Discrete Event Dyn Syst
DOI 10.1007/s10626-013-0170-6

A tutorial on event-based optimization—a new optimization framework

Li Xia · Qing-Shan Jia · Xi-Ren Cao

Received: 17 January 2013 / Accepted: 23 August 2013

Abstract In many practical systems, the control or decision making is triggered by certain events. The performance optimization of such systems is generally different from the traditional optimization approaches, such as Markov decision processes or dynamic programming. The goal of this tutorial is to introduce, in an intuitive manner, a new optimization framework called event-based optimization. This framework has a wide applicability to aforementioned systems. With performance potential as building blocks, we develop two intuitive optimization algorithms to solve the event-based optimization problem. The optimization algorithms are proposed based on

This work was supported in part by the 111 International Collaboration Project (B06002), National Natural Science Foundation of China (60704008, 60736027, 61174072, 61203039, 61221003, 61222302, 90924001), Tsinghua National Laboratory for Information Science and Technology (TNList) Cross-discipline Foundation, the Specialized Research Fund for the Doctoral Program of Higher Education (20070003110, 20120002120009). This work was also supported in part by the Collaborative Research Fund of the Research Grants Council, Hong Kong Special Administrative Region, China, under Grant No. HKUST11/CRF/10.

L. Xia (✉) · Q.-S. Jia
Center for Intelligent and Networked Systems (CFINS), Department of Automation, TNList, Tsinghua University, Beijing 100084, China
e-mail: xial@tsinghua.edu.cn

Q.-S. Jia
e-mail: jiaqs@tsinghua.edu.cn

X.-R. Cao
Department of Finance and the Key Laboratory of System Control and Information Processing of Ministry of Education, Department of Automation, Shanghai Jiao Tong University, Shanghai, China
e-mail: xrcao@sjtu.edu.cn

X.-R. Cao
Institute of Advanced Study, Hong Kong University of Science and Technology, Hong Kong, China
e-mail: eecao@ust.hk

Published online: 11 September 2013

an intuitive principle, and theoretical justifications are given with a performance sensitivity based approach. Finally, we provide a few practical examples to demonstrate the effectiveness of the event-based optimization framework. We hope this framework may provide a new perspective to the optimization of the performance of event-triggered dynamic systems.

Keywords Event-based optimization · Performance potential · Discrete-event dynamic systems · Sensitivity-based approach

1 Introduction

In many practical systems, such as engineering, social, and financial systems, the control decisions are only made when certain events happen. This is either because of the discrete nature of the sensor detection and digital computing equipments, or the limitation of computation power, which makes state-based control infeasible due to the huge state spaces.

An event can be understood in the sense of everyday life as something physically happening; e.g., a request arrives at a communication line, a customer arrives at a server, a stock price reaches certain value, a sensor detects an intruding object, an estimated error goes beyond certain level, etc. In admission control, actions (accept or reject) are taken only when a request or customer arrives at a network. In a sensor network, actions are taken only when one of the sensors detects some abnormal situations. In a material handling problem, actions are taken when the inventory level falls below certain threshold. In a stock market, a trader sells or buys stocks when she/he observes that the price of a stock follows some trend; etc.

Mathematically, the concept of an event is the same as the event used in probability. In a stochastic system, an event is a $\mathcal{F}_t$-measurable set in the probability space, with $\mathcal{F}_t$ being the σ field generated by the stochastic process up to time t. Strictly speaking, any such event can be chosen as the event in our sense; but the number of $\mathcal{F}_t$ measurable events is too big (much bigger than the number of states) and choosing such events makes analysis infeasible and defeats our original purpose. Therefore, to properly define events, we need to strike a balance between the generality on one hand and the applicability on the other hand.

In our event-based setting, we formally define an event as a set of state transitions. This definition depends only on the changes happening at a particular time instant, and it does not require recording the history to determine an event and therefore reflects most of the practical situations. It can also be easily generalized to represent a finite sequence of state transitions. We shall see that in many real problems, the number of events that may happen is usually much smaller than that of the states. With this definition, state aggregation and time aggregation (Cao et al. 2002) can be treated as two special cases of the events.

The main difficulty in developing the event-based optimization theory and algorithms lies in the fact that the sequence of events is usually not Markovian. The standard optimization approach such as dynamic programming does not apply to such problems. In Cao (2005, 2007) and Cao and Zhang (2008), the event-based optimization is studied with a sensitivity-based approach. In this approach, performance optimization is based on two fundamental sensitivity equations, one for performance

difference and the other for performance derivative. It has been proved that policy iteration in Markov decision processes (MDPs) and the Hamilton-Jacobi-Bellman (H-J-B) optimality equation can be derived easily from the performance difference equation (Chapter 4 in Cao 2007), and gradient-based optimization (perturbation analysis Cao 1994; Ho and Cao 1991; Leahu et al. 2013; Xia and Cao 2012; Yao and Cassandras 2012, or PA for short, or policy gradient Cao 2007; Marbach and Tsitsiklis 2001) is based on the performance derivative equation (Cao 2007). It has also been shown that most results in MDP, including the bias and n-bias optimality, can be derived with this approach in an intuitive way. Furthermore, the performance difference equation can usually be derived based on intuition and the approach is based on the most basic fact of comparing two objects; this approach can be applied to cases where dynamic programming fails to work.

In this tutorial, we try to explain the event-based optimization algorithms in an intuitive way, and provide theoretical justifications with the sensitivity-based approach. We will also give a few practical examples to illustrate the application of this approach. We work on a discrete time model although a continuous-time counterpart exists. We start in Section 2 with a brief review of the sensitivity based approach applied to the optimization of the long-run average performance for the standard MDP problems. The approach is based on a fundamental quantity called performance potential, and its variation, the Q-factor. We explain the ideas in an intuitive way and a theoretical justification is also provided. In Section 3, we first formulate the event, event-based policy, and event-based optimization. Then, following the same idea as in MDP, we introduce the event-based Q-factor and provide two iterative algorithms. Because of the missing of Markov property, the optimality of these algorithms may not hold exactly. With the performance difference equation, we are able to provide conditions under which the algorithms lead to the exact optimal policy. The algorithms are based on sample paths and therefore is of an online nature. In Section 4, we provide a few examples to show the application of the event-based formulation. In two of the examples, the required condition holds and therefore the algorithms lead to the exact optimal policy. Other examples illustrate how these algorithms perform in cases where the required condition does not hold.

There are a number of advantages of the event-based formulation. First, in many applications, the number of events requiring control decisions is much smaller than that of states. Therefore, computation may be dramatically saved. Second, compared with the state-based formulation, events may capture the structure of a system and therefore may be more efficient by exploring these information. Third, the event-based setting allows actions at different states being correlated, a violation of the assumption of the standard dynamic programming. Finally, this formulation may be applied to a number of existing subjects such as multilevel (hierarchical) control (Parr 1998), state and time aggregation (Cao et al. 2002), options (Barto and Mahadevan 2003), singular perturbation (Altman et al. 2004), and partially observed MDPs (Wang and Cao 2011), etc., by formulating different events to capture the different features of these problems.

The limitation of the approach is that, as will be shown in Section 3, the aggregated potentials in the performance difference equation may depend on both policies under comparison, and therefore, in general it requires some special condition for the approach to be exact. This is an inherited drawback that limits the application of the algorithms, and further research to overcome this limitation is needed.

2 Brief overview of optimization in Markov systems

In this section, we give a brief discussion on how to optimize the performance of a Markov system in a direct and intuitive way. This discussion can be viewed as the motivation of the event-based optimization. The results explained here will be studied in a more rigorous way in the subsequent sections.

2.1 Problem statement and intuitive explanations

We first introduce the notations for a generic discrete time Markov chain. A Markov chain is denoted as $\boldsymbol{X} = \{X_l, l = 0, 1, \dots\}$, where X_l is the system state at time l. The system state is discrete and finite. The state space is defined as $\mathcal{S} := \{1, 2, \dots, S\}$, where S is the size of the state space. The Markov chain transits from state i to state j with probability $p(j|i)$, $i, j \in \mathcal{S}$. The state transition probability matrix is denoted as $P = [p(j|i)]_{i,j\in\mathcal{S}}$. Obviously, we have $Pe = e$, where e is an S-dimensional column vector whose elements are all 1. The steady-state distribution is denoted as an S-dimensional row vector π and it satisfies $\pi P = \pi$. The reward function is an S-dimensional column vector denoted as f. The long-run average performance of the Markov system is denoted as η and for ergodic chains, we have

$$\eta = \pi f = \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l) = \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} E[f(X_l)|X_0], \tag{1}$$

where the last equality holds with probability 1 (w.p. 1) and it is independent of X_0.

In a Markov decision process, at each state i, we can choose an action a from the action space $\mathcal{A}$ and apply it to the system. The action affects the transition probability which is denoted as $p^a(j|i)$, $i, j \in \mathcal{S}$ and $a \in \mathcal{A}$. Let A_l be the action applied to the system at time l. A policy d is defined as a mapping from S to A, i.e., $d : \mathcal{S} \to \mathcal{A}$, which specifies that action $d(i)$ is chosen when the state is i. The policy space is denoted as $\mathcal{D}$. In this paper, we focus the attention on the stationary and deterministic policies, since the optimality always can be attained in such policy space (Puterman 1994). Thus, action a at state i under policy d is also denoted as $a = d(i)$, $i \in \mathcal{S}$, $a \in \mathcal{A}$, and $d \in \mathcal{D}$. A policy affects the value of the transition probability matrix P and the reward function f. The transition probability matrix under policy d is denoted as $P^d = [p^{d(i)}(j|i)]_{i,j\in\mathcal{S}}$. The reward function under policy d is denoted as f^d, which is an S-dimensional column vector and its elements are denoted as $f(i, d(i))$, $i \in \mathcal{S}$. We also assume that the Markov system is always ergodic, i.e.,

Assumption 1 The Markov chains under all policies are ergodic.

The steady-state distribution of the Markov chain under policy d is denoted as π^d, which is an S-dimensional row vector. The long-run average performance is denoted as η^d and we have

$$\begin{aligned} \eta^d = \pi^d f^d &= \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l, d(X_l)) \\ &= \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} E[f(X_l, d(X_l))|X_0], \end{aligned} \tag{2}$$

where the last equality holds as the system is ergodic. The optimization problem is to find the optimal policy $d^* \in \mathcal{D}$ such that the corresponding performance η^{d^*} is maximal among all the policies. That is,

$$d^* = \arg\max_{d \in \mathcal{D}} \eta^d. \tag{3}$$

The problem above is a standard discrete-time infinite-horizon MDP. First, we discuss the situation where the values of P^d and f^d are known. At state i, a different action a induces a different reward $f(i, a)$. A greedy control rule is the one that always chooses the action a that has the maximal reward $f(i, a)$, i.e., $d(i) = \arg\max_{a \in A} f(i, a)$; this is called a *myopic policy*. However, this rule considers only the one-step reward. It does not consider the long-term effect of an action. An action with a good one-step reward may not be good because it may make the system visit the "bad" states more probably in the future. One direct idea is that if we can quantify the long-term effect of a state, not just the one-step reward, then we may get a better policy than the myopic one. As shown in what follows, the *performance potential* is such a quantity.

First, from Eq. 2, one may guess that $f(X_0, A_0) + E[f(X_1, A_1)|X_0]$ is better than $f(X_0, A_0)$ to represent the "long-term contribution" of state X_0 with action A_0 to the performance; and $f(X_0, A_0) + E[f(X_1, A_1)|X_0] + E[f(X_2, A_2)|X_0]$ is even better; and so on. Then, can we use

$$g_N^d(i) = \sum_{l=0}^{N-1} E[f(X_l, A_l)|X_0 = i] \tag{4}$$

for $N \gg 1$ to represent the long-term effect? The answer is YES. In fact, we have the following result (it becomes obvious after reading the next subsection).

Proposition 1 *Given a sample path under any policy d, estimate the $g_N^d(i)$'s for all $i \in \mathcal{S}$ on the sample path, with $N \gg 1$ being a fixed large integer. Suppose that N is sufficiently large and the estimates are accurate enough. Let h be a policy defined as*

$$h(i) := \arg\max_{a \in \mathcal{A}} \left\{ \sum_{j=1}^{S} p^a(j|i) g_N^d(j) + f(i, a) \right\}, \quad \forall\, i \in \mathcal{S}. \tag{5}$$

Then we have $\eta^h \geq \eta^d$.

As we will see in the next subsection, $g_N^d(i)$ is an estimate of the *performance potential* $g^d(i)$, which quantifies the long-term contribution of the initial state i to the average performance η^d. Figure 1 illustrates how $g_4^d(3)$ can be estimated on a sample path; the estimate equals the average of a series of $g_{4,1}^d(3), g_{4,2}^d(3), \ldots$, from the sample path, where $g_{4,k}^d(3)$ denotes the kth sample of $g_4^d(3)$. Other more efficient algorithms for estimating $g_N^d(i)$ exist. Please see the Chapter 3 of Cao (2007) for example.

Proposition 1 indicates that given a non-optimal policy d, we may find a better policy h by analyzing the sample path under policy d. By repeating this improvement

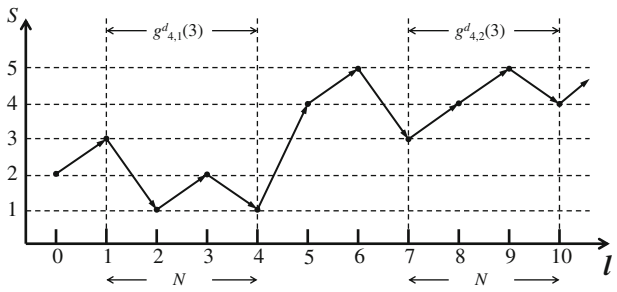


Fig. 1 The estimation of performance potential based on sample path

procedure, we may find an optimal policy. This is the *policy iteration* algorithm of MDP.

Procedure 1 Policy iteration of MDP based on the performance potential estimation.

- Choose an arbitrary policy as the initial policy d.
- For the sample path under policy d, estimate the performance potential based on the formula $g_N^d(i) = E\{\sum_{t=0}^{N-1} f(X_{l+t}, A_{l+t})|X_l = i\}$ for all $i \in \mathcal{S}$, N is fixed and $N \gg 1$. Suppose the estimates of g_N^d's are accurate enough.
- Update the policy as $h(i) := \arg\max_{a\in\mathcal{A}} \left\{\sum_{j=1}^{S} p^a(j|i)g_N^d(j) + f(i,a)\right\}$, for all $i \in \mathcal{S}$, where if $d(i)$ attains the maximum, then set $h(i) := d(i)$.
- If $d = h$, stop and output $d^* := d$; Otherwise, set $d := h$ and go to step 2.

Proposition 2 *When the policy iteration algorithm (Procedure* 1*) stops, it stops at an optimal policy d^*, which satisfies the following H-J-B equation*

$$\sum_{j=1}^{S} p^{d^*(i)}(j|i)g_N^{d^*}(j) + f(i, d^*(i)) = \max_{a\in\mathcal{A}} \left\{\sum_{j=1}^{S} p^a(j|i)g_N^{d^*}(j) + f(i,a)\right\}. \tag{6}$$

Procedure 1 in fact is an implementation of policy iteration for MDP. By Proposition 1, the system performance improves in every iteration. By Proposition 2, if the policy space is finite and the estimates of g_N^d's are accurate enough, the algorithm will stop at the optimal policy within finite number of iterations.

In Procedure 1, we assume that P^d and f^d are known. However, in practice, the structure of the Markov model may be unknown. The optimizer can observe only the sample paths of system state X_l. After an action A_l is adopted, a system reward $f(X_l, A_l)$ will be observed. The optimizer has to find a better policy based on the data observed.

When the values of P^d and f^d are unknown, we cannot directly use Eq. 5 to do policy iteration. However, if we know (e.g., by estimation) the value of

$$Q_N^d(i,a) := \sum_{j=1}^{S} p^a(j|i)g_N^d(j) + f(i,a) \tag{7}$$

for all $a \in \mathcal{A}$, then we can choose $h(i) := \arg\max_{a\in\mathcal{A}} Q_N^d(i,a)$, which is equivalent to Eq. 5. Therefore, the problem becomes how to estimate the value of $Q_N^d(i,a)$.

We may rewrite Eq. 7 as $Q_N^d(i,a) = E\{g_N^d(X_1) + f(X_0, A_0)|X_0 = i, A_0 = a\}$. Since $g_N^d(i) = E\{\sum_{t=0}^{N-1} f(X_{l+t}, d(X_{l+t}))|X_l = i\}$, we have

$$Q_N^d(i,a) = E\left\{ f(X_0, A_0) + \sum_{l=1}^{N} f(X_l, d(X_l)) \Big| X_0 = i, A_0 = a \right\}. \tag{8}$$

With Eq. 8, the sample path based estimation algorithms for $Q_N^d(i,a), i \in \mathcal{S}, a \in \mathcal{A}$, can be developed. However, a sample path of the system under policy d cannot visit all of the state-action pairs (i,a) as d is deterministic (only one action $d(i)$ is chosen when the system is at state i). To estimate $Q_N^d(i,a)$ for all $a \in \mathcal{A}$, we need to use randomizing techniques to make the sample path visit all the possible actions a. One of such techniques is to use the d_ϵ policy, i.e., at state i we adopt the action $a = d(i)$ with probability $1-\epsilon$, and evenly adopt any other actions $a \neq d(i), a \in \mathcal{A}$, with probability $\epsilon/(|\mathcal{A}|-1)$, where ϵ is a very small positive number, $|\mathcal{A}|$ is the size of the action space (by default, $|\mathcal{A}| > 1$). When ϵ is small enough, the sample path under d_ϵ is close to the original deterministic policy d, yet it visits all the possible state-action pairs. The small probability ϵ is added to help exploring the system behavior of other actions.[1]

Thus, a sample path generated by policy d_ϵ visits all the state-action pairs (i,a). We can use the average of the summation of the rewards of the subsequent $N+1$ epochs as an estimate of $Q_N^d(i,a)$ in Eq. 8. When the sample path is long enough, we can obtain accurate estimates for all $Q_N^d(i,a)$'s, $i \in \mathcal{S}$ and $a \in \mathcal{A}$. With the values of $Q_N^d(i,a)$, we can obtain an improved policy h by $h(i) := \arg\max_{a \in \mathcal{A}} Q_N^d(i,a)$, which is similar to the policy iteration. The optimization procedure is summarized as follows.

Procedure 2 Policy iteration of MDP based on Q-factors $Q_N^d(i,a)$.

- Choose an arbitrary policy as the initial policy d.
- At every time l, adopt the action $a = d(X_l)$ with probability $1-\epsilon$, and adopt other actions $a \neq d(X_l)$ and $a \in \mathcal{A}$ with probability $\frac{\epsilon}{|\mathcal{A}|-1}$, where $\epsilon \ll 1$ is a very small positive number.
- Estimate $Q_N^d(i,a) = E\{\sum_{t=0}^{N} f(X_{l+t}, A_{l+t})|X_l = i, A_l = a\}$, for all $i \in \mathcal{S}$ and $a \in \mathcal{A}$, where N is a fixed large integer. Run the sample path long enough to obtain the accurate estimates for all $Q_N^d(i,a)$'s.
- Generate a newly improved policy h as, $h(i) := \arg\max_{a \in \mathcal{A}} Q_N^d(i,a)$ for all $i \in \mathcal{S}$, where if $d(i)$ attains the maximum, then set $h(i) := d(i)$.
- If $d = h$, stop and output d as the optimal policy; Otherwise, set $d := h$ and go to step 2.

Procedure 2 is an alternative version of policy iteration when the forms of P^d and f^d are unknown. The estimation of $Q_N^d(i,a)$ is illustrated by Fig. 2, in which we set $N = 4$. At time $l = 1, 6$, the sample path visits the state $X_l = 3$ and action $A_l = a_1$. We use $Q_{4,1}^d(3,a_1)$ and $Q_{4,2}^d(3,a_1)$ as the estimates of $Q_4^d(3,a_1)$. However,

[1] If policy d is a random policy that picks each action with positive probability (i.e., $d(i)$ is a probability distribution over $\mathcal{A}$ that specifies the probabilities of the actions adopted when state i is visited), then a sample path under policy d visits all the state-action pairs $(i,a), i \in \mathcal{S}, a \in \mathcal{A}$. Therefore, $Q_N^d(i,a)$ for all the state-action pairs $(i,a), i \in \mathcal{S}, a \in \mathcal{A}$, can be estimated without using the randomizing techniques.

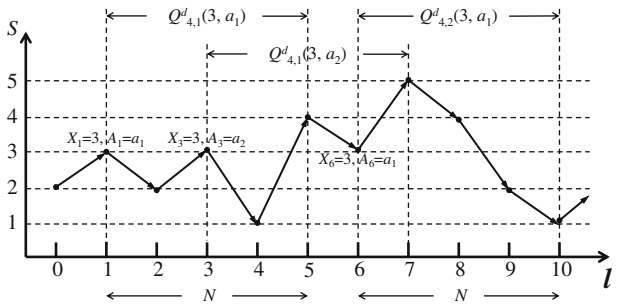


Fig. 2 The estimation procedure of Q-factor $Q^d_N(i, a)$ based on sample path

at time $l = 3$, although the state is $X_l = 3$, the action is $A_l = a_2$ which is not a_1. The corresponding data can be used to estimate $Q^d_4(3, a_2)$, not $Q^d_4(3, a_1)$. (cf. Fig. 1 for the estimation of performance potentials.)

In fact, $Q^d_N(i, a)$ is an estimate of the so-called *Q-factor* $Q^d(i, a)$, which quantifies the long-term effect of the state-action pair (i, a) on the system performance. Thus, the physical meaning of $Q^d_N(i, a)$ is similar to that of the performance potential $g^d_N(i)$. They qualify either the effect of the state, or that of the state-action pair. There are other ways to estimate $Q^d_N(i, a)$. One of them is to use an iterative formula $Q^d_N(X_l, A_l) = f(X_l, A_l) - \eta^d + E\{Q^d_N(X_{l+1}, A_{l+1})\}$, which is usually more efficient than that in Procedure 2. More details can be found in Chapter 6 of Sutton and Barto (1998) and Chapter 6 of Cao (2007). In Procedure 2, we simply use the average as the estimates, because it is directly from the physical meaning of Q-factors, and more efficient algorithms can be developed based on this basic form (see Chapter 6.2 in Cao 2007). Moreover, Procedure 2 is based on a sample path and is online implementable.

2.2 Theoretical justifications and motivations

In this subsection, we provide some theoretical justifications for the optimization procedures intuitively presented in the previous subsection. We use the direct-comparison based approach to justify the reasonability of these optimization procedures (Cao 2007). This approach can be easily extended to the event-based optimization discussed in the next section. For simplicity, we omit some rigorous mathematical proofs. Interested readers can refer to the related literature (see Chapter 4.1 in Cao 2007 for direct-comparison based approach and see Bertsekas and Tsitsiklis 1996; Puterman 1994; Sutton and Barto 1998, for dynamic programming and reinforcement learning).

Procedure 1 is a policy iteration algorithm together with the estimation of performance potentials. Here, we give some brief discussion on the key points. Performance potential $g(i)$ is a fundamental metric in this theory and it quantifies the contribution of an initial state i to the average performance η. The definition of $g^d(i)$ under policy d, $i \in \mathcal{S}$ and $d \in \mathcal{D}$, is

$$g^d(i) := \lim_{L\to\infty} E\left\{\sum_{l=0}^{L}[f(X_l, d(X_l)) - \eta^d]|X_0 = i\right\}, \tag{9}$$

which is the solution to the Poisson equation (with $g^d = (g^d(1), \cdots, g^d(N))^T$)

$$(I - P^d)g^d + \eta^d e = f^d, \tag{10}$$

where I is an identity matrix and e is a column vector whose elements are all 1. Please see page 45 of Cao (2007) for more details.

From Eq. 9, we can use

$$E\left\{\sum_{l=0}^{N-1}[f(X_l, d(X_l)) - \eta^d]|X_0 = i\right\} \tag{11}$$

with $N \gg 1$ as an estimate of $g^d(i)$. Note that $g^d(i)$ is a relative metric, meaning that if g^d satisfies Eq. 10, so does $g^d + ce$ for any constant c. Thus, we may add $N\eta^d$ to Eq. 11 and get the following estimate for $g^d(i)$

$$g_N^d(i) = E\left\{\sum_{l=0}^{N-1} f(X_l, d(X_l))|X_0 = i\right\}, \qquad \text{with } N \gg 1. \tag{12}$$

For any two policies d and h of a Markov system, the performance difference equation, which is the core of the direct-comparison based approach, can be easily established by left-multiplying both sides of the Poisson equation (Eq. 10) with π^h (Cao and Chen 1997; Cao 2007)

$$\begin{aligned}
\eta^h - \eta^d &= \pi^h[(P^h - P^d)g^d + (f^h - f^d)] \\
&= \sum_{i=1}^{S} \pi^h(i) \left\{\sum_{j=1}^{S}[(p^{h(i)}(j|i) - p^{d(i)}(j|i))g^d(j)] + f(i, h(i)) - f(i, d(i))\right\} \\
&= \sum_{i=1}^{S} \pi^h(i) \left\{\left[\sum_{j=1}^{S} p^{h(i)}(j|i)g^d(j) + f(i, h(i))\right]\right. \\
&\qquad \left. - \left[\sum_{j=1}^{S} p^{d(i)}(j|i)g^d(j) + f(i, d(i))\right]\right\}.
\end{aligned} \tag{13}$$

In the above equation, the performance potential $g^d(j)$ can be calculated by solving the Poisson equation (Eq. 10) if P and f are known, or estimated on a sample path by Eqs. 11 or 12. Since $\pi^h(i)$ is always positive for ergodic systems, we have

$$\eta^h \geq \eta^d, \tag{14}$$

if

$$\sum_{j=1}^{S} p^{h(i)}(j|i)g^d(j) + f(i, h(i)) \geq \sum_{j=1}^{S} p^{d(i)}(j|i)g^d(j) + f(i, d(i)), \quad \forall i \in \mathcal{S}; \tag{15}$$

the inequality in Eq. 14 holds if the inequality holds for at least one i in Eq. 15. This justifies Procedure 1.

For Procedure 2, we note that $Q_N^d(i,a)$ is in fact an estimate of $Q^d(i,a)$. The Q-factor under policy d is defined as Bertsekas and Tsitsiklis (1996); Sutton and Barto (1998)

$$Q^d(i,a) := \sum_{j=1}^{S} p^a(j|i)g^d(j) + f(i,a), \quad i \in \mathcal{S}, a \in \mathcal{A}, \tag{16}$$

which becomes Eq. 7 if $g^d(j)$ is replaced by $g_N^d(j)$. Therefore, we can use Eq. 8 to estimate the value of Q-factor $Q^d(i,a), i \in \mathcal{S}, a \in \mathcal{A}$. When the estimate of $Q^d(i,a)$ is accurate enough, the policy update at step 4 of Procedure 2 is equivalent to

$$h(i) := \arg\max_{a\in\mathcal{A}} \left\{ Q^d(i,a) \right\} = \arg\max_{a\in\mathcal{A}} \left\{ \sum_{j=1}^{S} p^a(j|i)g^d(j) + f(i,a) \right\}. \tag{17}$$

This is exactly the policy iteration implemented at step 3 of Procedure 1.

More accurate algorithms based on Q-factors exist, the most popular one is the Q-learning algorithm. These algorithms usually combine the estimation step and the update step together to save the computation. Details can be referred to the literature (Bertsekas and Tsitsiklis 1996; Cao 2007; Sutton and Barto 1998).

We have briefly summarized some main results in the stochastic optimization. Intuitively, we can use the average sum of the rewards at the subsequent steps after visiting a state to represent the long-term effect of this state on the long-run average performance. From this, policy iteration and H-J-B equation can be easily derived based on the performance difference equation. This intuition motivates to develop optimization methods for newly emerging problems, such as event-based optimization. In many real world problems, actions can be applied to the system only when certain events happen. At other epochs, no action can be chosen and the system operates according to its natural scheme. In other words, actions are triggered by certain events. Since the event occurrence is much rarer compared to the system state, the number of epochs of decision making is much less than that in a standard MDP. Moreover, in such systems, the system state is usually unobservable, and we can observe only the events and (or) system rewards. This problem does not fit well the framework of the classical MDP theory. In the next section, we will show that following the intuitions explained in the previous section for the standard MDP and the direct-comparison method, we can develop an approach to the event-based optimization problems.

3 Event-based optimization

In this section, we first give a formal description of the event-based optimization. Then, with the intuitions explained in the previous section, we design an intuitive approach to the event-based optimization problem. Finally, we briefly present the theory of the event-based optimization and describe the conditions under which the intuitive approach leads to an optimal event-based policy.

3.1 Problem description and mathematical formulation

It is well known that the framework of classical MDP is very general and it can model many problems. However, good generality may lose efficiency, especially for the following specific issues. First, the state space of a practical system is usually very large, so is the number of performance potentials or Q-factors. It makes the standard optimization methods of MDP inefficient, even infeasible, in practice. Second, in a standard Markov model, a system is represented by the state transition probabilities, which is very general and may not catch the special structure of specific problems to save computations. For example, by a simple Markov model for a queueing network, it may not be straightforward to find its physical structure by the transition probabilities alone; therefore, it may not be easy to develop algorithms that explore the queueing structure. Third, the policy iteration described in Procedure 1 and 2 requires that the actions should be selected independently at different states, i.e., the choice of action at one state does not depend on that at other states. However, in many practical problems, the actions at different states may be correlated. This makes the standard policy iteration of MDP (specifically, step 3 in Procedure 1 or step 4 in Procedure 2) inapplicable to these problems. The event-based optimization (or EBO for short) is one of the methods that may overcome the aforementioned issues of the standard MDP (of course, with some costs which will be clear later); with EBO, actions at different states can be related and policy iteration may still apply. We briefly introduce the formulation of EBO as follows.

Consider a discrete time Markov chain $\boldsymbol{X} = \{X_l, l = 0, 1, \ldots\}$. The notations of the basic elements of $\boldsymbol{X}$ are the same as those in Section 2. An event e is defined as a set of state transitions with certain common properties. That is, $e := \{\langle i, j\rangle : i, j \in \mathcal{S}$ and $\langle i, j\rangle$ has common properties$\}$, where $\langle i, j\rangle$ denotes the state transition from i to j. The concept of events can be used to describe the structure information of many problems. For example, in an admission control problem (Cao 2005), an event can be a customer arrival. In a two-level hierarchical control problem (Parr 1998), an event can be a transition of control modes on the higher level. Since an event is defined as state transitions, it can provide more information than system states. An event $\langle i, j\rangle$ reveals the information of not only the current state i, but also the next state j. Thus, with an event, we may have some information about the future dynamics of the system. We define the input states of event e as $I(e) := \{$all $i \in \mathcal{S} : \langle i, j\rangle \in e$ for some $j\}$. We further define the output states of an input state i of event e as $O_i(e) := \{$all $j \in \mathcal{S} : \langle i, j\rangle \in e\}$.

In EBO, we choose actions only when certain events happen. All of these events that can trigger a control form the event space $\mathcal{E}$. We assume the event space is finite. For simplicity, we denote $\mathcal{E} := \{e_\phi, e_1, e_2, \ldots, e_V\}$. In particular, we use "e_ϕ" to denote a"no-action" event, meaning when it occurs, no action is taken.

The action space is denoted as $\mathcal{A}$. An action taken at event e determines the state transition probability denoted as $p^a(j|i, e), i, j \in \mathcal{S}, e \in \mathcal{E}, a \in \mathcal{A}$, and $\langle i, j\rangle \in e$.

Example 1 Consider a discrete time single queue. Time is divided into small intervals with equal length and we denote the end of each period as $T_l, l = 0, 1, \ldots$. In each period, there is a customer arriving to the server with probability q, and no customer arriving with probability $1 - q$; if the server is not empty, then the server finishes its service to a customer with probability p, $1 > p > q > 0$. The state of the system

is the number of customers in the system, which is denoted as $X_l = 0, 1, 2, \ldots$. At each epoch T_l, if there is a customer arrival, we may take two actions: either accept the customer, denoted as action c, or reject the customer, denoted as action r. If there is a customer arriving in the lth period and meanwhile there are n customers in the server, then the corresponding state transition has three possible situations, i.e., $\langle X_l, X_{l+1}\rangle \in \{\langle n, n+1\rangle, \langle n, n\rangle, \langle n, n-1\rangle\}$. We denote the event of a customer arrival while there are n customers in the server as e_n, $n = 0, 1, \ldots$. Assume we can observe the event happening. The input state of event e_n is $I(e_n) = \{n\}$. The output state of an input state n of event e_n is $O_n(e_n) = \{n-1, n, n+1\}$ with $n > 0$ and $O_n(e_n) = \{n, n+1\}$ with $n = 0$. The event space is $\mathcal{E} = \{e_\phi, e_0, e_1, e_2, \ldots\}$, with e_ϕ being the no-action event. The action space is a two-element set $\mathcal{A} = \{c, r\}$. When an event e_n happens, if we take an acceptance action c, the state transition probability is $p^c(n|n, e_n) = p$ and $p^c(n+1|n, e_n) = 1 - p$; if we take a rejection action r, the state transition probability is $p^r(n|n, e_n) = 1 - p$ and $p^c(n-1|n, e_n) = p$ with $n > 0$, and $p^r(n|n, e_n) = 1$ with $n = 0$.

An event-based policy d is defined as a mapping from $\mathcal{E}$ to $\mathcal{A}$. That is, $d : \mathcal{E} \to \mathcal{A}$. Here, we consider only stationary and deterministic policies. The event-based policy space is denoted as $\mathcal{D}_e$. In many practical problems, the size of the event space is usually linear to the system size, which is much smaller than the size of the state space, i.e., $|\mathcal{E}| \ll |\mathcal{S}|$. It makes the complexity of EBO much smaller than that of standard MDP. The reward function under policy d is denoted as $f^d = f(i, d(e))$, with $d(e_\phi)$ indicating no action taken, and the associated long-run average performance is denoted as η^d. When an event e happens, we choose an action $a = d(e)$ according to the policy d, where $e \in \mathcal{E}$ and $d \in \mathcal{D}_e$. Our goal is to find the optimal policy d^* which maximizes the long-run average performance as follows.

$$d^* = \arg\max_{d\in\mathcal{D}_e} \{\eta^d\} = \arg\max_{d\in\mathcal{D}_e} \lim_{L\to\infty} E\left\{\frac{1}{L}\sum_{l=0}^{L-1} f^d(X_l)\right\}. \tag{18}$$

On a sample path of an EBO problem, an event may happen at some epoch and an associated action will be taken according to a policy d. Let l_k be the occurrence time of the kth event and E_{l_k} and A_{l_k} be the event happening and the action taken at time l_k. We know $A_{l_k} = d(E_{l_k})$, $k = 1, 2, \ldots$. Figure 3 illustrates a sample path of an EBO problem. In the figure, there are two types of events that can trigger actions; i.e., $\mathcal{E} = \{e_\phi, e_1, e_2\}$, $e_1 = \{\langle 2, 3\rangle, \langle 3, 4\rangle\}$, and $e_2 = \{\langle 5, 3\rangle\}$. We observe that a series of events happen at time $l_1 = 2$, $l_2 = 6$, and $l_3 = 9$, on the sample path, respectively. Thus,

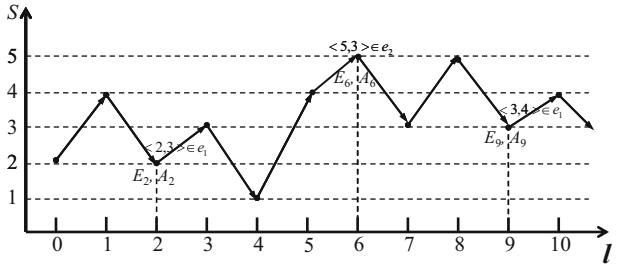


Fig. 3 The sample path of an event-based optimization

we have $E_2 = e_1$, $E_6 = e_2$, and $E_9 = e_1$. The actions are $A_2 = d(e_1)$, $A_6 = d(e_2)$, and $A_9 = d(e_1)$, respectively.

In Fig. 3, we make decisions only at epochs $l = 2, 6, 9$, which are much less than the case of MDP. Moreover, the actions A_2 and A_9 should be the same as $d(e_1)$ since the same event e_1 happens at time $l_1 = 2$ and $l_3 = 9$ and we cannot distinguish the system states, which are different at these two epochs, with $X_2 = 2$ and $X_9 = 3$. This means that we may require the actions at different states be identical in EBO. It is totally different from the requirement in the standard MDP, where the actions are selected independently for different states. Furthermore, we notice that when we observe an event, we receive some information not only for the current state, but also for the future state. For example, when we observe event e_1, it means that the current state is either 2 or 3 and the next state may be 3 or 4. Therefore, the event describes the structure information of the system and it can reveal some information about the future dynamics of the system.

This example illustrates the difference between the formulation of EBO and that of standard MDP. EBO provides a new optimization framework to study a category of practical problems.

3.2 Intuitive optimization

Motivated by the optimization procedures presented in Section 2, we now propose an optimization approach to EBO with a similar intuition. The theoretical analysis will be provided in the next subsection.

First, we assume that the transition probabilities $p^a(j|i, e)$ are known and the system state X_l is observable, $i, j \in \mathcal{S}$, $e \in \mathcal{E}$, $a \in \mathcal{A}$. Suppose that the current event-based policy is d and an event e happens at the initial time 0. That is, $l_0 = 0$ and $E_{l_0} = E_0 = e$. One myopic method for policy updating is to choose the action a for event e which has the maximal one-step reward $E\{f(X_0, A_0)|E_0 = e\}$, that is

$$a = \arg\max_{A_0 \in \mathcal{A}} E\{f(X_0, A_0)|E_0 = e\}, \tag{19}$$

where "E" denote the steady-state expectation, i.e., with the assumption that X_0 and E_0 have a steady-state distribution. However, the above updating method only considers the one-step reward and cannot guarantee to find the optimal policy. To improve this myopic method, we can use a long-term reward to replace the one-step reward to quantify the quality of the action A_0 for event E_0. Similar to the analysis in Procedure 1, we look forward N steps more and use the following term to quantify this long-term reward

$$E\left\{ f(X_0, A_0) + \sum_{l=1}^{N} f^d(X_l) \middle| E_0 = e \right\}, \quad N \gg 1. \tag{20}$$

With Eq. 12, we notice that the second term in the above formula $E\{\sum_{l=1}^{N} f^d(X_l)|E_0 = e\}$ is a conditional average performance potential $E\{g_N^d(X_1)|E_0 = e\}$. Now, we need the distribution of X_1 under the condition that $E_0 = e$. Let $\pi^d(i|e)$ be the steady-state conditional probability under policy d that the system is at state i when event e happens, $i \in I(e)$. Then we take action $a = d(e)$ and the system will transit to the next state j with probability $p^a(j|i, e)$,

where $i \in I(e)$, $j \in O_i(e)$. Therefore, under policy d the probability that the next state is j when event e happens is $\sum_{i \in I(e)} \pi^d(i|e) p^a(j|i, e)$.

Thus, Eq. 20 can be rewritten as below.

$$\sum_{i \in I(e)} \pi^d(i|e) \left\{ f(i, a) + \sum_{j \in O_i(e)} p^a(j|i, e) g_N^d(j) \right\}, \quad N \gg 1. \tag{21}$$

The above term (21) is similar to the term $\sum_{j=1}^{S} p^a(j|i) g_N^d(j) + f(i, a)$ in Procedure 1. Term (21) quantifies the "contribution" of action a at event e on the performance from a long-term point of view. Therefore, the principle for action selection is to choose the action with the maximal value of Eq. 21. Motivated by Proposition 1, we wish that the following proposition holds.

Proposition 3 *Given a sample path under any event-based policy d, estimate the $\pi^d(i|e)$'s and $g_N^d(i)$'s, $i \in \mathcal{S}$ and $e \in \mathcal{E}$, on the sample path (cf. Eq. 4), with $N \gg 1$ being a fixed large integer. Suppose that N is sufficiently large and the estimates are accurate enough. Let h be an event-based policy defined by*

$$h(e) := \arg\max_{a \in \mathcal{A}} \left\{ \sum_{i \in I(e)} \pi^d(i|e) \left[f(i, a) + \sum_{j \in O_i(e)} p^a(j|i, e) g_N^d(j) \right] \right\}, \quad \forall\, e \in \mathcal{E}. \tag{22}$$

Then we have $\eta^h \geq \eta^d$.

With Eq. 27 shown later, we will see that this proposition requires a condition that $\pi^d(i|e) = \pi^h(i|e)$ for any policy d and h, $i \in I(e)$, and $e \in \mathcal{E}$. We will discuss the details later.

The two terms $f(i, a)$ and $p^a(j|i, e)$ on the right-hand side of Eq. 22 are known parameters. Thus, if Proposition 3 holds, then, by estimating the values of $\pi^d(i|e)$ and $g_N^d(j)$ from sample paths, we can update the event-based policy effectively with Eq. 22. In this case, an intuitive optimization procedure for EBO may be proposed as follows.

Procedure 3 Intuitive optimization of EBO based on the performance potential estimation.

- Choose an arbitrary event-based policy as the initial policy d, $d \in \mathcal{D}_e$.
- On the sample path under policy d, estimate the performance potential based on the formula $g_N^d(i) = E\{\sum_{t=0}^{N-1} f^d(X_{l+t}) | X_l = i\}$, N is a fixed large integer. Meanwhile, estimate $\pi^d(i|e) = \lim_{K \to \infty} \sum_{k=1}^{K} \mathbf{1}_{(X_{l_k} = i)} \mathbf{1}_{(E_{l_k} = e)} / \sum_{k=1}^{K} \mathbf{1}_{(E_{l_k} = e)}$, where $\mathbf{1}_{(Y)}$ is an indicator function defined as $\mathbf{1}_{(Y)} = 1$ (or 0) when Y is true (or false). Suppose that the sample path is long enough to guarantee the estimation accuracy of g_N^d and $\pi^d(i|e)$.
- Update the policy as $h(e) := \arg\max_{a \in \mathcal{A}} \left\{ \sum_{i \in I(e)} \pi^d(i|e) \left[f(i, a) + \sum_{j \in O_i(e)} p^a(j|i, e) \right. \right.$ $\left. \left. g_N^d(j) \right] \right\}$ for all $e \in \mathcal{E}$, where if $d(e)$ already attains the maximum, then set $h(e) := d(e)$.
- If $d = h$, stop and output d as the optimal policy; Otherwise, set $d := h$ and go to step 2.

In the above optimization procedure, by Proposition 3, the event-based policy is iteratively improved. The optimality of this procedure, under some conditions, will

be theoretically justified later. However, this optimization procedure requires the system state X_l be observable and the transition probabilities $p^a(j|i,e)$ and reward function $f(i,a)$ be known for all $i, j \in \mathcal{S}$, $e \in \mathcal{E}$, and $a \in \mathcal{A}$. Moreover, this procedure needs to estimate $g_N^d(i)$ for every state. When the system size increases, the state space will grow explosively. The estimation of $g_N^d(i)$ becomes un-treatable, even the storage of $g_N^d(i)$ becomes a critical problem for the requirement of large storage space. Fortunately, we notice that the event space usually grows linearly to the system size. If we can aggregate the performance potentials based on events, we will be able to significantly reduce both the computation and storage complexity of the algorithm. This inspires us to develop the next optimization procedure.

Consider the situation where the system state X_l is not observable and the transition probabilities $p^a(j|i,e)$ and reward function $f(i,a)$ are unknown parameters. The system is running under an event-based policy d, $d \in \mathcal{D}_e$. We observe only the event happening E_{l_k} and the system reward $f^d(X_l)$, $l = 0, 1, \ldots$ and $k = 1, 2, \ldots$. This scenario is common for many practical systems, especially when the optimization is implemented online and we do not have too much information about the system model. Similar to the Q-factor in Procedure 2 for the MDP problem, we define the event-based Q-factor

$$Q_N^d(e,a) := \sum_{i \in I(e)} \pi^d(i|e) \left[f(i,a) + \sum_{j \in O_i(e)} p^a(j|i,e) g_N^d(j) \right], \tag{23}$$

for all $e \in \mathcal{E}$ and $a \in \mathcal{A}$. We can choose $h(e) := \arg\max_{a \in \mathcal{A}} Q_N^d(e,a)$, which is equivalent to Eq. 22, to improve the policy. Now, we discuss how to estimate $Q_N^d(e,a)$, $e \in \mathcal{E}$, $a \in \mathcal{A}$, based on the system sample path.

With Eq. 20, $Q_N^d(e,a)$ can be rewritten as follows.

$$Q_N^d(e,a) = E\left\{ f(X_0, A_0) + \sum_{l=1}^{N} f^d(X_l) \,\middle|\, E_0 = e, A_0 = a \right\}. \tag{24}$$

With Eq. 24, we can estimate $Q_N^d(e,a)$ by summating the following $N+1$ steps rewards under the condition that the action a is taken at the occurrence time of the current event e. All the terms on the right-hand side of Eq. 24 can be observed from the sample path. Therefore, we can estimate $Q_N^d(e,a)$ based on the sample path, without knowing the prior information of $p^a(j|i,e)$ and $f(i,a)$. However, similar to Procedure 2, we also have to adopt d_ϵ policy to guarantee that all of the event-action pairs (e,a) can be visited on a sample path. A detailed optimization method based on $Q_N^d(e,a)$ is described in the following procedure.

The above optimization procedure requires only the observations of event happenings and system rewards. We do not need to know the system state X_l and the values of $p^a(j|i,e)$. Thus, this procedure is online implementable. The estimation of $Q_N^d(e,a)$ is illustrated in Fig. 4. Events happen at $l = 1, 3, 6$ and the event types are all e_1. The actions adopted are a_1, a_2, a_1, respectively. Figure 4 illustrates the update of the estimate of $Q_N^d(e_1,a_1)$, where $N = 4$. We see that $Q_{4,1}^d(e_1,a_1)$ and $Q_{4,2}^d(e_1,a_1)$ are the samples of the estimate of $Q_4^d(e_1,a_1)$, while $Q_{4,1}^d(e_1,a_2)$ is a sample of the estimate of $Q_4^d(e_1,a_2)$. Furthermore, compared with Fig. 2, the number of estimated parameters $Q_N^d(e,a)$ is relatively small because the number of event types is much

Procedure 4 Intuitive optimization of EBO based on the factor $Q_N^d(e, a)$.

- Choose an arbitrary event-based policy d as the initial policy, $d \in \mathcal{D}_e$.
- At every event occurrence time l_k, adopt the action $a = d(E_{l_k})$ with probability $1 - \epsilon$, and adopt other actions $a \neq d(E_{l_k})$ and $a \in \mathcal{A}$ with probability $\frac{\epsilon}{|\mathcal{A}|-1}$, where $\epsilon \ll 1$ is a very small positive number.
- Estimate $Q_N^d(e, a) = E\{\sum_{t=0}^{N} f(X_{l_k+t}, A_{l_k+t}) | E_{l_k} = e, A_{l_k} = a\}$, for all $e \in \mathcal{E}$ and $a \in \mathcal{A}$, where N is a fixed large integer. Run the sample path long enough to obtain the accurate estimates for all $Q_N^d(e, a)$'s.
- Generate a new policy h by setting $h(e) := \arg\max_{a \in \mathcal{A}} Q_N^d(e, a)$ for all $e \in \mathcal{E}$; setting $h(e) := d(e)$ if $d(e)$ already attains the maximum.
- If $d = h$, stop and output d as the optimal policy; Otherwise, set $d := h$ and go to step 2.

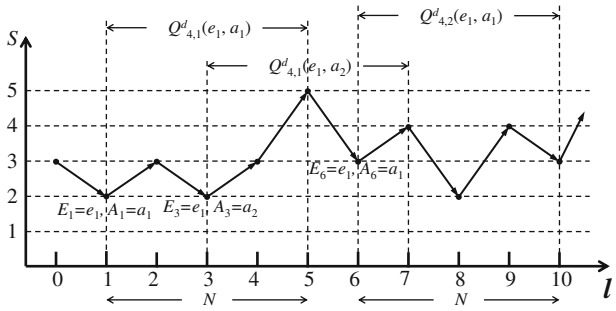


Fig. 4 The estimation procedure of factor $Q_N^d(e, a)$ based on the sample path of EBO

less than that of system states. Thus, the optimization framework of EBO has a better scalability compared with the MDP formulation.

We have introduced two possible optimization procedures of EBO based on the intuition similar to MDP. In the next subsection, we will briefly discuss the theoretical justifications for these optimization procedures and provide conditions under which they may be applied exactly.

3.3 Theory of event-based optimization

With the approach of direct comparison, in this section we briefly discuss the theory of EBO. Some conditions that are required for the optimality of Procedure 3 and 4 are also presented. Because of the space limit of the paper, the mathematical analysis is simplified. The audience interested in the details and rigorous proofs can refer to the literature (Cao 2005, 2007; Jia 2011).

First, we discuss the theoretical justification for Procedure 3. Consider an event-based policy d, $d \in \mathcal{D}_e$. At the steady state, the state transition probability under policy d is

$$p^d(j|i) = \sum_{e=1}^{V} \pi(e|i)\, p^{d(e)}(j|i, e), \tag{25}$$

where $\pi(e|i)$ is the conditional steady-state probability of event e happening when the state is i. $\pi(e|i)$ is usually determined by the system structure and it is independent of the policy (Cao 2007). Consider another event-based policy h, $h \in \mathcal{D}_e$. By using the difference equation (13) of Markov systems, we derive the following equation for the difference of η^h and η^d.

$$\begin{aligned}
&\eta^h - \eta^d \\
&= \pi^h[(P^h - P^d)g^d + (f^h - f^d)] \\
&= \sum_{i=1}^{S} \pi^h(i) \left\{ \sum_{j=1}^{S} \sum_{e=1}^{V} \pi(e|i)[p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) + [f^h(i) - f^d(i)] \right\} \\
&= \sum_{i=1}^{S} \left\{ \sum_{j=1}^{S} \sum_{e=1}^{V} \pi^h(i,e)[p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) + \pi^h(i)[f^h(i) - f^d(i)] \right\} \\
&= \sum_{e=1}^{V} \pi^h(e) \sum_{i \in I(e)} \pi^h(i|e) \left\{ \sum_{j \in O_i(e)} [p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) + [f^h(i) - f^d(i)] \right\},
\end{aligned} \tag{26}$$

The equation above quantifies the performance difference under any two event-based policies h and d. If the conditional probability $\pi^h(i|e)$ has the following property

$$\pi^h(i|e) = \pi^d(i|e), \quad \forall\, i \in I(e), e \in \mathcal{E}, \text{ for any two policies } h \text{ and } d, \tag{27}$$

then Eq. 26 can be rewritten as below.

$$\begin{aligned}
&\eta^h - \eta^d \\
&= \sum_{e=1}^{V} \pi^h(e) \sum_{i \in I(e)} \pi^d(i|e) \left\{ \sum_{j \in O_i(e)} [p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)]g^d(j) + [f^h(i) - f^d(i)] \right\}.
\end{aligned} \tag{28}$$

Since $\pi^h(e)$ is always positive for ergodic systems, if we choose $h(e) := \arg\max_{a \in \mathcal{A}} \left\{ \sum_{i \in I(e)} \pi^d(i|e) \left[\sum_{j \in O_i(e)} p^a(j|i,e)g^d(j) + f(i,a) \right] \right\}$ for all $e \in \mathcal{E}$, then from Eq. 28 we have $\eta^h \geq \eta^d$. Thus, with Eq. 28, Proposition 3 is justified and Procedure 3 can improve the policy iteratively. When the policy space $\mathcal{D}_e$ is finite, Procedure 3 reaches the optimal policy within a finite number of iterations. However, the optimality of Procedure 3 requires the condition (27). That is, the steady-state conditional distribution of state i given event e should be irrelevant to the event-based policy d. Details can be referred to Chapter 8 of the book (Cao 2007).

For Procedure 4, similar to the previous analysis of Procedure 2, we have the following analogous definition of $Q^d(e,a)$ under an event-based policy d

$$Q^d(e,a) := \sum_{i \in I(e)} \pi^d(i|e) \left[f(i,a) + \sum_{j \in O_i(e)} p^a(j|i,e)g^d(j) \right], \quad e \in \mathcal{E}, a \in \mathcal{A}. \tag{29}$$

Since ϵ is small enough, the policy d_ϵ in step 2 of Procedure 4 is close to the deterministic policy d, yet it can visit all of the possible event-action pairs. Thus, we can simply view the policy generated by step 2 of Procedure 4 as the policy d. Replacing $g^d(j)$ with $g_N^d(j)$, we can rewrite Eq. 29 as Eq. 23, and $Q_N^d(e,a)$ can be viewed as an estimate of $Q^d(e,a)$. When the estimate of $Q^d(e,a)$ is accurate enough, we choose $h(e) := \arg\max_{a\in\mathcal{A}} Q^d(e,a)$, for all $e \in \mathcal{E}$, according to step 4 of Procedure 4. With Eq. 29, we have

$$h(e) := \arg\max_{a\in\mathcal{A}} Q^d(e,a) = \arg\max_{a\in\mathcal{A}} \left\{ \sum_{i\in I(e)} \pi^d(i|e) \left[\sum_{j\in O_i(e)} p^a(j|i,e) g^d(j) + f(i,a) \right] \right\}. \tag{30}$$

Therefore, with Eq. 30 the policy iteration in Procedure 4 is the same as that in Procedure 3. Similarly, the optimality of policy iteration in Procedure 4 can also be guaranteed by the difference equation (28). Moreover, the condition (27) is also required when we apply Procedure 4 to find the optimal policy.

3.4 Discussion

EBO is a new optimization framework and it works well for the performance optimization of problems with event-triggered control. In EBO, the concept of event is used to capture the structure information of the system. In the framework of EBO, action is triggered by events, not by states. The event space is usually much smaller than the state space. Thus, the optimization complexity of EBO is much less than that of MDP. Moreover, EBO determines the actions according to events, not according to states. Thus, EBO does not require actions to be chosen independently at different states, which is a constraint of the standard MDP theory.

Procedures 3 and 4 are proposed based on an intuitive principle. We define a quantity to quantify the long-term effect of actions on the system performance, called the performance potential $g^d(i)$ or factor $Q^d(e,a)$. Using the difference equation (28), we can find the improved policy iteratively in Procedures 3 and 4. Procedure 3 is applicable when the transition probability $p^a(j|i,e)$ and reward function $f(i,a)$ are known and the system state X_l is observable; while Procedure 4 can be used when $p^a(j|i,e)$ and $f(i,a)$ are unknown and X_l is unobservable. In some sense, Procedure 4 can be viewed as an online version of Procedure 3.

However, the optimality of Procedures 3 and 4 requires the condition (27). This means that the conditional probability $\pi(i|e)$ is determined by the system structure and is irrelevant to the policies. For practical systems satisfying this condition, we can use Procedures 3 or 4 to find the optimal event-based policy. For systems not satisfying this condition, we cannot guarantee the convergence of the algorithms to the optimum. We have to study the effect of this condition on the final results. If this condition does not hold, we may still use Procedures 3 or 4 to obtain a near optimal policy which is acceptable in practice. The second application example in the next section partly demonstrates this idea.

Another approach to treat the problem of the strict constraint of condition (27) is to use the gradient-based optimization. Without loss of generality, we discuss a randomized policy which adopts policy d with probability $1-\delta$ and adopts policy h with probability δ, where $0 \le \delta \le 1$. We denote this randomized policy with a

superscript δ. The performance difference of the system with the original policy d and with this randomized policy δ is still quantified by Eq. 26. We rewrite Eq. 26 as below.

$$\begin{aligned}
&\eta^{\delta} - \eta^{d} \\
&= \sum_{e=1}^{V} \pi^{\delta}(e) \sum_{i \in I(e)} \pi^{\delta}(i|e) \left\{ \sum_{j \in O_i(e)} [p^{\delta(e)}(j|i,e) - p^{d(e)}(j|i,e)] g^{d}(j) + [f^{\delta}(i) - f^{d}(i)] \right\} \\
&= \sum_{e=1}^{V} \pi^{\delta}(e) \sum_{i \in I(e)} \pi^{\delta}(i|e) \left\{ \sum_{j \in O_i(e)} \delta [p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)] g^{d}(j) + \delta [f^{h}(i) - f^{d}(i)] \right\},
\end{aligned} \tag{31}$$

where the second equality holds by using the fact that the randomized policy δ generates the transition probability matrix $P^h\delta + P^d(1-\delta)$ and the reward function $f^h\delta + f^d(1-\delta)$. Letting $\delta \to 0$, we obtain the following derivative equation from Eq. 31

$$\begin{aligned}
&\frac{\partial \eta^{d}}{\partial \delta} \\
&= \sum_{e=1}^{V} \pi^{d}(e) \sum_{i \in I(e)} \pi^{d}(i|e) \left\{ \sum_{j \in O_i(e)} [p^{h(e)}(j|i,e) - p^{d(e)}(j|i,e)] g^{d}(j) + [f^{h}(i) - f^{d}(i)] \right\}.
\end{aligned} \tag{32}$$

With Eq. 32, we obtain the performance derivative with respect to a randomized policy δ from policy d and h. Since $\pi^d(e)$ is always positive and $\pi^d(i|e)$ can be calculated or estimated under the current policy d, we can choose a proper event-based policy h which makes the total value of the second summation of Eq. 32 maximally positive. This means that the performance derivative is positive. If we choose a proper step-size along this direction, we can obtain a better policy with an improved performance. This is the basic idea of the gradient-based optimization. Please note, Eq. 32 does not include $\pi^h(i|e)$, only includes $\pi^d(i|e)$ which is observable under the current system sample path. Therefore, even the condition (27) is not satisfied, we can still use the performance derivative Eq. 32 to do the gradient-based optimization. However, this gradient-based optimization approach suffers some intrinsic deficiencies, such as the trapping into local optimum and the difficulty of selecting proper step-sizes. For more details of the policy gradient-based optimization in Markov systems, please refer to Chapter 2.1 and Chapter 8.3.3 of Cao (2007).

4 Applications

In this section, we use some applications of EBO to demonstrate the effectiveness of this new optimization framework. First, we present an admission control problem. It is a small-size problem and it satisfies the condition (27). We use this academic example to demonstrate the correctness of EBO theory. Second, we use a practical

example, the activation policy optimization problem of wireless sensor networks, to demonstrate the effectiveness of EBO for practical applications. Third, we consider a material handling problem, in which the gradient-based policy iteration achieves a good performance. Fourth, we consider the portfolio management with unobservable stochastic volatility to demonstrate how the EBO approach can be applied to solve a partially observable MDP (POMDP).

4.1 Admission control problem

Consider a standard open Jackson network with M servers. The service rate of server k is denoted as μ_k, $k = 1, \ldots, M$. The service discipline is first come first serve. The external customer arrival is a Poisson process with rate λ. The network has a capacity N_c. That is, when the total number of customers in the network is N_c, any new arrivals will be rejected. The routing probability is denoted as $q_{kk'}$, $k, k' = 0, 1, \ldots, M$ and server 0 indicates the exterior of the network. Without loss of generality, we assume $q_{kk} = 0$, $k = 0, 1, \ldots, M$. The number of customers at server k is denoted as n_k, $n_k = 0, 1, \ldots, N_c$, $k = 1, \ldots, M$. The system state is denoted as $\boldsymbol{n} := (n_1, \ldots, n_M)$ and the state space is denoted as $\mathcal{S}$. The total number of customers in the network is denoted as $n := \sum_{k=1}^{M} n_k$, $n = 0, 1, \ldots, N_c$. The system reward function is denoted as $f(\boldsymbol{n})$, e.g., $f(\boldsymbol{n}) = R \cdot \sum_{k=1}^{M} \mu_k \mathbf{1}_{(n_k>0)} - C \cdot n$, which means that each service completion gets a reward R and each customer in the network pays a cost C per unit time. The detailed explanation about this problem can be referred to Cao (2005) and Xia (2012, 2013).

When an external customer arrives at the network, an admission controller determines if the customer will be accepted or rejected. If the customer is accepted, it will enter server k with probability q_{0k}, $k = 1, \ldots, M$. Otherwise, it will leave the network. The controller makes the admission decision based on the total number of customers in the network. More specifically, the controller accepts the arriving customers according to an admission probability $a(n), 0 \le a(n) \le 1$ and $n = 0, 1, \ldots, N_c$. The values of $a(n)$'s are decided by the controller. Therefore, the controller makes decision only when the event of customer arrival happens. At other state transition epochs, e.g., the transition of customers among servers, the controller cannot make admission decision.

This problem can be modeled as an EBO problem naturally. The event is defined as $e_n := \{\langle \boldsymbol{n}, \boldsymbol{n}_{+k} \rangle, \langle \boldsymbol{n}, \boldsymbol{n} \rangle : \boldsymbol{n} \in \mathcal{S}_n, k = 1, \ldots, M\}$, where $\boldsymbol{n}_{+k}$ is called the neighboring state of $\boldsymbol{n}$ and it is defined as $\boldsymbol{n}_{+k} = (n_1, \ldots, n_k + 1, \ldots, n_M)$, $\mathcal{S}_n$ is the set of states where the total number of customers equals n, and $n = 0, 1, \ldots, N_c$. The event e_n consists of two types of state transitions, $\langle \boldsymbol{n}, \boldsymbol{n}_{+k} \rangle$ means that the arriving customer is admitted to server k, $\langle \boldsymbol{n}, \boldsymbol{n} \rangle$ means that the arriving customer is rejected by the controller. Thus, e_n indicates the arrival event when the total number of customers is n. The event space is $\mathcal{E} := \{e_\phi, e_0, e_1, \ldots, e_{N_c}\}$. The action is the admission probability and the action space is $\mathcal{A} := \mathbb{R}[0, 1]$. The event-based policy is $d : \mathcal{E} \to \mathbb{R}[0, 1]$. Actually, policy d can also be represented by a series of admission probabilities $a(n)$'s, $n = 0, 1, \ldots, N_c$. Obviously, $a(N_c) = 0$ since the network capacity is N_c. The optimization goal is to find the optimal policy d^* or the optimal admission probabilities $a^*(n)$'s, $n = 0, 1, \ldots, N_c - 1$, which has the maximal long-run average performance.

First, we have to check if this problem satisfies the condition (27). Fortunately, we find that it does. That is, the conditional probability $\pi(\boldsymbol{n}|e_n)$ (further denoted as $\pi(\boldsymbol{n}|n)$ for simplicity) is independent of the policy d. It can be verified by using the property of the product-form solution of Jackson networks. The detailed proof is omitted for the limit of space. Interested audience can refer to Xia (2012, 2013).

We set the parameters of this problem as follows. $M = 3$, $N_c = 5$, $\mu_1 = 30$, $\mu_2 = 40$, $\mu_3 = 50$, and $\lambda = 50$. The routing probabilities are $(q_{0k})_{k=0,1,2,3} = (0, 0.3, 0.3, 0.4)$, $(q_{1k})_{k=0,1,2,3} = (0.2, 0, 0.4, 0, 4)$, $(q_{2k})_{k=0,1,2,3} = (0.5, 0.2, 0, 0.3)$, $(q_{3k})_{k=0,1,2,3} = (0.4, 0.2, 0.4, 0)$. The reward $R = 1$ and the cost $C = 15$.

This problem is a continuous-time Markov process in nature. However, the EBO framework in this paper handles the discrete-time Markov chain. Although we also have the continuous-time version of EBO, we convert this problem to a discrete-time Markov chain for consistency. We use the uniformization technique (Puterman 1994). First, we determine the infinitesimal generator B of this problem. Select β as the element of B with the maximal absolute value. We have $\beta = \lambda + \mu_1 + \mu_2 + \mu_3 = 170$. Thus, the original problem can be converted to a discrete-time Markov chain with the state transition probability matrix $P = I + B/\beta$, where I is an identity matrix with the same dimension as B. This Markov chain has the same steady-state distribution as that of the original Markov process.

Suppose the system state is observable. Since we know the values of the state transition probabilities, we use Procedure 3 to optimize this EBO problem. The state transition probability $p^a(j|i, e)$ is rewritten as $p^{a(n)}(\boldsymbol{n}'|\boldsymbol{n}, n)$, where $\langle \boldsymbol{n}, \boldsymbol{n}' \rangle \in e_n$ and $a(n) \in \mathbb{R}[0, 1]$. Obviously, the value of $p^{a(n)}(\boldsymbol{n}'|\boldsymbol{n}, n)$ is

$$p^{a(n)}(\boldsymbol{n}'|\boldsymbol{n}, n) = \begin{cases} 1 - a(n), & \text{if } \boldsymbol{n}' = \boldsymbol{n}; \\ a(n)q_{0k}, & \text{if } \boldsymbol{n}' = \boldsymbol{n}_{+k}, k = 1, 2, 3. \end{cases} \tag{33}$$

Since the reward function f is irrelevant to actions and $\pi(\boldsymbol{n}|n)$ is independent of d, the update formula in step 3 of Procedure 3 can be rewritten as below.

$$h(n) := \arg \max_{a(n)\in\mathbb{R}[0,1]} \sum_{\langle \boldsymbol{n}, \boldsymbol{n}' \rangle \in e_n} \pi(\boldsymbol{n}|n) p^{a(n)}(\boldsymbol{n}'|\boldsymbol{n}, n) g^d(\boldsymbol{n}') \tag{34}$$

Substituting Eq. 33 into the above equation and removing the constant terms, we have

$$h(n) := \arg \max_{a(n)\in\mathbb{R}[0,1]} a(n) \sum_{\boldsymbol{n}\in\mathcal{S}_n} \pi(\boldsymbol{n}|n) \sum_{k=1}^{3} q_{0k}[g^d(\boldsymbol{n}_{+k}) - g^d(\boldsymbol{n})]. \tag{35}$$

In the above equation, we only need to estimate the term $\sum_{\boldsymbol{n}\in\mathcal{S}_n} \pi(\boldsymbol{n}|n) \sum_{k=1}^{3} q_{0k}[g^d(\boldsymbol{n}_{+k}) - g^d(\boldsymbol{n})]$. If this term is positive, we choose $h(n) = 1$. Otherwise, we choose $h(n) = 0$. That is, the admission probability is either 1 or 0. It avoids the difficulty that the action space is continuous and infinite.

Therefore, we obtain a specific version of Procedure 3 for this admission control problem. We choose the initial policy d as $a(n) = 1$, for all $n = 0, 1, \ldots, 4$. We set $N = 100$ and $L = 2 \times 10^5$. Implementing this optimization procedure, we obtain the following optimization results.

In Fig. 5, the optimal admission probabilities are $a^*(0) = a^*(1) = a^*(2) = 1$ and $a^*(3) = a^*(4) = a^*(5) = 0$. The long-run average performance is estimated as $\eta^* = 25.4699$. The optimality of this result is verified by enumerating the system

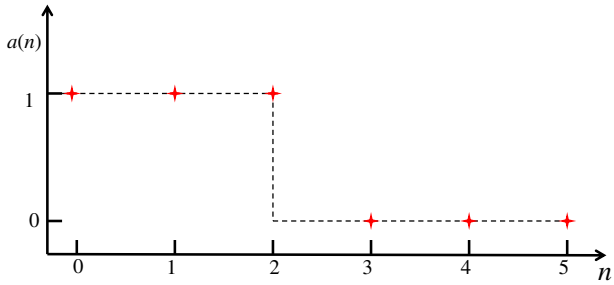


Fig. 5 The optimal admission control policy based on EBO optimization framework

performance of Jackson networks theoretically. Moreover, we see that the optimal admission policy actually has a threshold form. When $n < 3$, accept all of the arriving customers; Otherwise, reject all. This optimality structure is proved theoretically (Xia 2012, 2013).

4.2 Activation policy optimization of wireless sensor networks

In this subsection, we apply the aforementioned EBO method to solve the activation policy optimization problem in a wireless sensor network (WSN). Consider a WSN that is composed of N_s sensors. These sensors are uniformly randomly deployed in an area of interest (AoI) with unit size. An example of $N_s = 10$ is shown in Fig. 6. When activated, each sensor can detect the target of interest within the radius of r_d with probability p. Assume the detection error of the sensors are independent. So when a position is covered by m active sensors in the same time, the detection probability at that position is $1 - (1 - p)^m$. The sensors are powered by batteries. Assume the working time of a battery has an exponential distribution. An active sensor consumes power with rate R_d. When fully discharged, a sensor enters the sleeping mode and it is charged by solar power. Assume the time to fully charge a sensor has an exponential distribution with rate R_c. When fully charged, a sensor is either activated immediately, or stays at the ready state, depending on a centralized activation policy. The policy makes decision according to the ratio of active sensors among all the sensors. When a sensor is about to enter the sleeping mode, the activation policy also decides whether to turn on some of the ready sensors, depending on the ratio of the active sensors. We want to find an activation policy to maximize the average detection probability, where the average is taken both over time and over space.

We model the above problem as an EBO. Let $s_t = (s_t(1), \ldots, s_t(k), \ldots, s_t(N_s))$ be the state of the sensors at time t, where $s_t(k) = 0, 1, 2$ means sensor k is sleeping, active, or ready at time t, respectively. There are two types of events. The first group is the ready event, namely, a sensor is just fully charged. Depending on the ratio of active sensors, there are n events in this group. Let $e(k)$ denote the event that a sensor is just fully charged and the ratio of active sensors is in $((k-1)/n, k/n]$ for $k = 2, \ldots, n$ and the ratio is in $[0, 1/n]$ for $k = 1$. The second group is the sleeping event, namely, a sensor is just fully discharged. Depending on the ratio of active sensors, there are n events in this group. Let $e(k)$ denote the event that a sensor is just fully discharged, and the ratio of active sensors is $((k-n-1)/n, (k-n)/n]$ for $k = n+2, \ldots, 2n$ and the ratio is in $[0, 1/n]$ for $k = n+1$. So, there are totally $2n$ observable events that may trigger a decision making. For the first group of events, i.e.,

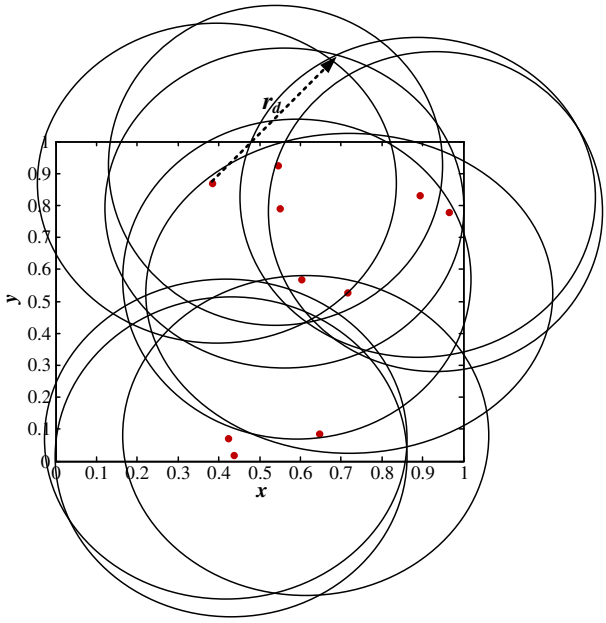


Fig. 6 An AoI with ten randomly deployed sensors

$k = 1, \ldots, n$, the feasible actions are either to activate the current sensor or not, i.e., $a = 1$ or $a = 0$, respectively. If $a = 1$, this sensor enters the active mode. Otherwise, it enters the ready mode. For the second group of events, i.e., $k = n + 1, \ldots, 2n$, the feasible actions are either to activate one of the ready sensors or not to activate any sensor, i.e., $a = 1$ or $a = 0$, respectively. The event-based policy is a row vector $d = [a(k)]_{k=1,\ldots,2n}$, where $a(k)$ is the action for event $e(k)$. The reward function $f^d(s_t)$ is the average detection probability over the AoI under the policy d. That is $f^d(s_t) = \int_0^1 \int_0^1 [1 - (1 - p)^{m(x,y,s_t)}]dxdy$, where (x, y) is a geographical point in the AoI and $m(x, y, s_t)$ is the number of active sensor nodes covering point (x, y) at state s_t. The objective function is the average detection probability both over time and over space, i.e., $\eta^d = \lim_{T\to\infty} \frac{1}{T} \int_0^T f^d(s_t)dt$.

We now apply a variation of Procedure 4 to this policy optimization problem. Consider the following parameter settings, i.e., $N_s = 10, r_d = 0.5, p = 0.9, R_c = 1, R_d = 2, n = 2$. There are $2^{2n} = 16$ policies in total. We use $N = 100$ to estimate the performance potentials, and then obtain the next policy from the policy iteration. However, the condition (27), $\pi^h(s|e) = \pi^d(s|e)$, does not hold for this problem. Moreover, under many policies in this problem, the system is not ergodic and sometimes the events are not ergodic. Therefore, we adopt a tabu-type randomized search technique to improve the searching ability of step 4 of Procedure 4. In other words, when the next policy is generated, if this policy has not been visited in the previous iterations, then it is taken. Otherwise, a new policy is randomly generated from the set of policies that have not been visited so far. We also record the best-so-far policy during this search process. When the policy iteration stops after a given number of iterations, the best-so-far policy is output. We start from 16 different initial policies and show the performance of the best-so-far policies in Figs. 7 and 8. Note that d_1 is the initial policy. For example, $d_1 = [0\ 1\ 0\ 1]$ means that under this policy if a

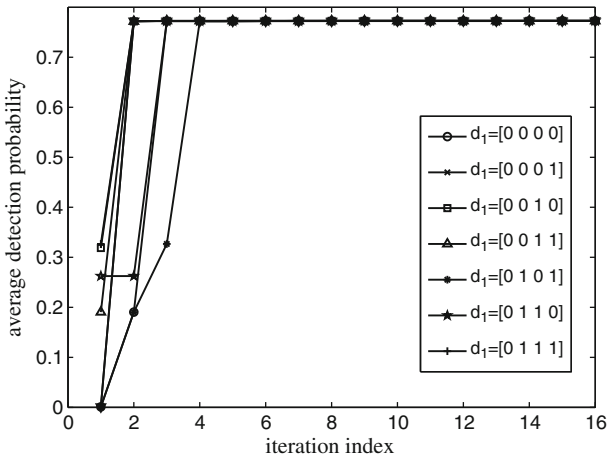


Fig. 7 A sample path of the randomized policy iteration, when $d_1(1) = 0$

sleeping sensor becomes fully charged, this sensor will be activated if more than 50 % of the sensors are active (indicated by $d_1(2) = 1$). Otherwise, this sensor will enter the ready mode (indicated by $d_1(1) = 0$). If some sensor becomes fully discharged, one of the ready sensors (if any) will be activated if more than 50 % of the sensors are active (indicated by $d_1(4) = 1$). Otherwise, no ready sensors will be activated (indicated by $d_1(3) = 0$). The algorithm stops at the optimal policy $d^* = [1\ 0\ 1\ 0]$ with the maximal average detection probability $\eta^* = 0.7752$.

From Figs. 7 and 8 we can see that no matter which policy is picked as the initial policy, the optimal policy is found within four iterations. One can easily calculate that it takes a purely random search 8.5 iterations to find the optimal policy on the average. This experiment demonstrates the effectiveness of the EBO optimization framework for practical applications.

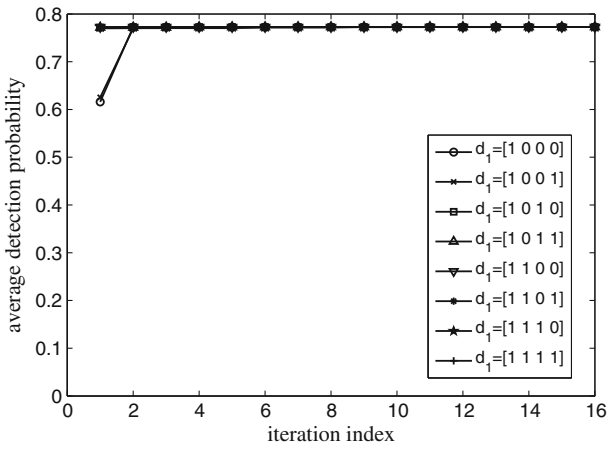


Fig. 8 A sample path of the randomized policy iteration, when $d_1(1) = 1$

4.3 Material handling problem

In this subsection, we consider an optimization problem in a material handling (MH) system of a general assembly (GA) line motivated by a practical car manufacturing system. As Fig. 9 illustrates, the GA line consists of a conveyor and series of assembling stations (Zhao et al. 2010a). Semi-products (say car bodies) are loaded onto the conveyor, going through all the stations where different parts (such as engine, seats, tires, etc.) are assembled, and then final products are produced. The conveyor is called a paced-moving conveyor since it moves step-by-step in every period of time (called cycle time) and transfers semi-products downstream synchronously when all stations complete their assembly operations. To have a smooth production, the parts are supplied by a driver of the MH system from a central docking area to the lineside buffers with a 2-dolly train. Each dolly can hold parts for only one lineside buffer, so that the driver can supply at most two buffers in each trip.

This practical problem can be formulated as an EBO (Zhao et al. 2010b). Consider a discrete-time version of the problem. The system state at time t is $\boldsymbol{s}_t = (\boldsymbol{n}_t, \boldsymbol{y}_t)$, where $\boldsymbol{n}_t$ is the stack vector of all the inventory levels, and $\boldsymbol{y}_t$ is the time units left for the driver to come back to the buffer to supply with the first and the second dollies. Note that for each state $\boldsymbol{s}_t$, we can calculate the estimated remaining lifetime (ERL) for each lineside buffer, which is the expected time length that the buffer can maintain without replenishment thereafter. Then we can define an event as $e(l_{(1)}, l_{(2)})$, which is the set of all the transitions from a state with the first and second minimal ERL being equal to $l_{(1)}$ and $l_{(2)}$, respectively. So the size of the event space is L^2, where L is the capacity of the lineside buffer. One can easily verify that this is much smaller than the size of the state space. According to the events, one may choose an action, which specifies the indexes of the buffer to supply with the first and the second dollies. Note that such an action can be made only when the driver is at the central docking area. The one-step cost function contains the parts, namely the penalty cost for starvation, the transportation cost for the driver, and the inventory holding cost of the buffers. We want to find an event-based policy to minimize the long-run average cost.

Unfortunately, Eq. 27 is not satisfied in this problem. We therefore apply the gradient-based approach to find a local optimum. Due to the space limit, we omit the rest details of this application and direct interested readers to (Zhao et al. 2010b). In the numerical results therein, it is shown that the system saves about 10.58 % costs on the average by using the policy found by our EBO approach comparing with the policy that is currently adopted in practice. The results also show that the

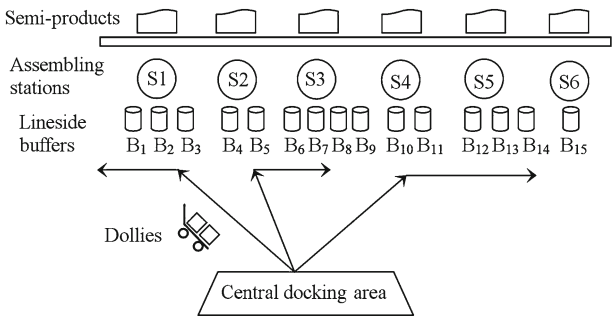


Fig. 9 A sketch of a general assembly line with material handling

EBO approach has consistently good performance under different system parameter settings.

4.4 Portfolio management with unobservable stochastic volatility

In this subsection, we briefly introduce the application of the EBO method to the portfolio management problem with unobservable stochastic volatility. In this example, condition (27) holds. Consider a discrete-time discrete-state version of the problem. In the beginning, one has a portfolio which is composed of a bond asset and a stock asset, the amount of which are denoted by x_0^b and x_0^s, respectively. At time $l+1$, the price of the bond asset is

$$p_{l+1}^b = f^b(p_l^b, r), \tag{36}$$

where f^b is a deterministic function and r is the risk free interest rate. The price of the stock asset is

$$p_{l+1}^s = f^s(p_l^s, \mu, V_l, B_l), \tag{37}$$

where f^s is a deterministic function, μ is the appreciation rate, V_l is the volatility at time l, and B_l is a random walk. Note that the volatility V_l also follows a random walk. In practice, we only observe x_l^b, x_l^s, p_l^b, and p_l^s, and cannot observe V_l. The objective is to maximize the long-run average return, i.e., $\lim_{l\to\infty} E\left\{\log\left(x_l^b p_l^b + x_l^s p_l^s\right)/l | x_0^b, x_0^s, p_0^b, p_0^s\right\}$.

We can formulate this problem as an EBO. Let $\left(p_l^b, p_l^s, V_l\right)$ be the system state. V_l is unobservable and all the other parts of the state are observable. There are different ways to define events. One example is to define the event as the current stock price crossing certain level. Another example is to define events as certain properties of a finite history of the stock prices (e.g., those specified by the statistical quantities used in the technical analysis). The action is the amount of stock and bond asset to hold. When $x_{l+1}^s > x_l^s$, this means that we buy the stock at price p_{l+1}^s and the bond asset will be decreased accordingly. When $x_{l+1}^s < x_l^s$, this means that we sell the stock at price p_{l+1}^s and the bond asset will be increased accordingly.

Note that the volatility V_l is independent of the actions taken by the user. This is because of the assumption that the effect of the transactions of a single user is too small to change the market. Therefore, the distribution of all the events is independent of the policies. This is another example in which the condition (27) naturally holds. One can then apply the aforementioned policy iteration to find the optimal event-based policy of this problem. For more details of this problem, please refer to Wang and Cao (2011).

As a final remark, condition (27) is required to guarantee the optimality of Procedure 3. This condition has to be checked on a case by case basis for any specific problems, just as what we did for the admission control problem in Section 4.1 and the portfolio management problem in Section 4.4. We can also use numerical calculation to verify if Eq. 27 is unsatisfied (find some counterexample of $\pi(i|e)$'s that Eq. 27 is unsatisfied), just like the problems in Sections 4.2 and 4.3.

5 Conclusion

In this paper, we discuss a new optimization framework called EBO which can naturally handle the performance optimization of event-triggered dynamic systems. Inspired by two intuitive optimization methods in the standard MDP, we propose two optimization approaches for the EBO framework. The optimization approaches are based on two fundamental quantities, performance potentials and Q-factors for events. These quantities quantify the long-term effect of states or event-action pairs on the system performance. Based on these quantities, we can improve the event-based policy iteratively. These quantities can be estimated from the system sample path. Thus, the EBO optimization approaches can be implemented with an online manner. The optimality of these optimization approaches are theoretically justified by using the difference equations for EBO. Some application examples are discussed to demonstrate the effectiveness of the EBO approach.

The optimality of EBO approaches requires the condition (27), that is the conditional probability of the system state when a certain event happens is irrelevant to the event-based policy. For the problems not satisfying this condition, the EBO approaches in this paper cannot guarantee to find the optimal policy. However, with appropriate techniques, such as the gradient-based optimization, it is still possible to find a near-optimal solution efficiently. How to relax this condition is an important research topic in the future.

6 Frequently asked questions

For researchers who are familiar with the Markov models, it is very natural to compare it with the proposed event-based model. Here are some frequently asked questions and answers to them.

1. Is event-based model similar to state aggregation?
   Answer: As defined, an event is a set of state transitions in the form of $\langle i, j\rangle$, not a set of states. An event can be used to represent a set of states. Let $S_0 \subseteq \mathcal{S}$ be any set of states, and define $e := \{\langle i, j\rangle : i \in S_0\}$. Then event e is equivalent to $i \in S_0$. Therefore, state aggregation is a special case of events; and events may contain more information than state aggregation.
   Next, every state transition is the simplest event and can be viewed as a fundamental event. Then we have a space of fundamental events consisting of $|\mathcal{S}| \times |\mathcal{S}|$ elements. Every event can be viewed as a subset of this fundamental event space. In this sense, an event can be viewed as an aggregation in this space of fundamental events.
2. Can we model events by augmenting the state at time l to represent both the states before and after a transition at l?
   Answer: Precisely, this means to define a Markov chain with $Y_l = \{X_{l-1}, X_l\}$. Indeed, we can model an event in Markov chain $\{X_l, l = 0, 1, \cdots\}$ by state aggregation in the state space of $\{Y_l, l = 0, 1, \cdots\}$. However, the effect of the controlled actions in the EBO model is different from that in the setting of MDP in the augmented Markov process $\{Y_l, l = 0, 1, \cdots\}$. In the augmented chain, an action at time l controls the transition from X_l to X_{l+1}; while in EBO, a transition at time l splits into two; an observation of event happening is made after the

first part of transition, followed by an action, which controls the second part of transition; all these happen at the same time instant (cf. the admission control example). In this sense, EBO may have more modeling power.

3. In EBO, the occurrence of an event, e.g, a customer arrival, can be modeled by the realization of some exogenous random events; thus, can we incorporate this feature in an MDP by an appropriate state expansion?
   Answer: In some cases, such as the admission control problem, this state expansion method works well; in some other cases, it is difficult to determine the exogenous random events; and in more cases, such exogenous random events may not exist. The EBO model explores the problem structure in a simple and natural way without enlarging the state space.
4. Is an event-based policy similar to restricting the policy space in a Markov model?
   Answer: In some cases, both are indeed similar. For example, policies based on state aggregation is a special case of restricting the policies to a subset in the policy space, and it is also a special case of the event-based policies. Another case is the one discussed in Item 3, where an event-based policy can be modeled by adding exogenous random events; in this case, an event-based policy is similar to restricting policies in the policy space on the expanded state space.
   However, in general, an event-based policy is different from restricting the policy space in a Markov model. As discussed in Item 2, when an event occurs, such as a customer arrives at a network, the state transition may not be completed, and the action based on the event may control the rest of the transition. Both event-based policy and restricting the policy space are two different models.

In summary, EBO is a model that tries to capture the structural information revealed in a physical event without expanding the state space. There also exist many other excellent approaches for the performance optimization of Markov systems; see e.g., Altman (1999), Bertsekas and Tsitsiklis (1996) and Whitt (1978, 1979).

References

Altman E (1999) Constrained Markov decision processes. Chapman & Hall/CRC

Altman E, Avrachenkov KE, Núñez-Queija R (2004) Perturbation analysis for denumerable Markov chains with application to queueing models. Adv. Appl. Probab. 36:839–853

Bertsekas DP, Tsitsiklis JN (1996) Neuro-dynamic programming. Athena, Belmont, MA

Barto A, Mahadevan S (2003) Recent advances in hierarchical reinforcement learning. In: Special issue on reinforcement learning. discrete event dynamic systems: theory and applications 13:41–77

Cao XR (1994) Realization probabilities—the dynamics of queueing systems. Springer, New York

Cao XR (2005) Basic ideas for event-based optimization of Markov systems. Discrete Event Dynamic Systems: Theory and Applications 15:169–197

Cao XR (2007) Stochastic learning and optimization—a sensitivity-based approach. Springer, New York

Cao XR, Chen HF (1997) Potentials, perturbation realization, and sensitivity analysis of Markov processes. IEEE Trans Autom Control 42:1382–1393

Cao XR, Zhang JY (2008) Event-based optimization of Markov systems. IEEE Trans Autom Control 53:1076–1082

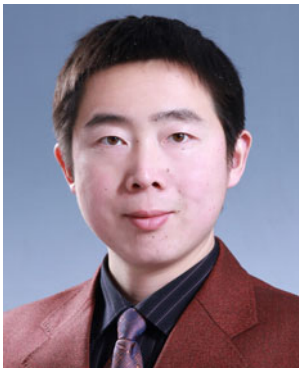
Cao XR, Ren ZY, Bhatnagar S, Fu M, Marcus S (2002) A time aggregation approach to Markov decision processes. Automatica 38:929–943

Ho YC, Cao XR (1991) Perturbation analysis of discrete event systems. Kluwer Academic Publishers, Norwell

Jia QS (2011) On solving event-based optimization with average reward over infinite stages. IEEE Trans Autom Control 56:2912–2917

Leahu H, Heidergott B, Hordijk A (2013) Perturbation analysis of waiting times in the G/G/1 queue. Discrete Event Dyn Syst Theory Appl 23:277–305
Marbach P, Tsitsiklis JN (2001) Simulation-based optimization of Markov reward processes. IEEE Trans Autom Control 46:191–209
Parr RE (1998) Hierarchical control and learning for Markov decision processes. Ph.D. Dissertation, University of California at Berkeley
Puterman ML (1994) Markov decision processes: discrete stochastic dynamic programming. Wiley
Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge, MA
Wang DX, Cao XR (2011) Event-based optimization for POMDP and its application in portfolio management. In: The Proceedings of the 18th IFAC World Congress, 28 Aug–2 Sept 2011. Milano, Italy, pp 3228–3233
Whitt W (1978) Approximation of dynamic programs, I. Math Oper Res 3:231–243
Whitt W (1979) Approximation of dynamic programs, II. Math Oper Res 4:179–185
Xia L (2012) Optimal control of customer admission to an open Jackson network. In: The Proceedings of the 31st Chinese Control Conference, 25–27 July 2013. Hefei, China, pp 2362–2367
Xia L (2013) Event-based optimization of admission control in open queueing networks. Discrete Event Dyn Syst Theory Appl. doi:10.1007/s10626-013-0167-1
Xia L, Cao XR (2012) Performance optimization of queueing systems with perturbation realization. Eur J Oper Res 218:293–304
Yao C, Cassandras CG (2012) Using infinitesimal perturbation analysis of stochastic flow models to recover performance sensitivity estimates of discrete event systems. Discrete Event Dyn Syst Theory Appl 22:197–219
Zhao Y, Yan CB, Zhao Q, Huang N, Li J, Guan X (2010a) Efficient simulation method for general assembly systems with material handling based on aggregated event-scheduling. IEEE Trans Autom Sci Eng 7(4):762–775
Zhao Y, Zhao Q, Guan X (2010b) Stochastic optimal control for a class of manufacturing systems based on event-based optimization. In: Huang J, Liu K-Z, Ohta Y (eds) Theory and applications of complex systems and robust control. Tsinghua University Press, Beijing, China, pp 63–83

Li Xia received the B.E. degree in automation, in July 2002, and the Ph.D. degree in control science and engineering, in July 2007, both from Tsinghua University, Beijing, China. From 2007 to 2009, he was a staff research member in IBM China Research. From 2009 to 2011, he was a postdoctoral research fellow in the King Abdullah University of Science and Technology (KAUST), Saudi Arabia. Presently, he is an assistant professor in the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University. He was a visiting scholar in the Hong Kong University of Science and Technology and a visiting scholar in University of Waterloo, Canada. He serve/served as a reviewer of international journals including IEEE Transactions on Automatic Control, IEEE Transactions on Automation Science and Engineering, European Journal of Operational Research, IIE Transactions, Journal of Optimization Theory and Applications, etc. He is an IEEE member and a member of Operation Research Society of China. His Research interests include the methodology research in stochastic analysis and optimization, queueing theory, discrete event systems, and the application research in building energy, scheduling and optimization of smart grid, wireless sensor networks, production systems, etc.



Qing-Shan Jia received the B.E. degree in automation in July 2002 and the Ph.D. degree in control science and engineering in July 2006, both from Tsinghua University, Beijing, China. He is an Associate Professor in the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University, and also a visiting scholar in Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. He was a visiting scholar at Harvard University in 2006, and a visiting scholar at the Hong Kong University of Science and Technology in 2010. His research interests include theories and applications of discrete event dynamic systems (DEDSąś) and simulation-based performance evaluation and optimization of complex systems.

Xi-Ren Cao received the M.S. and Ph.D. degrees from Harvard University, in 1981 and 1984, respectively. From 1984 to 1986, he was a research fellow at Harvard University. From 1986 to 1993, he worked as consultant engineer/engineering manager at Digital Equipment Corporation, Massachusetts, U.S.A. From 1993 to 2010, he was with the Hong Kong University of Science and Technology (HKUST), where he served as reader/professor/chair professor. Since July 2010, he is a chair professor of Shanghai Jiao Tong University and an affiliate member of the Institute for Advanced Study, Hong Kong University of Science and Technology. Dr. Cao owns three patents in data- and tele- communications and published three books in the area of performance optimization and discrete event dynamic systems. He received the Outstanding Transactions Paper Award from the IEEE Control System Society in 1987, the Outstanding Publication Award from the Institution of Management Science in 1990, Outstanding Service Award from IFAC in 2008, and the National Natural Science Award (2nd class), China, in 2009. He is a Fellow of IEEE (1996), a Fellow of IFAC (2008), and is/was the Chairman of IEEE Fellow Evaluation Committee of IEEE Control System Society, Editor-in-Chief of Discrete Event Dynamic Systems: Theory and Applications, Associate Editor at Large of IEEE Transactions of Automatic Control, and Board of Governors of IEEE Control Systems Society and on the Technical Board of IFAC. His current research areas include financial engineering, stochastic learning and optimization, performance analysis of economic systems, and discrete event dynamic systems.