Multiresolution Discrete Abstraction for Optimal Control

Yuichi Tazaki and Jun-ichi Imura

***Abstract*— This paper presents a method that computes an approximate solution to a class of optimal control problems in arbitrary precision based on a discrete abstraction technique. Using a discrete abstract model, one can obtain upper and lower bounds on the optimal cost of an optimal control problem. By making use of this property, the method initially constructs a coarse discrete model, and then it gradually increases the resolution of discretization to tighten the bound on the optimal cost until the required precision is satisfied. The effectiveness of the method is demonstrated in a path-planning problem on a 2D plane.**

I. Introduction

Discrete abstraction is a technique to construct a finite-state model that approximates the behavior of a dynamical system with continuous states. Control of continuous-state systems poses various difficulties such as the nonlinearity of the system's dynamics and the non-convexity of constraints. By making use of discrete abstract models, such control problems are converted into path-planning problems on finite automata, most of which can be solved using novel search techniques whose computational complexities are $O(N)$ (N is normally the length of prediction time steps). To date, various methods on the control system design using discrete models have been explored [1]-[6]. In particular, Munos [7] focuses on approximating value functions in optimal control by the discretization of the state space. However, the method does not provide a measure of how much the approximate solution is close to the optimal solution. The authors [9] have shown that discrete abstract models based on approximate bisimulation [8] provides an approximate solution to optimal control problems with a quantitative performance measure. One critical drawback is that the discretization is uniform, meaning that the number of symbolic states grow exponentially with respect to the state dimension.

In this paper, we propose a technique to compute an approximate solution to optimal control problems using multiresolution discrete models. Unlike most of the previous methods, the proposed method is an on-line method, which constructs a discrete model after an initial state is given. A basic idea underlying the proposed method is that one can compute upper and lower bounds on the optimal cost by making use of a discrete model. Moreover, one can tighten this bound by increasing the resolution of discretization under a certain rule. The proposed method repeats this procedure to obtain an approximate solution that is arbitrarily close to the exact solution. Multiresolution approaches have been investigated in the control and robotics field [10][11]. There is a fundamental difference, however, that they mainly focus on multiresolution representation of the input space, whereas we consider multiresolution discretization of the state space to obtain a finite-state model that is well suited to optimal control.

The rest of this paper is organized as follows. In Section II, we formulate the class of optimal control problems we consider. Next, in Section III, we review the representation of discrete models using mesh structures. Section IV describes how to bound the optimal cost using a discrete model. We also derive a method to actually compute this bound based on the forward dynamic programming. The tightness of the bound depends on the resolution of the mesh. Section V presents a simple algorithm that gradually increases the mesh resolution until a required precision on the cost bound is achieved. In Section VI, we apply the proposed technique to a simple path-planning problem of a 2D vehicle. Finally, concluding remarks are given in Section VII.

Notation) The symbol $[i_1 : i_2]$ denotes a sequence of integers $i_1, i_1+1, \dots, i_2$. Moreover, $x_{i_1:i_2}$ denotes an indexed sequence of variables $x_{i_1}, x_{i_1+1}, \dots, x_{i_2}$. To denote the $\max$ operation, we use $\max_{\mathcal{P}}(x) f(x)$ and $\max\,(f(x)\,|\,\mathcal{P}(x))$; both expressions denote the maximum value of $f(x)$ under the condition that the predicate $\mathcal{P}(x)$ is true. When we write $\langle f^*, x^* \rangle = \max_{\mathcal{P}}(x) f(x)$, f^* denotes the maximum and x^* denotes the maximizer. The same notation is used for the $\min$ operation.

II. Problem Setting

In this paper, we consider discrete-time systems with continuous state variables and control inputs. A discrete-time system is a tuple $\langle X, X_0, U, f \rangle$, where $X \subset \mathbb{R}^n$ is the set of states, $X_0 \subseteq X$ is the set of initial states, $U \subset \mathbb{R}^m$ is the set of control inputs and $f : X \times U \mapsto X$ is the state transition function. The state and the control input of the system at time $t \in \mathbb{Z}^+$ are expressed as $x_t \in X$ and $u_t \in U$, respectively. The state transition at time t is expressed as

$$x_{t+1} = f(x_t, u_t). \tag{1}$$

We assume that the sets X and U are both bounded. We denote an N-step trajectory of a system $\Sigma = \langle X, X_0, U, f \rangle$ by $\{x_t, u_t\}_{0:N}$ (be aware that u_N is not included). If the number of steps is apparent from the context, we simply write $\{x_t, u_t\}$.

For this class of systems, we consider the following finite-horizon optimal control problem.

Y. Tazaki is with the Department of Mechanical Science and Engineering, Nagoya University, Nagoya, Japan. tazaki@nuem.nagoya-u.ac.jp

J. Imura is with the Department of Mechanical and Environmental Informatics, Tokyo Institute of Technology, Meguro, Tokyo, Japan. imura@mei.titech.ac.jp

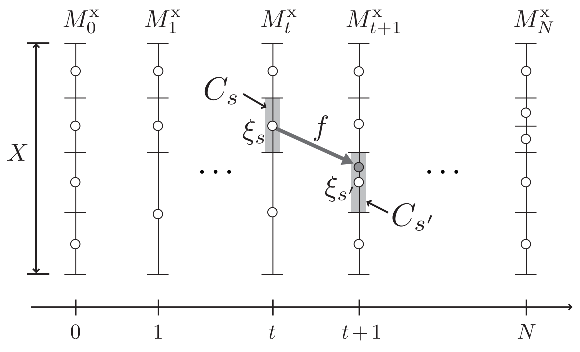


Fig. 1. Discrete obtained by quantizer embedding

Problem 1 *For a system $\Sigma = \langle X, X_0, U, f\rangle$ and a given initial state $x \in X_0$, find a trajectory $\{x_t, u_t\}_{0:N}$ that minimizes*

$$J(\{x_t, u_t\}_{0:N}) = \sum_{t=0}^{N-1} [\psi_t^{\mathrm{x}}(x_t) + \psi_t^{\mathrm{u}}(u_t)] + \psi_N^{\mathrm{x}}(x_N) \quad (2)$$

subject to (1) and $x_0 = x$.

In the above problem, $J : X^{N+1} \times U^N \mapsto \mathbb{R}$ defines the cost of a given trajectory. Here, the cost need not be positive. The functions $\psi_t^{\mathrm{x}} : X \mapsto \mathbb{R}$ and $\psi_t^{\mathrm{u}} : U \mapsto \mathbb{R}$ are assumed to be continuous. We do not consider constraint conditions explicitly; constraints should be encapsulated in the cost function as penalty terms. Our purpose is to obtain an approximate solution of this problem such that the difference between the cost of the approximate solution and the optimal solution is below a prescribed positive constant ϵ.

III. Representation of Discrete Abstract Models

A. Construction of Discrete Models by means of Quantizers

In this section, we review the notion of quantizer embedding [13], which is a systematic method to transform a continuous-state system into a finite automaton. First of all, we introduce a mesh structure. A mesh M is a finite collection of pairs of a node and a cell:

$$M = \{(\xi_s, C_s) \mid s \in \mathcal{S}\}. \quad (3)$$

Here, $\mathcal{S}$ denotes a finite set of identifiers. Each C_s $(s \in \mathcal{S})$ is called a cell and each $\xi_s \in C_s$ is called the node of the cell C_s. The cells form a partition of the mesh domain $\mathrm{dom}(M)$; that is, $C_{s_1} \cap C_{s_2} = \emptyset$ $(s_1 \neq s_2)$ and $\bigcup_s C_s = \mathrm{dom}(M)$. A mesh defines a quantization function defined below:

$$\begin{aligned} &Q[M] : \mathrm{dom}(M) \mapsto \{\xi_s\}_{s\in\mathcal{S}}, \\ &Q[M](x) = \xi_s \ \text{ if } \ x \in C_s. \end{aligned} \quad (4)$$

A quantization function $Q[M](\cdot)$ maps a given point x inside its domain to a node whose corresponding cell includes x.

Now, let us define a series of meshes for the state set X and the input set U:

$$\begin{aligned} M_t^{\mathrm{x}} &= \{(\xi_s, C_s) \mid s \in \mathcal{S}_t\} \ \ (t \in [0:N]), \\ M_t^{\mathrm{u}} &= \{(\xi_a, C_a) \mid a \in \mathcal{A}_t\} \ \ (t \in [0:N-1]). \end{aligned}$$

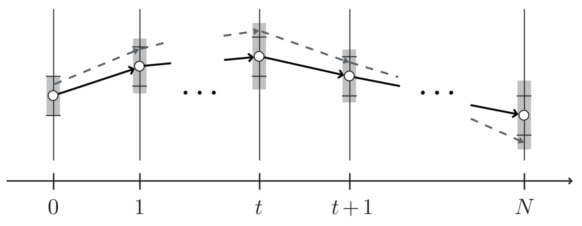


Fig. 2. Set of continuous trajectories associated with a symbolic trajectory

Here, $\mathrm{dom}(M_t^{\mathrm{x}}) = X$ and $\mathrm{dom}(M_t^{\mathrm{u}}) = U$. The symbol $\mathcal{S}_t$ denotes the set of cell identifiers of the state mesh M_t^{x} and $\mathcal{A}_t$ denotes that of the input mesh M_t^{u}. Note that cell identifiers do not overlap between different meshes. Therefore, if we take an s from $\mathcal{S}_t$, then a cell denoted by C_s uniquely identifies the cell of the state mesh M_t^{x}.

Using these meshes, we define a symbolic system $\hat{\Sigma}$ whose set of symbolic states and that of symbolic inputs at time t are given by $\mathcal{S}_t$ and $\mathcal{A}_t$, respectively, and the transition relation at time t is written as

$$s_t \xrightarrow{a_t} s_{t+1} \ \Leftrightarrow \ f(\xi_{s_t}, \xi_{a_t}) \in C_{s_{t+1}} \quad (5)$$

where $s_t \in \mathcal{S}_t$, $a_t \in \mathcal{A}_t$ and $s_{t+1} \in \mathcal{S}_{t+1}$. Note that the behavior of $\hat{\Sigma}$ is defined only in the time interval $[0 : N]$. Since each mesh is composed of finite cells, the system $\hat{\Sigma}$ can be seen as a finite automaton. Fig. 1 illustrates this transformation in the case of a system with a scalar-valued state no input. Each mesh M_t^{x} covers the entire state domain X. In the figure, $f(\xi_s)$ $(s \in \mathcal{S}_t)$ is included in $C_{s'}$ $(s' \in \mathcal{S}_{t+1})$. Therefore, a symbolic transition $s \to s'$ is defined.

It is also possible to obtain a finite automaton without explicit input quantization. Like the previous case, the set of symbolic states at time t is given by $\mathcal{S}_t$. The transition relation is defined as

$$\begin{aligned} &s_t \to s_{t+1} \quad \Leftrightarrow \quad u_t \in U_{s_t, s_{t+1}}, \\ &U_{s,s'} = \{u \mid f(\xi_s, u) \in C_{s'}\}. \end{aligned} \quad (6)$$

Here, a set of symbolic inputs is defined for each symbolic state s_t, and it is equivalent with the set of successor states; that is, for each $s \in \mathcal{S}_t$, we have $\mathcal{A}_{t,s} = \{s' \mid s \to s'\}$ instead of $\mathcal{A}_t$. In the later discussion, we consider the case with input quantization only. Nevertheless, the proposed method is applicable to the case without input quantization with minor modification.

As a particular class of meshes, we consider variable-resolution meshes composed of rectangular cells. A refinement of a mesh is done by selecting a cell and subdividing it into two sub-cells. Here, a subdivision can be made in one of n directions. For an example, in the 2-dimensional case, a cell can be divided either horizontally or vertically. Each of newly created cells has its node in its middle.

IV. Bounding Optimal Cost Using Discrete Models

A. Trajectory Mapping between Continuous and Discrete Models

In this section, we introduce transformation between trajectories of the continuous-state model and the trajectories of the discrete-state model. For a trajectory $\{x_t, u_t\}$ of Σ, consider a map $\Phi^{\Sigma\to\hat{\Sigma}}$ defined as

$$\{s_t, a_t\} = \Phi^{\Sigma\to\hat{\Sigma}}(\{x_t, u_t\}) \;\Leftrightarrow\; \begin{cases} x_0 \in C_{s_0}, \\ u_t + \phi(\xi_{s_t}, x_t) \in C_{a_t}, \\ s_t \xrightarrow{a_t} s_{t+1}. \end{cases} (t \in [0:N-1]) \tag{7}$$

The function $\Phi^{\Sigma\to\hat{\Sigma}}$ maps a trajectory of Σ to a trajectory of $\hat{\Sigma}$. Here, the function ϕ satisfies $\phi(x,\hat{x}) = -\phi(\hat{x},x)$ and serves as an error correction term that is to be added on the control input in order to compensate the error between the states of two systems, x_t and ξ_{s_t}. If ϕ is set as $\phi(\cdot,\cdot) \equiv 0$, then the same input trajectory is applied to both systems. One usage of ϕ is shown in Section VI. Similarly, consider a map $\Phi^{\hat{\Sigma}\to\Sigma}$ defined as

$$\{x_t, u_t\} \in \Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\}) \;\Leftrightarrow\; \begin{cases} x_0 \in C_{s_0}, \\ u_t = \xi_{a_t} + \phi(x_t, \xi_{s_t}), \\ x_{t+1} = f(x_t, u_t). \end{cases} (t \in [0:N-1]) \tag{8}$$

Here, $\Phi^{\hat{\Sigma}\to\Sigma}$ maps a trajectory of $\hat{\Sigma}$ to a set of trajectories of Σ. This is natural since Σ has infinitely many trajectories, while $\hat{\Sigma}$ has only finite trajectories. For these transformations, we have the following lemma.

Lemma 1 *For $\{s_t, a_t\} = \Phi^{\Sigma\to\hat{\Sigma}}(\{x_t, u_t\})$, $\{x_t, u_t\} \in \Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\})$ holds.*

Proof: Omitted. ■

Now, for a symbolic trajectory $\{s_t, a_t\}$, consider a set of trajectories given by $\Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\})$, and moreover, consider the projection of this set onto the set of states at time t. Let us denote this set by R_{s_t} to indicate that the set of continuous states is related to the symbolic state at the same time instant. Then, the series $\{R_{s_t}\}_{0:N}$ is determined in a recursive manner, starting from $t = 0$, as shown below. For s_0,

$$R_{s_0} = C_{s_0}. \tag{9}$$

Moreover, for $t = 0, 1, \ldots, N-1$,

$$\begin{aligned} R_{a_t} &= \{u \,|\, x \in R_{s_t}, u + \phi(\xi_{s_t}, x) \in C_{a_t}\}, \\ R_{s_{t+1}} &= \{f(x,u) \,|\, x \in R_{s_t}, u + \phi(\xi_{s_t}, x) \in C_{a_t}\}. \end{aligned} \tag{10}$$

Let us define these equations as maps

$$R_{a_t} = F^{\mathrm{u}}(R_{s_t}, (s_t, a_t)), \tag{11}$$

$$R_{s_{t+1}} = F^{\mathrm{x}}(R_{s_t}, (s_t, a_t)). \tag{12}$$

This recursive property is exploited in a dynamic programming-based approach described later.

Fig. 2 illustrates the relation between a symbolic trajectory and the set of continuous trajectories associated with it. The circles (symbolic states) connected by solid arrows depict a symbolic trajectory $\{s_t, a_t\}$ (symbolic inputs are not shown). For each time instant t, a gray band drawn behind the node ξ_{s_t} depicts R_{s_t}. Therefore, all continuous trajectories in $\Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\})$ lies within these bands. One sample trajectory is shown in dashes arrows.

Remark 1 *The discussion made here has a close relation between approximate bisimulation [8] of a continuous-state system and a discrete-state system. However, there is one fundamental difference; while approximate bisimulation considers arbitrary behavior, our aim is to approximate only the optimal behavior.*

B. Cost Bound

In the following, it is shown that the optimal cost in (2) is bounded by the solution of a class of optimal control problems defined on the discrete model. First, for a symbolic trajectory $\{s_t, a_t\}$, consider the maximum and minimum value of costs taken by continuous trajectories that are associated with $\{s_t, a_t\}$ by $\Phi^{\hat{\Sigma}\to\Sigma}$.

$$\overline{J}(\{s_t, a_t\}) = \max_{\{x_t,u_t\}\in\Phi^{\hat{\Sigma}\to\Sigma}(\{s_t,a_t\})} J(\{x_t, u_t\}), \tag{13}$$

$$\underline{J}(\{s_t, a_t\}) = \min_{\{x_t,u_t\}\in\Phi^{\hat{\Sigma}\to\Sigma}(\{s_t,a_t\})} J(\{x_t, u_t\}). \tag{14}$$

A set of continuous trajectories associated with a symbolic trajectory can be captured as uncertainty caused by quantization. From this point of view, $\overline{J}$ and $\underline{J}$ are interpreted as the worst-case cost and the best-case cost of $\{s_t, a_t\}$ under uncertainty. Furthermore, for a symbolic state $s \in \mathcal{S}_0$, consider the minimum values of $\overline{J}(\{s_t, a_t\}_{0:N})$ and $\underline{J}(\{s_t, a_t\}_{0:N})$ over all trajectories starting from s:

$$\langle \overline{J}^*(s), \{s_t, a_t\}^{\text{min-max}}(s) \rangle = \min_{\{s_t,a_t\}_{0:N}, s_0 = s} \overline{J}(\{s_t, a_t\}), \tag{15}$$

$$\langle \underline{J}^*(s), \{s_t, a_t\}^{\text{min-min}}(s) \rangle = \min_{\{s_t,a_t\}_{0:N}, s_0 = s} \underline{J}(\{s_t, a_t\}). \tag{16}$$

Let us call $\overline{J}^*(s)$ ($\{s_t, a_t\}^{\text{min-max}}(s)$) the min-max cost (trajectory) and $\underline{J}^*(s)$ ($\{s_t, a_t\}^{\text{min-min}}(s)$) the min-min cost (trajectory).

First, we obtain the following basic lemma.

Lemma 2 *Let $J^*(x)$ be the optimal cost of Problem 1 with an initial state $x \in X_0$. Moreover, let $s \in \mathcal{S}_0$ be a symbolic state satisfying $x \in C_s$. Then, the following statements hold.*
i) There exists a trajectory $\{s_t, a_t\}$ with $s_0 = s$ such that

$$\underline{J}(\{s_t, a_t\}) \leq J^*(x)$$

holds.
ii) For all trajectory $\{s_t, a_t\}$ with $s_0 = s$,

$$J^*(x) \leq \overline{J}(\{s_t, a_t\})$$

holds.

Proof: Let $\{x_t^*, u_t^*\}$ be the optimal trajectory and $\{s_t, a_t\} = \Phi^{\Sigma\to\hat{\Sigma}}(\{x_t^*, u_t^*\})$. Here we have $s_0 = s$. From Lemma 1, $\{x_t^*, u_t^*\} \in \Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\})$ holds. Therefore $\underline{J}(\{s_t, a_t\}) \le J(\{x_t^*, u_t^*\})$ holds and thus we obtain the first claim.

Next, suppose $\overline{J}(\{s_t, a_t\}) < J(\{x_t^*, u_t^*\})$ holds for some $\{s_t, a_t\}$ with $s_0 = s$. Then, a trajectory given by

$$\{x_t, u_t\} \in \Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\}),\, x_0 = x$$

satisfies $J(\{x_t, u_t\}) < J(\{x_t^*, u_t^*\})$, which contradicts the optimality of $\{x_t^*, u_t^*\}$. This proves the second claim. ■

Based on Lemma 2, the next theorem provides a bound on the optimal cost as well as a way to obtain a trajectory of Σ whose cost is inside this bound.

Theorem 1 *Let $x \in X_0$ be an initial state of Problem 1 and let $s \in \mathcal{S}_0$ satisfying $x \in C_s$. Then, the following inequality holds.*

$$\underline{J}^*(s) \le J^*(x) \le \overline{J}^*(s) \tag{17}$$

Moreover, a trajectory given by

$$\{\tilde{x}_t, \tilde{u}_t\} \in \Phi^{\hat{\Sigma}\to\Sigma}(\{s_t, a_t\}^{\textit{min-max}}(s)),\, \tilde{x}_0 = x \tag{18}$$

satisfies

$$J(\{\tilde{x}_t, \tilde{u}_t\}) \le J^*(x) + (\overline{J}^*(s) - \underline{J}^*(s)). \tag{19}$$

Proof: The first claim is a straightforward consequence of Lemma 2. For $\{\tilde{x}_t, \tilde{u}_t\}$, we have

$$\begin{aligned} J(\{x_t^*, u_t^*\}) &\le J(\{\tilde{x}_t, \tilde{u}_t\}) \\ &\le \overline{J}(\{s_t, a_t\}^{\text{min-max}}(s)) = \overline{J}^*(s) \end{aligned}$$

where the first inequality follows from the optimality of $\{x_t^*, u_t^*\}$, and the second from the definition of $\overline{J}$. From this inequality and (17), we obtain (19). ■

Theorem 1 states that the optimal cost is bounded by the min-max cost and the min-min cost, computed on the discrete model. In fact, (17) is the tightest bound on $J(\{x_t^*, u_t^*\})$ that is available based on Lemma 2 without knowing $\{x_t^*, u_t^*\}$ itself. The theorem also states that a suboptimal trajectory whose cost is within this bound is obtained using the min-max trajectory of the discrete model. Therefore, in order to obtain an approximate solution whose cost is below $J^*(x) + \epsilon$, we need to construct a mesh such that the difference between the min-max cost and the min-min cost is smaller than ϵ.

Remark 2 *In [12], an alternative way to obtain a bound on the optimal cost of a time-optimal control problem is derived based on symbolic abstraction. Comparison of these methods regarding computational complexity and the tightness of the bounds is an open question.*

C. Cost Computation Using Dynamic Programming

So far, all discussion has been made in the trajectory level. However, it is not always a trivial task to compute an upper-bound $\overline{J}$ and a lower-bound $\underline{J}$ for a given symbolic trajectory. Moreover, it is obviously inefficient to enumerate all trajectories of the discrete model in order to find the min-max/min-min trajectories, even though the trajectory space is finite. In the following, it is shown that an overestimate of the min-max cost and that of the min-min cost can be computed using a forward dynamic programming technique. At first, for each discrete state $s \in \mathcal{S}_0$, we compute

$$\overline{J}_s^* := \max_{x\in C_s} \psi_0^{\mathrm{x}}(x),\; \underline{J}_s^* := \min_{x\in C_s} \psi_0^{\mathrm{x}}(x), \tag{20}$$

$$R_s^{\text{min-max}} := C_s,\; R_s^{\text{min-min}} := C_s. \tag{21}$$

Next, for $t \in [1 : N]$ in the ascending order and for each $s' \in \mathcal{S}_t$, compute

$$\begin{aligned}\langle\, \overline{J}_{s'}^*,\, (s,a)_{s'}^{\text{min-max}} \,\rangle := \min_{(s,a)\text{ s.t. } s\xrightarrow{a}s'} \Big[\overline{J}_s^* \\ + \max_{u\in F^{\mathrm{u}}(R_s^{\text{min-max}},(s,a))} \psi_t^{\mathrm{u}}(u) + \max_{x\in F^{\mathrm{x}}(R_s^{\text{min-max}},(s,a))} \psi_{t+1}^{\mathrm{x}}(x)\Big], \end{aligned} \tag{22}$$

$$\begin{aligned}\langle\, \underline{J}_{s'}^*,\, (s,a)_{s'}^{\text{min-min}} \,\rangle := \min_{(s,a)\text{ s.t. } s\xrightarrow{a}s'} \Big[\underline{J}_s^* \\ + \min_{u\in F^{\mathrm{u}}(R_s^{\text{min-min}},(s,a))} \psi_t^{\mathrm{u}}(u) + \min_{x\in F^{\mathrm{x}}(R_s^{\text{min-min}},(s,a))} \psi_{t+1}^{\mathrm{x}}(x)\Big], \end{aligned} \tag{23}$$

$$R_{s'}^{\text{min-max}} := F^{\mathrm{x}}(R_s^{\text{min-max}}, (s,a)_{s'}^{\text{min-max}}), \tag{24}$$

$$R_{s'}^{\text{min-min}} := F^{\mathrm{x}}(R_s^{\text{min-min}}, (s,a)_{s'}^{\text{min-min}}). \tag{25}$$

Unlike conventional dynamic programming, the above procedure computes two different costs, the min-max cost and the min-min cost, as well as some other related quantities. For each s, $\overline{J}_s^*$ and $\underline{J}_s^*$ store the minimum value of the worst-case cost and the best-case cost, respectively, among all trajectories that ends at s. Moreover, the symbols $(s,a)_{s'}^{\text{min-max}}$ and $(s,a)_{s'}^{\text{min-min}}$ are used to keep track of transitions that lead to s' in the trajectories giving $\overline{J}_{s'}^*$ and $\underline{J}_{s'}^*$, respectively. Finally, $R_{s'}^{\text{min-max}}$ and $R_{s'}^{\text{min-min}}$ denote states taken by continuous trajectories that are associated with the symbolic trajectories giving $\overline{J}_s^*$ and $\underline{J}_s^*$, respectively. Once this computation is completed, one can obtain the min-max trajectory by finding $s \in \mathcal{S}_N$ that gives the smallest $\overline{J}_s^*$ and tracking $(s,a)_{s'}^{\text{min-max}}$ backwards. The min-min trajectory is also obtained in the same way.

Remark 3 *One should keep in mind that the min-max/min-min costs computed in the above procedure are overestimates of the ones defined in Section IV. By overestimate, we mean that the computed min-max cost may be larger and the min-min cost may be smaller than the true values. At the same time, the computed min-max/min-min trajectories may be different from the real ones. This is because the* $\max$ *and* $\min$ *operations on ϕ_t^{u} and ψ_t^{x} are evaluated separately, despite* *they do not always take the extreme values at every time instant in a single trajectory. Nevertheless, these can still be used to bound the optimal cost.*

Remark 4 *It is often difficult to compute the set operations F^{x} and F^{u} strictly and therefore one should rely on some set approximation techniques to overestimate $R_s^{\text{min-max}}$ and $R_s^{\text{min-min}}$. There is also no general solution to the evaluation of* $\max$ *and* $\min$ *operations on the cost terms ψ_t^{x} and ψ_t^{u}. One should consider efficient implementation for each concrete problem. One example of implementation is shown in Section VI.*

V. Mesh Refinement

In the previous section, it has been shown that the true optimal cost is bounded by the min-max/min-min costs calculated on the discrete model. In this section, we discuss the generation of a mesh whose cost gap, the difference between the min-max cost and the min-min cost, is less than ϵ. Our basic strategy is to begin with a coarse mesh and refine it repeatedly until the condition is satisfied. A key question is which part of the mesh to refine, in order to reduce the cost gap most efficiently. This time, we employ a simple rule; subdivide the largest cell on the min-max/min-min trajectories. This rule is based on the fact that the cost gap is caused by uncertainty in the min-max/min-min trajectories, and therefore refining the mesh along these trajectories reduces the cost gap. To choose one particular cell out of the cells on these trajectories, we simply choose the largest one. However, the largest cell may not always have the largest influence on the cost gap, thus there remains a room for improvement in the refinement rule.

To summarize, the overall procedure is written as follows.
Algorithm:

1) Initialize the mesh :
$$M_t^{\mathrm{x}} := \{(\mathbf{0}, X)\} \;\; (t \in [0:N]),$$
$$M_t^{\mathrm{u}} := \{(\mathbf{0}, U)\} \;\; (t \in [0:N-1])$$
2) Construct a discrete model $\hat{\Sigma}$ by quantizer embedding.
3) Compute min-max cost and min-min cost using (20) - (25).
4) Terminate if $\overline{J}^*(s) - \underline{J}^*(s) \le \epsilon$.
5) Subdivide the largest cell on the min-max/min-min trajectories. Go to Step 2.

Theorem 2 *The algorithm terminates in finite iterations.*

Proof: Consider a symbolic trajectory $\{s_t, a_t\}$. Since the cost function J is assumed continuous, for any $\epsilon > 0$, there exists a $\delta > 0$ such that

$$\begin{aligned} &\bar{r}(C_{s_t}, \xi_{s_t}) \le \delta \;\; \forall t \in [0:N], \\ &\bar{r}(C_{a_t}, \xi_{a_t}) \le \delta \;\; \forall t \in [0:N-1], \\ &\qquad\qquad \Rightarrow \;\; \overline{J}(\{s_t, a_t\}) - \underline{J}(\{s_t, a_t\}) \le \epsilon \end{aligned}$$

where $\bar{r}(C, \xi)$ denotes the radius of the smallest ball that is centered at ξ and includes C.

On the other hand, since we have

$$\overline{J}(\{s_t, a_t\}^{\text{min-max}}) \le \overline{J}(\{s_t, a_t\}^{\text{min-min}})$$

from the definition of min-max trajectory,

$$\begin{aligned} \overline{J}(\{s_t, a_t\}^{\text{min-max}}(s)) - \underline{J}(\{s_t, a_t\}^{\text{min-max}}(s)) \le \epsilon, \\ \overline{J}(\{s_t, a_t\}^{\text{min-min}}(s)) - \underline{J}(\{s_t, a_t\}^{\text{min-min}}(s)) \le \epsilon \end{aligned}$$

imply

$$\begin{aligned} &\overline{J}^*(s) - \underline{J}^*(s) \\ &\quad = \overline{J}(\{s_t, a_t\}^{\text{min-max}}(s)) - \underline{J}(\{s_t, a_t\}^{\text{min-min}}(s)) \quad \le \epsilon. \end{aligned}$$

Here, s denotes a symbolic state whose cell includes the initial state x. Therefore, if all cells in the min-max trajectory and the min-min trajectory are smaller than δ, the difference between the min-max cost and the min-min cost will be below ϵ and thus the algorithm will terminate. Moreover, the algorithm subdivides the largest cell in the min-max/min-min trajectories. These two facts indicate that the algorithm never subdivides cells smaller than δ. Since the state set and the input set are both bounded, we can conclude that the algorithm terminates, at the latest, when all cells in the mesh becomes smaller than δ. ■

VI. Example

In this section, we apply the proposed technique to the path-planning problem of a vehicle that moves on a 2D plane. We assume that the vehicle moves in a constant velocity v along the y-axis, while its motion along the x-axis is given by

$$x_{t+1} = x_t + u_t.$$

The symbol x_t denotes the vehicle's x-position at time t, u_t denotes the control input (amount of change in the x-coordinate in a single move). Figs. 3(a)-(d) show the workspace as well as meshes during the progress of refinement. We take the x-axis in the vertical direction and the y-axis in the horizontal direction. Since the vehicle moves in a constant velocity in the y direction, the state mesh at time t, M_t^{x}, can be overlaid on the workspace in the $y = vt$ position. The state set is set as $X = [-1, 1]$ and the step length is set as $N = 20$. In this example, the control input is not quantized explicitly.

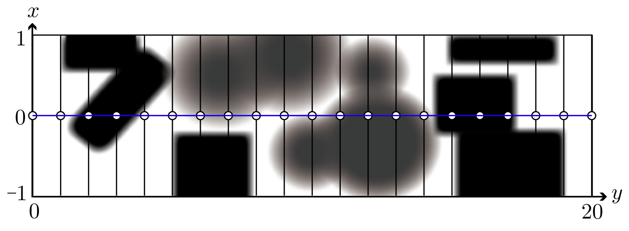
The state cost ψ_t^{x} reflects the difficulty of moving in certain locations in the workspace. In the figures, the state cost at each location is expressed by the intensity of the color at that location. Regions in white are assigned the cost of 0, while the regions in black are assigned the maximum cost of 10. The input cost is simply given as $\psi_t^{\mathrm{u}}(u) = u^2$.

For the input interface ϕ, we set $\phi(x, \hat{x}) = \hat{x} - x$. In this way, the function F^{x} is given by

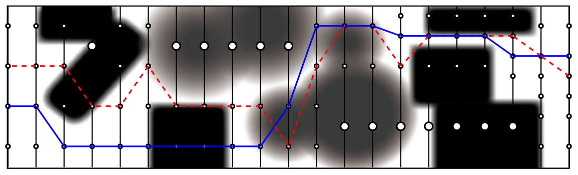
$$F^{\mathrm{x}}(R_s, (s, a)) = C_{s'} \;\; (s \xrightarrow{a} s'),$$

which means that the set of continuous states R_s always coincides with the cell C_s. On the other hand, the function F^{u} is given by

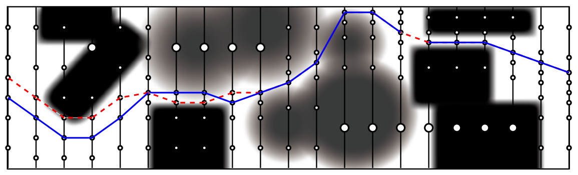
$$F^{\mathrm{u}}(R_s, (s, a)) = \{u \,|\, u = x' - x, \, x \in C_s, \, x' \in C_{s'}\}.$$



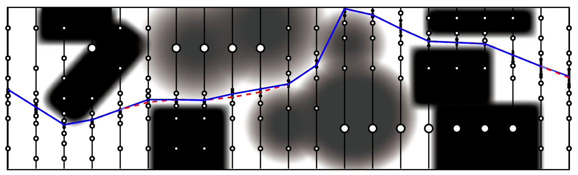
(a) Initial mesh



(b) After 60 refinements



(c) After 100 refinements



(d) After 200 refinements

Fig. 3. Mesh refinement process in path-planning problem

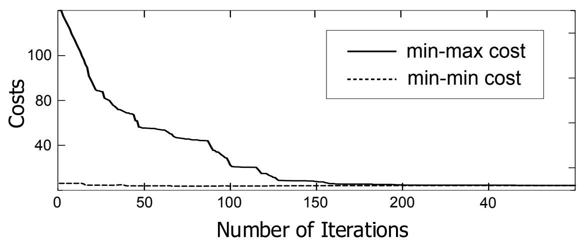


Fig. 4. Change of cost gap during refinement

Since the state and the control input are both scalar, sets are numerically expressed as ranges. The $\max$ and $\min$ operations on the state cost ψ_t^{x} are computed approximately by means of sampling, and those on the input cost $\psi_t^{\mathrm{x}}(u) = u^2$ are evaluated analytically.

Fig. 3(a) to Fig. 3(d) show the initial mesh, the meshes after 60, 100 and 200 refinements, respectively. Each small circle represents the node of a cell. A symbolic trajectory draw in solid lines depict the min-max trajectory whereas a trajectory in dashed lines depict the min-min trajectory. On the other hand, Fig. 4 shows the change of the min-max cost and the min-min cost in the course of refinement. We observe that in this particular example, the cost gap exhibits roughly exponential decrease. After 200 refinements, the cost gap becomes 0.41, which is less than 10% of the min-min cost of 4.23. The amount of computation time for 200 refinements was 390 milliseconds on a personal computer equipped with 1.2GHz CPU and 2.0GB memory.

VII. Conclusion

This paper has presented an online approach to optimal control that makes use of multiresolution discrete models. First, it has been shown that the optimal cost is bounded by the min-max cost and the min-min cost, computed on a discrete model based on a dynamic programming technique. Based on this theoretical result, the proposed algorithm refines a mesh repeatedly until this bound becomes precise enough. A heuristic rule on the selection of a cell to be subdivided in the mesh refinement algorithm is used, accompanied with a proof that the algorithm terminates in finite time. This time, we have considered discrete-time systems. In the future, the method should be extended to continuous-time systems so that the discretization of the time axis as well as the state space and the input space is handled in a uniform framework.

References

[1] J. Raisch and S.D. O'Young : Discrete Approximation and Supervisory Control of Continuous Systems, IEEE Trans. on Automatic Control, Vol. 43, No. 4, 569/573, 1998.

[2] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver and M.D. Lemmon : Supervisory Control of Hybrid Systems, In Proc. of IEEE, Special Issue in Hybrid Systems. P.J. Antsaklis, Ed., 1026/1049, July 2000.

[3] L.C.G.J.M. Habets, P.J. Collins and J.H. van Schuppen : Reachability and Control Synthesis for Piecewise-Affine Hybrid Systems on Simplices, IEEE Transactions on Automatic Control, Vol. 51, No. 6, 938/948, 2006.

[4] J. Imura and H. Matsushima : Simultaneous Optimization of Continuous Control Inputs and Discrete State Waypoints, 9th International Workshop on Hybrid Systems: Computation and Control (HSCC 2006), J.Hespanha and A.Tiwari (Eds.) LNCS 3927 Springer-Verlag, pp. 302-317, 2006.

[5] M. Kloetzer, C. Belta : A Fully Automated Framework for Control of Linear Systems from LTL Specifications, Hybrid Systems: Computation and Control (HSCC06), Lecture Notes in Computer Science 3927, Springer-Verlag, 333/347, 2006.

[6] A. Girard and G.J. Pappas : Hierarchical Control System Design Using Approximate Simulation, Automatica, 45(2), 566/571, 2009.

[7] R. Munos and A. Moore : Variable Resolution Discretization in Optimal Control, Machine Learning Journal, 49, pp 291-323, 2001.

[8] A. Girard and G.J. Pappas : Approximation Metrics for Discrete and Continuous Systems, IEEE Transactions on Automatic Control, 52(5), 782/798, 2007.

[9] Y. Tazaki and J. Imura : Finite Abstractions of Discrete-time Linear Systems and Its Application to Optimal Control, 17th IFAC World Congress (in CDROM), Seoul, Korea, July 6-11, 2008.

[10] A. Girard : Towards a Multiresolution Approach to Linear Control, IEEE Transactions on Automatic Control, 51(8), 1261/1270, 2006.

[11] S.R. Lindemann and S.M. LaValle : A Multiresolution Approach for Motion Planning Under Differential Constraints, IEEE International Conference on Robotics and Automation, pp.139-144, May 15-19, Orlando, FL, USA, 2006.

[12] M. Mazo and P. Tabuada : Approximate Time-Optimal Control via Approximate Alternating Simulations, American Control Conference, pp.1009-1014, Baltimore, Maryland, USA, June 30 - July 2, 2010.

[13] Y. Tazaki and J. Imura : Approximately Bisimilar Discrete Abstractions of Nonlinear Systems Using Variable-Resolution Quantizers, American Control Conference, pp.1015-1020, Baltimore, Maryland, USA, June 30 - July 2, 2010.