstract

We introduce a new and efficient numerical method for multicriterion optimal control and single criterion optimal control under integral constraints. The approach is based on extending the state space to include information on a "budget" remaining to satisfy each constraint; the augmented Hamilton-Jacobi-Bellman PDE is then solved numerically. The efficiency of our approach hinges on the causality in that PDE, i.e., the monotonicity of characteristic curves in one of the newly added dimensions. A semi-Lagrangian "marching" method is used to approximate the discontinuous viscosity solution efficiently. We compare this to a recently introduced "weighted sum" based algorithm for the same problem [25]. We illustrate our method using examples from flight path planning and robotic navigation in the presence of friendly and adversarial observers.

Mathematics subject classification: 90C29, 49L20, 49L25, 58E17, 65N22, 35B37, 65K05.
Key words: Optimal control, Multiobjective optimization, Pareto front, Vector dynamic programming, Hamilton-Jacobi equation, Discontinuous viscosity solution, Semi-Lagrangian discretization.

1. Introduction

In the continuous setting, deterministic optimal control problems are often studied from the point of view of dynamic programming; see, e.g., [1,8]. A choice of the particular control $\boldsymbol{a}(t)$ determines the trajectory $\boldsymbol{y}(t)$ in the space of system-states $\Omega \subset \boldsymbol{R}^n$. A running cost K is integrated along that trajectory and the terminal cost q is added, yielding the total cost associated with this control. A *value function u*, describing the minimum cost to pay starting from each system state, is shown to be the unique viscosity solution of the corresponding Hamilton-Jacobi PDE. Once the value function has been computed, it can be used to approximate optimal feedback control. We provide an overview of this classic approach in section 2.

However, in realistic applications practitioners usually need to optimize by many different criteria simultaneously. For example, given a vehicle starting at $\boldsymbol{x} \in \Omega$ and trying to "optimally" reach some target $\mathcal{T}$, the above framework allows to find the most fuel efficient trajectories and the fastest trajectories, but these generally will not be the same. A natural first step is to compute the total time taken along the most fuel-efficient trajectory and the total amount of fuel needed to follow the fastest trajectory. Computational efficiency requires a method for

computing this simultaneously for all starting states $\boldsymbol{x}$. A PDE-based approach for this task is described in section 3.1.

This, however, does not yield answers to two more practical questions: what is the fastest trajectory from $\boldsymbol{x}$ to $\mathcal{T}$ without using more than the specified amount of fuel? Alternatively, what is the most fuel-efficient trajectory from $\boldsymbol{x}$, provided the vehicle has to reach $\mathcal{T}$ by the specified time? We will refer to such trajectories as *constrained-optimal*.

One approach to this more difficult problem is the *Pareto optimization*: finding a set of trajectories, which are optimal in the sense that no improvement in fuel-efficiency is possible without spending more time (or vice versa). This defines a *Pareto front* – a curve in a time-fuel plane, where each point corresponds to time & fuel needed along some Pareto-optimal trajectory. This approach is generally computationally expensive, especially if a Pareto front has to be found for each starting state $\boldsymbol{x}$ separately. The current state of the art for this problem has been developed by Mitchell and Sastry [25] and described in section 3.2. Their method is based on the usual *weighted sum* approach to multiobjective optimization [24]. A new running cost K is defined as a weighted average of several competing running costs K_i's, and the corresponding Hamilton-Jacobi PDE is then solved to obtain one point on the Pareto front. The coefficients in the weighted sum are then varied and the process is repeated until a solution satisfying all constraints is finally found. Aside from the computational cost, the obvious disadvantage of this approach is that only a convex part of the Pareto front can be obtained by weighted sum methods [13], which may result in selecting suboptimal trajectories. In addition, recovering the entire Pareto front for each $\boldsymbol{x} \in \Omega$ is excessive and unnecessary when the real goal is to solve the problem for a fixed list of constraints (e.g., maximum fuel or maximum time available).

Our own approach (described in section 3.3) remedies these problems by systematically constructing the exact portion of Pareto front relevant to the above constraints *for all $\boldsymbol{x} \in \Omega$ simultaneously*. Given $\Omega \subset \boldsymbol{R}^n$ and r additional integral constraints, we accomplish this by solving a single augmented partial differential equation on a $(r + n)$-dimensional domain. Our method has two key advantages. First, it does not rely on any assumptions about the convexity of Pareto front. Secondly, the PDE we derive has a special structure, allowing for a very efficient marching method. Our approach can be viewed as a generalization of the classic equivalency of Bolza and Mayer problems [8]. The idea of accommodating integral constraints by extending the state space is not new. It was previously used by Isaacs to derive the properties of constrained-optimal strategies for differential games [21]. More recently, it was also used in infinite-horizon control problems by Soravia [37] and Motta & Rampazzo [26] to prove the uniqueness of the (lower semi-continuous) viscosity solution to the augmented PDE. However, the above works explored the theoretical issues only and, to the best of our knowledge, ours is the first practical numerical method based on this approach. In addition, we also show the relationship between optimality under constraints and Pareto optimality for feasible trajectories.

The computational efficiency of our method is deeply related to the general difference in numerical methods for time-dependent and static first-order equations. In optimal control problems, time-dependent HJB PDEs result from *finite-horizon* problems or problems with time-dependent dynamics and running cost. Static HJB PDEs usually result from *exit-time* or *infinite-horizon* problems with time-independent (though perhaps time-discounted) dynamics and running cost. In the time-dependent case, efficient numerical methods are typically based on time-marching. In the static case, a naive approach involves iterative solving of a system of discretized equations. Several popular approaches were developed precisely to avoid these

iterations either by space-marching (e.g., [29,34,40]), or by embedding into a higher-dimensional time-dependent problem (via Level Set Methods, e.g., [27]), or by treating one of the spatial directions as if it were time (resulting in a "paraxial" approximation; see, e.g., [28]). For reader's convenience, we provide a brief overview of these approaches in sections 2.3 and 2.4. Our key observation is that the augmented "static" PDE has *explicit causality*, allowing simple marching (similar to time-marching) in the secondary cost direction.

Our semi-Lagrangian method is described in section 3.4. Since the augmented PDE is solved on a higher-dimensional domain, any restriction of that domain leads to substantial computational savings. In section 3.5 we explain how this can be accomplished by solving static PDEs in Ω for each individual cost. This additional step also yields improved boundary conditions for the primary value function in $\mathbf{R}^{n+r}$.

In section 4 we illustrate our method using examples from robotic navigation (finding shortest/quickest paths, while avoiding (or seeking) exposure to stationary observers) and a testproblem introduced in [25] (planning a flight-path for an airplane to minimize the risk of encountering a storm while satisfying constraints on fuel consumption). Finally, in section 5 we discuss the limitations of our approach and list several directions for future research.

2. Single-criterion Dynamic Programming

2.1. Exit-time optimal control

To begin, we consider a general deterministic exit-time optimal control problem. This is a classic problem and our discussion follows the description in [1]. Suppose $\Omega \subset \mathbf{R}^n$ is an open bounded set of all possible "non-terminal" states of the system, while $\partial\Omega$ is the set of all terminal states. For every starting state $\boldsymbol{x} \in \Omega$, the goal is to find the cheapest way to leave Ω.

Suppose $A \in \mathbf{R}^m$ is a compact set of control values, and the set of admissible controls $\mathcal{A}$ consists of all measurable functions $\boldsymbol{a} : \mathbf{R} \mapsto A$. The dynamics of the system is defined by

$$\begin{aligned}
\boldsymbol{y}'(t) &= \boldsymbol{f}(\boldsymbol{y}(t), \boldsymbol{a}(t)), \\
\boldsymbol{y}(0) &= \boldsymbol{x} \in \Omega,
\end{aligned} \tag{2.1}$$

where $\boldsymbol{y}(t)$ is the system state at the time t, $\boldsymbol{x}$ is the initial system state, and $\boldsymbol{f} : \bar{\Omega} \times A \mapsto \mathbf{R}^n$ is the velocity. The exit time associated with this control is

$$T_{\boldsymbol{x},\boldsymbol{a}} = \min\{t \in \mathbf{R}_{+,0} | \boldsymbol{y}(t) \in \partial\Omega\}. \tag{2.2}$$

The problem description also includes the running cost $K : \bar{\Omega} \times A \mapsto \mathbf{R}$ and the terminal (non-negative, possibly infinite) cost $q : \partial\Omega \mapsto (\mathbf{R}_{+,0} \cup \{+\infty\})$. This allows to specify the total cost of using the control $\boldsymbol{a}(\cdot)$ starting from $\boldsymbol{x}$:

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = \int_0^{T_{\boldsymbol{x},\boldsymbol{a}}} K(\boldsymbol{y}(t), \boldsymbol{a}(t))\, dt \, + \, q(\boldsymbol{y}(T_{\boldsymbol{x},\boldsymbol{a}})).$$

We will make the following assumptions throughout the rest of this paper:

(A0) Function q is lower semi-continuous and $\min_{\partial\Omega} q < +\infty$;

(A1) Functions $\boldsymbol{f}$ and K are Lipschitz-continuous;

(A2) There exist constants k_1, k_2 such that $0 < k_1 \le K(\boldsymbol{x}, \boldsymbol{a}) \le k_2$ for $\forall \boldsymbol{x} \in \bar{\Omega}, \boldsymbol{a} \in A$;

(A3) For every $\boldsymbol{x} \in \Omega$, the *scaled vectogram*

$$V(\boldsymbol{x}) = \{\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})/K(\boldsymbol{x}, \boldsymbol{a}) \mid \boldsymbol{a} \in A\}$$

is a strictly convex set, containing the origin in its interior.

The key idea of *dynamic programming* is to introduce the value function $u(\boldsymbol{x})$, describing the minimum cost needed to exit Ω starting from $\boldsymbol{x}$:

$$u(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot) \in \mathcal{A}} \mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)). \tag{2.3}$$

Bellman's optimality principle [6] shows that, for every sufficiently small $\tau > 0$,

$$u(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot)} \left\{ \int_0^\tau K(\boldsymbol{y}(t), \boldsymbol{a}(t)) \, dt + u(\boldsymbol{y}(\tau)) \right\}, \tag{2.4}$$

where $\boldsymbol{y}(\cdot)$ is a trajectory corresponding to a chosen control $\boldsymbol{a}(\cdot)$. If $u(\boldsymbol{x})$ is smooth, a Taylor expansion of the above formula can be used to formally show that u is the solution of a static Hamilton-Jacobi-Bellman PDE:

$$\min_{\boldsymbol{a} \in A} \{K(\boldsymbol{x}, \boldsymbol{a}) + \nabla u(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})\} = 0, \qquad \text{for } \boldsymbol{x} \in \Omega;$$

$$u(\boldsymbol{x}) = q(\boldsymbol{x}), \qquad \text{for } \boldsymbol{x} \in \partial\Omega. \tag{2.5}$$

Unfortunately, a smooth solution to Eqn. 2.5 might not exist even for smooth $\boldsymbol{f}, K$, q, and $\partial\Omega$. Generally, this equation has infinitely many weak Lipschitz-continuous solutions, but the unique *viscosity solution* can be defined using additional conditions on smooth test functions [11, 12]. It is a classic result that the viscosity solution of this PDE coincides with the value function of the above control problem.

Under the above assumptions the value function $u(\boldsymbol{x})$ is locally Lipschitz-continuous on Ω, an optimal control $\boldsymbol{a}(\cdot)$ actually exists for every $\boldsymbol{x} \in \Omega$ (i.e., **min** can be used instead of **inf** in formula 2.3), and the minimizing control value $\boldsymbol{a}$ (in equation 2.5) is unique wherever $\nabla u(\boldsymbol{x})$ is defined [1]. The characteristic curves of this PDE are the optimal trajectories of the control problem. The points where ∇u is undefined are precisely those, where multiple characteristics intersect (or, alternatively, the points for which multiple optimal trajectories are available). We will let Γ be a set of all such points. By Rademacher's theorem, Γ has a measure zero in $\bar{\Omega}$.

We note that the assumptions (A2-A3) can be relaxed at the cost of additional technical details. For example, if $V(\boldsymbol{x})$ is not convex, the existence of an optimal control is not guaranteed even though the value function can still be recovered from the PDE (2.5) and there exist suboptimal controls whose cost is arbitrarily close to $u(\boldsymbol{x})$. On the other hand, if $V(\boldsymbol{x})$ does not contain the origin in its interior, then the *reachable set*

$$\mathcal{R} = \left\{ \boldsymbol{x} \in \Omega \mid \text{ there exists a control leading from } \boldsymbol{x} \text{ to } \partial\Omega \text{ in finite time} \right\}$$

need not be the entire Ω. In that case, $\partial\mathcal{R}$ is a *free boundary*. We refer readers to [1] for further details.

The above framework is also flexible enough to describe the task of optimally reaching some compact target $\mathcal{T} \subset \Omega$ without leaving Ω. To do that, we can simply pose the problem on a new domain $\Omega^{new} = \Omega \backslash \mathcal{T}$, defining the new exit cost q^{new} to be infinite on $\partial\Omega$ and finite on

$\partial\mathcal{T}$. The assumptions (A0-A3) guarantee the continuity of the value function on Ω even in the presence of state-constraints; $k_1 > 0$ and the fact that the origin is in the interior of $V(\boldsymbol{x})$ yield both Soner's "inward pointing" condition along the boundary of the constraint set (as in [36]) and the local controllability (as in [2], for example).

Remark 2.1. The continuity of u on $\bar{\Omega}$ is a more subtle issue requiring additional *compatibility conditions* on q even if that function is continuous; otherwise, the boundary conditions are satisfied by the value function "in viscosity sense" only [1]. However, due to our very strong controllability assumption (A3), the local Lipschitz-continuity of u in the interior is easy to show even if q is discontinuous, as allowed by (A0). Without assumption (A3) or its equivalent, discontinuous boundary data typically leads to discontinuities in the value function in the interior as well. Such is the case for the augmented PDE defined in section 3.3.

Before continuing with the general case we consider two particularly useful subclasses of problems.

Geometric dynamics. Suppose $A = \{\boldsymbol{a} \in \boldsymbol{R}^n \mid |\boldsymbol{a}| \le 1\}$ and, for all $\boldsymbol{x} \in \Omega, \boldsymbol{a} \in A\backslash\{0\}$,

$$K(\boldsymbol{x}, \boldsymbol{a}) \ge |\boldsymbol{a}|K(\boldsymbol{x}, \boldsymbol{a}/|\boldsymbol{a}|), \quad \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x}, \boldsymbol{a}/|\boldsymbol{a}|)\boldsymbol{a}, \quad \text{and } \boldsymbol{f}(\boldsymbol{x}, 0) = 0.$$

Then it is easy to show that the cost of any trajectory is reduced by traversing it as quickly as possible; i.e., we can redefine $A = S_{n-1} = \{\boldsymbol{a} \in \boldsymbol{R}^n \mid |\boldsymbol{a}| = 1\}$ without affecting the value function. In that case, the control value $\boldsymbol{a}$ is simply our choice for the direction of motion and f is the speed of motion in that direction. The equation (2.5) now becomes

$$\min_{\boldsymbol{a}\in S_{n-1}} \{K(\boldsymbol{x}, \boldsymbol{a}) + (\nabla u(\boldsymbol{x}) \cdot \boldsymbol{a})\,f(\boldsymbol{x}, \boldsymbol{a})\} = 0. \tag{2.6}$$

A further simplification is possible if the speed and cost are isotropic, i.e., $f(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x})$ and $K(\boldsymbol{x}, \boldsymbol{a}) = K(\boldsymbol{x})$. In this case, the minimizing control value is $\boldsymbol{a} = -\nabla u(\boldsymbol{x})/|\nabla u(\boldsymbol{x})|$ and (2.5) reduces to an *Eikonal PDE*:

$$|\nabla u(\boldsymbol{x})|\,f(\boldsymbol{x}) = K(\boldsymbol{x}). \tag{2.7}$$

The characteristics of the Eikonal equation coincide with the gradient lines of its viscosity solution. That special property can be used to construct particularly efficient numerical methods for this PDE; see section 2.4.

Time-optimal control. If $K(\boldsymbol{x}, \boldsymbol{a}) = 1$ for all $\boldsymbol{x}$ and $\boldsymbol{a}$, then $\mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = T_{\boldsymbol{x},\boldsymbol{a}} + q(\boldsymbol{y}(T_{\boldsymbol{x},\boldsymbol{a}}))$. Interpreting q as an *exit time penalty*, we can restate this as a problem of finding a *time-optimal control* starting from $\boldsymbol{x}$. The PDE (2.5) becomes

$$\max_{\boldsymbol{a}\in A}\{-\nabla u(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})\} = 1. \tag{2.8}$$

We note that the value function for *every* exit-time optimal control problem satisfying assumptions (A0-A3) can be reduced to this time-optimal control case by setting $K^{new} = 1$ and $\boldsymbol{f}^{new}(\boldsymbol{x}, \boldsymbol{a}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})/K(\boldsymbol{x}, \boldsymbol{a})$; a proof of this for the case of geometric dynamics can be found in [41].

A combination of these two special cases is attained when $K = 1$ and $f = 1$, leading to a PDE $|\nabla u(\boldsymbol{x})| = 1$. If the boundary condition is $q = 0$, then $u(\boldsymbol{x})$ is simply the distance from $\boldsymbol{x}$ to $\partial\Omega$.

2.2. Semi-Lagrangian discretizations

Suppose X is a grid or a mesh on the domain $\bar{\Omega}$ and for simplicity assume that $\partial\Omega$ is well-discretized by ∂X. We will use $M = |X|$ to denote the total number of meshpoints in X.

A natural first-order accurate semi-Lagrangian discretization of equation (2.5) is obtained by assuming that the control value is held fixed for some small time τ. If $U(\boldsymbol{x})$ is the approximation of the value function at the mesh point $\boldsymbol{x} \in X$, then the optimality principle yields the following system:

$$U(\boldsymbol{x}) = \min_{\boldsymbol{a}\in A} \left\{ \tau K(\boldsymbol{x}, \boldsymbol{a}) + U\left(\boldsymbol{x} + \tau \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})\right) \right\}, \qquad \forall \boldsymbol{x} \in X \backslash \partial\Omega;$$

$$U(\boldsymbol{x}) = q(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \partial X. \tag{2.9}$$

Since $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} = \boldsymbol{x} + \tau \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})$ usually is not a gridpoint, a first-order accurate reconstruction is needed to approximate $U(\tilde{\boldsymbol{x}})$ based on values of U at nearby meshpoints[1] . Discretizations of this type were introduced by Falcone for time-discounted infinite-horizon problems, which can be related to the above exit-time problem using the Kruzhkov transform [16, 17].

In case of geometric dynamics, another natural discretization is based on the assumption that the direction of motion is held constant until reaching a boundary of a simplex. For notational simplicity, suppose that $n = 2$ and $S(\boldsymbol{x})$ is the set of all triangles in X with a vertex at $\boldsymbol{x}$. For every $s \in S(x)$, denote its other vertices as $\boldsymbol{x}_{s1}$ and $\boldsymbol{x}_{s2}$. Suppose $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} = \boldsymbol{x} + \tau_{\boldsymbol{a}} f(\boldsymbol{x}, \boldsymbol{a})\boldsymbol{a}$ lies on the segment $\boldsymbol{x}_{s1}\boldsymbol{x}_{s2}$. Let

$$\Xi = \left\{ \xi = (\xi_1, \xi_2) \mid \xi_1 + \xi_2 = 1 \ \ \text{and} \ \ \xi_1, \xi_2 \geq 0 \right\}.$$

Then $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} = \xi_1 \boldsymbol{x}_{s1} + \xi_2 \boldsymbol{x}_{s2}$ for some $\xi \in \Xi$ and $U(\tilde{\boldsymbol{x}}_{\boldsymbol{a}}) = \xi_1 U(\boldsymbol{x}_{s1}) + \xi_2 U(\boldsymbol{x}_{s2})$. Alternatively, given $\tilde{\boldsymbol{x}}_{\xi} = \xi_1 \boldsymbol{x}_{s1} + \xi_2 \boldsymbol{x}_{s2}$, we can define

$$\boldsymbol{a}_\xi = (\tilde{\boldsymbol{x}}_\xi - \boldsymbol{x})/|\tilde{\boldsymbol{x}}_\xi - \boldsymbol{x}| \ \ \text{and} \ \ \tau_\xi = |\tilde{\boldsymbol{x}}_\xi - \boldsymbol{x}|/f(\boldsymbol{x}, \boldsymbol{a}_\xi).$$

This yields the following system of discretized equations:

$$U(\boldsymbol{x}) = \min_{s\in S(\boldsymbol{x})} \min_{\xi\in\Xi} \left\{ \tau_\xi K(\boldsymbol{x}, \boldsymbol{a}_\xi) + (\xi_1 U(\boldsymbol{x}_{s1}) + \xi_2 U(\boldsymbol{x}_{s2})) \right\}, \qquad \forall \boldsymbol{x} \in X \backslash \partial\Omega;$$

$$U(\boldsymbol{x}) = q(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \partial X. \tag{2.10}$$

Discretizations of this type were used by Gonzales and Rofman in [18]. Both discretizations (2.9) and (2.10) were also earlier derived by Kushner and Dupuis approximating continuous optimal control by controlled Markov processes [23]. In the appendix of [35] we demonstrated that (2.10) is also equivalent to a first-order upwind finite difference approximation on the same mesh X.

2.3. Causality, dimensionality & computational efficiency

We note that both of the above discretizations result in a system of M non-linear coupled equations. Finding a numerical solution to that system efficiently is an important practical problem in itself.

---

[1] If $\boldsymbol{x}$ is close to $\partial\Omega$, it is possible that $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} \notin \bar{\Omega}$. This can be handled by either enlarging X to cover some neighborhood of Ω (see, e.g., [4]) or by decreasing τ in such cases to make $\tilde{\boldsymbol{x}}_{\boldsymbol{a}} \in \partial\Omega$. The latter strategy is also adopted in our implementation of the method in section 3.4.

Suppose all meshpoints in X are ordered $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ and $U = (U_1, \ldots, U_M)$ is a vector of corresponding approximate values. The above discretized equations can be formally written as $U = \mathcal{F}(U)$ and one simple approach is to recover U by fixed-point iterations, i.e., $U^{k+1} = \mathcal{F}(U^k)$, where U^0 is an appropriate initial guess for U. This procedure is generally quite inefficient since each iteration has a $\mathcal{O}(M)$ computational cost and a number of iterations needed is at least $\mathcal{O}(M^{1/n})$.

A Gauss-Seidel relaxation of the above is a typical practical modification, where the entries of U^{k+1} are computed sequentially and the new values are used as soon as they become available: $U_i^{k+1} = \mathcal{F}_i(U_1^{k+1}, \ldots, U_{i-1}^{k+1}, U_i^k, \ldots, U_M^k)$. The number of iterations required to converge will now heavily depend on the PDE, the particular discretization and the ordering of the meshpoints. We will say that a discretization is *causal* if there exists an ordering of meshpoints such that the convergence is attained after a *single* Gauss-Seidel iteration. For example, if the dynamics of the control problem is such that one of the state components (say, y_1) is monotone increasing along any feasible trajectory $\boldsymbol{y}(t)$, then ordering all the meshpoints in ascending order by x_1 would guarantee that only one Gauss-Seidel iteration is needed. Such ordering is analogous to a simple time-marching (from the past into the future) used with discretizations of time-dependent PDEs (e.g., in recovering the value function for fixed-horizon optimal control problems). If a causal ordering is explicitly known a priori (as in the above example), we refer to such discretizations as *explicitly causal*.

Explicit causality is a desirable property since it results in computational efficiency. Suppose $u(\boldsymbol{x})$ is a viscosity solution of some boundary value problem. Historically, two approaches have sought to capitalize on explicit causality in solving more general static PDEs. First, it is often possible to formulate a different time-dependent PDE on the same domain Ω so that its viscosity solution ϕ is related to u as follows:

$$u(\boldsymbol{x}) = t \iff \phi(\boldsymbol{x}, t) = 0.$$

The PDE for ϕ is then solved by explicit time-marching; moreover, since only the zero level set of ϕ is relevant for approximating u, the Level Set Methods are applicable e.g., see [27,30]. Aside from increasing the dimensionality of the computational domain, this approach is also subject to additional CFL-type stability conditions, which often impact the efficiency substantially. Alternatively, in some applications (e.g., seismic imaging) a "paraxial" approximation results from assuming that all *optimal* trajectories must be monotone in one of the state components (say, y_1) even if the same is not true for all *feasible* trajectories. This leads to a time-like marching in x_1 direction (essentially solving a time-dependent PDE on an $(n-1)$-dimensional domain); see, e.g., [28]. This method is certainly computationally efficient, but if the assumption on monotonicity of optimal trajectories is incorrect, it does not converge to the solution of the original PDE.

2.4. Efficient methods for non-explicitly causal problems

Finding the right ordering without increasing the dimensionality has been the subject of much work in the last fifteen years. This task can be challenging for discretizations which are causal, but not explicitly causal.

For the isotropic control problems and the Eikonal PDE (2.7), an equivalent of discretization (2.10) on a Cartesian grid is *monotone-causal* in the sense that U_i cannot depend on U_j if $U_j > U_i$. This makes it possible to find the correct ordering of gridpoints (ascending in U)

at run-time, effectively de-coupling the system of discretized equations. This idea, the basis of Dijkstra's classic method for shortest paths on graphs [14], yields the computational complexity of $\mathcal{O}(M \log M)$. Tsitsiklis has introduced the first Dijkstra-like algorithm for semi-Lagrangian discretization on a Cartesian grid in [40]. A Dijkstra-like Fast Marching Method was introduced by Sethian for Eulerian upwind finite-difference discretizations of isotropic front propagation problems [29]. The method was later extended by Sethian and collaborators to higher-order accurate discretizations on grids and unstructured meshes, in $\boldsymbol{R}^n$ and on manifolds, and to related non-Eikonal PDEs; see [30], [33], and references therein. For anisotropic control problems, the discretization (2.10) generally is not causal, but the computational stencil can be expanded dynamically to regain the causality. This is the basis of Ordered Upwind Methods [34, 35, 41], whose computational complexity is $\mathcal{O}(\Upsilon M \log M)$, where Υ measures the amount of anisotropy present in the problem.

The idea of alternating the directions of Gauss-Seidel sweeps to "guess" the correct mesh point ordering was employed by Boue and Dupuis to speed up the convergence in [7]. For Eulerian discretizations of HJB PDEs, the same technique was later used as a basis of Fast Sweeping Methods by Tsai, Cheng, Osher, and Zhao [39, 43]. Even though a finite number of Gauss-Seidel sweeps is required in the Eikonal case, resulting in a computational cost of $\mathcal{O}(M)$, the actual number of sweeps is proportional to the maximum number of times an optimal trajectory may be changing direction from quadrant to quadrant. A computational comparison of fast marching and fast sweeping approaches to Eikonal PDEs can be found in [19, 20].

3. Multi-criterion Optimal Control

Unlike the classical case considered in section 2, in realistic applications there is often more than one criterion for optimality. For a system controlled in $\Omega \subset \boldsymbol{R}^n$, there may be a number of different running costs $K_0, \ldots, K_r$ and a number of different terminal costs $q_0, \ldots, q_r$ (all of them satisfying assumptions (A0-A3)) resulting in $(r + 1)$ different definitions $\mathcal{J}_0, \ldots, \mathcal{J}_r$ of the total cost for a particular control. A common practical problem is to find a control $\boldsymbol{a}^*(\cdot)$ minimizing $\mathcal{J}_0(\boldsymbol{x}, \boldsymbol{a}^*(\cdot))$ but subject to constraints $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}^*(\cdot)) \leq B_i$ for all $i = 1, \ldots, r$.

We will refer to a control minimizing $\mathcal{J}_0$ without any regard to constraints as *primary-optimal*. A control minimizing some $\mathcal{J}_j$ (for $j > 0$) without any regard to other constraints will be called *j-optimal* (or simply *secondary-optimal*, when j is clear from the context). A control minimizing $\mathcal{J}_0$ subject to all of the above constraints on $\mathcal{J}_i$'s will be called *constrained-optimal*.

For a fixed $\boldsymbol{x} \in \Omega$, we will say that a control $\boldsymbol{a}_1(\cdot)$ is *j-dominated* by a control $\boldsymbol{a}_2(\cdot)$ if $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}_2(\cdot)) \leq \mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}_1(\cdot))$ for all $i = 0, \ldots, r$ and $\mathcal{J}_j(\boldsymbol{x}, \boldsymbol{a}_2(\cdot)) < \mathcal{J}_j(\boldsymbol{x}, \boldsymbol{a}_1(\cdot))$. We will also say that $\boldsymbol{a}_1(\cdot)$ is *dominated* by $\boldsymbol{a}_2(\cdot)$ if it is j-dominated for some $j \in \{0, \ldots, r\}$. E.g., the constrained-optimal control $\boldsymbol{a}^*(\cdot)$ described above might be dominated, but will not be 0-dominated by any control. Any control $\boldsymbol{a}^{**}(\cdot)$ dominating $\boldsymbol{a}^*(\cdot)$ will have the same *primary cost* $\mathcal{J}_0$ and will also satisfy the same constraints; moreover, it will even spend less in at least one of the *secondary costs* $\mathcal{J}_1, \ldots, \mathcal{J}_r$.

We will say that $\boldsymbol{a}(\cdot)$ is *Pareto-optimal* for $\boldsymbol{x}$, if it is not dominated by any other control. In that case, its vector of costs corresponds to a point on a Pareto-front $\mathcal{P}(\boldsymbol{x})$ in a $\mathcal{J}_0, \ldots, \mathcal{J}_r$ space; see Figure 3.2A.

Our goal is to find an efficient numerical method for approximating the costs associated with Pareto-optimal controls. The efficiency requires solving this problem for all $\boldsymbol{x} \in \Omega$ simultaneously.

We begin by showing how to compute the total cost $\mathcal{J}_i$ incurred by using a control optimal with respect to a different running cost (section 3.1). We then describe a recent method due to Mitchell and Sastry for recovering a convex portion of the Pareto front by *scalarization* (section 3.2). Finally, in section 3.3 we describe a new method for solving fully this problem by augmenting the system state to include the "budget remaining" in each secondary cost.

3.1. Total cost along "otherwise optimal" trajectories

We will use $u_i(\boldsymbol{x})$ to denote the value function with respect to $\mathcal{J}_i$ if all other costs are ignored. As explained in section 2.1, $u_i(\boldsymbol{x})$ can be recovered as a viscosity solution of the PDE (2.5), if we set $K = K_i$ and $q = q_i$.

Given a different value function u derived for some other cost $\mathcal{J}$, we will define a restricted set of $\mathcal{J}$-optimal controls

$$\mathcal{A}_{u,\boldsymbol{x}} = \left\{ \boldsymbol{a}(\cdot) \in \mathcal{A} \mid \mathcal{J}(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = u(\boldsymbol{x}) \right\}.$$

As explained in section 2.1, if $\mathcal{J}$ satisfies the assumptions (A0-A3), then $\mathcal{A}_{u,\boldsymbol{x}}$ will be non-empty for every $\boldsymbol{x} \in \Omega$ and will contain a single element for every $\boldsymbol{x} \in \Omega \backslash \Gamma$.

We will use $v_i(\boldsymbol{x})$ to denote a $\mathcal{J}$-optimality-restricted value function with respect to $\mathcal{J}_i$:

$$v_i(\boldsymbol{x}) = \inf_{\boldsymbol{a}(\cdot) \in \mathcal{A}_{u,\boldsymbol{x}}} \mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}(\cdot)). \tag{3.1}$$

This notation relies on a fixed choice of $\mathcal{J}$ and u, and we will specify $\mathcal{J}$ explicitly in each case to avoid ambiguity. For the cases when $\mathcal{J} = \mathcal{J}_j$ and $u = u_j$, we will use v_{ij} instead of v_i. According to this definition, we also have $v_{ii} = u_i$.

The following properties of Pareto fronts follow from the above definitions. The proofs are simple and we omit them for the sake of brevity.

Property 3.1. $v_i(\boldsymbol{x}) \geq u_i(\boldsymbol{x})$ *for* $\forall x \in \bar{\Omega}$ *and for any choice of* $\mathcal{J}$ *satisfying assumptions* $(A0 - A3)$.

Property 3.2. *Suppose* $u(\boldsymbol{x})$ *is the value function corresponding to some* $\mathcal{J}$ *that satisfies assumptions* $(A0 - A3)$. *If* K_i *and* q_i *also satisfy* $(A0 - A2)$, *then* $v_i(\boldsymbol{x})$ *is lower semi-continuous on* Ω *and continuous wherever* $u(\boldsymbol{x})$ *is differentiable.*

Property 3.3. *Let* $\mathcal{U}(\boldsymbol{x}) = \left\{ (J_0, \ldots, J_r) \in \boldsymbol{R}^{r+1} \mid J_i \geq u_j(\boldsymbol{x}), j = 0, \ldots, r \right\}$. *Then for any* $\boldsymbol{x} \in \bar{\Omega}$, *we have*
1. $\mathcal{P}(\boldsymbol{x}) \subset \mathcal{U}(\boldsymbol{x})$ *and*
2. $(v_{i0}(\boldsymbol{x}), \ldots, v_{ir}(\boldsymbol{x})) \in \mathcal{P}(\boldsymbol{x})$ *for all* $i = 0, \ldots, r$.

Property 3.4. *If* $r = 1$, *then* $\mathcal{P}(\boldsymbol{x}) \subset [v_{00}(\boldsymbol{x}), v_{01}(\boldsymbol{x})] \times [v_{11}(\boldsymbol{x}), v_{10}(\boldsymbol{x})]$.

Suppose u is a smooth solution to the PDE (2.5). Then $\Gamma = \emptyset$ and for every $\boldsymbol{x} \in \Omega$ there exists a unique optimal/minimizing control value $\boldsymbol{a}^* = \boldsymbol{a}^*(\boldsymbol{x}, \nabla u(x)) \in A$. The local rate of increase of $\mathcal{J}_i$ along the corresponding trajectory is $K_i(\boldsymbol{x}, \boldsymbol{a}^*)$. This yields a system of $(r+1)$ linear PDEs

$$\begin{aligned} \nabla v_i(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}^*) &= -K_i(\boldsymbol{x}, \boldsymbol{a}^*), & \forall \boldsymbol{x} \in \Omega; \\ v_i(\boldsymbol{x}) &= q_i(\boldsymbol{x}), & \forall \boldsymbol{x} \in \partial\Omega; \quad i = 0, \ldots, r. \end{aligned} \tag{3.2}$$
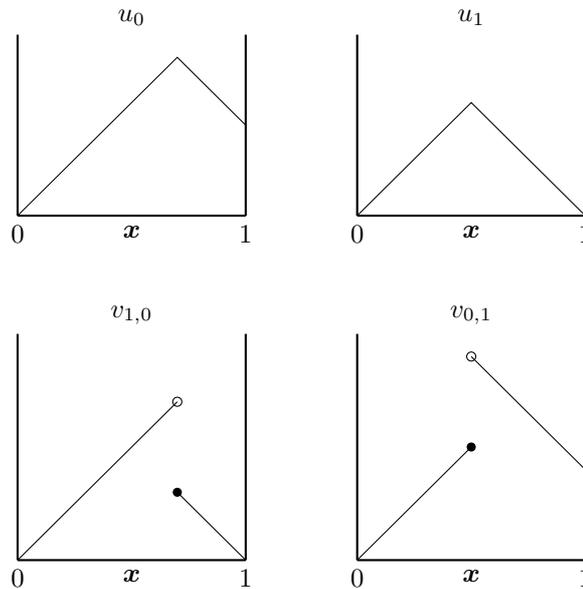
Fig. 3.1. *Cost along "otherwise optimal" trajectories. A simple one-dimensional example:* $\bar{\Omega} = [0, 1]$, $f = K_0 = K_1 = 1$, *and* $q_0(0) = q_1(0) = q_1(1) = 0$, *but* $q_0(1) = 0.4$. *Similar discontinuities may occur even if* $q_0 = q_1$ *(when* $K_0 \neq K_1$*).*

This system is coupled to a nonlinear equation (2.5), since $\boldsymbol{a}^*$ is generally not available a priori unless ∇u is already known.

In the Eikonal case (when $A = S_{n-1}$, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x})\boldsymbol{a}$ and $K(\boldsymbol{x}, \boldsymbol{a}) = K(\boldsymbol{x})$), the optimal direction of motion is

$$\boldsymbol{a}^* = -\frac{\nabla u(\boldsymbol{x})}{|\nabla u(\boldsymbol{x})|} = -\nabla u(\boldsymbol{x})\frac{f(\boldsymbol{x})}{K(\boldsymbol{x})},$$

and the system (3.2) can be rewritten as

$$\nabla v_i(\boldsymbol{x}) \cdot \nabla u(\boldsymbol{x}) = K_i(\boldsymbol{x}, \boldsymbol{a}^*)K(\boldsymbol{x})/f^2(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \Omega;$$

$$v_i(\boldsymbol{x}) = q_i(\boldsymbol{x}), \qquad\qquad\qquad\qquad \forall \boldsymbol{x} \in \partial\Omega; \qquad i = 0, \dots, r. \qquad (3.3)$$

If u is not smooth, the functions v_i may become discontinuous (see Figure 3.1) and a generalized solution is needed to define $v_i(\boldsymbol{x})$ at any points $\boldsymbol{x} \in \Gamma$. Luckily, Bellman's optimality principle provides an alternative definition to resolve this ambiguity:

$$v_i(\boldsymbol{x}) = \lim_{\tau \to 0^+} \min_{\boldsymbol{a}^* \in A_{u,\boldsymbol{x}}} \left\{ \tau K(\boldsymbol{x}, \boldsymbol{a}^*) + v_i\left(\boldsymbol{x} + \tau \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}^*)\right) \right\},$$

where $A_{u,\boldsymbol{x}} \subset A$ is the set of minimizing control values in (2.5). If $\boldsymbol{x} \notin \Gamma$, then the $A_{u,\boldsymbol{x}}$ consists of a single element and this formulation is equivalent to (3.2). Whether or not $\boldsymbol{a}^*$ is unique, the above formula yields the lower semi-continuity of v_i. It can also be used (with a fixed small $\tau > 0$) to derive a first-order semi-Lagrangian discretization of (3.2).

We note that the key technical idea employed above (solving a linear equation along the characteristics of another PDE) is well-known and useful in many applications. A common version of it arises whenever there is a need to "propagate a constant" from the boundary along
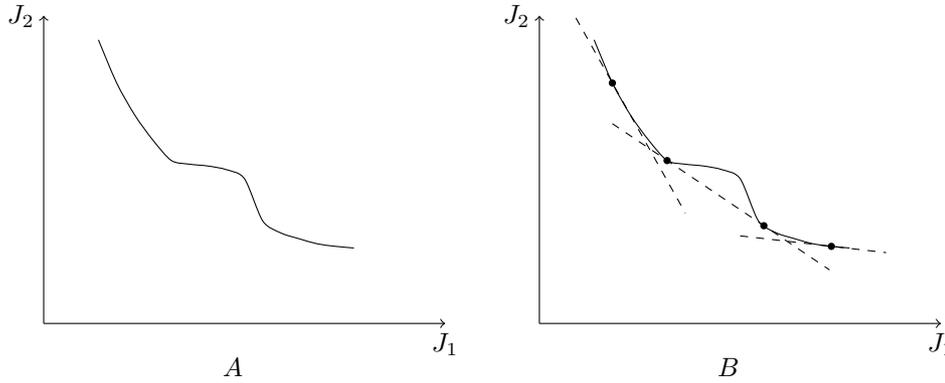
Fig. 3.2. Pareto Front (left) and its reconstruction using the "weighted sums" method (right). An optimum found for each $\lambda \in \Lambda$ yields a point on a convex part of Pareto front and the vector λ will be orthogonal to a support hyperplane to the front at that point (shown by a dashed line). In weighted sums method, an envelope of all support hyperlplanes is used to approximate the Pareto front, but misses all non-convex parts of the front [13].

the characteristics of some PDE. For example, this is the essential idea behind the "velocity extension equation" in [31] and the "escape equations" in [32]. A slightly less general version of our equation (3.3) (for isotropic costs/dynamics with $f = 1$ and $q_i = 0$) has also been previously used in [25].

3.2. Weighted sums method and Pareto fronts

Scalarization is a popular approach where a point on Pareto-front is recovered by minimizing an "aggregate objective function". The *method of weighted sums* defines that aggregate function as a convex linear combination of the original objectives [24].

A recent method based on this approach was introduced by Mitchell & Sastry for multiobjective optimal control in the case when $f = 1$ and all costs are isotropic [25]. Here we describe a slightly generalized version of their method.

Let

$$\Lambda = \left\{ \lambda = (\lambda_0, \ldots, \lambda_r) \mid \sum_{j=0}^{r} \lambda_j = 1 \quad \text{and} \quad \lambda_i \geq 0 \text{ for all } i \right\}$$

and suppose $\tilde{\Lambda}$ is some mesh discretizing Λ. Given $\lambda \in \Lambda$, define

$$K_\lambda(\boldsymbol{x}, \boldsymbol{a}) = \sum_{i=0}^{r} \lambda_i K_i(\boldsymbol{x}, \boldsymbol{a}); \qquad q_\lambda(\boldsymbol{x}, \boldsymbol{a}) = \sum_{i=0}^{r} \lambda_i q_i(\boldsymbol{x}, \boldsymbol{a}).$$

Solve Eq. (2.5) for $K = K_\lambda$ and $q = q_\lambda$. Having found u, solve the system (3.2) to obtain v_i's. The resulting point $(v_0(\boldsymbol{x}), \ldots, v_r(\boldsymbol{x}))$ will belong to the Pareto front $\mathcal{P}(\boldsymbol{x})$.

The above procedure is applied repeatedly for all meshpoints $\lambda \in \tilde{\Lambda}$ to obtain a mesh approximating $\mathcal{P}(\boldsymbol{x})$. Since finding each point on the Pareto front involves solving one non-linear and $(r + 1)$ linear PDEs, any restriction of the computational domain $\Omega \subset \boldsymbol{R}^n$ is worthwhile. This can be often accomplished by finding $u_1, \ldots, u_r$ first and excluding all $\boldsymbol{x}$ for which $u_i(\boldsymbol{x}) > B_i$ for some $i \in \{1, \ldots, r\}$.

Aside from the computational cost of the above procedure, this approach suffers from two usual problems associated with the method of weighted sums. First, a uniform mesh on Λ often results in a highly non-uniform mesh on $\mathcal{P}(\boldsymbol{x})$. Secondly, the weighted sum method can approximate only a convex part of the Pareto front [13]; see Figure 3.2B. This may result in selecting suboptimal trajectories. Mitchell & Sastry acknowledge that, for non-convex fronts, "this method may fail to detect a feasible path even if one exists", but they report that "non-convexity has not been a problem" in their numerical experiments. They further suggest that "neighboring values of λ can be used to bound the error in the convex approximation" of non-convex portions of the front. We believe that the latter procedure can be very inaccurate, especially since the Pareto front is frequently discontinuous for multiobjective optimal control problems. See Figure 4.5 for an example of non-convexities actually present in the test-problem introduced in [25].

In addition, we note that recovering the entire Pareto front for each $\boldsymbol{x} \in \Omega$ is excessive and unnecessary when the real goal is to solve the problem for a fixed list of constraints $B_1, \ldots, B_r$. If instead of using a mesh $\tilde{\Lambda}$, we attempt changing λ adaptively, it is not generally clear in what direction should λ be perturbed in order to satisfy the constraints.

Despite the above limitations, this technique can be useful in many applications, whenever there is a need to produce at least some of the Pareto-optimal trajectories efficiently. E.g., a similar scalarization approach has been independently used by Kim and Hespanha in path planning (to minimize the risk of detection/interception) for groups of UAVs [22].

3.3. An augmented PDE on an expanded state space

We propose an alternative approach based on augmenting the system-state space to reflect the budget remaining in each of the secondary costs. This is a generalization of the idea classically used to recast a Bolza problem as a problem with zero running cost by adding an extra dimension to the state space [8].

Suppose $b_i \in [0, B_i]$ is the "budget" remaining in the secondary cost $\mathcal{J}_i$. We define an extended state variable $\hat{\boldsymbol{x}} = (\boldsymbol{x}, b_1, \ldots, b_r)$ and the extended state space $\Omega_e = \Omega \times (0, B_1) \times \ldots \times (0, B_r)$. The *outflow boundary* and the *inflow boundary* of this domain are

$$\mathcal{B}_{\text{out}} = \left\{ \hat{\boldsymbol{x}} \in \partial \bar{\Omega}_e \mid \boldsymbol{x} \in \Omega, \text{ and } b_i \in (0, B_i] \text{ for } i = 1, \ldots, r \right\} \quad \text{and} \quad \mathcal{B}_{\text{in}} = \partial \Omega_e \backslash \mathcal{B}_{\text{out}}.$$

We note that $\mathcal{B}_{\text{out}}$ is the part of the boundary where at least one of the budgets is at the maximal level ($\exists j$ such that $b_j = B_j$). For the case $r = 1$, $\mathcal{B}_{\text{in}}$ coincides with the so called *parabolic boundary* [15]; see Figure 3.3. We will also refer to a *feasible subset* of $\mathcal{B}_{\text{in}}$:

$$\mathcal{B}_{\text{f}} = \left\{ \hat{\boldsymbol{x}} \in \mathcal{B}_{\text{in}} \mid \boldsymbol{x} \in \partial \Omega, \text{ and } b_i \geq q_i(\boldsymbol{x}) \text{ for } i = 1, \ldots, r \right\}.$$

The extended state at the time t will be denoted as $\hat{\boldsymbol{y}}(t) = (\boldsymbol{y}(t), \beta_1(t), \ldots, \beta_r(t)) \in \bar{\Omega}_e$, and the extended dynamics is now defined by

$$\boldsymbol{y}'(t) = \boldsymbol{f}(\boldsymbol{y}(t), \boldsymbol{a}(t)); \tag{3.4a}$$

$$\beta_i'(t) = -K_i(\boldsymbol{y}(t), \boldsymbol{a}(t)), \qquad i = 1, \ldots, r; \tag{3.4b}$$

$$\hat{\boldsymbol{y}}(0) = \hat{\boldsymbol{x}} = (\boldsymbol{x}, b_1, \ldots, b_r) \in \Omega_e. \tag{3.5}$$

As before, the exit time corresponding to a particular control is defined by (2.2), but we will use $T = T_{\hat{\boldsymbol{x}}, \boldsymbol{a}}$ to simplify the notation.

The total cost of this control is defined as

$$\hat{\mathcal{J}}(\hat{\boldsymbol{x}}, \boldsymbol{a}(\cdot)) = \begin{cases} \int_0^T K_0(\boldsymbol{y}(t), \boldsymbol{a}(t))\, dt \ + \ q_0(\boldsymbol{y}(T)) & \text{if } \hat{\boldsymbol{y}}(T) \in \mathcal{B}_{\mathrm{f}}, \\ +\infty & \text{otherwise.} \end{cases} \tag{3.6}$$

The (possibly infinite) value function of the new control problem is defined, as usual: $w(\hat{\boldsymbol{x}}) = w(\boldsymbol{x}, b_1, \ldots, b_r) = \inf \hat{\mathcal{J}}(\hat{\boldsymbol{x}}, \boldsymbol{a}(\cdot))$.

If the value function is smooth, the standard argument (based on Taylor-expanding $w(\hat{\boldsymbol{y}}(\tau))$ in the optimality principle), shows that w satisfies the PDE

$$\min_{\boldsymbol{a} \in A} \left\{ K_0(\boldsymbol{x}, \boldsymbol{a}) + \nabla_{\boldsymbol{x}} w \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) \ - \ \sum_{i=1}^r K_i(\boldsymbol{x}, \boldsymbol{a}) \frac{\partial w}{\partial b_i} \right\} \ = \ 0. \tag{3.7}$$

with the boundary conditions

$$w(\hat{\boldsymbol{x}}) = \begin{cases} q_0(\boldsymbol{x}) & \text{on } \mathcal{B}_{\mathrm{f}}; \\ +\infty & \text{on } \mathcal{B}_{\mathrm{in}} \backslash \mathcal{B}_{\mathrm{f}}. \end{cases} \tag{3.8}$$

However, w can be not only non-smooth, but also discontinuous inside Ω_e since the boundary data is discontinuous and no local controllability (in the directions $b_1, \ldots, b_r$) is present; see Remark 2.1. Nevertheless, w can still be interpreted as a unique *discontinuous viscosity solution* [37] of equation (3.7), despite the fact that Soner's "inward pointing" condition is clearly violated on $\mathcal{B}_{\mathrm{out}}$ [26].

As in the single-objective case, if the minimization in $\boldsymbol{a}$ can be performed analytically, the augmented PDE can be rewritten in a simpler form. For example, in the fully isotropic case,

$$K_0(\boldsymbol{x}) \ - \ |\nabla_{\boldsymbol{x}} w| f(\boldsymbol{x}) \ - \ \sum_{i=1}^r K_i(\boldsymbol{x}) \frac{\partial w}{\partial b_i} \ = \ 0, \tag{3.9}$$
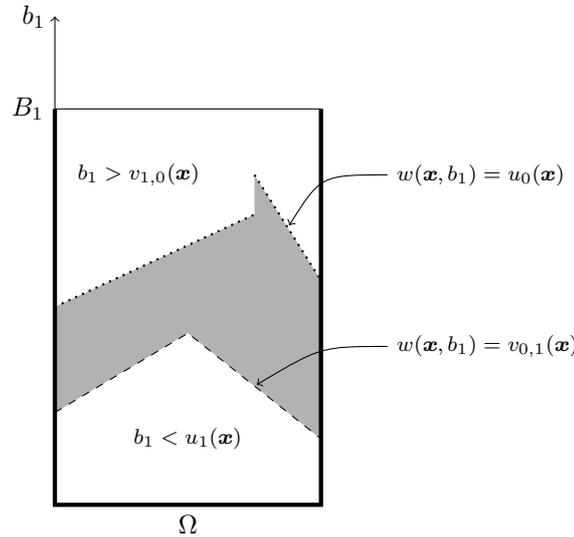

Fig. 3.3. A sketch of the domain $\bar{\Omega}_e$ for the case $r = 1$. The thick line shows the inflow boundary $\mathcal{B}_{\mathrm{in}}$. The dashed line is the graph of $u_1(\boldsymbol{x})$ (the minimum b_1 needed to leave Ω starting from $\boldsymbol{x}$). $\mathcal{B}_{\mathrm{f}}$ is the portion of $\mathcal{B}_{\mathrm{in}}$ above the dashed line. The dotted line shows the points where $b_1 = v_{1,0}(\boldsymbol{x})$. At that level and higher, the primary-optimal controls are feasible and $w(\boldsymbol{x}, b_1) = u_0(\boldsymbol{x})$. Above the dotted line, the constraint is slack. The PDE (3.7) has to be solved numerically in the shaded region.

which in the case $r = 0$ reduces to the usual Eikonal equation (2.7).

The following properties of w follow from the above definitions. The proofs are simple or standard and we omit most of them (except for the last three) for the sake of brevity.

Property 3.5. *The value function $w : \bar{\Omega}_e \mapsto (\mathbf{R} \cup \{+\infty\})$ is lower-semicontinuous; see [37] and references therein.*

Property 3.6. *The value function w is monotone non-increasing in each b_i. As a result, if $\boldsymbol{b} \geq \boldsymbol{c}$ componentwise, then $w(\boldsymbol{x}, \boldsymbol{b}) \leq w(\boldsymbol{x}, \boldsymbol{c})$.*

Property 3.7. *If for some i, $b_i < u_i(\boldsymbol{x})$, then $w(\boldsymbol{x}, \boldsymbol{b}) = +\infty$.*

Property 3.8. $w(\boldsymbol{x}, \boldsymbol{b}) \geq u_0(\boldsymbol{x}) = w\left(\boldsymbol{x}, v_{10}(\boldsymbol{x}), \ldots, v_{r0}(\boldsymbol{x})\right).$

Property 3.9. *w is Lipschitz continuous along every characteristic.*

Property 3.10. *The characteristics of PDE (3.7) have the following properties:*
1. *All characteristics are monotone-increasing in all b_i's.*
2. *Projections of characteristics on Ω yield constrained-optimal trajectories.*
3. *Any characteristic starting on $\mathcal{B}_f$ leads into $\bar{\Omega}_e$.*
4. *Any characteristic starting on $\mathcal{B}_{out}$ leads out of $\bar{\Omega}_e$.*

We will say that a control $\boldsymbol{a}(\cdot)$ is *feasible* for $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b})$ if $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}(\cdot)) \leq b_i$ for all $i = 1, \ldots, r$. We will say that $\hat{\boldsymbol{x}} \in \bar{\Omega}_e$ is a *feasible point* if it has at least one feasible control (i.e., if $w(\hat{\boldsymbol{x}}) < \infty$). We will say that a point $\hat{\boldsymbol{x}}$ is *i-tightly-constrained* if for every constrained-optimal control $\boldsymbol{a}^*(\cdot)$ we have $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}^*(\cdot)) = b_i$. Otherwise, we will call $\hat{\boldsymbol{x}}$ *i-slack*. We will also say that $\hat{\boldsymbol{x}}$ is *totally slack* if there exists a constrained-optimal control $\boldsymbol{a}^*(\cdot)$ such that $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}^*(\cdot)) < b_i$ for all $i = 1, \ldots, r$.

Property 3.11. *The point $(w(\boldsymbol{x}, \boldsymbol{b}), \boldsymbol{b})$ lies on the Pareto front $\mathcal{P}(\boldsymbol{x})$ in $\mathbf{R}^{r+1}$ if and only if $(\boldsymbol{x}, \boldsymbol{b})$ is i-tightly-constrained for all $i = 1, \ldots, r$. Moreover, if $(b_0, \ldots, b_r) \in \mathcal{P}(\boldsymbol{x})$ and $b_i \leq B_i$ for all $i = 1, \ldots, r$, then $w(\boldsymbol{x}, b_1, \ldots, b_r) = b_0$.*

Proof. Suppose $\boldsymbol{a}^*(\cdot)$ is the constrained-optimal control for $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b})$. If $\hat{\boldsymbol{x}}$ is i-slack, then $\mathcal{J}_i(\boldsymbol{x}, \boldsymbol{a}^*(\cdot)) < b_i$ and this does not contribute a point on Pareto front.

If $\hat{\boldsymbol{x}}$ is i-tightly-constrained for all $i = 1, \ldots, r$, then $\boldsymbol{a}^*(\cdot)$ cannot be dominated by any control. (If it were 0-dominated, that would contradict its constrained-optimality. If it were i-dominated for $i > 0$, that would contradict the fact that $\hat{\boldsymbol{x}}$ is i-tightly-constrained.)

Finally, suppose $\boldsymbol{a}(\cdot)$ is a control that realizes the cost vector $(b_0, \ldots, b_r) \in \mathcal{P}(\boldsymbol{x})$ with $b_i \leq B_i$ for all $i = 1, \ldots, r$. By the definition of w, we have

$$w(\boldsymbol{x}, b_1, \ldots, b_r) \leq \mathcal{J}_0(\boldsymbol{x}, \boldsymbol{a}(\cdot)) = b_0.$$

If we had $w(\boldsymbol{x}, b_1, \ldots, b_r) < b_0$ this would imply the existence of a control 0-dominating $\boldsymbol{a}(\cdot)$, which would contradict its Pareto-optimality. So, $w(\boldsymbol{x}, b_1, \ldots, b_r) = b_0$. $\square$

Property 3.12. *If the point $(\boldsymbol{x}_1, \boldsymbol{b})$ is totally slack, then w is locally Lipschitz-continuous in its first argument.*

Proof. First, we note that there exists some open neighborhood of $\boldsymbol{x}_1$ such that $(\boldsymbol{x}_2, \boldsymbol{b})$ is totally slack for every $\boldsymbol{x}_2$ from that neighborhood. Otherwise there would exist a sequence of i-tightly-constrained $\boldsymbol{x}_j$'s converging to $\boldsymbol{x}_1$, which can be used to show that $(\boldsymbol{x}_1, \boldsymbol{b})$ is i-tightly-constrained as well.

If $(\boldsymbol{x}_1, \boldsymbol{b})$ is totally slack, then, starting from any point $\boldsymbol{x}_2$ close to $\boldsymbol{x}_1$, the i-th cost of reaching $\boldsymbol{x}_1$ is bounded above by $|\boldsymbol{x}_1 - \boldsymbol{x}_2| \max(K_i/|\boldsymbol{f}|)$, where the maximum is taken over all $\boldsymbol{x}$ and all $\boldsymbol{a}$ such that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})/K_i(\boldsymbol{x}, \boldsymbol{a}) \in \partial V_i(\boldsymbol{x})$. By the assumption (A3), V_i contains the origin in its interior; so, there exists a constant L such that the i-th cost of reaching $\boldsymbol{x}_1$ from $\boldsymbol{x}_2$ is bounded by $L|\boldsymbol{x}_1 - \boldsymbol{x}_2|$ for all i. Suppose we travel from $(\boldsymbol{x}_2, \boldsymbol{b})$ to $(\boldsymbol{x}_1, \tilde{\boldsymbol{b}})$, where $\tilde{\boldsymbol{b}} \leq \boldsymbol{b}$. If $\boldsymbol{x}_2$ is sufficiently close to $\boldsymbol{x}_1$, then $(\boldsymbol{x}_1, \tilde{\boldsymbol{b}})$ is still totally slack, and any constrained-optimal control for $(\boldsymbol{x}_1, \boldsymbol{b})$ will be still feasible for $(\boldsymbol{x}_1, \tilde{\boldsymbol{b}})$. Thus,

$$w(\boldsymbol{x}_2, \boldsymbol{b}) \leq L|\boldsymbol{x}_1 - \boldsymbol{x}_2| + w(\boldsymbol{x}_1, \boldsymbol{b}).$$

To complete the proof, we repeat the argument switching the roles of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$.   □

Property 3.13. *If $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b})$ is totally slack, $\boldsymbol{a}^*(\cdot)$ is a constrained-optimal control for $\hat{\boldsymbol{x}}$ and $\boldsymbol{y}^*(t)$ is the corresponding trajectory in $\bar{\Omega}$, then $\boldsymbol{y}^*(t)$ is also a characteristic of problem (2.5) with $K = K_0$ and $q = q_0$.*

Proof. Briefly, if $(\boldsymbol{x}, \boldsymbol{b})$ is totally slack, then any sufficiently small perturbation of $\boldsymbol{a}^*(\cdot)$ will also be feasible. Since $\boldsymbol{a}^*(\cdot)$ is constrained-optimal, the function $\boldsymbol{y}^*(t)$ is a local minimizer of $\mathcal{J}_0$ and will be a solution of the Euler-Lagrange equation in $\boldsymbol{R}^n$ (see, e.g., [15]). By Pontryagin's maximum principle, it will also be a characteristic of the corresponding HJ PDE on $\bar{\Omega}$. This is an interesting fact, since the characteristics of (2.5) yield locally primary-optimal (unconstrained) trajectories.   □

Property 3.10.1 is the basis for explicitly causal discretizations of the augmented PDE, which enables an efficient numerical treatment (by "marching" in any direction b_i). Properties 3.7 and 3.8 can be used to reduce the computational domain, as shown in Figure 3.3 and further explained in section 3.5. Property 3.11 can be used to extract the relevant part of the Pareto front from the values of w.

3.4. Numerical method for the augmented PDE

We consider a Cartesian grid $\hat{X}$ on $\bar{\Omega}_e$. For simplicity, we will assume the same grid spacing h in all spatial dimensions and grid spacing $\Delta b_1, \ldots, \Delta b_r$ in each of the constraint/secondary cost directions. Let $\hat{h} = \max\{h, \max_i \Delta b_i\}$. Our goal is to construct an approximate solution W converging to the lower semi-continuous value function w as $\hat{h} \to 0$.

If the minimization in $\boldsymbol{a}$ can be performed analytically (e.g., in the fully isotropic case of equation (3.9)), a natural Eulerian framework scheme may be obtained by using upwind finite differences to approximate the partial derivatives of w. However, this approach, even when feasible, would be subject to CFL-type stability conditions, which would potentially have a significant impact on the computational cost of the scheme. (A simple example to illustrate this: suppose the $\boldsymbol{f}$ and K_0 are isotropic, $r = 1$, $q_1 = 0$ on $\partial\Omega$, and $K_1 = 1$, making $\mathcal{J}_1$ be the total time to $\partial\Omega$ along a given trajectory. If upwind finite differences are used in a given b_1-slice to approximate $\nabla_{\boldsymbol{x}} w$ and the forward-difference is used to approximate $\frac{\partial w}{\partial b_1}$, this results in a scheme suitable for explicit forward "time"-marching in b_1-direction, but not surprisingly requires a standard CFL condition $\max_{\boldsymbol{x}} f(\boldsymbol{x}) \leq h/\Delta b_1$ for stability.)

Instead, we chose to implement a semi-Lagrangian discretization of the augmented PDE (3.7). In addition to improved stability properties, the resulting scheme is also easy to extend to unstructured meshes in $\bar{\Omega}_e$ and is applicable when the minimization in $\boldsymbol{a}$ cannot be handled analytically (which is often the case for control-theoretic problems). Our discretization is a variant of (2.9). Given a point $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b}) = (x_1, \ldots, x_n, b_1, \ldots, b_r)$, we define a new system state $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ as follows:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \tau_{\boldsymbol{a}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}); \qquad \tilde{b}_i = b_i - \tau_{\boldsymbol{a}} K_i(\boldsymbol{x}, \boldsymbol{a}), \;\; \text{for } i = 1, \ldots, r. \tag{3.10}$$

Here the obvious dependence of the new state on the choice of control $\boldsymbol{a}$ is omitted for the sake of notational simplicity.

$$W(\boldsymbol{x}, b_1, \ldots, b_r) = \min_{\boldsymbol{a} \in A} \left\{ \tau_{\boldsymbol{a}} K_0(\boldsymbol{x}, \boldsymbol{a}) + W(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}}) \right\}, \quad \forall (\boldsymbol{x}, b_1, \ldots, b_r) \notin \mathcal{B}_{\text{in}}. \tag{3.11}$$

To ensure that the discretized equations allow efficient marching in the direction b_1, it is sufficient to take

$$\tau_{\boldsymbol{a}} = \Delta b_1 / K_1(\boldsymbol{x}, \boldsymbol{a}), \tag{3.12}$$

which guarantees that $\tilde{b}_1 = b_1 - \Delta b_1$ for any choice of $\boldsymbol{a}$. This means that the new position $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ will be in the previous "b_1-slice", in which the values of W were already computed.

Of course, $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ will usually not be a gridpoint and $W(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ has to be approximated using the values of W at nearby gridpoints. Our implementation uses a standard tensor product of linear interpolations in all $\boldsymbol{x}$ and $\boldsymbol{b}$ variables; see, e.g., [9].

For example, if $n = 2$ and $r = 1$, this yields a bilinear interpolation. Suppose $\tilde{\boldsymbol{x}}$ lies in a cell with vertices $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_4$ (enumerated clockwise, starting from the lower left corner). Let $(\gamma_1, \gamma_2) = (\tilde{\boldsymbol{x}} - \boldsymbol{x}_1)/h$. Then

$$\begin{aligned} W(\tilde{\boldsymbol{x}}, \tilde{b}_1) = &\gamma_1 \left(\gamma_2 W(\boldsymbol{x}_3, \tilde{b}_1) + (1 - \gamma_2) W(\boldsymbol{x}_2, \tilde{b}_1) \right) \\ &+ (1 - \gamma_1) \left(\gamma_2 W(\boldsymbol{x}_4, \tilde{b}_1) + (1 - \gamma_2) W(\boldsymbol{x}_1, \tilde{b}_1) \right). \end{aligned} \tag{3.13}$$

Remark 3.1. If the value function is smooth on the cell, the resulting interpolation error is $\mathcal{O}(\tilde{h}^2)$, where $\tilde{h}$ is the size of that cell (i.e., $\tilde{h} = \max\{h, \max_{i>1} \Delta b_i\}$). However, in the worst case, w may be discontinuous, resulting in a $\mathcal{O}(1)$ interpolation error. (We note that the property 3.9 ensures the Lipschitz continuity of w on the characteristic itself, but the interpolation cell containing $\tilde{\boldsymbol{x}}$ might still include a discontinuity.)

The convergence of semi-Lagrangian schemes to discontinuous viscosity solutions is more subtle. In [4,5] Bardi, Falcone and Soravia have proven that, on any compact subset where w is continuous, the semi-Lagrangian approximation converges to w uniformly, provided $\tilde{h}/\tau \to 0$ as $\tau \to 0$. The resulting schemes were successfully used to approximate discontinuous viscosity solutions both in the context of infinite-horizon optimal control and in differential games. For viscosity solutions continuous on the entire domain, a different proof [5] yields the convergence (and the convergence rate estimates) under a milder assumption that $\tilde{h}/\tau$ remains bounded as $\tau \to 0$.

Since we are setting τ for each $\boldsymbol{x}$ and $\boldsymbol{a}$ separately, the above condition for convergence to discontinuous solutions becomes: $\tilde{h}/(\min \tau_{\boldsymbol{a}}) \to 0$ as $(\max \tau_{\boldsymbol{a}}) \to 0$. If $\tau_{\boldsymbol{a}}$ is selected according to (3.12), this condition requires that the grid refinement should be performed in such a way that $\tilde{h} = o(\Delta b_1)$. (An alternative approach would be to keep $\tilde{h} = \mathcal{O}(\Delta b_1)$, but pick $\tau_{\boldsymbol{a}}$ so that $\tau_{\boldsymbol{a}} K_1(\boldsymbol{x}, \boldsymbol{a}) = b_1 - \tilde{b}_1 = m \Delta b_1$, where the integer $m \to \infty$ and $m \Delta b_1 \to 0$ as $\hat{h} \to 0$.) However,

for the examples considered in this paper, the numerical evidence suggests convergence even with $m = 1$ (i.e., as prescribed by formula (3.12)) and with $\tilde{h} = \mathcal{O}(\Delta b_1)$. The discontinuities are smeared in a narrow band with the width of that smearing proportional to Δb_1; e.g., see the convergence study for a simple example in section 4.1. We note that any higher order accurate semi-Lagrangian scheme would need to use a larger stencil to interpolate $W(\tilde{x}, \tilde{b})$, and an ENO or WENO reconstruction would be needed to handle the discontinuities.

Even though it is not necessary in principle, our current implementation assumes the geometric dynamics (defined in section 2.1). The minimization in formula (3.11) is performed numerically using the standard "golden section search" algorithm. Once W is computed, optimal controls and trajectories are recovered by following the characteristics of PDE (3.7) numerically.

3.5. Reducing the computational domain & initialization

Given the high dimensionality of $\bar{\Omega}_e$, any reduction of the domain is likely to result in substantial savings in the computational cost. Two such reductions are obviously worthwhile; see Figure 3.3. In both cases we recover a surface consisting of special characteristics of (3.7) by efficiently solving other PDEs on lower-dimensional domains. We note that a similar approach was previously proposed in [42] for time-optimal control in the case of time-dependent dynamics.

First, by property 3.8, if $w(\boldsymbol{x}, \boldsymbol{b}) = u_0(\boldsymbol{x})$, a primary-optimal control is already feasible and further increase in secondary budgets will not yield any reduction of w. The n-dimensional surface on which this happens can be found a priori (by numerically approximating functions v_{i0} for $i = 1, \ldots r$).

Secondly, only a subset of $\bar{\Omega}_e$ is feasible: if the initial budget-vector $\boldsymbol{b}$ is insufficient to reach $\partial\Omega$ starting from $\boldsymbol{x}$, then $w(\boldsymbol{x}, \boldsymbol{b}) = +\infty$. We formally define the Minimal Feasible Level (MFL) as a graph of the function

$$w_1(\boldsymbol{x}, b_2, \ldots, b_r) = \min\left\{ b_1 \mid w(\boldsymbol{x}, b_1, b_2, \ldots, b_r) < +\infty \right\}.$$

As described below, the MFL can be efficiently recovered for any r by solving a sequence of PDEs on lower-dimensional domains. However, for $r = 1$, this task is particularly simple, since in that case the MFL coincides with the graph of $u_1(\boldsymbol{x})$, and the latter can be approximated by solving the PDE (2.5) numerically on Ω. As explained in [37], the augmented PDE (3.7) then has to be solved on the epigraph of u_1. Thus, using the results in section 3.1, the value[1] of w on the MFL is provided by $v_{01}(\boldsymbol{x})$.

Remark 3.2. To represent MFL on the grid $\hat{X}$, our current implementation uses the smallest integer i such that $u_1(\boldsymbol{x}) \leq i\Delta b_1$ and then initializes $W(\boldsymbol{x}, i\Delta b_1) = v_{01}(\boldsymbol{x})$. This procedure is conservative (in the sense that it overestimates the minimal necessary budget), but it also introduces additional errors of order $O(\Delta b_1)$; see the convergence study in section 4.1. A better implementation can be built by locally altering the interpolation procedure near the MFL.

For $r > 1$ the approximation of the MFL is more subtle. In [37], Soravia suggested solving an augmented PDE on the set $\{(\boldsymbol{x}, \boldsymbol{b}) \in \bar{\Omega}_e \mid b_i \geq u_i(\boldsymbol{x}), \ i = 1, \ldots r\}$. However, we note that w

---

[1] We emphasize that the MFL does not really provide any additional boundary conditions for the equation (3.7) (since this surface consists of characteristics of that PDE). But for a semi-Lagrangian discretization, the numerical values on the MFL are needed – for an $\hat{x}$ just above that surface, the optimal $\tilde{x}$ might well lie in a cell one of whose corners is on the MFL. We note that any $\tilde{x}$ falling below the MFL is obviously non-optimal; when such situation arises during the minimization in equation (3.11), we use $W(\tilde{x}) = +\infty$.
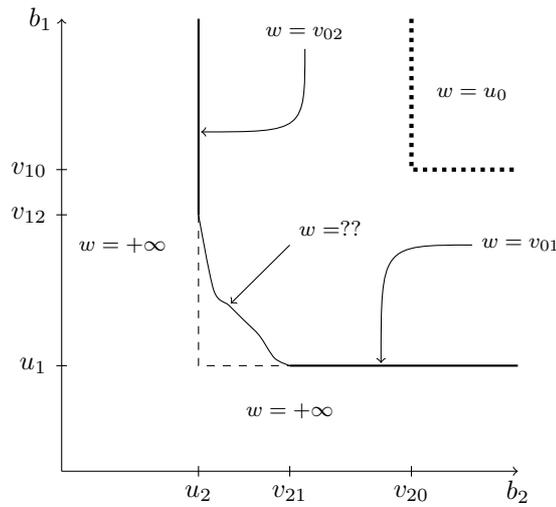
Fig. 3.4. *Two secondary costs diagram for a single point $\boldsymbol{x} \in \Omega$. The boundary of the feasible set ($\boldsymbol{x}$'s MFL) is shown by a solid line. To solve the PDE using an efficient semi-Lagrangian scheme, it is necessary to find the entire MFL (including its curved and possibly non-convex part) and pre-initialize w on that boundary. The dotted line shows the boundary of the set where the unconstrained (primary-optimal) controls are feasible.*

need not be finite everywhere on that set. This is already evident for $r = 2$. Indeed, a control optimal according to $\mathcal{J}_1$ usually incurs a higher $\mathcal{J}_2$-cost than the control optimal according to the latter (and vice versa). As a result, $w(\boldsymbol{x}, u_1(\boldsymbol{x}), u_2(\boldsymbol{x}))$ is usually $+\infty$. This situation is illustrated in Figure 3.4 (i.e., the region between the dashed and solid lines is not feasible). If $b_2 \geq v_{21}(\boldsymbol{x})$, then $w(\boldsymbol{x}, u_1(\boldsymbol{x}), b_2) = v_{01}(\boldsymbol{x})$; this allows to recover a part of the MFL quite easily (the thick solid lines in Figure 3.4). However, recovering the rest of the MFL requires more computational effort.

Suppose $r = 2$ and $w_1(\boldsymbol{x}, b_2)$ is the minimum budget b_1 sufficient to reach $\partial\Omega$ from $\boldsymbol{x}$ with $\mathcal{J}_2 \leq b_2$. It is clear that the PDE for w should be solved on the epigraph of w_1. Since w_1 optimizes $\mathcal{J}_1$ subject to a constraint on $\mathcal{J}_2$, we note that w_1 plays the same role for the pair of costs $(\mathcal{J}_1, \mathcal{J}_2)$ as w played for the triple $(\mathcal{J}_0, \mathcal{J}_1, \mathcal{J}_2)$. Thus, the same argument used in section 3.3 shows that w_1 can be recovered as a viscosity solution of a PDE similar to (3.7), but posed on $\bar{\Omega} \times [0, B_2]$; i.e.,

$$\min_{\boldsymbol{a} \in A} \left\{ K_1(\boldsymbol{x}, \boldsymbol{a}) + \nabla_{\boldsymbol{x}} w_1 \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) - K_2(\boldsymbol{x}, \boldsymbol{a}) \frac{\partial w_1}{\partial b_2} \right\} = 0. \tag{3.14}$$

To solve the latter PDE efficiently on an $(n + 1)$-dimensional domain, we first need to approximate its own n-dimensional "MFL_1". Luckily, this is equivalent to the one-secondary-cost problem already considered above. Thus, this new MFL_1 is the graph of $u_2(\boldsymbol{x})$ and the value of w_1 on the MFL_1 is provided by $v_{12}(\boldsymbol{x})$. While computing w_1, we can also integrate K_0 along w_1's optimal trajectories (as described in section 3.1) to find the values of w on its MFL (i.e., on the graph of w_1 in $\bar{\Omega} \times [0, B_1] \times [0, B_2]$). This is the approach used to compute the example in section 4.8.

For the general case ($r > 2$), the same procedure can be applied recursively. More precisely, let $w_i(\boldsymbol{x}, b_{i+1}, \ldots, b_r)$ be the minimum budget b_i sufficient to reach $\partial\Omega$ from $\boldsymbol{x}$ with $\mathcal{J}_j \leq b_j$ for all $j > i$. According to this definition, $w_r(\boldsymbol{x}) = u_r(\boldsymbol{x})$ and, for $\forall i < r$, w_i is finite only on the

epigraph of w_{i+1}. Focusing only on the costs $\mathcal{J}_i, \ldots \mathcal{J}_r$, the argument of section 3.3 shows that w_i is the (unique lower semi-continuous) viscosity solution of

$$\min_{\boldsymbol{a} \in A} \left\{ K_i(\boldsymbol{x}, \boldsymbol{a}) + \nabla_{\boldsymbol{x}} w_i \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) - \sum_{j=i+1}^{r} K_j(\boldsymbol{x}, \boldsymbol{a}) \frac{\partial w_i}{\partial b_j} \right\} = 0 \qquad (3.15)$$

on $\bar{\Omega} \times [0, B_{i+1}] \times \ldots \times [0, B_r]$. In this $(n + r - i)$-dimensional domain, the MFL for w_i (i.e., the MFL_i) is provided by a graph of w_{i+1}, and the values of w_i on MFL_i can be computed by integrating K_i along the optimal trajectories of w_{i+1}. Finally, the graph of w_1 is the MFL for $w = w_0$, and the latter solves the PDE (3.7).

Remark 3.3. Both techniques presented in this section allow a significant reduction of the computational domain. The net effect of the first technique (based on considering only those gridpoints where $w(\boldsymbol{x}, \boldsymbol{b}) \geq u_0(\boldsymbol{x})$) depends on how large the budget bounds B_is are compared to the values of $v_{i0}(\boldsymbol{x})$. If $r = 1$, the percent of excluded gridpoints can be found by averaging the ratio $[\max\{B_1 - v_{10}(\boldsymbol{x}), 0\} / B_1]$ over Ω.

The net effect of the second technique (based on finding the MFL) depends on the degree of "primary-non-optimality" of "secondary optimal" trajectories. If $r = 1$, the percent of gridpoints additionally excluded (after the MFL is computed) can be found by averaging the ratio $[\min\{B_1, u_1(\boldsymbol{x})\} / \min\{B_1, v_{10}(\boldsymbol{x})\}]$ over Ω. Based on the experimental evidence, the resulting efficiency gains are quite substantial. E.g., in the examples of sections 4.1 and 4.2.1, the MFL allowed additional exclusion of 50% and 93% of the remaining gridpoints respectively. In the example of section 4.8, where $r = 2$, the corresponding reduction was 76%, confirming that the recursive procedure to compute the MFL when $r > 1$ is worthwhile.

3.6. Selecting τ & optimal ordering

Formula (3.10) for the new state after using control $\boldsymbol{a}$ for $\tau_{\boldsymbol{a}}$ seconds is based on the assumption that, for every fixed $\boldsymbol{a}$, $\boldsymbol{f}$ and all K_i's are approximately constant near $\boldsymbol{x}$. There are two advantages to this formula. First, it allows to compute $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ very quickly. Second, taking $\tau_{\boldsymbol{a}} = \Delta b_1 / K_1(\boldsymbol{x}, \boldsymbol{a})$ ensures that the new state is in the previous b_1-slice; i.e., $\tilde{b}_1 = b_1 - \Delta b_1$. This allows for the explicit causality and marching in the b_1 direction. In section 4.9, we will refer to this implementation as Algorithm 1 (see Figure 3.5A).

There are also two obvious drawbacks to this discretization. First, this linear approximation is poor wherever $\boldsymbol{f}$ and K_i's vary significantly near $\boldsymbol{x}$. Second, the local error in formula (3.11) is generally $\mathcal{O}(\tau_{\boldsymbol{a}}^2)$ even for a smooth w and perfectly known $W(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$; thus, it would be preferable to select the smallest $\tau_{\boldsymbol{a}}$ that still allows explicit marching in the direction b_1.

To address the first of these drawbacks, we have implemented Algorithm 2. Given the current state $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b})$ and a particular control value $\boldsymbol{a}$, we start with the ray from $\hat{\boldsymbol{x}}$ in the direction $[\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}), -K_1(\boldsymbol{x}, \boldsymbol{a}), \ldots, -K_r(\boldsymbol{x}, \boldsymbol{a})]$. We find the first intersection of that ray with an $(n + r - 1)$-dimensional cell of the grid. If that cell is fully in the previous b_1-slice, we can interpolate, as in Algorithm 1. Otherwise, we re-evaluate $\boldsymbol{f}$, and K_i's at the new point and follow the new ray, repeating the process until we reach the previous b_1-slice (see Figure 3.5B). This procedure is computationally more expensive than Algorithm 1, since we might need to traverse through several $(n + r)$-dimensional cells before finding $\tilde{\boldsymbol{x}}$, especially when $\Delta b_1 \min(K_1) > h \min |\boldsymbol{f}|$. However, it is much more suitable for problems with rapidly
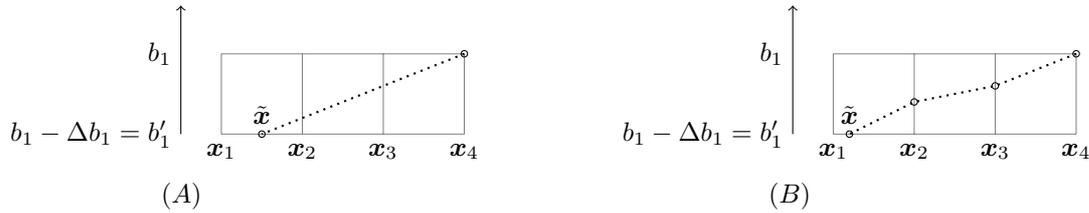
Fig. 3.5. Choices of τ illustrated for $r = 1$ and $n = 1$. Let $\hat{\boldsymbol{x}} = (\boldsymbol{x}_4, b_1)$. A trajectory corresponding to a particular $\boldsymbol{a}$ is shown by a dotted line. Algorithm 1 is illustrated on the left: set $\tau_{\boldsymbol{a}} = \Delta b_1 / K_1(\boldsymbol{x}, \boldsymbol{a})$ and assume that $\boldsymbol{f}$ and K_i's don't change. This ensures $\tilde{b}_1 = b_1'$ and gives a direct formula for $\tilde{\boldsymbol{x}} \in [\boldsymbol{x}_1, \boldsymbol{x}_2]$. Algorithm 2 is illustrated on the right: $\boldsymbol{f}$ and K_i's are re-evaluated at each intersection with 1-dimensional cells. Algorithm 3 is based on the fact that, if $W(\boldsymbol{x}_3, b_1)$ has been already computed, then there is no need to continue beyond the first intersection. In that case, a much smaller $\tau_{\boldsymbol{a}}$ already guarantees the causality, $\tilde{\boldsymbol{x}} = \boldsymbol{x}_3$, and $\tilde{b}_1 \in [b_1', b_1]$.

varying $\boldsymbol{f}$ and K_i's. (Indeed, in section 4.9 we consider several numerical examples, where these functions are actually discontinuous in $\boldsymbol{x}$.)

To alleviate the second problem, we have implemented Algorithm 3. The idea is to select the smallest $\tau_{\boldsymbol{a}}$ needed for the causality. Our implementation uses the smallest $\tau_{\boldsymbol{a}}$ sufficient to guarantee that $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{b}})$ lies in an $(n + r - 1)$-dimensional cell, all vertices of which have already had values of W previously computed (making interpolation possible). This will certainly be the case if that cell lies fully in the previous b_1-slice (as in Algorithm 2), but it may also happen earlier; see Figure 3.5B for an example. We note that all relevant intersection points are already computed in Algorithm 2. If one of them is suitable in the above sense, we don't need to continue beyond it (thus reducing both the computational cost and the local truncation error). In the worst case, we still need to trace this piecewise-linear trajectory up to its intersection with the previous b_1-slice (as in Algorithm 2). As a result, the computational cost of this Algorithm is always somewhat less than that of Algorithm 2 though obviously higher than that of Algorithm 1; see section 4.9.

Remark 3.4. We note that both the accuracy and the efficiency of Algorithm 3 clearly depend on the order of computing W's *within* the same b_1-slice. (The location of $\tilde{\boldsymbol{x}}$ in Figure 3.5B will be different depending on whether we have already computed $W(\boldsymbol{x}_3, \boldsymbol{b}_1)$ prior to $W(\boldsymbol{x}_4, \boldsymbol{b}_1)$.) Our current implementation uses a simple *lexicographic ordering* in secondary costs and then in spatial directions: assuming that $\hat{\boldsymbol{x}} = (\boldsymbol{x}, \boldsymbol{b}) = (x_1, \ldots, x_n, b_1, \ldots, b_r)$, we compute within each b_1-slice in the direction b_2, within each (b_1, b_2)-slice in the direction b_3, ... , within each $(\boldsymbol{b})$-slice in the direction x_1, within each $(\boldsymbol{b}, x_1)$-slice in the direction x_2, etc. While the preference for the secondary-cost (over spatial) directions is clearly motivated by the explicit causality (i.e., K_i's are positive), our fixed ordering of the spatial directions is arbitrary and hardly optimal. In the future, we would like to investigate the effect of using different orderings in spatial directions (e.g., in the $r = 1$ case, sorting $\hat{\boldsymbol{x}}$'s by W values found in the previous b_1-slice).

We emphasize that the goal of finding a good ordering within a b_1-slice is simply to reduce the local truncation error and to speed up each b_1-step.

4. Numerical Experiments

We illustrate our numerical method with several examples. Our approach requires to choose a primary objective function and treat other objectives as secondary. All feasible controls will satisfy the constraints $\mathcal{J}_i \leq B_i$ for $i = 1, \ldots, r$. We then minimize $\mathcal{J}_0$ along these feasible paths.

All of the examples in this section are computed for a two-dimensional system state and we assume that $\bar{\Omega} = [0, 1] \times [0, 1]$. In all of our examples (except the convergence study in section 4.1), the goal is to reach a single target point. We will thus assume that $q_i = 0$ at the target and $q_i = +\infty$ on the rest of $\partial\Omega$ for $i = 0, \ldots, r$. Wherever not explicitly specified, the target is at the point $(1, 1)$.

Most of the examples (except for section 4.2.2) are fully isotropic in cost & dynamics, and the results are obtained by solving a variant of equation (3.9). For the isotropic case, the dynamics of the system simplifies as

$$\boldsymbol{y}'(t) = f(\boldsymbol{y}(t)) \begin{bmatrix} \cos(\alpha(t)) \\ \sin(\alpha(t)) \end{bmatrix},$$

where $\boldsymbol{y}(0) = \boldsymbol{x} \in \Omega \subset \boldsymbol{R}^2$ and $f : \bar{\Omega} \mapsto \boldsymbol{R}$ is the speed, and $\alpha \in [0, 2\pi]$ is the control parameter.

In each test problem, we plot constrained-optimal paths for different resource vectors $\boldsymbol{b}$. We also show *secondary-optimal trajectories* (recovered from u_i's, for $i = 1, \ldots, r$) even in cases when these trajectories do not satisfy other integral constraints.

We note that the observability examples in subsections 4.5 - 4.8 use piecewise continuous running costs and/or speed functions. Even though this violates assumption (A1), the value function is also well-defined in this case and the semi-Lagrangian discretization (based on discretizing Bellman's optimality principle) appear to converge to it correctly. A detailed discussion of viscosity solutions to HJB PDEs with discontinuous Lagrangian can be found in [38]. A different class of observability-influenced path planning problems is considered in [10].

All figures presented in this section were computed using Algorithm 1 (in subsections 4.1 - 4.3, where the running costs and speeds are continuous) and Algorithm 3 (in subsections 4.5 - 4.8).

4.1. A simple example: convergence study

We use a two-dimensional generalization of example in Figure 3.1 to test the convergence of our method. We assume that $f = 1$, and $K_0 = K_1 = 1$. Unlike in all other examples of this section, here we assume that there are two possible exit points on the boundary $A_1 = (0, 0.5)$ and $A_2 = (1, 0.5)$ and define

$$q_0(\boldsymbol{x}) = \begin{cases} 1.5, & \text{if } \boldsymbol{x} = A_1; \\ 0, & \text{if } \boldsymbol{x} = A_2; \\ \infty, & \text{otherwise.} \end{cases} \qquad q_1(\boldsymbol{x}) = \begin{cases} 0, & \text{if } \boldsymbol{x} = A_1; \\ 0, & \text{if } \boldsymbol{x} = A_2; \\ \infty, & \text{otherwise.} \end{cases}$$

As a result, $\mathcal{J}_1$ is simply the pathlength of the chosen trajectory, all (constrained optimal) trajectories are straight lines leading to A_1 or A_2, and the analytical expression for the discontinuous solution w is readily available:

$$w(\boldsymbol{x}, b_1) = \begin{cases} |\boldsymbol{x} - A_2|, & \text{if } |\boldsymbol{x} - A_2| \leq b_1; \\ |\boldsymbol{x} - A_1| + 1.5, & \text{if } |\boldsymbol{x} - A_1| \leq b_1 < |\boldsymbol{x} - A_2|; \\ \infty, & \text{otherwise.} \end{cases}$$

Contour plot of Constrained Value function at slice no: 30 (401*401*41)

L_1 error computed on Ω_f			
h \ Δb_1	1/10	1/20	1/40
1/40	0.0161431	0.0132897	0.0217214
1/80	0.0117587	0.0078472	0.0081641
1/160	0.0088235	0.0051967	0.0044650
1/320	0.0075405	0.0034618	0.0024226
1/640	0.0069347	0.0026052	0.0014743

L_∞ error computed on $\Omega'(\Delta b_1)$			
h \ Δb_1	1/10	1/20	1/40
1/40	0.0039922	0.0114681	0.3558851
1/80	0.0016614	0.0024012	0.0533905
1/160	0.0007978	0.0008307	0.0038505
1/320	0.0004434	0.0003989	0.0004491
1/640	0.0002202	0.0002217	0.0001994

Fig. 4.1. *Convergence to a discontinuous solution (section 4.1). Left: the level sets of $w(x, 0.75)$. Right: the L_1 and L_∞ errors computed for various $(h, \Delta b_1)$ combinations.*

We use this example to test the convergence of our method numerically. (See Figure 4.1.)

We recall that there are typically four different sources of error in semi-Lagrangian schemes: (1) due to approximating the boundary conditions (in our case – also the approximation of the MFL) on the grid; (2) due to interpolation at the foot of the characteristic (e.g., as in formula (3.13)); (3) due to approximating the characteristic and (4) due to approximate integration of the solution along that characteristic.

Since in this example the characteristics are straight lines and K_0 is constant, the last two of these four sources of error are absent. Of the remaining two, the first should clearly be decreasing with Δb_1 (since errors of order $\mathcal{O}(\Delta b_1)$ are introduced when representing the MFL on the grid; see Remark 3.2). Assuming that the solution is smooth, each interpolation error is $\mathcal{O}(h^2)$ (due to our use of bilinear interpolation (3.13)). However, the number of interpolations is inversely proportional to Δb_1 and the cumulative effect of interpolation errors is larger when the total number of b_1-slices increases. Thus, if h is held constant, decreasing Δb_1 will eventually result in an increase in the overall error; this can be seen in the top two rows of the corresponding L_1-errors table. Of course, in more general problems, where characteristics are not straight lines and K_i's are not constant, the third and fourth sources of errors would normally prevent this phenomenon.

To verify the convergence, we include two tables of error measurements for various $(h, \Delta b_1)$ combinations. The first table shows errors measured in L_1 norm on Ω_f (the subset of Ω_e on which w is finite though not necessarily continuous). For discontinuous solutions, convergence in L_∞ norm is generally possible only for numerical methods that explicitly track the location of discontinuities. Since no such explicit tracking is performed here, we can only demonstrate L_∞-convergence away from discontinuities. The theoretical results in [3–5] guarantee that, if $h = o(\tau)$ as $\tau = \Delta b_1 \to 0$, then the semi-Lagrangian schemes uniformly converge to the viscosity solution w on any compact subset on which w is continuous. For each value of b_1, we define

the discontinuity set of w in the corresponding b_1-slice:

$$\mathcal{D}_{b_1} = \left\{ (\boldsymbol{x}, b_1) \in \Omega_e \mid w \text{ is discontinuous at } (\boldsymbol{x}, b_1) \right\},$$

and for each Δb_1 we define a subset of Ω_e to study the convergence:

$$\Omega'(\Delta b_1) = \left\{ \hat{\boldsymbol{x}} = (\boldsymbol{x}, b_1) \in \Omega_f \mid \text{distance}(\boldsymbol{x}, \mathcal{D}_{b_1}) \geq \frac{3}{2} \Delta b_1 \right\}.$$

The second table shows L_∞ errors measured on $\Omega'(\Delta b_1)$. Since the width of the "excluded band" is defined for each column of the second table separately, the initial errors in the second and third columns are significantly larger, but quickly decrease as $h \to 0$. In this example, our numerical results suggest the convergence even for $h = \mathcal{O}(\tau)$.

4.2. Fastest paths (with restriction on pathlength)

Here we consider two different examples (one isotropic with obstacles, the other anisotropic without obstacles, both inhomogeneous), in which the goal is to minimize the time-to-target subject to constraints on the maximal allowable pathlength. In the absence of obstacles, we use $K_0(\boldsymbol{x}) = 1$ and $K_1(\boldsymbol{x}, \boldsymbol{a}) = |\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a})|$ to ensure that $\mathcal{J}_0$ and $\mathcal{J}_1$ are respectively the time and the pathlength along the corresponding trajectory. If obstacles are present, we set $K_0 = +\infty$ inside them, to ensure that all trajectories passing through them have infinite cost $\mathcal{J}_0$.

4.2.1. Isotropic dynamics/cost in the presence of obstacles

Here we suppose that the dynamics is isotropic; i.e., $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x})\boldsymbol{a}$, where $\boldsymbol{a} \in S_1 = A$. For every $\boldsymbol{x} = (x, y) \in \Omega$ outside of obstacles, we will assume that

$$f(x, y) = 1 + 0.8 \sin(4\pi x) \sin(6\pi y) \quad \text{and} \quad K_1(x, y) = f(x, y).$$
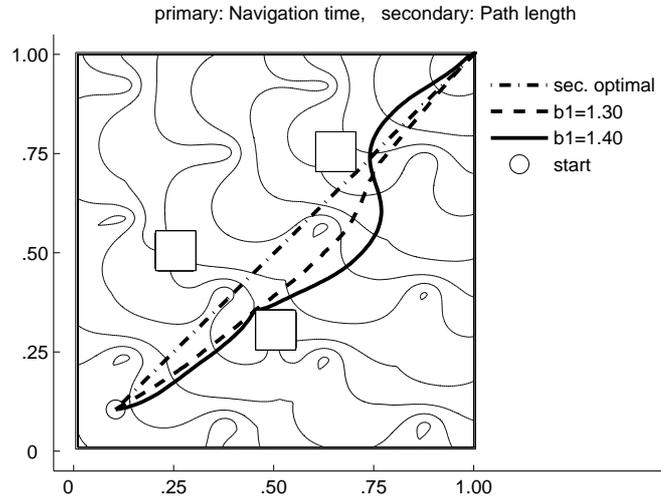
Fig. 4.2. Fastest paths, isotropic dynamics (constrained by pathlength) and obstacles. Optimal trajectories and the level curves of u_0.

Consider a goal of finding the fastest path to the top right corner $(1, 1)$ subject to restriction on a maximum allowable pathlength and in the presence of three rectangular obstacles. The computations are performed on a $301 \times 301 \times 301$ grid and several trajectories are shown in Figure 4.2 superimposed on top of the level curves of u_0. Secondary-optimal trajectory (dotted line) is the shortest path. Bold solid line shows a trajectory computed for a large maximum pathlength. The constraint is slack in this case and the resulting trajectory is in fact primary-optimal; hence its orthogonality to the level curves of u_0 (since for unconstrained isotropic problems the characteristics coincide with the gradient lines of u_0). The dashed line shows a constrained-optimal trajectory for a smaller budget (binding constraint).
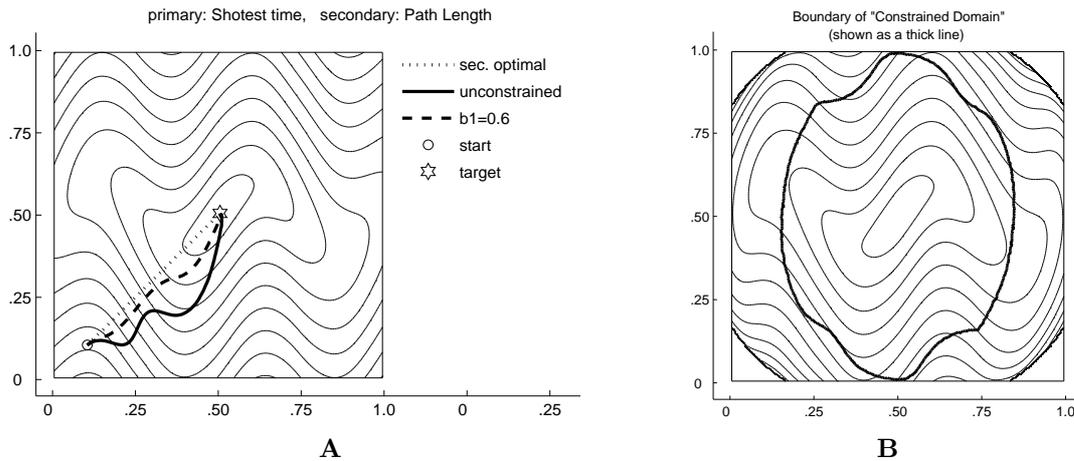
Fig. 4.3. Anisotropic dynamics, fastest paths (constrained by pathlength). Left: level sets of $w(\boldsymbol{x}, 1.5)$ (corresponding to level sets of u_0 in Figure 6A of [34]) and optimal trajectories from $(0.1, 0.1)$ to $(0.5, 0.5)$. Right: level sets of $w(\boldsymbol{x}, 0.6)$.

4.2.2. Anisotropic inhomogeneous dynamics

The following example of anisotropic dynamics was previously used in [34, 35], where it was motivated by problems in anisotropic seismic imaging. This belongs to a class of anisotropic geometric dynamics problems, i.e., $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{a}) = f(\boldsymbol{x}, \boldsymbol{a})\boldsymbol{a}$, where $\boldsymbol{a} \in S_1 = A$. Suppose $C : [0, 1] \mapsto \boldsymbol{R}$ is a smooth function. We are interested in defining f so that, for every $\boldsymbol{x} = (x, y) \in \Omega$, the "velocity profile" $V(\boldsymbol{x}) = \{f(\boldsymbol{x}, \boldsymbol{a})\boldsymbol{a} \mid \boldsymbol{a} \in S_1\}$ is an ellipse whose major/minor semi-axis have lengths F_2 and F_1 respectively and the major semi-axis is aligned with the graph of C (i.e., parallel to its tangent at the point x). This is attained by setting

$$f(\boldsymbol{x}, \boldsymbol{a}) = F_2 \left(1 + \left(\begin{bmatrix} p \\ q \end{bmatrix} \cdot \boldsymbol{a}\right)^2\right)^{-1/2}, \qquad \text{where} \quad \begin{bmatrix} p \\ q \end{bmatrix} = \frac{\sqrt{\left(\frac{F_2}{F_1}\right)^2 - 1}}{\sqrt{1 + \left(\frac{dC}{dx}(x)\right)^2}} \begin{bmatrix} \frac{dC}{dx}(x) \\ -1 \end{bmatrix}.$$

The derivation can be found in [35] and [41]. We use $C(x) = 0.1225 \sin(4\pi x)$, $F_2 = 0.8$, $F_1 = 0.2$ and compute the optimal (fastest) trajectories to the center of Ω. The unconstrained problem leads to an anisotropic static Hamilton-Jacobi PDE for u_0, which can be efficiently solved using Ordered Upwind Methods; the level sets of u_0 for the above parameters can be found in Figure 6A of [34]). Here we compute pathlength-constrained min-time trajectories for the same
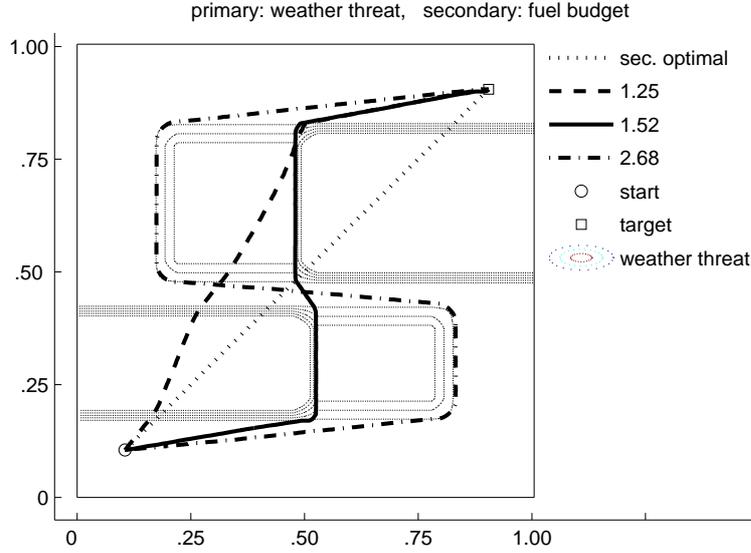
Fig. 4.4. Path planning for an airplane striving to be away from the region having high weather threat probability. The optimal paths are shown for the same constraint values used in Figure 3 of [25].

Table 4.1: Numerical parameters for the optimal flight-path example (Figure 4.4).

Grid points in system-domain	: 201×201
Grid points along secondary budget	: 401
Speed of the vehicle	: 1
Primary cost (Weather threat)	: explained in 4.1
Secondary cost(Fuel rate)	: 1
Starting point for optimal path	: $(0.1, 0.1)$
Target point for optimal path	: $(0.9, 0.9)$

example on a $201 \times 201 \times 301$ grid. Figure 4.3A shows optimal trajectories to the center starting from the point $(0.1, 0.1)$ superimposed on the level sets of $w(\boldsymbol{x}, 1.5)$. This is a large "budget": since $b_1 = 1.5 > v_{10}(\boldsymbol{x})$ for $\forall \boldsymbol{x} \in \Omega$, we have $w(\boldsymbol{x}, 1.5) = u_0(\boldsymbol{x})$ and all unconstrained-optimal trajectories are feasible. (We note that these optimal trajectories do not follow the gradient lines of u_0 since the speed $f(\boldsymbol{x}, \boldsymbol{a})$ is anisotropic.) On the other hand, for $b_1 = 0.6$, parts of the domain are not reachable (note that $w = +\infty$ in the corners of the domain in Figure 4.3B). The thick line separates the part of Ω where $w(\boldsymbol{x}, 0.6) = u_0(\boldsymbol{x})$; since the starting point $(0.1, 0.1)$ is outside that set, its constrained-optimal trajectory is different from the unconstrained-optimal.

4.3. Optimal flight-path: minimizing weather threat

Here, we consider a test example introduced by Mitchell and Sastry [25]: finding an optimal path for an airplane flying from one city to another. In one of their examples, weather threat is the primary objective to be minimized while the fuel minimization is the secondary objective. The running fuel cost K_1 and the speed of the airplane f are both taken to be unity in this example. Hence, the secondary objective, fuel cost is same as the path length.

As mentioned in [25], weather threat at a location, intuitively, is the probability of encountering a storm. Assuming the probability at all locations to be independent, the total

probability of encountering a storm during the flight is one minus the product of the probabilities of not encountering the storm at all locations along the path. As we require running costs of integral type, the logarithm of these probabilities are used to make weather-threat cost additive in nature. The numerical parameters for this simulation are listed in Table 4.1. Figure 4.4 shows optimal paths corresponding to different bounds on fuel budget, as well as the contours of weather threat cost. The latter is taken to be unity everywhere in $\bar{\Omega}$ except within the two rectangular bars where it jumps to a relatively higher value discontinuously. The magnitude of weather threat is 12 in the darker part of each of the rectangular bars and 4 in the other part of the bar. Weather threat function is further smoothened to remove the discontinuity and make it Lipschitz continuous.

As expected, the secondary-optimal path (the shortest path) follows a straight line between the two cities. Bounds on fuel budget associated with other constrained-optimal paths are shown in the legend. The Figure shows that, as the fuel budget increases, the airplane chooses a path through the region less susceptible to weather threat. These optimal paths match closely those in [25].

Figure 4.5 shows the Pareto front for this example. Pareto front is generated using two different approaches. In the first approach, fuel budget is taken as secondary objective while in the other approach weather threat is the secondary objective. As h and Δb_1 decrease, these two versions of Pareto front look more and more similar. The non-convex parts of this front have not been found in [25]; this illustrates the advantage of our approach as compared to the "weighted-sum scalarization" based techniques.
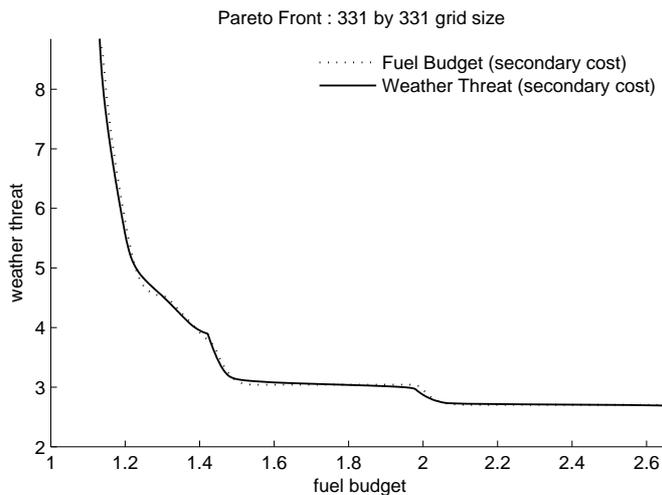

Fig. 4.5. Pareto front for a particular destination for the airplane problem. A convexification of this front can be found in Figure 4 in [25].

4.4. Computation of non-visible region

The remaining examples deal with robotic navigation in domains with obstacles in the presence of friendly and adversarial observers. The observer's position is assumed to be known and static and Ω is split into parts directly visible (Ω_v) or invisible (Ω_i) to each observer due to obstacles. The observability running cost is then set to be high on Ω_v and low on Ω_i for an

enemy observer (and the opposite of this for a friendly observer).

To enable the computation of constrained-optimal paths, we need to first determine Ω_v and Ω_i. While this can be accomplished by many methods, we use the efficient technique based on a Fast Marching Method and described in [30].

We first solve the Eikonal equation $|\nabla\psi_1(\boldsymbol{x})|f(\boldsymbol{x}) = 1$ with $f = 1$ on $\bar{\Omega}$ and the boundary condition $\psi_1 = 0$ at the observer's location. We then solve the Eikonal PDE for ψ_2 with the same boundary condition but with $f = 0$ inside the obstacles. The region with $\psi_2 > \psi_1$ defines the non-visible region. In practice, we use a (heuristically adjusted) threshold on the difference of ψ_2 and ψ_1. Thick grey lines are used in the following figures to show the boundaries of Ω_i.
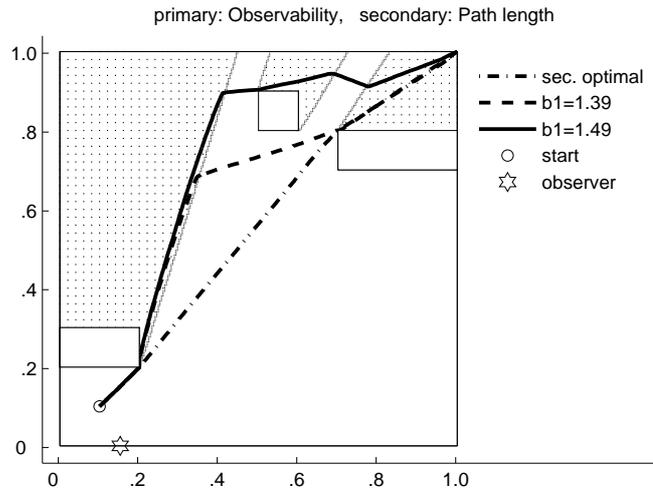
Fig. 4.6. Optimal path for a robot navigating to minimize the exposure to a static enemy observer.

Table 4.2: Numerical parameters for the exposure-minimization example (Figure 4.6).

Grid points in system-domain	: 251×251
Grid points along secondary budget	: 301
Speed of the vehicle	: 1
Primary cost (Observability cost)	: 10 in the visible region
	: 0.1 in the non-visible region
Secondary cost(Fuel rate)	: 1
Starting point for optimal path	: (0.1, 0.1)
Target point for optimal path	: (1.0, 1.0)
Observer location	: (0.15, 0)

4.5. Avoiding the observer

Here, we find optimal paths for a robot navigating in a region containing stationary obstacles and a stationary enemy observer. The robot strives to be minimally exposed to the enemy at the same time making sure to avoid the obstacles and stay within the specified fuel budget. Observability by the enemy becomes the primary cost ($\mathcal{J}_0$) to be minimized. As long as the secondary running cost K_1 remains positive, our numerical method can solve the PDE (3.7) efficiently even with with $K_0 = 0$ observability cost in Ω_i. However, this leads to infinitely many
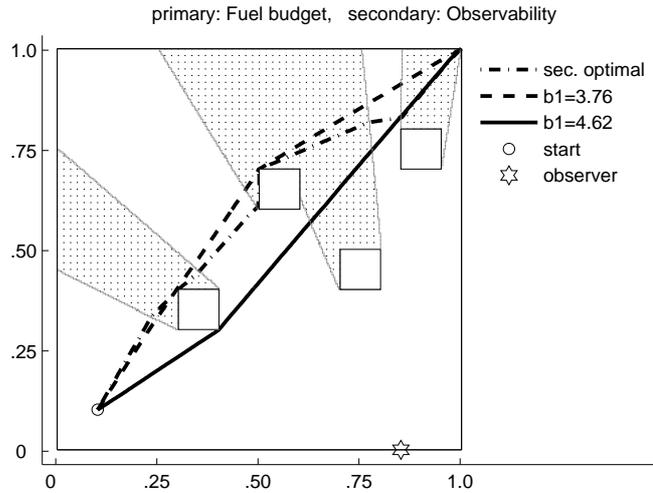
Fig. 4.7. Optimal path for a robot minimizing its fuel budget/path length under the maximum enemy exposure constraint. The enemy exposure budget of 4.62 is large enough to allow following the primary-optimal trajectory (thick line).

Table 4.3: Numerical parameters for path length minimization constrained by exposure (Figure 4.7).

Grid points in system-domain	: 301×301
Grid points along secondary budget	: 501
Speed of the vehicle	: 1
Primary cost (Fuel rate/Path length)	: 1
Secondary cost(Observability)	: 1 in the non-visible region
	: 5 in the visible region
Starting point for optimal path	: (0.1, 0.1)
Target point for optimal path	: (1.0, 1.0)
Observer location	: (0.85, 0)

possible optimal paths since any portion of the path inside Ω_i would not contribute anything to $\mathcal{J}_0$. To remove this arbitrariness and non-uniqueness, we assigned a small non-zero observability cost $K_0 = 0.1$ on Ω_i. Table 4.2 shows the other numerical parameters used in this experiment.

Figure 4.6 shows the non-visible region with its boundary as thick solid lines. The small rectangles represent obstacles. Optimal paths corresponding to different bounds B_1 on fuel budget are also plotted. Since the speed of motion is constant, the running costs are piecewise constant, and the obstacles are polygonal, it is easy to prove that all constraint-optimal paths are piecewise linear. As expected, the primary-optimal paths creep along the boundary of non-visible region. In this example, the target lies in Ω_i; the robot therefore follows the straight line path after it enters that component of Ω_i.

4.6. Minimizing fuel consumption/path length, constrained by enemy observability

Here, we find the optimal path for a robot with the same two objectives as in the previous section. But now the path length (or fuel consumption) is the primary cost and the enemy observability is secondary. The numerical parameters are shown in Table 4.3. Figure 4.7 shows the optimal paths for this example. The secondary-optimal path shown as a dashed line is the

least exposed to the enemy. As expected, the primary-optimal paths become shorter as we relax the constraint (or, alternatively, increase the "budget") for the maximum observability.

4.7. Striving to be observed

Given a stationary friendly observer, the goal in many applications is to minimize the total time outside of direct visibility while moving to the target. This is more or less the opposite of the problem considered in section 4.5. The total fuel available (or the maximum path length) is still treated as a constraint. The numerical parameters are shown in Table 4.4.

Figure 4.8 plots the optimal paths corresponding to different bounds on path length. When the bound on path length is tight, the robot has no option but to navigate through Ω_i. As we relax this bound, the robot finds a path which is always exposed to the observer.
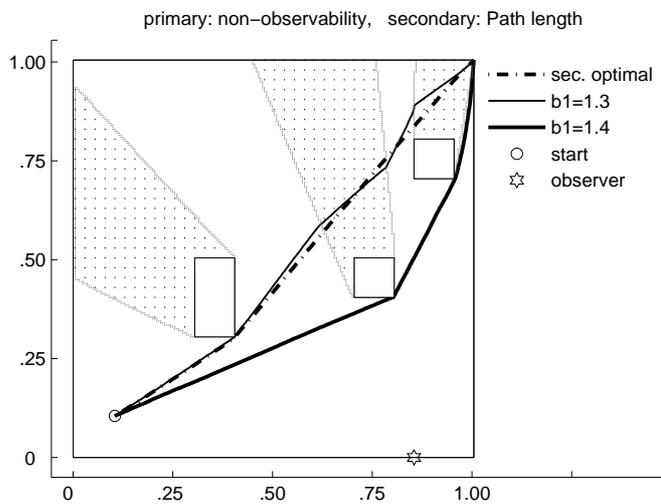
Fig. 4.8. Optimal path for a robot in the presence of a friendly observer.

Table 4.4: Numerical parameters for minimizing the non-exposure constrained by path length (Figure 4.8).

Grid points in system-domain	: 201 × 201
Grid points along secondary budget	: 301
Speed of the vehicle	: 1
Primary cost(Non-observability)	: 5 in non-visible region
	: 1 in visible region
Secondary cost (Fuel rate)	: 1
Starting point for optimal path	: (0.1, 0.1)
Target point for optimal path	: (1.0, 1.0)
Observer location	: (0.85, 0)

4.8. Path length minimization subject to two integral constraints

In this last example, we consider a problem of finding constrained-optimal paths in the presence of obstacles and two observers. The goal is to minimize the path-length subject to constraints on the amounts of time the robot can be visible to the enemy observer and invisible

to the friendly observer. Given two secondary costs, the numerical domain is four-dimensional. As explained in section 3.5, we first solve the PDE (3.14) on $\bar{\Omega} \times [0, B_2]$ to find the feasibility surface. We then march in the direction b_1 to solve for the value function w. Two constrained-

primary: Shortest path,   b1: Observability by Enemy, b2: Non–observability by Friend
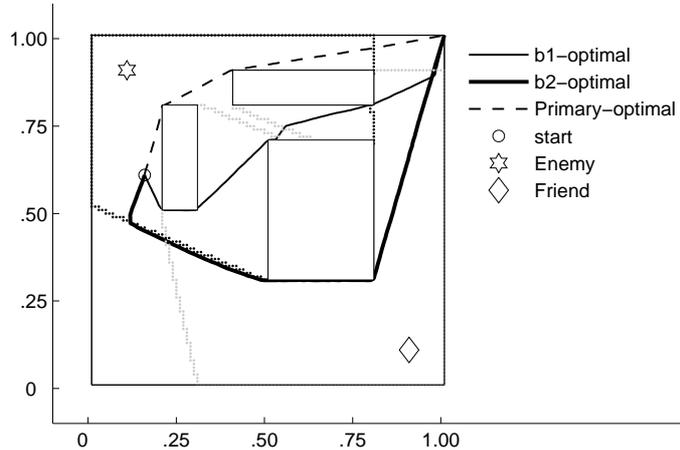
primary: Shortest path,   b1: Observability by Enemy, b2: Non–observability by Friend
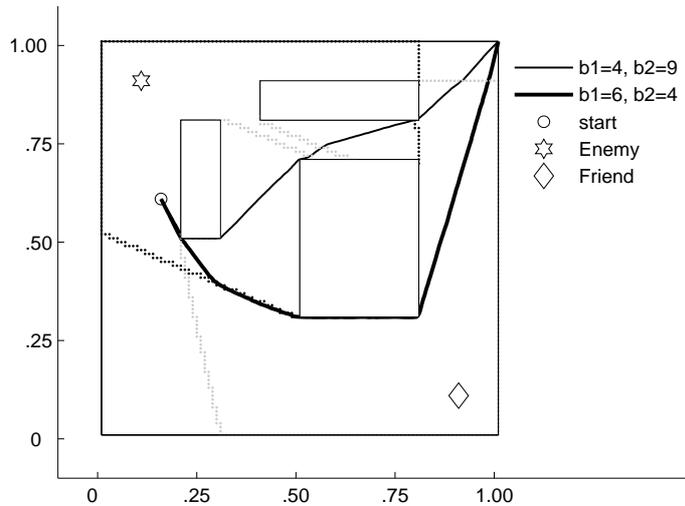
Fig. 4.9. A two-secondary costs example: fuel-optimality under constraints on visibility by enemy and non-visibility by friend. Dotted lines show the boundaries of visibility for both observers. Primary and secondary-optimal trajectories (top) and constrained-optimal trajectories (bottom).

Table 4.5: Numerical parameters for the two-secondary-costs example (Figure 4.9).

Grid points in system-domain	: 101×101
Grid points along each secondary budget	: 301
Speed of the vehicle	: 1
Primary cost	: path length
Secondary cost 1	: visibility to enemy
Secondary cost 2	: invisibility to friend
Visibility cost values: 1 and 10 for both enemy and observer.	

optimal trajectories are shown in Figure 4.9.

4.9. Discussion of computational complexity

Since our semi-Lagrangian discretization of the PDE (3.7) is explicitly causal, the computational complexity of the methods is $\mathcal{O}(M)$, where M is the number of gridpoint on $\bar{\Omega}_e$.

A careful restriction of the problem to a feasible subdomain yields an efficient numerical method. For the case $r = 1$, computations on a 3-dimensional grid are quite fast on an average laptop. We have used Dell Inspiron 1505 laptop with 2 GHz Intel Centrino processor, and 1 GB RAM. On a grid with $M = 201^3$ gridpoints our instrumented (and unoptimized) code took 21, 43, and 39 seconds for Algorithms 1, 2, and 3 of section 3 respectively. For $r = 2$ and $M = 101^4$, Algorithm 2 runs in less than 30 minutes on the same laptop.

In principle, only two b_1-slices of the grid are needed in RAM to enable efficient marching. However, our current implementation allocates the entire grid. As a result, the last example (involving $r = 2$ and $M = 101^2 \times 301^2$ gridpoints and $B_1 = 11$, $B_2 = 9$) was computed on a machine with 64 GB RAM though the memory footprint of the program is ≈ 10 GB. The initialization (by solving for w_1 on a $101^2 \times 301$ grid) took 17 seconds. The Algorithms 1, 2, and 3 then took 27, 67, and 55 minutes respectively.

We note that the computational time is heavily dependent on the values of B_1 and B_2. There are two reasons for this. First, the tighter constraints make a larger part of $\bar{\Omega}_e$ non-feasible, resulting in a big reduction in computational cost for all three algorithms. Second, in Algorithms 2 and 3, τ_a is usually dependent on ratios between h and Δb_i's. This also influences the number of $(n + r)$-dimensional cells traversed before reaching a suitable interpolation point $(\tilde{x}, \tilde{b})$; see section 3.6. E.g., for $r = 1$, when $\Delta b_1 \ll h$ the interpolation is performed after traversing a single cell. Decreasing B_i while holding constant the number of gridpoints in that direction decreases Δb_i proportionally. To illustrate this point, the problem of section 4.8 on the same grid ($M = 101^2 \times 301^2$), but with $B_1 = 4$ and $B_2 = 9$ is much less computationally expensive: Algorithms 1,2, and 3 now take only 20, 27, and 26 minutes respectively on the same computer.

In comparing these execution times to those reported in [25], it is important to keep in mind that Mitchell and Sastry have used an upwind finite-difference discretization of the Eikonal PDE (i.e., isotropic control problems only). In contrast, our semi-Lagrangian implementation is also suitable for much more general (anisotropic and/or non-small-time-controllable) problems, including those, where the minimization in (3.7) cannot be performed analytically.

We also note that our current implementation is simple but non-optimal since the grid is stored as a multidimensional array and all the domain-reduction procedures described in section 3.5 are performed only after the memory for the grid has been allocated. The "excluded" gridpoints are marked (to avoid computing W), but still take some computational time (in enumerations, input/output operations, etc.) A more efficient implementation could be clearly built to allocate the memory for non-excluded gridpoints only, but this would require using non-array data structures to represent the grid.

5. Conclusions

We have introduced a new numerical method for multiobjective optimal control and single-objective optimal control in the presence of integral constraints. Our approach is based on expanding the state space to include constraint-budgets and then solving an augmented PDE,

whose explicit causality allows for a non-iterative (marching) numerical method. We have also shown the connection between the integral-constrained single-objective problem and the task of finding all Pareto-optimal controls. Our method was illustrated with a number of test-problems for two-dimensional optimal control with one and two additional integral constraints. We have used a flight-path bad weather avoidance example introduced in [25] as well as several examples of optimal robotic navigation in the presence of friendly and adversarial observers.

It is a commonly accepted practical rule that lower-dimensional computations are much less expensive than the higher-dimensional ones (simply because the number of gridpoints grows exponentially with the dimension). However, this simple rule of thumb ignores more subtle issues: How many PDEs need to be solved on each domain? How many times do we need to solve each PDE? Does the lower-dimensional approach adequately capture the high-dimensional picture? Which PDE can be solved with a more efficient numerical method?
These questions reflect the difference between our approach and the prior method by Mitchell and Sastry [25]. Their method is based on solving a system of $(r + 2)$ PDEs (all but one of them linear, the remaining non-linear PDE with monotone causality, enabling a Dijkstra-like numerical method) on $\Omega \subset \boldsymbol{R}^n$ but with an r-dimensional parameter space, which requires solving this system repeatedly $\mathcal{O}(2^r)$ times. Our approach leads to a single PDE with explicit causality (enabling time-like marching) but on a $(n + r)$-dimensional domain. Even more importantly, our approach allows recovering the entire relevant part of Pareto front, including the non-convex parts of it, which are inaccessible using the weighted sums method employed in [25].

The explicit causality of the augmented PDE and a careful restriction of the problem to a feasible subdomain yield an efficient numerical method. However, the memory requirements of our method are more extensive since at least two b_1-slices of the grid have to be kept in memory at all times for efficient marching. This is an $(n + r - 1)$-dimensional grid, in contrast with an n-dimensional grid used by the method in [25]. Another disadvantage of our approach is the fact that the local truncation errors are $\mathcal{O}(\hat{h})$ rather than $\mathcal{O}(h)$. As a result, the quality of reconstruction of optimal controls and trajectories also depends on Δb_i's, whereas the method in [25] can provide a good trajectory reconstruction for each λ regardless of coarseness of the mesh imposed on Λ.

In the future we would like to build semi-Lagrangian and Eulerian higher-order accurate methods based on our approach. We also intend to explore the use of adaptive grids and unstructured meshes (since the constrained-optimal controls can be sensitive to small changes in available budgets). It will also be relevant to experiment with the efficiency/accuracy implications of the choice of marching direction − any other b_i can be chosen since all K_i's are assumed to be positive, but the time $\tau_{\boldsymbol{a}}$ in the semi-Lagrangian discretization is currently based on K_1.

Several other natural extensions should be possible without breaking the explicit causality of the augmented PDE. We intend to extend our method to

1. problems, where K_i's do not have to be positive for $i > 1$;
2. problems with running costs (and exit costs) dependent on the budgets still remaining;
3. optimal stochastic control subject to integral constraints;
4. differential games subject to integral constraints.

The theoretical framework for considering secondary costs of varying sign has been developed in [37]. We believe that as long as $K_i > 0$ holds at least for one $i \geq 1$, the computational efficiency will not be adversely affected. If all secondary K_i's (but not K_0) are allowed to change sign, this will require a method based on a more subtle monotone causality in w.

References

[1] M. Bardi and I. Capuzzo Dolcetta, Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations, Birkhäuser Boston, 1997.

[2] M. Bardi & M. Falcone, An approximation scheme for the minimum time function, *SIAM J. Control Optim.*, **28** (1990), 950-965.

[3] M. Bardi, S. Bottacin, and M. Falcone, Convergence of discrete schemes for discontinuous value functions of pursuit-evasion games, in G.J.Olsder (ed.), "New Trends in Dynamic Games and Applications", Birkhauser, (1995), 273-304.

[4] M. Bardi, M. Falcone, and P. Soravia, Fully discrete schemes for the value function of pursuit-evasion games, "Advances in Dynamic Games and Applications", T. Basar and A. Haurie (eds.), Birkhauser, (1994), 89-105.

[5] M. Bardi, M. Falcone, and P. Soravia, Numerical methods for pursuit-evasion games via viscosity solutions in "Stochastic and Differential Games: Theory and Numerical Methods", M. Bardi, T.E.S. Raghavan, and T. Parthasarathy (eds.), Birkhauser, (1999), 105-175.

[6] Bellman, R., Dynamic Programming, Princeton University Press, New Jersey, 1957.

[7] Boué, M. and Dupuis, P., Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control, *SIAM J. Numer. Anal.*, **36**:3 (1999), 667-695.

[8] A. Bressan and B. Piccoli, Introduction to the Mathematical Theory of Control, AIMS on Applied Math., Vol.2, 2007.

[9] E. Carlini, M. Falcone, and R. Ferretti, An efficient algorithm for Hamilton-Jacobi equations in high dimension, *Comput. Visual Sci.*, **7** (2004), 15-29.

[10] L.T. Cheng and Y.H. Tsai, Visibility optimization using variational approaches, *Commun. Math. Sci.*, **3**:3 (2005), 425-451.

[11] M.G. Crandall, L.C. Evans, and P-L.Lions, Some Properties of Viscosity Solutions of Hamilton-Jacobi Equations, *Tran. AMS*, **282** (1984), 487-502.

[12] Crandall, M.G. and Lions, P-L., Viscosity solutions of Hamilton-Jacobi equations, *Tran. AMS*, **277** (1983), 1-43.

[13] Das, I. and Dennis, J.E., A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems, *Struct. Optim.*, **14** (1997), 63-69.

[14] E.W. Dijkstra, A note on two problems in connection with graphs, *Numer. Math.*, **1** (1959), 269-271.

[15] Evans, L.C., Partial Differential Equations, American Mathematical Society, 1998.

[16] M. Falcone, The Minimum Time Problem and Its Applications to Front Propagation, in "Motion by Mean Curvature and Related Topics", Proceedings of the International Conference at Trento, 1992, Walter de Gruyter, New York, 1994.

[17] M. Falcone, A Numerical Approach to the Infinite Horizon Problem of Deterministic Control Theory, *Appl. Math. Opt.*, **15** (1987), 1-13; corrigenda **23** (1991), 213-214.

[18] Gonzales, R. and Rofman, E., On Deterministic Control Problems: an Approximate Procedure for the Optimal Cost, I, the Stationary Problem, *SIAM J. Control Optim.*, **23**:2 (1985), 242-266.

[19] Gremaud, P.A. and Kuster, C.M., Computational Study of Fast Methods for the Eikonal Equation, *SIAM J. Sci. Comput.*, **27** (2006), 1803-1816.

[20] S.-R. Hysing and S. Turek, The Eikonal equation: Numerical efficiency vs. algorithmic complexity on quadrilateral grids, In Proceedings of Algoritmy 2005, 22-31, 2005.

[21] Isaacs, R., Differential games. A mathematical theory with applications to warfare and pursuit, control and optimization, John Wiley & Sons Inc., New York, 1965.

[22] Kim, J. and Hespanha, J.P., Discrete approximations to continuous shortest-path: application to minimum-risk path planning for groups of UAVs, Proceedings of the 42nd IEEE Conference on Decision and Control (9-12 Dec., 2003), **2** (2003), 1734-1740.

[23] H.J. Kushner and P.G. Dupuis, Numerical Methods for Stochastic Control Problems in Continuous Time, Academic Press, New York, 1992.

[24] R.T. Marler and J.S. Arora, Survey of multi-objective optimization methods for engineering, *Struct. Mutidiscip. Optim.*, **26**:6 (2004), 369-395.

[25] Mitchell, I.M. and Sastry, S., Continuous path planning with multiple constraints, Proceedings of the 42nd IEEE Conference on Decision and Control, 5502-5507, 2003.

[26] M. Motta and F. Rampazzo, Multivalued dynamics on a closed domain with absorbing boundary. Applications to optimal control problems with integral constraints, *Nonlinear Anal-Theor.*, **41**:5-6 (2000), 631-647.

[27] S. Osher, A level set formulation for the solution of the Dirichlet problem for Hamilton-Jacobi equations, *SIAM J. Math. Anal.*, **24** (1993), 1145-1152.

[28] Qian, J. and Symes, W.W., Paraxial Eikonal solvers for anisotropic Quasi-P traveltimes, *J. Comput. Phys.* **173** (2001), 256-278.

[29] J.A. Sethian, A Fast Marching Level Set Method for Monotonically Advancing Fronts, *Proc. Nat. Acad. Sci.*, **93**:4 (1996), 1591-1595.

[30] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences, Cambridge University Press, 1996.

[31] D. Adalsteinsson and J. A. Sethian, The fast construction of extension velocities in level set methods, *J. Comput. Phys.*, **148** (1999), 2.22.

[32] Fomel, S. and Sethian, J.A., Fast-phase space computation of multiple arrivals, *Proc. Nat. Acad. Sci.*, **99** (2002), 7329-7334.

[33] J.A. Sethian and A. Vladimirsky, Fast Methods for the Eikonal and Related Hamilton–Jacobi Equations on Unstructured Meshes, *Proc. Nat. Acad. Sci.*, **97**:11 (2000), 5699-5703.

[34] J.A. Sethian and A. Vladimirsky, Ordered Upwind Methods for Static Hamilton-Jacobi Equations, *Proc. Nat. Acad. Sci.*, **98**:20 (2001), 11069-11074.

[35] J.A. Sethian and A. Vladimirsky, Ordered upwind methods for static Hamilton-Jacobi equations: theory & algorithms, *SIAM J. Numer. Anal.*, **41**:1 (2003), 325-363.

[36] Soner, M.H. (1986) Optimal control problems with state-space constraints, *SIAM J. Control Optim.*, **24**, 552-562 and 1110-1122.

[37] Soravia, P., Viscosity solutions and optimal control problems with integral constraints, *Systems Control Lett.*, **40** (2000), 325-335.

[38] Soravia, P., Boundary value problems for Hamilton-Jacobi equations with discontinuous Lagrangian, *Indiana U. Math. J.*, **51**:2 (2002), 451-477.

[39] Tsai, Y.-H.R., Cheng, L.-T., Osher, S., & Zhao, H.-K., Fast sweeping algorithms for a class of

Hamilton-Jacobi equations, *SIAM J. Numer. Anal.*, **41**:2 (2003), 659-672.

[40] J.N. Tsitsiklis,   Efficient algorithms for globally optimal trajectories, Proceedings, IEEE 33rd Conference on Decision and Control, pp. 1368–1373, Lake Buena Vista, Florida, December 1994.

[41] A. Vladimirsky,   Fast Methods for Static Hamilton-Jacobi Partial Differential Equations, Ph.D. Dissertation, Dept. of Mathematics, Univ. of California, Berkeley, 2001.

[42] A. Vladimirsky,   Static PDEs for time-dependent control problems, *Interfaces Free Bound.*, **8**:3 (2006), 281-300.

[43] Zhao, H.K.,   Fast sweeping method for Eikonal equations, *Math. Comput.*, **74** (2005), 603-627.