Approximate Dynamic Programming Recurrence Relations for a Hybrid Optimal Control Problem

W. Lu[§], S. Ferrari[§], R. Fierro[†] and T. A. Wettergren[‡]

[§]Laboratory for Intelligent Systems and Control (LISC), Department of Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA;
[†]Multi-Agent, Robotics, Hybrid, and Embedded Systems (MARHES), Laboratory Department of Electrical and Computer Engineering University of New Mexico, Albuquerque, NM, USA;
[‡]Naval Undersea Warfare Center, Newport, RI, USA.

ABSTRACT

This paper presents a hybrid approximate dynamic programming (ADP) method for a hybrid dynamic system (HDS) optimal control problem, that occurs in many complex unmanned systems which are implemented via a hybrid architecture, regarding robot modes or the complex environment. The HDS considered in this paper is characterized by a well-known three-layer hybrid framework, which includes a discrete event controller layer, a discrete-continuous interface layer, and a continuous state layer. The hybrid optimal control problem (HOCP) is to find the optimal discrete event decisions and the optimal continuous controls subject to a deterministic minimization of a scalar function regarding the system state and control over time. Due to the uncertainty of environment and complexity of the HOCP, the cost-to-go cannot be evaluated before the HDS explores the entire system state space; as a result, the optimal control, neither continuous nor discrete, is not available ahead of time. Therefore, ADP is adopted to learn the optimal control while the HDS is exploring the environment, because of the online advantage of ADP method. Furthermore, ADP can break the curses of dimensionality which other optimizing methods, such as dynamic programming (DP) and Markov decision process (MDP), are facing due to the high dimensions of HOCP.

Keywords: Approximate dynamic programming (ADP), hybrid systems, optimal control

1. INTRODUCTION

Recently, developments in the unmanned system consisting of robots and sensors, such as surveilling intruders, detecting and classifying targets, raises the degree of functionality, reconfigurability, and redundancy of the system itself. A three-layer hybrid architecture, one of hybrid dynamic systems (HDS), has been proposed in[1] to characterize the unmanned system's both continuous dynamics and discrete event behavior, where the events can be defined as reaching some HDS state that has specific properties than other state, or as the signal causing the sudden change of HDS state. A HDS has the ability of coordinating a variety of subsystems within their unique structures, and has the benefit of allowing more flexibility in dynamic models. The HDS optimal control problem (HOCP) usually requires discrete decisions and continuous controls, and its hybrid nature has been recognized by several authors.[1,2] An hybrid modeling approach in a mobile multi-agent network was recently developed,[3] that is very effective at maintaining a desired formation or connectivity in a sensor network. Furthermore, a hybrid modeling framework for robust maneuver-based motion planning in nonlinear systems with symmetries was proposed by Sanfelice.[4] The hybrid systems with autonomous or controlled events are reviewed in.[5] The HOCP is to find the optimal continuous controller and the optimal discrete decisions subject to a deterministic minimization of a scalar objective function regarding the HDS state, event, controls, and decisions over time. The objective functions in a discrete form or in a mixture of continuous-discrete form are used by several authors in.[6,7] In this paper, the objective function of the mixture form is adopted since it is consistent with the hybrid

Further author information: (Send correspondence to W.L.)
W.L. and S.F.: E-mail: {wenjie.lu, sferrari}@duke.edu, Telephone: 1 919 660 5305
R.F.: E-mail: rfierro@ece.unm.edu, Telephone: 1 505 277 4125
T.W.: E-mail: t.a.wettergren@ieee.org

Unmanned Systems Technology XIV, edited by Robert E. Karlsen, Douglas W. Gage, Charles M. Shoemaker, Grant R. Gerhart, Proc. of SPIE Vol. 8387, 83870C
© 2012 SPIE · CCC code: 0277-786X/12/$18 · doi: 10.1117/12.919286

nature of HDS and HOCP. The objective function includes the integration of the cost regarding continuous state and control over time, and the summation of all event costs (or rewards). A method based on the fixed mode sequence of events and event decisions are used in[7] and a two stage optimization method is adopted in.[8] Existing hybrid and distributed control approaches are reviewed comprehensively in Cassandras's book.[9]

Since the HDS is described by a multiple layer structure, and is controlled by the interaction between continuous state control and discrete event decisions, the high dimensionality of HOCP makes the problem intractable. The existing approaches solve HOCP by either decreasing the dimensions of the hybrid system through imposing more constraints, such as fixing event decision sequences, or by introducing a two stage optimization method, which obtains the event decisions first, and then calculates the continuous control based on decision sequence. As a result, the optimality of the solution is restricted, and more importantly, the methods are not adaptive, and can not be applied online when the environment and/or underlying HDS model are uncertain or unknown. Therefore, to overcome above limitations of existing approaches, a hybrid approximate dynamic programming (ADP)[10–12] is extended and used in this paper to solve HOCP. ADP is widely used in various fields, such as natural gas storage valuation,[13] budget planning,[14] and inventory allocation.[15] ADP uses a Bellman equation[16, 17] to describe the value function (also called cost-to-go) of HOCP, similar to dynamic programming (DP).[6, 18, 19] The DP algorithm, generally, uses a backward method to solve the Bellman equation by discretizing the system state. The algorithm can save some computational burden by cutting unnecessary choices of state to be visited, however, it still suffers from the curses of dimensionality, and becomes intractable when the system state is continuous and dimension is high. Furthermore, the backward method can not be applied to the online problem when the Bellman equation can not be solved by back propagating the value of last state to previous state since the knowledge of last state is not available before controls and decisions are executed, and the environment is explored. Different from DP, the ADP method adopts a forward structure which can be applied to the online problem. The ADP method updates the control law and the decisions by the reward obtained and the reward expected from executing an action to the system. As a result, the imperfect model of the system itself, due to the uncertainty of the system and the environment, can be adapted. A critic neural network is applied to approximate the gradient of the value function regarding system state based on the knowledge obtained online, and to generate the optimal control for the continuous control. More importantly, ADP approximates the value function (or the gradient of the value function) to decrease the number of parameters needed to be inferred, therefore reducing the computational burden. ADP works well in the problems with high dimensions without imposing unrealistic constraints to lower the dimensions.

In this paper, hybrid value functions are introduced to describe the Bellman function in HOCP. As a result, the ADP method alone is not sufficient. Considering the discrete nature of events and event decisions, a Markov decision process (MDP)[20]is used, together with ADP, to solve HOCP. MDP is a discrete time (stochastic) control process, and solves optimization problems where the performance is determined by the decisions. Value iteration or policy iteration[21] methods can be used to obtain the optimal decisions when the transition and reward functions are given. The MDP and ADP method are coordinated to learn the discrete event decisions and continuous controls which are explained in Section 5. The existing continuous controls and discrete event decisions are designed based on the knowledge of the plant dynamics and reward that the HDS obtains at each event. However, these designs can not be adaptive to the scenario when the HDS comes to an unmodeled event or disturbance, or when the parameters describing the plant are incorrect. Using the same control and decision design would result in a poor performance. In this paper the advantages of HDS and ADP are combined to obtain an adaptive continuous control and set of discrete decision designs subject to minimizing an objective function.

The paper is organized as follows. Section 2 describes the hybrid optimal control problem (HOCP) and assumptions. The background on approximate dynamic programming is reviewed in Section 3. Sections 4 and 5 presents hybrid value functions and the ADP method for HOCP. Conclusions and future work are described in Section 6.

2. HYBRID SYSTEM OPTIMAL PROBLEM FORMULATION

The hybrid optimal control problem (HOCP) arises from a variety of fields, such as mobile manipulator systems, unmanned robotic sensor planning, and autonomous assemble lines. In these applications, the discrete decisions, or actions, and continuous controls are crucial to the performance of the hybrid system. For example, in the mobile manipulator system, the performance is determined by the decisions on the manipulator actions, and by the continuous controls to the manipulator to finish the actions. A well-known three-layer hybrid framework[1] is adopted to characterize the structure of the HOCP. As shown in Fig. 1, the first layer contains a discrete event controller (DEC) which provides the discrete decision or action, $\mathbf{a} \in \mathcal{A} \subset \mathbb{R}$, to the second layer based on event, $\boldsymbol{\xi} \in \mathcal{E} \subset \mathbb{R}$, which is given by the second layer. In this framework, $\mathcal{A}$ is the 1-dimensional set of admissible discrete action inputs, while $\mathcal{E}$ is a 1-dimensional finite discrete set of events. The second layer functions as a discrete-continuous interface, mapping between the continuous state, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^n$, and the discrete event space, $\mathcal{E}$, where, $\mathcal{X}$ is an n-dimensional continuous state space. The interface provides parameters to the third layer, which consists of a continuous state plant (CSP) and a continuous controller (CSC). The controller gives the continuous control, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$ to the system, where, $\mathcal{U}$ is the m-dimensional space of admissible continuous control inputs. It is assumed that the discrete event $\boldsymbol{\xi}$ and continuous state $\mathbf{x}$ are fully observable without error. The second layer and the third layer are combined and denoted as a discrete event plant (DEP). Then, the DEC provides the discrete event action $\mathbf{a}(k)$ to the DEP based on the discrete event variable $\boldsymbol{\xi}(k)$ given by DEP, where k is the event serial number. The DEC providing the control for kth event $\boldsymbol{\xi}$ is defined as a function of $\boldsymbol{\xi}(k)$, and is denoted as

$$\mathbf{a}(k) = \phi[\boldsymbol{\xi}(k)] \tag{1}$$

where $\phi : \mathcal{E} \to \mathcal{A}$. In the mobile manipulator system , DEC gives the decision of which action to take at event $\boldsymbol{\xi}(k)$. Due to uncertainty of each action benefit, DEC is learned and adapted online.

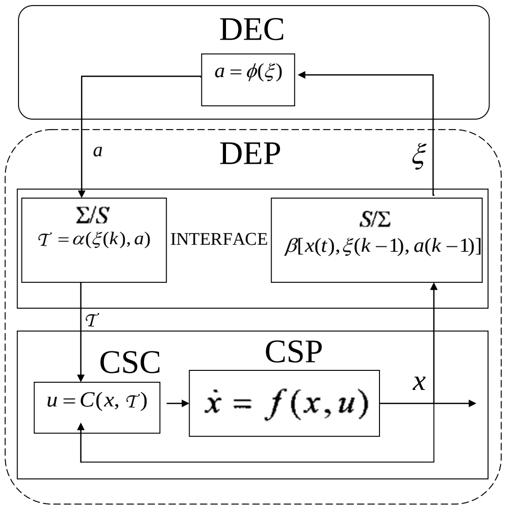

Figure 1. Hybrid System

According to the hybrid system, the time horizon $[t_0,\ \infty]$ can be partitioned by $t_1 < t_2 < \cdots$ into time intervals $\Delta_0, \Delta_1, \cdots$, such that $\Delta_k = (t_k, t_{k+1}]$, where t_k is the time when kth event $\boldsymbol{\xi}(k)$ happens. Before defining the interface in the second layer, let $\mathscr{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_N\}$ denote the set of simply connected convex subsets of $\mathcal{X}$, such that $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset, \forall\, i, \forall\, j,\ i \neq j$. Each $\mathcal{T}_i$ represents a subspace of interest in $\mathcal{X}$ that the system intends to visit. In the problem of controlling a mobile manipulator system, the reward of visiting $\mathcal{T}_i$ is obtained when $\mathbf{x} \in \mathcal{T}_i$. Similar to the internally forced switching system,[22] the kth event value $\boldsymbol{\xi}(k)$ is triggered when the system continuous state visits a subset $\mathcal{T} \in \mathscr{T}/\alpha[\boldsymbol{\xi}(k-1)]$, i.e., $\mathbf{x}(t) \in \mathcal{T}$, and the value is given as $\alpha^{-1}(\mathcal{T})$. Where, $\alpha :\ \mathcal{E} \to \mathscr{T}$ is a bijection, the operator "/" denotes the complementary set of $\alpha[\boldsymbol{\xi}(k-1)]$, and $t > t_{k-1}$. The following assumption is adopted to guarantee the value of $\boldsymbol{\xi}(k)$ is determined by the discrete event decision $\mathbf{a}(k-1)$ given $\boldsymbol{\xi}(k-1)$, i.e.,

$$\boldsymbol{\xi}(k) = \psi[\boldsymbol{\xi}(k-1), \mathbf{a}(k-1)] \tag{2}$$

Assumption 2.1. *$\exists$ $\mathbf{u}(\cdot)$ over Δ_{k-1}, s.t., $\mathbf{x}(t_k) \in \alpha[\psi(\boldsymbol{\xi}(k-1), \mathbf{a}(k-1))]$. $\forall$ $t_{k-1} < t < t_k$,$\nexists$ $\mathbf{u}(\cdot)$, s.t., $\mathbf{x}(t) \in \mathscr{T}/\alpha[\boldsymbol{\xi}(k-1)]$.* According to assumption (2.1), the time (t_k) when the kth event $\boldsymbol{\xi}(k)$ happens is determined by the continuous control $\mathbf{u}(t)$, $t \in \Delta_{k-1}$, given t_{k-1}.

From above definitions, the interface in the second layer can be fully defined. The interface includes two mappings: a) the mapping from the discrete event set $\mathcal{E}$ to the continuous space $\mathcal{X}$, Σ/S: α; b) the mapping from the continuous space $\mathcal{X}$ to the event set $\mathcal{E}$, S/Σ: β, that are defined as

$$\Sigma/S: \quad \mathcal{T} = \alpha[\psi(\boldsymbol{\xi}(k-1), \mathbf{a}(k-1))] \tag{3}$$

$$S/\Sigma: \quad \beta[\mathbf{x}(t), \boldsymbol{\xi}(k-1), \mathbf{a}(k-1)] = \begin{cases} \boldsymbol{\xi}(k) = \alpha^{-1}(\mathcal{T}) & \text{if } \mathbf{x}(t) \in \mathcal{T} \\ \boldsymbol{\xi}(k-1) & \text{if } \mathbf{x}(t) \notin \mathcal{T} \end{cases} \tag{4}$$

The discrete event decision $\boldsymbol{\xi}$ is determined by the event value $\boldsymbol{\xi}$ given by the interaction between $\mathbf{x}$ and $\mathcal{T}$, while the continuous control $\mathbf{u}$ for CSP is calculated based on $\mathcal{T}$, continuous state $\mathbf{x}$, which will be explained later.

In the third layer, the CSP is described by a dynamic function in the continuous space, $\mathcal{X}$, with continuous control input $\mathbf{u}$, and it is defined as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{5}$$

which assumes that the dynamics of CSP stays the same for any $\mathbf{x} \in \mathcal{X}$ for the sake of simplicity. This assumption can be relaxed by defining a dynamic function set, $\{f_{\boldsymbol{xi}}\}$, one of which is used according to the event value $\boldsymbol{\xi}$. As shown in Fig. 1, the CSC provides $\mathbf{u}$ to the CSP based on the $\mathcal{T}$ and the current CSP state $\mathbf{x}(t)$ to bring $\mathbf{x}$ to $\mathcal{T}$, and is given by

$$\mathbf{u}(t) = B(\mathbf{x}(t), \mathcal{T}). \tag{6}$$

According to (1), $\mathcal{T}$ is a function of $\boldsymbol{\xi}(k)$ and $\mathbf{a}(k)$ when $t \in \Delta_k$, then the function (6) can be written as

$$\begin{aligned} \mathbf{u}(t) &= B[\mathbf{x}(t), \mathcal{T}] \\ &= B[\mathbf{x}(t), \alpha(\phi(\boldsymbol{\xi}(k), \mathbf{a}(k)))] \\ &= C[\mathbf{x}(t), \boldsymbol{\xi}(k), \mathbf{a}(k)] \end{aligned} \tag{7}$$

which is designed such that $\mathbf{u}$ leads the system state $\mathbf{x}$ to the subset of interest, $\mathcal{T}$, given by $\alpha[\phi(\boldsymbol{\xi}(k), \mathbf{a}(k))]$. At the same time, the continuous control $\mathbf{u}$ (7), together with discrete event decision $\mathbf{a}$ (1), are optimized in order to minimize an objective function, which includes the integration of the cost regarding continuous state $\mathbf{x}(t)$ and control $\mathbf{u}(t)$ over $[t_0,\ \infty]$, and the summation of all event costs considering $\boldsymbol{\xi}$ and $\mathbf{a}$.

In this paper, a widely used objective function for HOCP[23, 24] is adopted and is given by

$$J \triangleq \int_{t_0}^{\infty} \gamma^k \mathscr{L}[\mathbf{x}(\tau), \mathbf{u}(\tau)]d\tau + \sum_{k=1} \gamma^k L[\boldsymbol{\xi}(k), \mathbf{a}(k)] \tag{8}$$

subject to the hybrid system illustrated in Fig. 1. The power k of discount factor γ, which is positive and less than 1, for $\mathscr{L}$ is determined by subscript of Δ_k that τ belongs to. The cost evaluation functions, $\mathscr{L}$ and L, in equation (8) are the functions of the continuous state (discrete event) and the continuous control (discrete decision), and they are defined separately as

$$\mathscr{L}(\mathbf{x}, \mathbf{u}) \triangleq \mathbf{x}^T\mathbf{H}\mathbf{x} + 2\mathbf{x}^T\mathbf{P}\mathbf{u} + \mathbf{u}^T\mathbf{Q}\mathbf{u} \tag{9}$$

$$L(\boldsymbol{\xi}, \mathbf{a}) \triangleq +\theta(\boldsymbol{\xi}, \mathbf{a}) \tag{10}$$

where, $\mathbf{H}$, $\mathbf{P}$, and $\mathbf{Q}$ are predefined matrices, and $\theta : \mathcal{E} \times \mathcal{X} \to \mathbb{R}$. In the mobile manipulator system, $\theta(\boldsymbol{\xi}, \mathbf{a})$ can be defined as the amount of product transported due to the event $\boldsymbol{\xi}$ and $\mathbf{a}$. In this paper, it is assumed that functions α, ψ and f are known. Generally, the HOCP is given as:

Problem 2.2. *Find the optimal continuous state control $\mathbf{u}$ and the discrete event decision $\mathbf{a}$, such that J is minimized given initial $\mathbf{x}$ and $\boldsymbol{\xi}$, subject to the hybrid system in Fig. 1.*

One of the applications involves controlling the mobile manipulator system. This example is used in the simulation section to demonstrate the method proposed in this paper. In this scenario, the $\mathscr{L}$ in (8) is considered as the energy cost for traveling, while L is considered is the negative reward by transporting products. For simplicity, the robot is a point mass and its state consists of position and velocity $[x, y, \dot{x}, \dot{y}]$. Let $T_i \subset \mathbb{R}^2$ denote the projection of $\mathcal{T}_i \in \mathscr{T}$ on to the $x - y$ plane. While $\mathcal{T}_i = T_i \times [-v_{max}\ v_{max}] \times [-v_{max}\ v_{max}]$, where v_{max} is positive and predefined. The setting of $\mathcal{T}_i$ comes from the applications where the product carried by the robot can only be examined when the robot's speed is under a threshold. The ADP designed to learn the optimal control for the mobile manipulators is discussed in Section 4 and 5, while the background on ADP is given in next section.

3. BACKGROUND ON APPROXIMATE DYNAMIC PROGRAMMING

Approximate dynamic programming (ADP)[10, 25] was proposed to solve the dimensionality curses that dynamic programming (DP)[18, 19] and Markov decision process (MDP)[18, 26] are facing when the system dimension is high or the system state is continuous and fine discretization is required. DP solves a wide spectrum of optimal control problems by backward methods for a finite time problem. MDP can obtain the optimal decision for each discrete state by the well known value iteration or policy iteration method for an infinite time problem. ADP, DP and MDP adopt the Bellman equation to describe the objective function (8) in recursive form by discretizing the time horizon. For a non-hybrid optimal control problem, the discrete value function used in the Bellman equation is given as

$$V(\kappa) \triangleq \sum_{j=\kappa} \gamma^{j-\kappa} \mathscr{L}[\mathbf{x}(j), \mathbf{u}(j)] \tag{11}$$

where j and κ are time indices from discretizing the time horizon, while $\mathscr{L}$ and γ are a quadratic function and discount factor, respectively. Let $V^*(\kappa)$ denote the optimal value function conditioned on optimal control. Then, using the Bellman equation, the recursive form of $V^*(\kappa)$, is written as

$$V^*(\kappa) = \mathscr{L}[\mathbf{x}(\kappa), \mathbf{u}(\kappa)] + \gamma \sum_{j=\kappa+1} \gamma^{j-\kappa-1} \mathscr{L}[\mathbf{x}(j), \mathbf{u}(j)] \tag{12}$$

$$= \mathscr{L}[\mathbf{x}(\kappa), \mathbf{u}(\kappa)] + V^*(\kappa+1) \tag{13}$$

where $\mathscr{L}[\mathbf{x}(\kappa), \mathbf{u}(\kappa)]$ is considered as the instant reward or cost. The DP method employs a backward procedure to solve the Bellman equation for a finite time problem given a initial system state and a goal state. Without losing generality, $V^*(\kappa_f)$ of the goal state is always set as zero, and is back propagated into (12) to obtain the optimal value $V^*(\kappa)$, as well as the decision associated with each time step κ. The DP algorithm requires the perfect knowledge of system dynamics and the environment (which is usually static). As a result, it can not be applied to the online problem when the value function V^* is impossible to be calculated before the decisions or the controls are executed and the environment is explored, sometimes, as well as the system itself, due to the uncertainty of the system model. Furthermore, at each time step κ, the value of $V^*(\kappa)$ on each possible state $\mathbf{x}$ is needed, which results in a curse of dimensionality when n is large. The DP algorithm can save some computational burden by cutting the unnecessary choice of states, however, it still becomes intractable when the $\mathbf{x}$ is continuous and a fine time discretization is needed. Similar to DP, MDP also suffers from the dimensionality curses since it forms a state vector which includes all the possible system states and a state transition matrix of system state, which is conditioned on decisions.

The ADP method can solve the curses of dimensionality by approximating the value function (or the gradient of the value function regarding the state) with fewer parameters than the DP or MDP uses, through a multiple layer structure, such as aggregation functions,[27] or through a reinforcement learning technique,[28] such as Q-learning,[29] supporting vector machine,[30] and neural network.[31] The sigmoid neural network learning technique[32] is used in Section 5 to approximate the gradient of the value function regarding the state, namely, the critic network,[33] with which an auxiliary neural network can be trained to obtain the continuous control $\mathbf{u}$ given the state $\mathbf{x}$ and the system dynamics. Another advantage of ADP is its online learning and adaptive ability, since it uses a forward algorithm,[10] and updates the critic network, as well as the controls, based on the difference between the reward (or cost) obtained and the reward (or cost) expected. Furthermore, the ADP can optimize the controls without having perfect knowledge of the system itself, and overcome the uncertainty of the environment. One ADP algorithm based on Q-learning[10] is summarized in the Algorithm 1 to learn the value function through iterations. Let $V^i(\mathcal{X})$ denote the value for all states $\mathbf{x} \in \mathcal{X}$ with the assumption that $\mathcal{X}$ is countable. The index i denotes the serial number of ith iteration. The algorithm starts with an initial guess of the value function, possibly generated by a potential function.[34] In each iteration, an initial state is randomly generated, and then a state trajectory is calculated by solving the Bellman equation. After that, with the information obtained along the trajectory, the value of $V(\mathbf{x}^i(t))$ for each visited state can be calculated. At last, the value of $V(\mathbf{x}^i(t))$ is updated by Q-learning method, as shown in Algorithm 1, where α_{i-1} is the learning rate which is a function of time.

This algorithm can be extended to the case where the $\mathcal{X}$ is continuous by adopting a neural network[31] to approximate the value function. The Algorithm in 1 can be modified and extended to solve an online optimal problem by replacing "Randomly choose initial state $\mathbf{x}_0^i$" as "current state $\mathbf{x}$" following the Gauss-Seidel variation.[10] The ADP method introduced in this section is for solving non-hybrid optimal control problems. In Sections 4, the hybrid value functions and their recurrence relations are developed for HOCP, while in Section 5, the MDP and ADP method are mixed to obtain the discrete event decisions $\mathbf{a}$ and the continuous controls $\mathbf{u}$.

Algorithm 1 ADP algorithm based on Q-learning

```
Require: Initialize $V^i(\mathcal{X})$ and set $i = 1$
  while $i \leq N_{max}$ do
    Randomly choose initial state $\mathbf{x}_0^i$.
    Solve : $V^i(t) = \max_{\mathbf{u}_t}\{\delta t \mathscr{L}[\mathbf{x}(t), \mathbf{u}(t)] + V^i(t+\delta t)\}$
    Record $\mathbf{x}^i(t)$ visited
    i=i+1;
    Update $V^i(\mathcal{X})$ as $V^i(\mathbf{x}) = \begin{cases} (1-\alpha_{i-1})V^{i-1}(\mathbf{x}^i(t)) + \alpha_{i-1}V^i(t) & \text{if } \mathbf{x} = \mathbf{x}^i(t) \\ (1-\alpha_{i-1})V^{i-1}(\mathbf{x}) & \text{otherwise} \end{cases}$
  end while
```

4. HYBRID VALUE FUNCTION

From Section 3, a recursive value function is required by the ADP method, hence, in this section, a hybrid value function and its recursive form are developed for the hybrid system, and will be used in section 5 to learn the controls $\mathbf{a}$ and $\mathbf{u}$. As explained in Section 2, the time horizon $[t_0,\ \infty]$ can be partitioned into time intervals $\Delta t_k = [t_k,\ t_{k+1})$, therefore, it's reasonable to rewrite the objective function as

$$J = \sum_{k=1} \gamma^k \left\{ \int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau), \mathbf{u}(\tau)] d\tau + L[\boldsymbol{\xi}(k), \mathbf{a}(k)] \right\} \tag{14}$$

Before applying approximate dynamic programming, we have several remarks and a further assumption to decompose the HOCP into subproblems.

Remark 4.1. *The functions (2) indicates that the set* $\{\boldsymbol{\xi}(k-1), \boldsymbol{\xi}(k-2), \ldots, \boldsymbol{\xi}(1)\}$ *is determined by the discrete event decision set* $\{\mathbf{a}(k-1), \mathbf{a}(k-2) \ldots, \mathbf{a}(0)\}$ *given the initial* $\boldsymbol{\xi}(0)$

Remark 4.2. *Following remark 4.1,* $\{\mathcal{T}(k), \mathcal{T}(k-1), \ldots, \mathcal{T}(1)\}$ *is also determined by the discrete event decision set* $\{\mathbf{a}(k), \mathbf{a}(k-1) \ldots, \mathbf{a}(0)\}$ *given the initial* $\boldsymbol{\xi}(0)$*. Notice that* $\mathbf{x}(t_k) \in \mathcal{T}(k)$.

Before introducing the assumption, let $D(\mathcal{T}_i,\ \mathcal{T}_j)$ define the distance between two subset $\mathcal{T}_i$ and $\mathcal{T}_j$ as

$$D(\mathcal{T}_i,\ \mathcal{T}_j) = \min \|\mathbf{x_i} - \mathbf{x_j}\|_\infty, \mathbf{x_i} \in \mathcal{T}_i, \mathbf{x_j} \in \mathcal{T}_j \tag{15}$$

where $\|\cdot\|_\infty$ denote the infinite norm, and let $d(\mathcal{T})$ define the diameter of a subset $\mathcal{T}$

$$d(\mathcal{T}) = \max \|\mathbf{x_i} - \mathbf{x_j}\|_\infty, \mathbf{x_i} \in \mathcal{T}_i, \mathbf{x_j} \in \mathcal{T}_i \tag{16}$$

Furthermore, Let $c(\mathcal{T})$ denote the center of the inscribed sphere of $\mathcal{T}$. To further approximate the problem, the following assumption is adopted.

Assumption 4.3. *$\forall\ \boldsymbol{\xi}$, $\forall\ \mathbf{a}$, $d[\alpha(\boldsymbol{\xi})]/D[\alpha(\boldsymbol{\xi}),\alpha(\phi(\boldsymbol{\xi},\mathbf{a})] < \eta$, where η is sufficient small, such that $\mathbf{x}(t_k) \approx c(\mathcal{T}(k))$.* Then, with remark 4.2, we have the following approximations

$$\{\mathbf{x}(k),\mathbf{x}(k-1),\ldots,\mathbf{x}(2)\} \approx \{c(\mathcal{T}(k)),c(\mathcal{T}(k-1)),\ldots,c(\mathcal{T}(2))\} \tag{17}$$

and the value of set $\{\mathbf{x}(k),\mathbf{x}(k-1),\ldots,\mathbf{x}(2)\}$ is determined by the discrete event decision set $\{\mathbf{a}(k),\mathbf{a}(k-1)\ldots,\mathbf{a}(1)\}$ given the initial $\boldsymbol{\xi}(1)$. Therefore,

Remark 4.4. *Given $\boldsymbol{\xi}(k)$ and $\mathbf{a}(k)$, the value of $\int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau$ is a function of $\mathbf{u}(t),\mathbf{x}(t)$ with $t \in \Delta_k$, and it is irrelevant to $\mathbf{u}(t),\mathbf{x}(t)$ with $t \notin \Delta_k$.*

Since the hybrid nature of HOCP, its value function also has a hybrid structure based on $\boldsymbol{\xi}$ and $\mathbf{a}$. To simplify the problem, let G denote the first value function of $\boldsymbol{\xi}(k)$, defined as

$$G[\boldsymbol{\xi}(k)] \triangleq \sum_{j=k} \gamma^{j-k}\left\{\int_{t_j}^{t_{j+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau + L[\boldsymbol{\xi}(j),\mathbf{a}(j)]\right\} \tag{18}$$

Let $\mathbf{u}(t_i:t_j)$ denote the continuous state control sequence from t_i to t_j, where $i<j$. While, let $\mathbf{a}(i:j)$ denote the discrete event decision sequence, where $i<j$. With remark 4.4, the optimal value function G^* and its recursive form are defined and obtained separately by

$$\begin{aligned} G^*[\boldsymbol{\xi}(k)] &= \min_{\mathbf{a}(k:\infty),\mathbf{u}(t_k:\infty)} \sum_{j=k} \gamma^{j-k}\left\{\int_{t_j}^{t_{j+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau + L[\boldsymbol{\xi}(j),\mathbf{a}(j)]\right\} && (19)\\ &= \min_{\mathbf{a}(k),\mathbf{u}(t_k:\infty)} \left\langle \int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau + L[\boldsymbol{\xi}(k),\mathbf{a}(k)] \right.\\ &\qquad \left. +\gamma \min_{\mathbf{a}(k+1:\infty),\mathbf{u}(t_{k+1}:\infty)} \sum_{j=k+1} \gamma^{j-k-1}\left\{\int_{t_j}^{t_{j+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau + L[\boldsymbol{\xi}(j),\mathbf{a}(j)]\right\}\right\rangle\\ &= \min_{\mathbf{a}(k),\mathbf{u}(t_k:\infty)} \left\{\int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau + L[\boldsymbol{\xi}(k),\mathbf{a}(k)] + \gamma G^*[\boldsymbol{\xi}(k+1)]\right\} && (20)\end{aligned}$$

Again by remark 4.4, $\mathbf{u}(t_k:t_{k+1})$ only have impact on $\mathbf{x}$ in Δ_k given $\mathbf{a}(k)$ and $\boldsymbol{\xi}(k)$, then the recursive form of G^* can be rewritten as

$$G^*[\boldsymbol{\xi}(k)] = \min_{\mathbf{a}(k)}\left\{\gamma G^*[\boldsymbol{\xi}(k+1)] + L[\boldsymbol{\xi}(k),\mathbf{a}(k)] + \min_{\mathbf{u}(t_k:t_{k+1})|\mathbf{a}(k),\boldsymbol{\xi}(k)} \int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau\right\} \tag{21}$$

The value of the third term in equation (21), $\min_{\mathbf{u}(t_k:t_{k+1})|\mathbf{a}(k),\boldsymbol{\xi}(k)} \int_{t_k}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau$, is required in order to evaluate the $G^*[\boldsymbol{\xi}(k)]$, and it is determined by the $\mathbf{u}(t_k:t_{k+1})$ given $\boldsymbol{\xi}(k)$ and $\mathbf{a}(k)$. Therefore a second value function V of $\mathbf{x}(t)$,$\boldsymbol{\xi}(k)$, and $\mathbf{a}(k)$ is defined as

$$V[\mathbf{x}(t)|\boldsymbol{\xi}(k),\mathbf{a}(k)] \triangleq \int_{t}^{t_{k+1}} \mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau \tag{22}$$

where, $t\in\Delta_k$. The notation $V[\mathbf{x}(t)|\boldsymbol{\xi}(k),\mathbf{a}(k)]$ for the value function V is used instead of $V[\mathbf{x}(t),\boldsymbol{\xi}(k),\mathbf{a}(k)]$ in order to emphasize the conditionality on $\boldsymbol{\xi}(k)$ and $\mathbf{a}(k)$. To calculate the integral in (22), the time interval Δ_k is further discretized into even time intervals with ϵ length, such that, $\epsilon \ll (t_{k+1}-t_k),\ \ \forall\ k$. Let $\kappa\in\{1,2,\ldots,N_k\}$ denote the instant $t_k+\epsilon(\kappa-1)$, where $N_k=(t_{k+1}-t_k)/\epsilon$. By discretizing the time horizon, the value function (22) can be rewritten as

$$V[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)] = \epsilon\sum_{j=\kappa}^{N_k} \mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] \tag{23}$$

and the optimal value function V^* and its recursive form are given by

$$V^*[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)] = \min_{\mathbf{u}(\kappa:N_k)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \epsilon \sum_{j=\kappa}^{N_k} \mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] \tag{24}$$

$$\begin{aligned} &= \min_{\mathbf{u}(\kappa:N_k)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \left\{ \epsilon\mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] + \epsilon \sum_{j=\kappa+1}^{N_k} \mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] \right\} \\ &= \min_{\mathbf{u}(\kappa)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \left\{ \epsilon\mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] + \min_{\mathbf{u}(\kappa+1:N_k)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \epsilon \sum_{j=\kappa+1}^{N_k} \mathscr{L}[\mathbf{x}(j),\mathbf{u}(j)] \right\} \\ &= \min_{\mathbf{u}(\kappa)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \{\epsilon\mathscr{L}[\mathbf{x}(t),\mathbf{u}(t)] + V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\} \end{aligned} \tag{25}$$

According to the problem formulation (2), the value functions G (24) and V (21) are not known, due to lack of the knowledge about the environment regarding subsets of interests or parameters related to energy cost, as well as due to the uncertainty of the hybrid system itself. Therefore, the continuous control (7) and discrete decision (1) can not be obtained ahead of time. In the next section, the ADP method is employed to learn the value functions implicitly and control laws while exploring the environment and gaining the knowledge of the hybrid system.

5. RECURRENCE RELATIONS FOR HYBRID APPROXIMATE DYNAMIC PROGRAMMING

In this section, the recurrence relations for hybrid approximate dynamic programming are developed. From the analysis from previous section, the continuous optimal control is given as

$$C^*[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)] = \text{argmin}_{\mathbf{u}(\kappa)|\mathbf{a}(k),\boldsymbol{\xi}(k)} \{\epsilon\mathscr{L}[\mathbf{x}(t),\mathbf{u}(t)] + V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\} \tag{26}$$

which can be obtained by setting

$$\frac{\partial V^*(\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k))}{\partial \mathbf{u}(t)} = \frac{\partial \{\epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)] + V^*(\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k))\}}{\partial \mathbf{u}(\kappa)} = 0 \tag{27}$$

While the optimal discrete decision is given as

$$\begin{aligned} \phi^*[\boldsymbol{\xi}(k)] &= \text{argmin}_{\mathbf{a}(k)} \left\{ \gamma G^*[\boldsymbol{\xi}(k+1)] + L[\boldsymbol{\xi}(k),\mathbf{a}(k)] + \min_{\mathbf{u}(t_k:t_{k+1})|\mathbf{a}(k),\boldsymbol{\xi}(k)} \int_{t_k}^{t_{k+1}} \epsilon\mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau \right\} \\ &= \text{argmin}_{\mathbf{a}(k)} \{\gamma G^*[\boldsymbol{\xi}(k+1)] + L[\boldsymbol{\xi}(k),\mathbf{a}(k)] + V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\} \end{aligned} \tag{28}$$

which can be solved by MDP when V^* is available. In the remainder of this section, the methods to learn the value function V^* and continuous optimal control law C^* are presented first by introducing recurrence relations. Then, the discrete decision law ϕ^* is obtained based on the knowledge of V^*.

The continuous control sequence $\mathbf{u}(t_k : t_{k+1})$ is optimized to minimize the value function $V[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),\mathbf{a}(k)]$, where t_{k+1} is free, since it is determined by the $\mathbf{x}(t)$ and $\mathcal{T}$, according to the interface definition (4). With remark (4.2), the subproblem in Δ_k becomes a open end time problem,[35] where the initial state is given by $c[\alpha(\boldsymbol{\xi}(k))]$ and the terminal state is given by $\alpha[\psi(\boldsymbol{\xi}(k),\mathbf{a}(k))]$. Expand the derivative given in (27) as

$$\begin{aligned} \frac{\partial \{\epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)] + V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\}}{\partial \mathbf{u}(\kappa)} &= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)} + \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{u}(\kappa)} \\ &= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)} + \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa+1)} \frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{u}(\kappa)} \\ &= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)} + \lambda[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)] \frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{u}(\kappa)} \end{aligned} \tag{29}$$

where $\lambda[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]$ denotes $\frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa+1)}$, which is calculated by

$$\lambda[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)] = \frac{\partial V^*[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa)} \tag{30}$$

$$= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{x}(\kappa)} + \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa)} \tag{31}$$

$$\begin{aligned}
&= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{x}(\kappa)} + \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)}\frac{\partial \mathbf{u}(\kappa)}{\partial \mathbf{x}(\kappa)} + \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa+1)}\frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{x}(\kappa)} \\
&\quad + \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa+1)}\frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{u}(\kappa)}\frac{\partial \mathbf{u}(\kappa)}{\partial \mathbf{x}(\kappa)} \\
&= \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{x}(\kappa)} + \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)}\frac{\partial \mathbf{u}(\kappa)}{\partial \mathbf{x}(\kappa)} + \lambda[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{x}(\kappa)} \\
&\quad + \lambda[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{u}(\kappa)}\frac{\partial \mathbf{u}(\kappa)}{\partial \mathbf{x}(\kappa)}
\end{aligned} \tag{32}$$

With the recurrence relations shown in equation (30), a sigmoid neural network, denoted as NN_λ, can be trained to approximate $\lambda[\mathbf{x}|\boldsymbol{\xi}(k),\mathbf{a}(k)]$. By applying the least mean square (LMS)[32] training algorithm, the update for weights of NN_λ, w_λ, is given as

$$\Delta w_\lambda = -\varepsilon\left(\lambda[\mathbf{x}(\kappa)|\boldsymbol{\xi}(k),\mathbf{a}(k)] - \frac{\partial V^*[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial \mathbf{x}(\kappa)} - \frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{x}(\kappa)}\right)\frac{\partial \lambda[\mathbf{x}(\kappa),\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial w_\lambda} \tag{33}$$

where ε is the learning rate, and the term in parentheses is obtained by subtracting the RHS from the LHS of (30). Then a second neural network, denoted as $\text{NN}_\mathbf{u}$ is used to approximate $\mathbf{u} = C(\mathbf{x}(\kappa),\boldsymbol{\xi}(k),\mathbf{a}(k))$ with criteria that the derivative (29) equals zero. Again, by applying the LMS training algorithm, the update for weights of NN_u, w_u, is given as

$$\Delta w_u = -\rho\left(\frac{\partial \epsilon\mathscr{L}[\mathbf{x}(\kappa),\mathbf{u}(\kappa)]}{\partial \mathbf{u}(\kappa)} + \lambda[\mathbf{x}(\kappa+1)|\boldsymbol{\xi}(k),\mathbf{a}(k)]\frac{\partial \mathbf{x}(\kappa+1)}{\partial \mathbf{u}(\kappa)}\right)\frac{\partial C[\mathbf{x}(\kappa),\boldsymbol{\xi}(k),\mathbf{a}(k)]}{\partial w_u} \tag{34}$$

where ρ is the learning step. The training of NN_λ and NN_u are executed spontaneously, the corresponding algorithm is summarized in Algorithm 2.[36]

Algorithm 2 Train NN_λ and NN_u

```
Require: Initialize $\lambda$ and $C(\mathbf{x},\boldsymbol{\xi},\mathbf{a})$
  while $\mathbf{x}(\kappa) \in \alpha[\psi(\boldsymbol{\xi}(k),\mathbf{a}(k))]$ do
    $\mathbf{u}(\kappa) = C(\mathbf{x}(\kappa),\boldsymbol{\xi}(k),\mathbf{a}(k))$
    $\mathbf{x}(\kappa+1) = f(\mathbf{x}(\kappa),\mathbf{u}(\kappa))$
    $\lambda(\kappa+1) = NN_\lambda(\mathbf{x}(\kappa+1))$
    Update $\text{NN}_\lambda$ and $\text{NN}_u$ according to (33) and (34)
    $\kappa = \kappa+1$
  end while
```

The subproblem can be solved by training NN_λ and NN_u, however, the value $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),(a)(k)]$ is still needed for solving the optimal discrete event decision. Due to the uncertainty of the environment of the system or the lack of the system model, the value of $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),(a)(k)]$ is calculated based the executed controls and the visited state by

$$V[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),\mathbf{a}(k)] \triangleq \int_{t_k}^{t_{k+1}} \epsilon\mathscr{L}[\mathbf{x}(\tau),\mathbf{u}(\tau)]d\tau \tag{35}$$

With the knowledge of $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),(a)(k)]$, the MDP is adopted to obtain and update the decision policy $\phi(\boldsymbol{\xi})$. According to the recurrence relations of the optimal value function G^* (19), the terms $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k),\mathbf{a}(k)]$

and $L[\boldsymbol{\xi}(k), \mathbf{a}(k)]$ can be combined as $\Omega[\boldsymbol{\xi}(k), \mathbf{a}(k)]$. By the remark (4.4) $\mathbf{x}(t_k) \approx c[\alpha(\boldsymbol{\xi}(k))]$. Therefore, the decision $\mathbf{a}$ optimization problem becomes an MDP, given by

$$G^*[\boldsymbol{\xi}(k)] = \min_{\mathbf{a}(k)} \{\Omega[\boldsymbol{\xi}(k), \mathbf{a}(k)] + \gamma G^*[\boldsymbol{\xi}(k+1)]\} \tag{36}$$

Since the size of $\mathcal{E}$ is not large, the value iteration or policy iteration method is sufficient enough to obtain the optimal decision function ϕ when $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k), \mathbf{a}(k)]$ is given. However, since the $V^*[\mathbf{x}(t_k)|\boldsymbol{\xi}(k), \mathbf{a}(k)]$ is learned online and can only be approximated after corresponding state is visited, the discrete event decision policy is myopic; therefore, a random decision is taken to encourage the system explore the unknown space to have better knowledge the system state.

6. CONCLUSION AND FUTURE WORK

The advantage of the hybrid system allows a multiple-layer structure to model the modern unmanned system with the discrete event decisions and the continuous control inputs. The hybrid value functions and their recurrence relations are presented in this paper to obtain the optimal decisions and controls by MDP and ADP. Due to the online nature of ADP, the decisions and controls can adapt to uncertainties of the environment and the hybrid system. The control, the critic neural networks, and decision policy learn newly available information online while retaining their baseline performance. In the future, the method will be tested to solve a mobile manipulator system control problem.

Acknowledgment

This work was supported by NSF grants ECCS #1027775 and ECCS #1028506.

REFERENCES

[1] Fierro, R. and Lewis, F., "A framework for hybrid control design," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* **27**, 765 –773 (nov 1997).

[2] Ferrari, S., Fierro, R., and Tolic, D., "A geometric optimization approach to tracking maneuvering targets using a heterogeneous mobile sensor network," in [*Proc. of the 2009 Conference on Decision and Control*], (2009).

[3] Zavlanos, M. and Pappas, G., "Distributed hybrid control for multiple-pursuer multiple-evader games," in [*Hybrid Systems: Computation and Control*], Bemporad, A., Bicchi, A., and Buttazzo, G., eds., *Lecture Notes in Computer Science* **4416**, 787–789 (2007).

[4] Sanfelice, R. and Frazzoli, E., "A hybrid control framework for robust maneuver-based motion planning," in [*American Control Conference, 2008*], 2254 –2259 (june 2008).

[5] Branicky, M., Borkar, V., and Mitter, S., "A unified framework for hybrid control: model and optimal control theory," *Automatic Control, IEEE Transactions on* **43**, 31 –45 (jan 1998).

[6] Borrelli, F., Baoti, M., Bemporad, A., and Morari, M., "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica* **41**(10), 1709 – 1721 (2005).

[7] Seatzu, C., Corona, D., Giua, A., and Bemporad, A., "Optimal control of continuous-time switched affine systems," *Automatic Control, IEEE Transactions on* **51**, 726 – 741 (may 2006).

[8] Xu, X. and Antsaklis, P. J., "Results and perspectives on computational methods for optimal control of switched systems," in [*Proceedings of the 6th international conference on Hybrid systems: computation and control*], *HSCC'03*, 540–555 (2003).

[9] Christos, Cassandras John, L., [*Stochastic Hybrid Systems*], CRC Press (2006).

[10] Powell, W. B., [*Approximate Dynamic Programming : Solving the Curses of Dimensionality*], Wiley-Interscience (2007).

[11] Powell, W. B., "What you should know about approximate dynamic programming," *Naval Research Logistics (NRL)* **56**(3) (2009).

[12] Murray, J., Cox, C., Lendaris, G., and Saeks, R., "Adaptive dynamic programming," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on* **32**, 140 – 153 (may 2002).

[13] Lai, G., Margot, F., and Secomandi, N., "An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation," *Oper. Res.* **58**, 564–582 (May 2010).

[14] Gao, L. and Zhang, Z., "Approximate dynamic programming approach to network-level budget planning and allocation for pavement infrastructure," in [*Transportation Research Board 88th Annual Meeting*], (2009).

[15] Hugo Simao, W. P., "Approximate dynamic programming for management of high-value spare parts," *Journal of Manufacturing Technology Management* **20**, 147 – 160 (May 2009).

[16] Bellman, R. E. and Dreyfus, S. E., [*Applied Dynamic Programming*], Princeton University Press, Princeton, NJ (1962).

[17] Al-Tamimi, A., Lewis, F., and Abu-Khalaf, M., "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* **38**, 943 –949 (aug. 2008).

[18] Howard, R. A., [*Dynamic Programming and Markov Processes*], The M.I.T. Press (1960).

[19] Bertsekas, D. P., [*Dynamic Programming and Optimal Control, Vols. I and II*], Athena Scientific, Belmont, MA (1995).

[20] Ji, S., Parr, R., and Carin, L., "Nonmyopic multiaspect sensing with partially observable markov decision processes," *IEEE Transactions on Signal Processing* **55**(1), 2720–2730 (2007).

[21] Bishop, C. M., [*Pattern Recognition and Machine Learning (Information Science and Statistics)*], Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006).

[22] Xu, X. and Antsaklis, P. J., "Results and perspectives on computational methods for optimal control of switched systems," in [*Proceedings of the 6th international conference on Hybrid systems: computation and control*], 540–555 (2003).

[23] Hedlund, S. and Rantzer, A., "Optimal control of hybrid systems," in [*IN PROCEEDINGS OF THE 38TH IEEE CONFERENCE ON DECISION AND CONTROL*], 3972–3977 (1999).

[24] Branicky, M., Borkar, V., and Mitter, S., "A unified framework for hybrid control: model and optimal control theory," *Automatic Control, IEEE Transactions on* **43**, 31 –45 (jan 1998).

[25] SI, J., [*Handbook of learning and approximate dynamic programming*], Hoboken, NJ, IEEE Press (2007).

[26] Fox, M., Ghallab, M., Infantes, G., and Long, D., "Robot introspection through learned hidden markov models," *Artificial Intelligence* **170**, 59–113 (2006).

[27] Grabisch, M., Marichal, J.-L., Mesiar, R., and Pap, E., [*Aggregation Functions (Encyclopedia of Mathematics and its Applications)*], Cambridge University Press, New York, NY, USA, 1st ed. (2009).

[28] Kollar, T. and Roy, N., "Trajectory optimization using reinforcement learning for map exploration," *The International Journal of Robotics Research* **27**(2), 175–196 (2008).

[29] Watkins, C. J. C. H. and Dayan, P., "Q-learning," *Machine Learning* **8**, 279–292 (1992).

[30] Russell, S. and Norvig, P., [*Artificial Intelligence A Modern Approach*], Prentice Hall, Upper Saddle River, NJ (2003).

[31] Specht, D. F., "Applications of probabilistic neural networks," *Proceedings of SPIE* **1294**, 344–353 (1990).

[32] F. L. Lewis, S. J. and Yesildirek, A., [*Neural Network Control of Robot Manipulators and Nonlinear Systems*], Taylor & Francis (1999).

[33] Prokorov, D. and Wunsch, D., "Adaptive critic designs," *IEEE transactions on Neural Network* **8**(5), 997–1007 (1997).

[34] Latombe, J. C., [*Robot Motion Planning*], Kluwer Academic Publishers (1991).

[35] Plumer, E., "Optimal control of terminal processes using neural networks," *Neural Networks, IEEE Transactions on* **7**, 408 –418 (mar 1996).

[36] Prokhorov, D. and Wunsch, D.C., I., "Adaptive critic designs," *Neural Networks, IEEE Transactions on* **8**, 997 –1007 (sep 1997).