Optimal Control Using Bisimulations: Implementation

Mireille Broucke[†], Maria Domenica Di Benedetto[‡†], Stefano Di Gennaro[‡], and Alberto Sangiovanni-Vincentelli[†]

[†] Dept. of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA 94720, USA. Tel: +11 510 642-1792; Fax: +11 510 643-5052;
[‡] Dip. di Ingegneria Elettrica, Università di L'Aquila, Poggio di Roio, 67040 L'Aquila, Italy.
mire,marika,alberto@eecs.berkeley.edu,digennar@dis.uniroma1.it

Abstract. We consider the synthesis of optimal controls for continuous feedback systems by recasting the problem to a hybrid optimal control problem which is to synthesize optimal enabling conditions for switching between locations in which the control is constant. We provide a single-pass algorithm to solve the dynamic programming problem that arises, with added constraints to ensure non-Zeno trajectories.

1 Introduction

In this paper we continue our investigation of the application of hybrid systems and bisimulation to optimal control problems. In the first paper [3] we developed a discrete method for solving an optimal control problem based on hybrid systems and bisimulation. We showed that the value function of the discrete problem converges to the value function of the continuous problem as a discretization parameter δ tends to zero. In this paper we focus on the pragmatic question of how the discretized problem can be efficiently solved.

Following the introduction of the concept of viscosity solution [11, 5], Capuzzo-Dolcetta [4] introduced a method for obtaining approximations of viscosity solutions based on time discretization of the Hamilton-Jacobi-Bellman (HJB) equation. The approximations of the value function correspond to a discrete time optimal control problem, for which an optimal control can be synthesized that is piecewise constant. Finite difference approximations were also introduced in [6] and [13]. In general, the time discretized approximation of the HJB equation is solved by finite element methods. Gonzales and Rofman [10] introduced a discrete approximation by triangulating the domain of the finite horizon problem they considered, while the admissible control set is approximated by a finite set. Gonzales and Rofman's approach is adapted in several papers, including [8]. The approach of [14] uses the special structure of an optimal control problem to obtain a single-pass algorithm to solve the discrete problem, thus bypassing the expensive iterations of a finite element method. The essential property needed to find a single pass algorithm is to obtain a partition of the domain so that the cost-to-go value from any equivalence class of the partition is determined from knowledge of the cost-to-go from those equivalence classes with strictly smaller cost-to-go values. In this paper we obtain a partition of the domain provided by a bisimulation partition. *The combination of the structure of the bisimulation partition and the requirement of non-Zeno trajectories enables us reproduce the essential property of [14], so that we obtain a Dijkstra-like algorithmic solution.* Our approach has complexity $O(N \log N)$ if suitable data structures are used, where N is the number of locations of the finite automaton.

While the objective is to solve a continuous optimal control problem, the method can be adapted to solve directly the problem of optimal synthesis of enabling conditions for hybrid systems. In that spirit, [1] investigates games on timed automata and obtains a dynamic programming formulation as well.

2 Optimal control problem

$cl(A)$ denotes the closure of set A. $\|\cdot\|$ denotes the Euclidean norm. $\mathcal{X}(\mathbb{R}^n)$ denotes the sets of smooth vector fields on $\mathbb{R}^n$. $\phi_t(x_0, \mu)$ denotes the trajectory of $\dot{x} = f(x, \mu)$ starting from x_0 and using control $\mu(\cdot)$.

Let U be a compact subset of $\mathbb{R}^m$, Ω an open, bounded, connected subset of $\mathbb{R}^n$, and Ω_f a compact subset of Ω. Define $\mathcal{U}_m$ to be the set of measurable functions mapping $[0, T]$ to U. We define the minimum hitting time $T : \mathbb{R}^n \times \mathcal{U}_m \to \mathbb{R}^+$ by

$$T(x, \mu) := \begin{cases} \infty & \text{if } \{t \mid \phi_t(x, \mu) \in \Omega_f \ \} = \emptyset \\ \min\{t \mid \phi_t(x, \mu) \in \Omega_f\} & \text{otherwise.} \end{cases} \tag{1}$$

A control $\mu \in \mathcal{U}_m$ specified on $[0, T]$ is *admissible* for $x \in \Omega$ if $\phi_t(x, \mu) \in \Omega$ for all $t \in [0, T]$. The set of admissible controls for x is denoted $\mathcal{U}_x$. Let $\mathcal{R} := \{ \ x \in \Omega \mid \exists \mu \in \mathcal{U}_x .\ T(x, \mu) < \infty \ \}$. We consider the following optimal control problem. Given $y \in \Omega$,

$$\text{minimize} \qquad J(y, \mu) = \int_0^{T(y,\mu)} L(x(s), \mu(s)) ds + h(x(T(y, \mu))) \tag{2}$$

$$\text{subject to} \qquad \dot{x} = f(x, \mu), \qquad a.e.\ t \in [0, T(y, \mu)] \tag{3}$$

$$x(0) = y \tag{4}$$

among all admissible controls $\mu \in \mathcal{U}_y$. $J : \mathbb{R}^n \times \mathcal{U}_m \to \mathbb{R}$ is the *cost-to-go* function, $h : \mathbb{R}^n \to \mathbb{R}$ is the *terminal cost*, and $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the *instantaneous cost*. At $T(y, \mu)$ the terminal cost $h(x(T(y, \mu)))$ is incurred and the dynamics are stopped. The control objective is to reach Ω_f from $y \in \Omega$ with minimum cost.

The *value function* or optimal cost-to-go function $V : \mathbb{R}^n \to \mathbb{R}$ is given by

$$V(y) = \inf_{\mu \in \mathcal{U}_y} J(y, \mu)$$

for $y \in \Omega \setminus \Omega_f$, and by $V(y) = h(y)$ for $y \in \Omega_f$. V satisfies the *Hamilton-Jacobi-Bellman* equation

$$-\inf_{u \in U} \Big\{ L(x, u) + \frac{\partial V}{\partial x} f(x, u) \Big\} = 0 \tag{5}$$

at each point of $\mathcal{R}$ at which it is differentiable. The HJB equation is an infinitesimal version of the equivalent *Dynamic Programming Principle* (DPP) which says that

$$\begin{aligned} V(x) &= \inf_{\mu \in \mathcal{U}_x} \left\{ \int_0^t L(\phi_s(x,\mu),\mu(s))ds \ + \ V(\phi_t(x,\mu)) \right\}, \ x \in \Omega \setminus \Omega_f \\ V(x) &= h(x) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad x \in \Omega_f. \end{aligned}$$

Because the HJB equation may not have a C^1 solution it has not been possible to obtain a rigorous foundation for solutions in the usual sense. The correct concept for solutions is that of viscosity solutions [11, 5], which provide the unique solution of (5) without differentiability. We showed in [3] that under assumptions of Lipschitz continuity of f,L, and h, and non-Zenoness and transversality with Ω_f of ϵ-optimal trajectories, that a particular discrete approximation $\hat{V}$ of the value function converges to the viscosity solution of HJB.

3 From hybrid automata to finite automata

In [3] we proposed a mapping from the continuous optimal control problem (2)-(4) to a hybrid optimal control problem. The first step is to restrict the class of controls over which the cost function is minimized to piecewise constant controls taking values in a set $\Sigma_\delta \subseteq U$. $\Sigma_\delta \subseteq U$ is a finite approximation of U having a mesh size $\delta := \sup_{u \in U} \min_{\sigma \in \Sigma_\delta} \|u - \sigma\|$. Next we restrict the continuous behavior to the set of vector fields $\{f(x,\sigma)\}_{\sigma \in \Sigma_\delta}$. If we associate each vector field to a location of a hybrid automaton and, additionally, define a location reserved for when the target is reached, we obtain a hybrid automaton

$$H := (\Sigma \times \mathbb{R}^n, \Sigma_\delta, D, E_h, G, R)$$

which has the following components:

State set $\Sigma \times \mathbb{R}^n$ is a finite set $\Sigma = \Sigma_\delta \cup \{\sigma_f\}$ of control locations and n continuous variables $x \in \mathbb{R}^n$. σ_f is a terminal location when the continuous dynamics are stopped (in the same sense that the dynamics are stopped in the continuous optimal control problem).

Events Σ_δ is a finite set of control event labels.

Vector fields $D : \Sigma \to \mathcal{X}(\mathbb{R}^n)$ is a function assigning an autonomous vector field to each location; namely $D(\sigma) = f(x,\sigma)$.

Control switches $E_h \subset \Sigma \times \Sigma$ is a set of control switches. $e = (\sigma, \sigma')$ is a directed edge between a source location σ and a target location σ'. If $E_h(\sigma)$ denotes the set of edges that can be enabled at $\sigma \in \Sigma$, then $E_h(\sigma) := \{(\sigma,\sigma') \mid \sigma' \in \Sigma \setminus \sigma\}$ for $\sigma \in \Sigma_\delta$ and $E_h(\sigma_f) = \emptyset$. Thus, from a source location not equal to σ_f, there is an edge to every other location (but not itself), while location σ_f has no outgoing edges.

Enabling conditions $G : E_h \to \{g_e\}_{e \in E_h}$ is a function assigning to each edge e an enabling (or guard) condition $g_e \subset \mathbb{R}^n$.

The enabling conditions are unknown and must be synthesized algorithmically. (See [3] for how the enabling conditions are extracted once the discrete problem is solved.) Trajectories of H evolve in σ-steps and t-steps. σ-steps occur when H changes locations (and the control changes value, since there are no self-loops) and t-steps occur when the continuous state evolves according to the dynamics of a location as time passes. The reader is referred to [3] for precise statements. A hybrid trajectory is *non-Zeno* if between every two non-zero duration t-steps there are a finite number of σ-steps and zero duration t-steps.

Let λ represent an arbitrary time interval. A *bisimulation* of H is an equivalence relation $\simeq \subset (\Sigma_\delta \times \mathbb{R}^n) \times (\Sigma_\delta \times \mathbb{R}^n)$ such that for all states $p_1, p_2 \in \Sigma_\delta \times \mathbb{R}^n$, if $p_1 \simeq p_2$ and $\sigma \in \Sigma_\delta \cup \{\lambda\}$, then if $p_1 \xrightarrow{\sigma} p_1'$, there exists p_2' such that $p_2 \xrightarrow{\sigma} p_2'$ and $p_1' \simeq p_2'$.

One sees that $\simeq$ encodes σ-steps and t-steps of H in a time abstract form by partitioning $\Sigma_\delta \times \mathbb{R}^n$. If $\simeq$ has a finite number of equivalence classes, then they form the states of a finite automaton A. If $q := [(\sigma, x)]$ and $q' := [(\sigma', x')]$ are two different equivalence classes of $\simeq$, then A has an edge $q \to q'$ if there exists $(\sigma, y) \in q$ and $(\sigma', y') \in q'$ such that $(\sigma, y) \to (\sigma', y')$ is a σ-step or t-step of H. We define the set of interesting equivalence classes of $\simeq$, denoted Q, as those that intersect $\Sigma_\delta \times cl(\Omega)$, and we identify a distinguished point $(\sigma, \xi) \in q$ for each $q \in Q$, denoted $q = [(\sigma, \xi)]$.

Consider the class of non-deterministic automata with cost structure represented by the tuple

$$A = (Q, \Sigma_\delta, E, obs, Q_f, \hat{L}, \hat{h}).$$

Q is the state set just defined, and Σ_δ is the set of control labels as before. $obs : E \to \Sigma_\delta$ is a map that assigns a control label to each edge and is given by $obs(e) = \sigma'$, where $e = (q, q')$, $q = [(\sigma, \xi)]$ and $q' = [(\sigma', \xi')]$. Q_f is an over (or under) approximation of Ω_f, $Q_f = \{q \in Q \mid \exists x \in \Omega_f \ . \ (\sigma, x) \in q \ \}$. $E \subseteq Q \times Q$ is the transition relation of A and is defined assuming that each enabling condition is initially the entire region Ω. The identity map is implemented in A by an over-approximation in terms of equivalence classes of $\simeq$. That is, for $\sigma \neq \sigma'$, $([\sigma, x)], [(\sigma', x')]) \in E$ if the projections to $\mathbb{R}^n$ of $[\sigma, x)]$ and $[(\sigma', x')]$ have non-empty intersection. This over-approximation introduces non-determinacy in A. Let

$$\tau_q = \sup_{(\sigma,x),(\sigma,y) \in q} \{ \ t \mid y = \phi_t(x, \sigma) \ \}.$$

Let $e = (q, q')$ with $q = [(\sigma, \xi)]$ and $q' = [(\sigma', \xi')]$. $\hat{L} : E \to \mathbb{R}$ is the *discrete instantaneous cost* given by

$$\hat{L}(e) := \begin{cases} \tau_q L(\xi, \sigma) & \text{if} \quad \sigma = \sigma' \\ 0 & \text{if} \quad \sigma \neq \sigma'. \end{cases} \qquad (6)$$

$\hat{h} : Q \to \mathbb{R}$ is the *discrete terminal cost* given by

$$\hat{h}(q) := h(\xi).$$

A transition or *step* of A from $q \in Q$ to $q' \in Q$ with observation $\sigma' \in \Sigma_\delta$ is denoted $q \xrightarrow{\sigma'} q'$. If $\sigma \neq \sigma'$ the transition is referred to as a *control switch*, and it is forced. $\sigma = \sigma'$ the transition is referred to as a *time step*. If $E(q)$ is the set of edges that can be enabled from $q \in Q$, then for $\sigma \in \Sigma_\delta$, $E_\sigma(q) = \{e \in E(q) \mid obs(e) = \sigma\}$. If $|E_\sigma(q)| > 1$, then we say that $e \in E_\sigma(q)$ is *unobservable* in the sense that when control event σ is issued, it is unknown which edge among $E_\sigma(q)$ is taken. (Note that unobservability of edges refers strictly to the discrete automaton A, whereas in H one may be able to reconstruct which edge was taken using continuous state information). If $\sigma = \sigma'$, then $|E_\sigma(q)| = 1$, by the uniqueness of solutions of ODE's and by the definition of bisimulation.

A *control policy* $c : Q \to \Sigma_\delta$ is a map assigning a control event to each state; $c(q) = \sigma$ is the control event issued when the state is at q. A *trajectory* π of A over c is a sequence $\pi = q_0 \xrightarrow{\sigma_1} q_1 \xrightarrow{\sigma_2} q_2 \xrightarrow{\sigma_3} \ldots$, $q_i \in Q$. Let $\Pi_c(q)$ be the set of trajectories starting at q and applying control policy c, and let $\tilde{\Pi}_c(q)$ be the set of trajectories starting at q, applying control policy c, and eventually reaching Q_f. If for every $q \in Q$, $\pi \in \Pi_c(q)$ is non-Zeno then we say c is an *admissible control policy*. The set of all admissible control policies for A is denoted $\mathcal{C}$.

A control policy c is said to have a *loop* if A has a trajectory $q_0 \xrightarrow{c(q_0)} q_1 \xrightarrow{c(q_1)} \ldots \xrightarrow{c(q_{m-1})} q_m = q_0$, $q_i \in Q$. A control policy has a *Zeno loop* if it has a loop made up of control switches and/or zero duration time steps (i.e. $\tau_q = 0$) only.

Lemma 1. *A control policy c for non-deterministic automaton A is admissible if and only if it has no Zeno loops.*

Proof. First we show that a non-deterministic automaton with non-Zeno trajectories has a control policy without Zeno loops. For suppose not. Then a trajectory starting on a state belonging to the loop can take infinitely many steps around the loop before taking a non-zero duration time step. This trajectory is not non-Zeno, a contradiction. Second, we show that a control policy without Zeno loops implies non-Zeno trajectories. Suppose not. Consider a Zeno trajectory that takes an infinite number of control switches and/or zero duration time steps between two non-zero duration time steps. Because there are a finite number of states in Q, by the Axiom of Choice, one of the states must be repeated in the sequence of states visited during the control switches and/or zero duration time steps. This implies the existence of a loop in the control policy. Either each step of the loop is a control switch, implying a Zeno loop; or the loop has one or more zero duration time steps. But the bisimulation partition permits zero duration time steps only if $\tau_q = 0$, which implies a Zeno loop. □

Example 1. Consider the automaton in Figure 1. If we are at q_1 and the control $\sigma'\sigma'\sigma$ is issued, then three possible trajectories are $q_1 \xrightarrow{\sigma'} q_3 \xrightarrow{\sigma'} q_4 \xrightarrow{\sigma} q_2$, $q_1 \xrightarrow{\sigma'} q_4 \xrightarrow{\sigma'} q_5 \xrightarrow{\sigma} q_2$, or $q_1 \xrightarrow{\sigma'} q_3 \xrightarrow{\sigma'} q_4 \xrightarrow{\sigma} q_1$. The first trajectory has a zero duration time step. The control is inadmissible since the last trajectory has a Zeno loop.

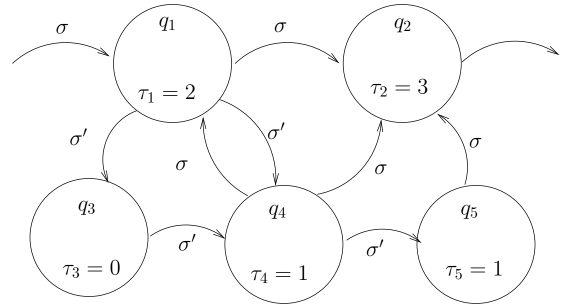


Fig. 1. Fragment of automaton with a zero duration time step.

4 Dynamic programming

In this section we formulate the dynamic programming problem on A. This involves defining a cost-to-go function and a value function that minimizes it over control policies suitable for non-deterministic automata.

Let $\pi = q_0 \overset{\sigma_1}{\to} q_1 \dots q_{N-1} \overset{\sigma_N}{\to} q_N$, where $q_i = [(\sigma_i, \xi_i)]$ and π takes the sequence of edges $e_1 e_2 \dots e_N$. We define a *discrete cost-to-go* $\hat{J} : Q \times \mathcal{C} \to \mathbb{R}$ by

$$\hat{J}(q,c) = \begin{cases} \max_{\pi \in \tilde{\Pi}_c(q)} \left\{ \sum_{j=1}^{N_\pi} \hat{L}(e_j) + \hat{h}(q_{N_\pi}) \right\} & \text{if} \quad \Pi_c(q) = \tilde{\Pi}_c(q) \\ \infty & \text{otherwise} \end{cases}$$

where $N_\pi = \min\{j \geq 0 \mid q_j \in Q_f\}$. We take the maximum over $\tilde{\Pi}_c(q)$ because of the non-determinacy of A: it is uncertain which among the (multiple) trajectories allowed by c will be taken so we must assume the worst-case situation. The *discrete value function* $\hat{V} : Q \to \mathbb{R}$ is

$$\hat{V}(q) = \min_{c \in \mathcal{C}} \hat{J}(q,c)$$

for $q \in Q \setminus Q_f$ and $\hat{V}(q) = \hat{h}(q)$ for $q \in Q_f$. We showed in [3] that $\hat{V}$ satisfies a DPP that takes into account the non-determinacy of A and ensures that optimal control policies are admissible. Let $\mathcal{A}_q$ be the set of control assignments $c(q) \in \Sigma_\delta$ at q such that c is admissible.

Proposition 1. *$\hat{V}$ satisfies*

$$\hat{V}(q) = \min_{c(q) \in \mathcal{A}_q} \left\{ \max_{e=(q,q') \in E_{\sigma'}(q)} \left\{ \hat{L}(e) + \hat{V}(q') \right\} \right\}, \quad q \in Q \setminus Q_f \tag{7}$$

$$\hat{V}(q) = \hat{h}(q), \qquad q \in Q_f. \tag{8}$$

5 Non-deterministic Dijkstra algorithm

The dynamic programming solution (7)-(8) can be viewed as a shortest path problem on a non-deterministic graph subject to all optimal paths satisfying a non-Zeno condition. We propose an algorithm which is a modification of the Dijkstra algorithm for deterministic graphs [7]. First we define notation. F_n is the set of states that have been assigned a control and are deemed "finished" at iteration n, while U_n are the unfinished states. At each n, $Q = U_n \cup F_n$. $\Sigma_n(q) \subseteq \Sigma_\delta$ is the set of control events at iteration n that take state q to finished states exclusively. $\tilde{U}_n$ is the set of states for which there exists a control event that can take them to finished states exclusively. $\tilde{V}_n(q)$ is a tentative cost-to-go value at iteration n. B_n is the set of "best" states among $\tilde{U}_n$.

The non-deterministic Dijkstra (NDD) algorithm first determines $\tilde{U}_n$ by checking if any q in U_n can take a step to states belonging exclusively to F_n. For states belonging to $\tilde{U}_n$, an estimate of the value function $\tilde{V}$ following the prescription of (7) is obtained: among the set of control events constituting a step into states in F_n, select the event with the lowest worst-case cost. Next, the algorithm determines B_n, the states with the lowest $\tilde{V}$ among $\tilde{U}_n$, and these are added to F_{n+1}. The iteration counter is incremented until it reaches $N = |Q|$. It is assumed in the following description that initially $\hat{V}(q) = \infty$ and $c(q) = \emptyset$ for all $q \in Q$.

Procedure NDD:

$\mathsf{F_1 = Q_f;\ U_1 = Q - Q_f;}$
for each $\mathsf{q \in Q_f}$, $\mathsf{\hat{V}(q) = \hat{h}(q)}$;

for $\mathsf{n = 1}$ to N, do
    for each $\mathsf{q \in U_n}$,
        $\mathsf{\Sigma_n(q) = \{\sigma' \in \Sigma_\delta \mid \text{if } q \xrightarrow{\sigma'} q', \text{then } q' \in F_n\}}$;
    $\mathsf{\tilde{U}_n = \{q \in U_n \mid \Sigma_n(q) \neq \emptyset\}}$;
    for each $\mathsf{q \in \tilde{U}_n}$,
        $\mathsf{\tilde{V}_n(q) = \min_{\sigma' \in \Sigma_n(q)}\{\max_{e=(q,q') \in E_{\sigma'}(q)}\{\hat{L}(e) + \hat{V}(q')\}\}}$;
    $\mathsf{B_n = \text{argmin}_{q \in \tilde{U}_n}\{\tilde{V}_n(q)\}}$;
    for each $\mathsf{q \in B_n}$,
        $\mathsf{\hat{V}(q) = \tilde{V}_n(q)}$;
        $\mathsf{c(q) = \text{argmin}_{\sigma' \in \Sigma_n(q)}\{\max_{e=(q,q') \in E_{\sigma'}(q)}\{\hat{L}(e) + \hat{V}(q')\}\}}$;
    endfor
    $\mathsf{F_{n+1} = F_n \cup B_n;\ U_{n+1} = Q - F_{n+1}}$;
endfor

We prove that algorithm NDD is *optimal*; that is, it synthesizes a control policy so that each $q \in Q$ reaches Q_f with the best worst-case cost. We observe a few properties of the algorithm. First, if all states of Q can reach Q_f then $Q - Q_f = \cup_n B_n$. Second, as in the deterministic case, the algorithm computes $\hat{V}$ in order of level sets of $\hat{V}$. In particular, $\hat{V}(B_n) \leq \hat{V}(B_{n+1})$. Finally, we need the following property.

Lemma 2. *For all $q \in Q$ and $\sigma' \in \Sigma_\delta$,*

$$\hat{V}(q) \leq \max_{e=(q,q')\in E_{\sigma'}(q)} \{\hat{L}(e) + \hat{V}(q')\}.$$

Proof. Fix $q \in Q$ and $\sigma' \in \Sigma_\delta$. There are two cases.
Case 1.

$$\hat{V}(q) \leq \max_{e=(q,q')\in E_{\sigma'}(q)} \{\hat{V}(q')\}.$$

In this case the result is obvious.
Case 2.

$$\hat{V}(q) > \max_{e=(q,q')\in E_{\sigma'}(q)} \{\hat{V}(q')\}. \tag{9}$$

We observed above that q belongs to some B_n. Suppose w.l.o.g. that $q \in B_j$. Together with (9) this implies $q' \in F_j$ for all q' such that $q \xrightarrow{\sigma'} q'$. This, in turn, means that $\sigma' \in \Sigma_j(q)$ and according to the algorithm

$$\hat{V}(q) = \tilde{V}_j(q) \leq \max_{e=(q,q')\in E_{\sigma'}(q)} \{\hat{L}(e) + \hat{V}(q')\}$$

which proves the result. □

Theorem 1. *Algorithm NDD is optimal and synthesizes a control policy with no Zeno loops.*

Proof. First we prove optimality. Let $V(q)$ be the optimal (best worst-case) cost-to-go for $q \in Q$ and $\overline{Q} = \{q \in Q \mid V(q) < \hat{V}(q)\}$. Let $l(\pi_q)$ be the number of edges taken by the shortest optimal (best worst-case) trajectory π_q from q. Define $\overline{q} = \arg\min_{q\in\overline{Q}}\{l(\pi_q)\}$. Suppose that the best worst-case trajectory starting at $\overline{q}$ is $\pi_{\overline{q}} = \overline{q} \xrightarrow{\sigma'} \overline{\overline{q}} \to \ldots$. We showed in the previous lemma that

$$\hat{V}(\overline{q}) \leq \max_{e=(\overline{q},q')\in E_{\sigma'}(\overline{q})} \{\hat{L}(e) + \hat{V}(q')\} \leq \hat{L}(e) + \hat{V}(\overline{\overline{q}}).$$

Since $\pi_{\overline{q}}$ is the best worst-case trajectory from $\overline{q}$ and by the optimality of $V(\overline{q})$

$$V(\overline{q}) = \max_{e=(\overline{q},q')\in E_{\sigma'}(\overline{q})} \{\hat{L}(e) + V(q')\} = \hat{L}(e) + \hat{V}(\overline{\overline{q}}).$$

Since $\pi_{\overline{q}}$ is the shortest best worst-case trajectory, we know that $\overline{\overline{q}} \notin \overline{Q}$, so $V(\overline{\overline{q}}) = \hat{V}(\overline{\overline{q}})$. This implies $\hat{V}(\overline{q}) \leq \hat{L}(e) + V(\overline{\overline{q}}) = V(\overline{q})$, a contradiction.

To prove that the algorithm synthesizes a policy with no Zeno loops we argue by induction. The claim is obviously true for F_1. Suppose that the states of F_n have been assigned controls forming no Zeno loops. Consider F_{n+1}. Each state of B_n takes either a time step or a control switch to F_n so there cannot be a Zeno loop in B_n. The only possibility is for some $q \in B_n$ to close a Zeno loop with states in F_n. This implies there exists a control assignment that allows an edge from F_n to q to be taken; but this is not allowed by NDD. Thus, F_{n+1} has no Zeno loops. □

Remarks:

1. It is intuitively reasonable that the algorithm cannot synthesize a controller with Zeno loops. This worst-case behavior would show up in the value function, forcing it to be infinite for states that can reach the loop.
2. When we say that the algorithm is optimal, we mean the algorithm determines the best worst-case cost to take each state to the target set. In fact, (see remark below) the hybrid system or continuous system using the synthesized controller may perform better than worst case.
3. The non-deterministic automaton predicts more trajectories than what either the continuous system or the hybrid system can exhibit. Indeed, the automaton may exhibit a trajectory that reaches the target set using only control switches, and thus accruing zero cost. This is not of concern. Such a trajectory is an artifact of the non-determinacy of the automaton, and is not used in the determination of the value function, which accounts only for worst-case behavior, nor is it exhibited in either the hybrid system or the continuous system when the control policy synthesized by Algorithm NDD is used.
4. Related to the previous remark is that the non-deterministic automaton may also predict worst-case behavior which is not exhibited by the continuous system. It would appear that a discrepancy will develop between the cost-to-go obtained by applying the synthesized controller to the continuous system and the cost-to-go predicted by the nondeterministic automaton. This error is incurred every time a control switch is taken and is effectively an error in predicting the state and has an upper bound of δ at each iteration. This error was accounted for in our proof of convergence of the method, and the convergence result essentially depends on the fact that only a finite number of control switches occur [3].

6 Example

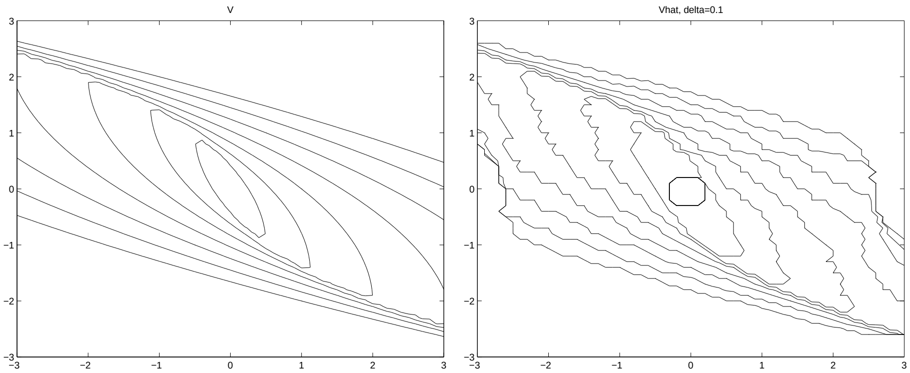
We apply our method to the time optimal control problem of a double integrator

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = u.$$

Given the set of admissible controls $U = \{u \ : \ |u| \leq 1\}$, we select $\Omega = (-1,1) \times (-1,1)$ and $\Omega_f = \overline{B}_\epsilon(0)$, the closed epsilon ball centered at 0. The cost-to-go function is $J(x,\mu) = \int_0^{T(x,\mu)} dt$. The bang-bang solution obtained using Pontryagin's maximum principle is well known to involve a single switching curve. The continuous value function V is shown in Figure 2(a).

To construct the hybrid automaton H we select $\Sigma_\delta = \{-1,1\}$. H is show in Figure 3. The state space is $\{\sigma_{-1} = -1, \sigma_1 = 1, \sigma_f\} \times \mathbb{R}^n$. $g_{e_{-1}}$ and g_{e_1} are unknown and must be synthesized, while $g_{e_2} = g_{e_3} = \Omega_f$.

A first integral for vector field $\dot{x}_1 = x_2$, $\dot{x}_2 = 1$ is $x_1 - \frac{1}{2}x_2^2 = c_1$, $c_1 \in \mathbb{R}$. For $\dot{x}_1 = x_2$, $\dot{x}_2 = -1$ a first integral is $x_1 + \frac{1}{2}x_2^2 = c_2$, $c_2 \in \mathbb{R}$. We select a transverse foliation (see [2]) for each vector field, given by $x_2 = c_3$.



(a) V

(b) $\hat{V}$ for $\varDelta = 0.1$.

Fig. 2. Continuous and discrete value functions for double integrator

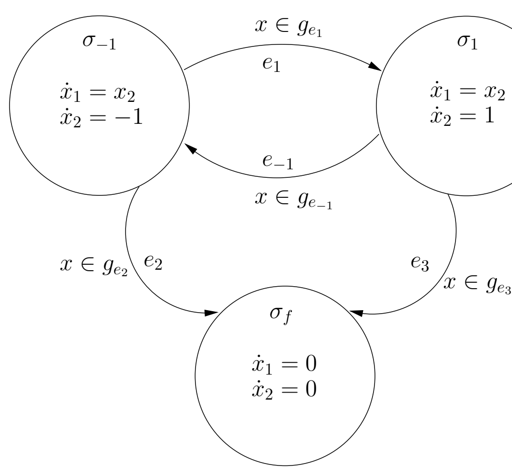


Fig. 3. Hybrid automaton for time optimal control of a double integrator system

We define Q, Q_f, E, $\hat{L}$ and $\hat{h}$ for automaton A derived from H in Figure 3. Q can be visualized using Figure 4.

The states $q \in Q$ are of the form $(\sigma, [x])$ with $\sigma \in \{\sigma_{-1}, \sigma_1\}$. For the case $\sigma = \sigma_1$ with $c_1, c_2 \in \mathbb{R}$, $[x]$ is either an open subset of $\mathbb{R}^2$ bounded by the leaves $c_1 < x_1 - \frac{1}{2}x_2^2 < c_1 + \Delta$ and $c_2 < x_2 < c_2 + \Delta$; or an open interval in a horizontal leaf $x_1 - \frac{1}{2}x_2^2 = c_1$, $c_2 < x_2 < c_2 + \Delta$; or an open interval in a vertical leaf $c_1 < x_1 - \frac{1}{2}x_2^2 < c_1 + \Delta$, $x_2 = c_2$; or a point $x_1 - \frac{1}{2}x_2^2 = c_1$, $x_2 = c_2$. Analogous expressions can be written for $\sigma = \sigma_{-1}$. In Figure 4, $\Delta = 0.25$, $c_1 \in [-1, 1]$ and $c_2 \in [-1, 1]$. If we identify equivalence classes $(\sigma, [x])$ by their Euclidean coordinates (c_1, c_2) directly, then Q_f, shown in Figure 4 as the regions inside the dotted lines, includes states $(\sigma, [x])$, where $[x]$ satisfies $c_1, c_2 \in (-\Delta, \Delta)$.

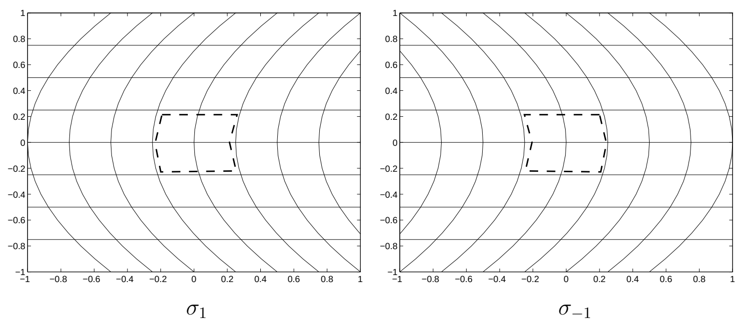
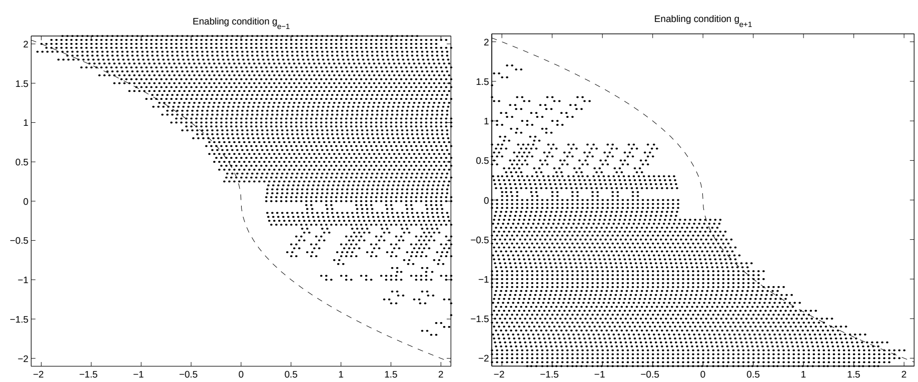


Fig. 4. Partitions for states σ_1 and σ_{-1} of the hybrid automaton of Figure 3

Let us consider the edges corresponding to control switches of A. $q = (\sigma_1, [x]) \in Q$ has an outgoing edge to $q' = (\sigma_{-1}, [y]) \in Q$ if $[x] \cap [y] \neq \emptyset$. For example, for $q = (\sigma_1, [x])$ and $[x]$ satisfying $c_1 \in (-.25, -.5)$ and $c_2 = .25$, there are three outgoing edges from q to q_i',$i = 1, \ldots, 3$, with $[y]$ satisfying $c_2 = .25$ and $c_1 \in (-.5, -.25)$, $c_1 = -.25$, and $c_1 \in (-.25, 0)$, respectively. Similarly, for $q = (\sigma_1, [x])$ and $[x]$ satisfying $c_1 \in (-.5, -.25)$ and $c_2 \in (.75, 1)$, there are five outgoing edges from q to q_i',$i = 1, \ldots, 5$, with $[y]$ satisfying $c_2 \in (.75, 1)$ and $c_1 \in (-.25, 0)$, $c_1 = 0$, $c_1 \in (0, .25)$, $c_1 = .25$ and $c_1 \in (.25, .5)$, respectively. Edges corresponding to time steps of A can be determined from visual inspection of Figure 4. For example, for $q = (\sigma_1, [x])$ with $[x]$ satisfying $c_1 \in (-.25, -.5)$ and $c_2 = .25$, there is an outgoing edge from q to $q' = (\sigma_1, [y])$ with $[y]$ satisfying $c_1 \in (-.25, -.5)$ and $c_2 \in (.25, .5)$.

The results of algorithm NDD are shown in Figure 2(b) and Figure 5. In Figure 5 the dashed line is the smooth switching curve for the continuous problem. The black dots identify equivalence classes where NDD assigns a control switch. Considering $g_{e_{-1}}$ we see that the boundary of the enabling condition in the upper left corner is a jagged approximation using equivalence classes of the smooth switching curve. Initial conditions in the upper left corner just inside the enabling condition must switch to a control of $u = -1$, otherwise the trajectory will increase in the x_2 direction and not reach the target. Initial conditions in the upper left corner just outside the enabling condition must allow time to pass until they reach the enabling condition, for if they switched to $u = -1$ they would be unable to reach the target. Hence the upper left boundary of the enabling condition is crisp. The lower right side of the enabling condition which has islands of time steps shows the effect of the non-determinacy of automaton A. These additional time steps occur because it can be less expensive to take a time step than to incur the cost of the *worst case* control switch. Indeed consider an initial condition in Figure 5(a) which lies in an equivalence class that takes a time step but should take a control switch according to the continuous optimal control. Such a point will move up and to the left before it takes a control switch. By moving slightly closer to the target, the worst-case cost-to-go incurred in a control switch is reduced. Notice that all such initial conditions eventually take a control switch. This phenomenon of extra time steps is a function of the mesh size δ: as δ decreases there are fewer extra time steps. Finally we note that the two enabling conditions have an empty intersection, as expected in order to ensure non-Zeno trajectories.



(a) $g_{e_{-1}}$

(b) g_{e_1}

Fig. 5. Enabling conditions

Figure 6 shows trajectories of the closed-loop system using the controller synthesized by NDD. The bold lines are the trajectories, the central hatched region is an enlarged target region, and the shaded areas are the equivalence classes visited during the simulation.

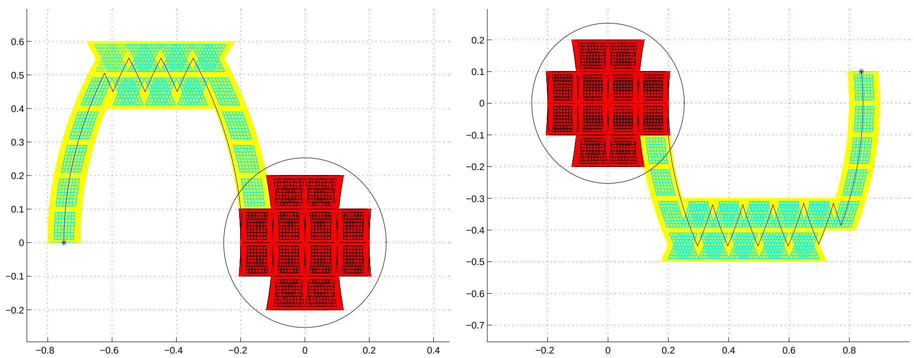


Fig. 6. Trajectories of the closed-loop system

7 Conclusion

In this paper we developed an efficient single-pass algorithm to solve a dynamic programming problem on a non-deterministic graph that arises in the solution of a continuous optimal control problem using hybrid systems and bisimulation. We have seen that the single-pass nature of the solution depends on the partitioning method. An area for future investigation is exploring other partition methods in relation to the efficiency of the algorithmic solution of the dynamic programming problem. This would include partitions that are not bisimulations, especially when analytical expressions for first integrals are difficult to obtain.

We have developed a prototype tool for the synthesis of hybrid optimal controls based on bisimulation. The algorithm has complexity $O(N \log N)$ where N is the number of states of the automaton. The number of states is exponential in the dimension of the continuous state space. In the "vanilla" version of our approach, the automaton is constructed before running the Djikstra-like algorithm. To improve the speed and the memory usage of the algorithm, we plan to build the automaton on the fly while algorithm NDD is executing. In addition, we plan to apply the approach to solving a number of optimal control problems arising in automotive engine control.

References

1. E. Asarin and O. Maler. As soon as possible: time optimal control for timed automata. In *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, eds., LNCS 1569, Springer-Verlag, pp. 19-30, 1999.
2. M. Broucke. A geometric approach to bisimulation and verification of hybrid systems. In *Hybrid Systems: Computation and Control*, F. Vaandrager and J. van Schuppen, eds., LNCS 1569, Springer-Verlag, pp. 61-75, 1999.

3. M. Broucke, M.D. Di Benedetto, S. Di Gennaro, A. Sangiovanni-Vincentelli. Theory of optimal control using bisimulations. In *Hybrid Systems: Computation and Control*, N. Lynch and B. Krogh, eds., LNCS 1790, Springer-Verlag, pp. 89-102, 2000.
4. I. Capuzzo Dolcetta. On a discrete approximation of the Hamilton-Jacobi equation for dynamic programming. *Applied Math. Optim.*, vol. 10, pp. 367-377, 1983.
5. M.G. Crandall, P.L. Lions. Viscosity solutions of Hamilton-Jacobi equations. *Trans. Amer. Math. Soc.*, vol. 277, no. 1, pp. 1-42, 1983.
6. M.G. Crandall, P.L. Lions. Two approximations of solutions of Hamilton-Jacobi equations. *Mathematics of Computation*, vol. 43, no. 176, pp. 1-19, July 1984.
7. E.W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik* 1, p. 269-271, 1959.
8. M. Falcone. A numerical approach to the infinite horizon problem of deterministic control theory. *Applied Mathematics and Optimization*, 15, pp. 1-13, 1987.
9. W.H. Fleming, R.W. Rishel. *Deterministic and stochastic optimal control.* Springer-Verlag, New York, 1975.
10. R. Gonzales and E. Rofman. On deterministic control problems: an approximation procedure for the optimal cost. I: the stationary problem. *SIAM J. Contr. Optim.*, vol. 23, no. 2, pp. 242-266, 1985.
11. P.L. Lions. *Generalized solutions of Hamilton-Jacobi equations*. Pitman, Boston, 1982.
12. L. Polymenakos, D. Bertsekas, and J. Tsitsiklis. Implementation of efficient algorithms for globally optimal trajectories. *IEEE Trans. AC*, vol.43, no.2, pp. 278-83, Feb. 1998.
13. P.E. Souganidis. Approximation schemes for viscosity solutions of Hamilton-Jacobi equations. *Journal of Differential Equations*, vol. 59, no. 1, p. 1-43, August 1985.
14. J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528-1538, September 1995.