2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Sample Path Sharing in Simulation-Based Policy Improvement

Di Wu[1], Qing-Shan Jia[1], and Chun-Hung Chen[2]

Abstract— **Simulation-based policy improvement (SBPI) has been widely used to improve given base policies through simulation. The basic idea of SBPI is to estimate all the Q-factors for a given state using simulation, and then select the action that achieves the minimal cost. It is therefore of great importance to efficiently use the given budget in order to select the best action with high probability. Different from existing budget allocation algorithms that estimate Q-factors by independent simulation, we share the sample paths to improve the probability of correctly selecting the best action. Our method can be combined with equal allocation, Successive Rejects, and optimal computing budget allocation to enhance their probabilities of correct selection as well as to achieve better policies in SBPI. Such improvement depends on the overlap in reachable states under different actions. Numerical results show that with such overlap, combining our method with equal allocation, Successive Rejects and optimal computing budget allocation produces higher probability of selection as well as better policies in SBPI.**

keywords— **Discrete event dynamic system, simulation-based optimization, optimal computing budget allocation.**

I. INTRODUCTION

Simulation-based policy improvement (SBPI) has been widely used in practice to improve given base policies through simulation. It is an important approximate solution methodology to approach large-scale Markov decision process. SBPI improves the current policy by iteration in an online fashion. For a known current state, a Q-factor is used to measure the cost of taking an action in the current stage, followed by the optimal policy in the future stages. Since the optimal policy is not known in advance, the Q-factor is usually estimated through simulation. Based on the estimates, the action that achieves the minimum is selected, which completes the decision-making for the current stage. When the system evolves to the next stage, similar procedures can be followed. Due to the randomness in the system dynamics, a large number of sample paths are usually required to estimate the Q-factors. Therefore, it is of great practical interest to study how to best utilize the given computing budget so that the best action can be selected with high probability.

*This work was supported in part by the National Science Foundation of China under grants (Nos. 61174072, 61222302, 90924001, 91224008, and U1301254), the National 111 International Collaboration Project (No. B06002), and the TNList Funding for Cross Disciplinary Research.

[1] Di Wu and Qing-Shan Jia are with Center for Intelligent and Networked Systems (CFINS), Department of Automation, TNLIST, Tsinghua University, Beijing 100084, China. Email: woody10074026@gmail.com jiaqs@tsinghua.edu.cn

[2] Chun-Hung Chen is with Department of Systems Engineering & Operations Research, George Mason University, 4400 University Drive, MS 4A6, Fairfax, VA 22030, USA. He is also with National Taiwan University. Email: cchen9@gmu.edu

A brief literature review on the existing studies of the computing budget allocation in SBPI will be given in section II. In those work independent simulation of Q-factors for different actions are usually used. In this paper we consider how to share the sample paths to improve the probability of correct selection. We make the following contributions. First, we develop an algorithm to aggregate the sample paths according to the states that are visited in the simulation. Second, we analyze the performance of this algorithm and show under which cases the algorithm is effective. Third, through numerical experiments we demonstrate that our algorithm can be easily combined with other computing budget allocation procedures such as equal allocation, Successive Rejects, and optimal computing budget allocation (OCBA). We also discuss the limitation of our algorithm.

The rest of this paper is organized as follows. We briefly review the related works in section II, and mathematically formulate the problem in section III. Our method is presented in section IV. Numerical results can be found in section V, and a brief conclusion is given in section VI.

II. RELATED WORKS

Markov decision process (MDP) has provided a general framework for modeling many problems in control, decision making, and optimization [1]. It is well known that exact solution methods such as the traditional policy iteration and value iteration suffer from the curse of dimensionality. Many methods have been developed and can be classified into two groups. In the first group, the methods explore the system structure to solve the problem exactly and fast, such as state aggregation [2], [3], time aggregation [4], [5], and action elimination [6], [7]. In the second group, the methods solve the problem approximately, such as neuro-dynamic programming [8], reinforcement learning [9], approximate dynamic programming [10], event-based optimization [11]–[14], and receding horizon approach [15].

Rollout is a simulation-based policy improvement (SBPI) method that was originally developed to tackle deterministic optimization problems [16] and later extended to stochastic optimization problems [17]. SBPI improves given base policies through simulation, and has been successfully applied to many problems including the quiz problems, wireless sensor network [18], water source management [19], and engine maintenance problems [20]. Because simulation could be very time-consuming and a large number of replications is usually used in SBPI, it is of great practical interest to seek a method that efficiently uses the sample paths so that the best action is selected with high probability.

There have been many studies on computing budget allocation in simulation-based optimization, such as ranking and selection (R&S) including both the indifference-zone formulation [21] and the subset selection formulation [22], [23], multi-comparison procedure [24], hit and run [25], [26], COMPASS [27], and nested partitions [28], just to name a few. Excellent reviews are available in [29]–[31]. In this paper, we combine our method with two of the existing allocation algorithms, namely Successive Rejects [32] and OCBA [33]–[36].

We develop an algorithm to aggregate the sample paths according to the states that are visited in the simulation. This introduces correlation among the estimation of Q-factors. By combining our algorithm with Successive Rejects and OCBA, we achieve higher probability of correct selection for a single stage, and better policy in the long run. This will be demonstrated by numerical results in section IV.

III. PROBLEM FORMULATION

Consider a stationary and finite-horizon MDP with discrete state space $\mathcal{S}$ and discrete action space $\mathcal{A}$. The objective is to minimize the overall cost within T stages. Assume we have a base policy. SBPI then uses this base policy to improve the decision making in each stage. We provide more details for a single stage in the following.

At stage k, the state s_k becomes available. Let $\mathcal{A} = \{1, \ldots, n\}$ be the set of feasible actions. The Q-factor for each action $a \in \mathcal{A}$ is defined as

$$Q(s_k, a) = c(s_k, a) + \sum_{s' \in \mathcal{S}} P(s'|s_k, a) v(s'), \tag{1}$$

where $c(s_k, a)$ is the deterministic immediate cost if action a is taken at state s_k, $P(s'|s_k, a)$ is the one-step transition probability to s' if action a is taken at s_k, and $v(s')$ is the value function that represents the future cost under the optimal policy, which is defined as

$$v(s') = \mathbf{E}\left[\sum_{t=k+1}^{T} c(s_t, \pi^*(s_t)) \middle| s_{k+1} = s' \right], \tag{2}$$

where π^* denotes the optimal policy. The action to be taken under policy π^* at state s_k is

$$\pi^*(s_k) = \arg\min_{a \in \mathcal{A}} Q(s_k, a). \tag{3}$$

In practice, however, the optimal policy π^* is not known in advance (otherwise we would have already solved the problem and implemented policy π^*) and we usually replace it with a given base policy π^b in Eq. (1). In this way, we obtain the following equations

$$\hat{Q}(s_k, a) = c(s_k, a) + \sum_{s' \in \mathcal{S}} P(s'|s_k, a) \hat{v}(s'), \tag{4}$$

where

$$\hat{v}(s') = \mathbf{E}\left[\sum_{t=k+1}^{T} c(s_t, \pi^b(s_t)) \middle| s_{k+1} = s' \right]. \tag{5}$$

Let π^{PI} denote the policy that is generated by using SBPI and the base policy π^b. Then we have

$$\pi^{PI}(s_k) = \arg\min_{a \in \mathcal{A}} \hat{Q}(s_k, a). \tag{6}$$

Note that $\hat{Q}$ in Eq. (4) is an expectation, which can only be accurately evaluated by infinite number of replications, i.e.,

$$\hat{Q}(s_k, a) = \lim_{N_a \to \infty} \frac{1}{N_a} \sum_{i=1}^{N_a} \left[c(s_k, a) + \sum_{t=k+1}^{T} c(s_t, \pi^b(s_t)|\xi_i) \right], \tag{7}$$

where ξ_i represents the randomness in the ith simulation. This may be approximated by a finite number of replications, i.e.,

$$\tilde{Q}(s_k, a) = \frac{1}{N_a} \sum_{l=1}^{N_a} \left[c(s_k, a) + \sum_{t=k+1}^{T} c(s_t, \pi^b(s_t)|\xi_l) \right], \tag{8}$$

for $a = 1, \ldots, n$. Then action that is observed as the best action is

$$a_1 = \arg\min_{a=1,\ldots,n} \tilde{Q}(s_k, a). \tag{9}$$

Note that a_1 may not minimize $\hat{Q}(s_k, a)$. So we define the probability of correct selection (PCS) as

$$PCS = \Pr\left\{ \hat{Q}(s_k, a_1) \le \hat{Q}(s_k, a), a \in \mathcal{A}, a \ne a_1 \right\}. \tag{10}$$

The problem we consider here is how to maximize PCS under a fixed given computing budget, i.e.,

$$\max_{N_i, i=1,\ldots,n} PCS, \text{ s.t. } \sum_{i=1}^{n} N_i = N, \tag{11}$$

where N is the total available computing budget.

IV. SAMPLE PATH SHARING

Conventionally, all the sample paths generated by the existing computing budget allocation algorithms are directly averaged to obtain estimates of Q-factors according to Eq. (8). In this section, we present an algorithm to aggregate the sample paths to generate better estimates of Q-factors and therefore achieves a higher PCS. Our method can be combined with any allocation strategy.

A. Sample path generation

There have been many studies on how to efficiently allocate a fixed budget to different solution candidates. In this paper we consider two of them. The first one is Successive Rejects (SR), which was developed to handle the multi-armed bandit problem. In our settings, the fixed number of budget corresponds to "the number of pulls", and the best action corresponds to "the best arm". SR eliminates the action that is observed as the worst after each round of simulation. The budget allocated to each action at the end of the kth round can be calculated by

$$n(k) = \left\lceil \frac{1}{\overline{log}(N)} \frac{N - n}{n + 1 - k} \right\rceil \quad \forall k = 1, \ldots, n-1, \tag{12}$$

where n is the number of feasible actions and $\overline{log}(N) = 0.5 + \sum_{k=2}^{n} 1/k$. Note that SR can be implemented offline because it does not utilize any new information during simulation.

OCBA was developed to address simulation-based optimization problems in general. Its basic idea is to allocate the computing budget so that the best solution candidate is separated from the rest of the solution candidates with the highest probability. A systematic introduction to OCBA is now available in [37].

B. Estimation of reachable states

Note that when different actions are taken at s_k, the same state may be visited in the next stage in different sample paths. We define set of reachable states in the next stage if action a is taken at state s as

$$R_a(s) = \{s' | s' \in S, P(s'|s,a) > 0\}. \tag{13}$$

Then the set of reachable states under different actions is

$$R(s) = \cup_{a\in\mathcal{A}} R_a(s), \tag{14}$$

and Eq. (4) can be rewritten as

$$\hat{Q}(s_k,a) = c(s_k,a) + \sum_{s'\in R(s_k)} P(s'|s_k,a)\hat{v}(s'). \tag{15}$$

As a result, for each $s' \in R(s_k)$, we can aggregate the sample paths that share the same s' to obtain a more accurate estimate of $\hat{v}(s')$ and use Eq. (15) to estimate $\hat{Q}$. In practice, we may not know $R(s_k)$ in advance and it must be estimated from the given N sample paths. We denote the estimate as $\tilde{R}_N(s_k)$, which is a subset of $R(s_k)$, and our method can then be applied to $\tilde{R}_N(s_k)$.

C. Estimation of one-step transition probabilities

In practice we may not know the one-step transition probabilities. This can be addressed in two ways. First, when single-step simulation is possible, we can perform extra one-step simulation to estimate the one-step transition probabilities. It should not be time-consuming because only one-step simulation is required in each replication. Second, when the states that are visted in the next stage can be stored in the simulation, we can estimate the transition probability directly from the existing sample paths. We focus on this second way in the following discussion. Denote $m(s',a)$ as the number of sample paths with $s_{k+1} = s'$ when action a is taken at s_k. The estimate of $P(s'|s_k,a)$ is

$$\tilde{P}(s'|s_k,a) = \frac{m(s',a)}{\sum_{s\in R(s_k)} m(s,a)}. \tag{16}$$

D. Estimation of value function

Assume that we can store the states that are visited in the next stage during all the simulation. In this case, each sample path that starts from the same state $s_{k+1} = s'$ and follows the same base policy π^b in the rest of the stages provides an estimate of $v(s')$. The average of all such estimates then provide an estimate of $v(s')$. Now we have

$$\tilde{Q}(s_k,a) = c(s_k,a) + \sum_{s'\in\tilde{R}_N(s_k)} \tilde{P}(s'|s_k,a)\tilde{v}(s'). \tag{17}$$

The above procedures are summarized in Algorithm 1, where $\tilde{v}_i(s')$ is the estimate of $v(s')$ in the ith sample path, and $B(s')$ is the sum of $\tilde{v}_i(s')$.

Algorithm 1 Sample path sharing

1: **Input:** N sample paths starting from $s_k = s$.
2: Step 0. Set $\hat{R}(s) = \varnothing, m = \varnothing, B = \varnothing$.
3: Step 1.
for $i = 1,\dots,N$ **do**
Get the $s_{k+1} = s'$ and action a for the ith sample path
if $s' \notin \hat{R}(s)$
$\hat{R}(s) = \hat{R}(s) \bigcup s'$,
$m(s',a) = 0$,
$B(s') = 0$.
end if
$m(s',a) = m(s',a) + 1$,
$B(s') = B(s') + \tilde{v}_i(s')$.
end for
4: Step 2.
Estimate $P(s'|s,a)$ using Eq. (16).
5: Step 3.
for each $s' \in \hat{R}(s)$ **do**
$\tilde{v}(s') = B(s')/\sum_{a\in\mathcal{A}} m(s',a)$.
end for
6: Step 4. Calculate $\tilde{Q}(s,a)$ using Eq. (17).
7: **Output:** $a_1 = \arg\min_{a=1,\dots,n} \tilde{Q}(s,a)$

E. Discussion

In this subsection, we briefly discuss why and when our method may achieve higher PCS. First we need to justify the use of Eq. (4) rather than Eq. (8) in estimating Q-factors. The estimation error of $\hat{Q}$ contains two parts, namely the estimation error of P and the estimation error of $\hat{v}$. In Eq. (8) the two terms P and $\hat{v}$ are estimated based on the sample paths under the same action a. In Eq. (4) the two terms are estimated using all the sample paths that are obtained through the simulation under all the actions. When there are overlap among the sample paths under different actions, Eq. (4) leads to smaller sample variance in P and $\hat{v}$ than that in Eq. (8). In particular, the sample-path sharing may have good performance under the following two conditions.

Condition 1. $|R(s)| \ll N$. Because one and only one state in $R(s)$ is visited in each sample path. When $|R(s)|$ is much smaller than N, this implies that most states in $R(s)$ will be visited by multiple times. This leads to more accurate estimation of P and $\hat{v}$.

Condition 2. $|R(s,a) \cap R(s,a')| \approx |R(s)|$ for any $a, a' \in \mathcal{A}$. This means that there are a lot of overlap between the reachable states under different actions. This condition suggests that more sample paths can be aggregated to get better estimates of P and $\hat{v}$.

Note that when the above two conditions are not satisfied, our algorithm simply recovers the naive estimation of $\hat{Q}$ as in Eq. (8). In order to see this, consider the extreme case in which $R(s,a) \cap R(s,a') = \varnothing$ for any $a, a' \in \mathcal{A}$. This means that the sample paths under different actions reach different states in the next stage. So there is not any sample paths to share. This is also demonstrated by numerical results in the next section.

V. NUMERICAL RESULTS

We demonstrate the performance of our methods by conducting three experiments. In the first experiment, we compare the PCS of 6 algorithms (EA, SR, OCBA, EA-Sharing, SR-Sharing and OCBA-Sharing) on a 10-state MDP. In the second experiment, we slightly modify the transition probabilities in the first experiment and re-examine the performance of our method. In the third experiment, we compare the policies obtained from 4 algorithms (EA, OCBA, EA-Sharing and OCBA-Sharing) based on a finite-state controllable random walk.

In the first two experiments, we compare the performance of the following methods.

Method 1: Equal Allocation (EA). EA equally allocates the computing budget among different actions.

Method 2: Successive Rejects (SR). SR successively eliminates the action that is observed as the worst in each iteration and eventually outputs the action that is observed as the best.

Method 3: OCBA. OCBA sequentially allocates the computing budget among different actions.

Method 4: EA-Sharing (EA-S). EA-S equally allocates the computing budget among different action candidates and then uses Algorithm 1 to aggregate the sample paths.

Method 5: SR-Sharing (SR-S). SR-S uses SR to allocate the computing budget among the action candidates and then uses Algorithm 1 to aggregate the sample paths.

Method 6: OCBA-Sharing (OCBA-S). OCBA-S uses OCBA to allocate the computing budget among the action candidates and then uses Algorithm 1 to aggregate the sample paths.

A. Experiment 1

Consider an MDP with 10 states as that is shown in Fig. 1, where the curves indicate reachable states and the numbers beside the curves represent the one-step transition probabilities. Note that red curves stand for the probabilities that can be controlled by taking different actions. There are five actions available in stage 1, i.e., $\mathcal{A} = \{1, \ldots, 5\}$. One can verify that action 1 is the best. The cost function is $c(s, a) = 0, s \neq 1$, and $c(1, a) = 1$, and we want to minimize the total cost over $T = 100$ stages. The base policy always picks action 1 at state 1. Suppose the initial state is 1. We use the six methods to allocate the computing budget when $N = 100, 200, \ldots, 1000$. Their performance are estimated by 10000 replications. The parameters for OCBA is $n_0 = 10$ and $\Delta = 10$. When the exact values of the transition probabilities are used in the methods, the results are shown in Fig. 2. When the transition probabilities are estimated, the results are shown in Fig. 3. We make the following remarks.

Remark 1. In Fig. 2, our algorithm significantly improves the PCS. This is demonstrated by comparing EA with EA-S, SR with SR-S, and OCBA with OCBA-S. Actually after sharing the sample paths, the PCS of different methods are close to each other.

Remark 2. In Fig. 3, due to the estimation error of the transition probabilities, these PCSs are smaller than those in Fig. 2. But our algorithm still improves the PCS and this advantage increases with N.

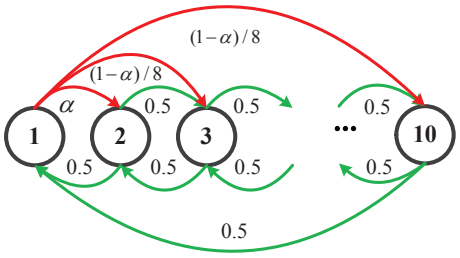

Fig. 1. A Markov chain with 10 states.

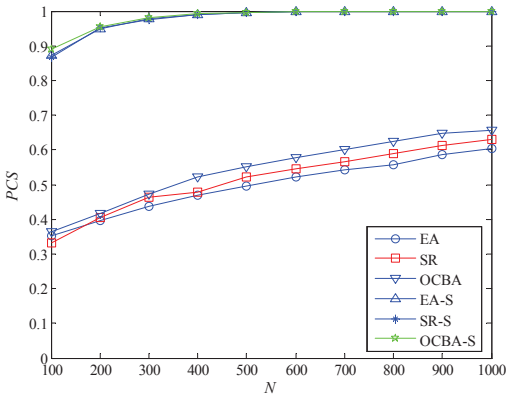

Fig. 2. PCS based on accurate transition probabilities in experiment 1.

B. Experiment 2

In the second experiment, we follow the same settings as that in the first experiment except for some slight modifications in the transition probabilities. The one-step transition probability matrix now is

$$\begin{aligned} P(s' = 2i - 1 | s = 1, a = i) &= 0.5, \\ P(s' = 2i | s = 1, a = i) &= 0.5, \\ P(s' | s = 1, a = i) &= 0, \text{otherwise}. \end{aligned} \tag{18}$$

In this case, only two states can be reached when an action is taken. Furthermore, the reachable states under different actions are completely disjoint, namely $R(1, a) \cap R(1, a') = \varnothing$ for $a, a' \in \mathcal{A}$ and $a \neq a'$. In other words, when different actions are taken, the state will always transit to different states, and there is no overlap between different sample paths in terms of s_{k+1}. One can verify that action 3 is the best action. The results are summarized in Fig. 4. We make the following remarks.

Remark 3. Fig. 4 shows that when there is not any overlap in the reachable states under different actions, our algorithm neither improves nor degrades PCS. It simply recovers the case where the sample paths are not aggregated at all.

C. Experiment 3

In the third experiment, a finite-state controllable random walk is used to evaluate the policies that are obtained from 4 methods (EA, OCBA, EA-S and OCBA-S). To be more specific, let $\mathcal{S} = \{-10, -9, \ldots, 10\}$ be the state space. If $s =$

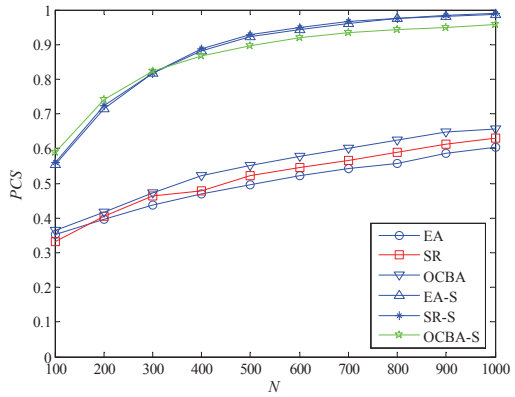

Fig. 3. PCS based on estimated transition probabilities in experiment 1.

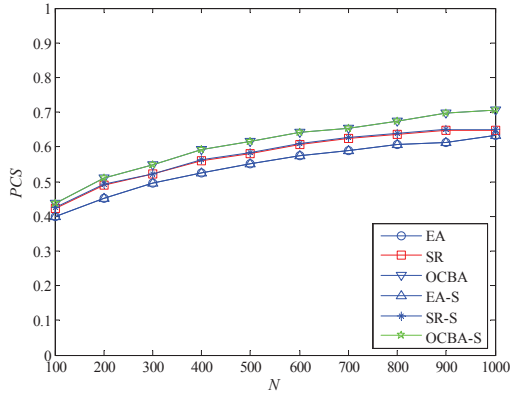

Fig. 4. PCS based on estimated transition probabilities in experiment 2.

10 (or -10), the system will jump to state 9 (or -9) for sure in the next step. For all the other states, there are three available actions $\mathcal{A} = \{-1, 0, 1\}$. We have for $i \in \mathcal{S}$ and $|i| < 10$,

$$\begin{aligned} \Pr\{s' = i+1 | s = i, a = 0\} &= 0.5, \\ \Pr\{s' = i+1 | s = i, a = -1\} &= 0.2, \\ \Pr\{s' = i+1 | s = i, a = 1\} &= 0.8. \end{aligned}$$

In other words, actions -1 and 1 tend to move the system to the left and the right, respectively, while action 0 does not have any preference but just pushing the system away from the current state. The cost function is $c(s_k, a) = |s_k|$. Suppose the state is at 0 in the beginning. The base policy picks action 0 regardless of the current state. We want to minimize the total cost over $T = 100$ stages. Intuitively, a "sensible" policy should always take action 1 for negative states and take action -1 for positive states. This encourages the system to stay at state 0 with high probability and leads to a low total cost in the long run.

During each iteration, every visited state from -9 to 9 will be given a budget of 100 (no action can be taken at state -10 or 10). We estimate the performance of the 4 policies by 1000 replications. The policies are plotted in Fig. 5, where the horizontal axis is the state space and the vertical axis is the action space. The estimated total cost of the policies that

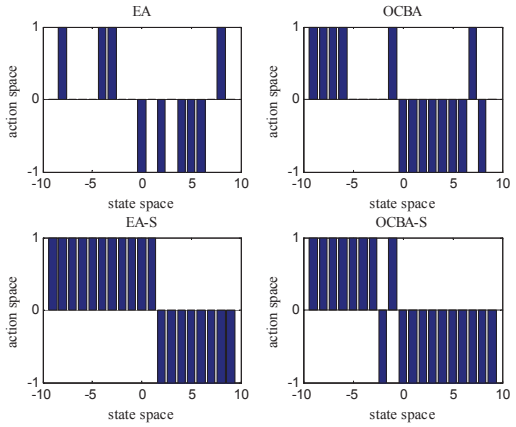

Fig. 5. Policies obtained from the 4 methods in experiment 3.

are obtained by these methods are summarized in Table I. We make the following remarks.

TABLE I
PERFORMANCES UNDER THE 4 POLICIES IN EXPERIMENT 3.

	EA	**OCBA**	**EA-S**	**OCBA-S**
Total Cost	186 ± 67	188 ± 88	156 ± 19	159 ± 42

Remark 4. Table I shows that the policies that are obtained from using EA-S and OCBA-S achieve smaller cost than EA and OCBA. This is consistent with Fig. 5 in which both EA-S and OCBA-S tend to attract the system around state 0.

Remark 5. The variances of the total costs of the policies that are obtained by EA-S and OCBA-S are also smaller than that of EA and OCBA, respectively.

Remark 6. Note that the set of reachable states from a common state under different actions are identical. So both aforementioned conditions are satisfied in this problem. This explains the good performance of our algorithm.

VI. CONCLUSION

In this paper, we consider the problem of computing budget allocation in simulation-based policy improvement. Different from existing methods that estimate Q-factors by independent simulation under different action candidates, we develop a sample path sharing procedure to aggregate the sample paths according to the states that are visited in the next stage. Our method can be easily combined with other computing budget allocation procedures. Numerical results show that our method can be combined with equal allocation, Successive Rejects, and OCBA to improve the probability of correct selection in a single stage, and to output better policies. We plan to improve our method to consider the estimation error of the transition probabilities when these values are unknown. Note that a special issue on event-based control and optimization is upcoming [38], which includes seven interesting papers covering various recent progress in this area [39]–[45]. It is a future work to apply our method to event-based optimization [39]. Also, one can aggregate the

sample paths according to the states that have been visited in each stage and simulation as long as there are enough memory. This may lead to a higher PCS and a better policy.

REFERENCES

[1] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: John Wiley and Sons, Inc., 1994.

[2] Z. Ren and B. H. Krogh, "State aggregation in markov decision processes," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Dec. 2002, pp. 3819–3824.

[3] Q.-S. Jia, "On state aggregation to approximate complex value functions in large-scale markov decision processes," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 333–344, 2011.

[4] X. R. Cao, Z. Y. Ren, S. Bhatnagar, M. Fu, and S. Marcus, "A time aggregation approach to markov decision processes," *Automatica*, vol. 38, no. 6, pp. 929–943, 2002.

[5] T. Sun, Q. C. Zhao, and P. B. Luh, "Incremental value iteration for time aggregated markov decision processes," *IEEE Transactions on Automatic Control*, vol. 52, pp. 2177–2182, 2007.

[6] L. Xia, Q. Zhao, and Q.-S. Jia, "A structure property of optimal policies for maintenance problems with safety-critical components," *IEEE Transactions on Automation Science and Engineering*, vol. 5, no. 3, pp. 519–531, 2008.

[7] Q.-S. Jia, "A structural property of optimal policies for multi-component maintenance problems," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 677–680, 2010.

[8] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

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

[11] X. R. Cao, "A basic formula for online policy gradient algorithms," *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 696–699, May 2005.

[12] ——, *Stochastic Learning and Optimization: A Sensitivity-Based Approach*. New York, NY: Springer, 2007.

[13] Q.-S. Jia, "On solving optimal policies for finite-stage event-based optimization," *IEEE Transactions on Automatic Control*, vol. 56, no. 9, pp. 2195–2200, 2011.

[14] ——, "On solving event-based optimization with average reward over infinite stages," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2912–2917, Dec. 2011.

[15] H. S. Chang and S. I. Marcus, "Approximate receding horizon approach for markov decision processes: Average reward case," *Journal of Mathematical Analysis and Applications*, vol. 286, no. 2, pp. 636–651, 2003.

[16] D. P. Bertsekas, J. N. Tsitsiklis, and C. Wu, "Rollout algorithms for combinatorial optimization," *Heuristics*, vol. 3, pp. 245–262, 1997.

[17] D. P. Bertsekas and D. A. Castañon, "Rollout algorithms for stochastic scheduling problems," *Journal of Heuristics*, vol. 5, pp. 89–108, 1999.

[18] Q.-S. Jia, "A rollout method for finite-stage event-based decision processes," in *Proceedings of the 2010 Workshop on Discrete Event Systems*, Berlin, Germany, 2010, pp. 257–262, aug. 30 - Sept. 1.

[19] Y. Zhao, X. Chen, Q.-S. Jia, X. Guan, S. Zhang, and Y. Jiang, "Long-term scheduling for cascaded hydro energy systems with annual water consumption and release constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 4, pp. 969–976, 2010.

[20] Q.-S. Jia, "Efficient computing budget allocation for simulation-based policy improvement," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 2, pp. 342–352, Apr. 2012.

[21] R. E. Bechhofer, "A single-sample multiple decision procedure for ranking means of normal populations with known variances," *The Annals of Mathematical Statistics*, vol. 25, pp. 16–39, 1954.

[22] S. S. Gupta, "On a decision rule for a problem in ranking means," Ph.D. dissertation, University of North Carolina, Chapel Hill, NC, 1956.

[23] ——, "On some multiple decision (ranking and selection) rules," *Technometrics*, vol. 7, pp. 225–245, 1965.

[24] C. W. Dunnett, "A multiple comparison procedure for comparing several treatments with a control," *Journal of the American Statistical Association*, vol. 50, no. 272, pp. 1096–1121, Dec. 1955.

[25] R. L. Smith, "Efficient monte carlo procedures for generating points uniformly distributed over bounded region," *Operations Research*, vol. 32, pp. 1296–1308, 1984.

[26] Z. B. Zabinsky, R. L. Smith, J. F. McDonald, H. E. Romeijn, and D. E. Kaufman, "Improving hit-and-run for global optimization," *Journal of Global Optimization*, vol. 3, no. 2, pp. 171–192, 1993.

[27] L. J. Hong and B. L. Nelson, "Discrete optimization via simulation using compass," *Operations Research*, vol. 54, no. 1, pp. 115–129, 2006.

[28] L. Shi and S. Olafsson, "Nested partitions method for global optimization," *Operations Research*, vol. 48, no. 3, 2000.

[29] R. E. Bechhofer, T. J. Santner, and D. Goldsman, *Design and Analysis of Experiments for Statistical Selection, Screening and Multiple Comparisons*. New York, NY: Wiley, 1995.

[30] J. R. Swisher, S. H. Jacobson, and E. Yücesan, "Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey," *ACM Transactions on Modeling and Computer Simulation*, vol. 13, pp. 134–154, 2003.

[31] S.-H. Kim and B. L. Nelson, "Selecting the best system: Theory and methods," in *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, Eds. Piscataway, NJ: IEEE, 2003, pp. 101–112.

[32] J.-Y. Audibert, S. Bubeck *et al.*, "Best arm identification in multi-armed bandits," *COLT 2010-Proceedings*, 2010.

[33] C.-H. Chen, "A lower bound for the correct subset-selection probability and its application to discrete event system simulations," *IEEE Transactions on Automatic Control*, vol. 41, pp. 1227–1231, 1996.

[34] H.-C. Chen, C.-H. Chen, and E. Yücesan, "Computing efforts allocation for ordinal optimization and discrete event simulation," *IEEE Transactions on Automatic Control*, vol. 45, no. 5, pp. 960–964, May 2000.

[35] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems - Theory and Applications*, vol. 10, pp. 251–270, 2000.

[36] C.-H. Chen and E. Yücesan, "An alternative simulation budget allocation scheme for efficient simulation," *International Journal of Simulation and Process Modeling*, vol. 1, pp. 49–57, 2005.

[37] C.-H. Chen and L. H. Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. Singapore: World Scientific, 2010.

[38] Q.-S. Jia and K. H. Johansson, "Guest editorial: Event-based control and optimization," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-014-0181-y

[39] L. Xia, Q.-S. Jia, and X.-R. Cao, "A tutorial on event-based optimization - a new optimization framework," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-013-0170-6

[40] L. Xia, "Event-based optimization of admission control in open queueing networks," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-013-0167-1

[41] L. Li and M. Lemmon, "Weakly coupled event triggered output feedback system in wireless networked control systems," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-013-0165-3

[42] Y. Sun and X. Wang, "Stabilizing bit-rates in networked control systems with decentralized event-triggered communication," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-013-0169-z

[43] M. C. F. Donkers, P. Tabuada, and W. P. M. H. Heemels, "Minimum attention control for linear systems - a linear programming approach," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-012-0155-x

[44] G. A. Kiener, D. Lehmann, and K. H. Johansson, "Actuator saturation and anti-windup compensation in event-triggered control," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-012-0151-1

[45] A. Molin and S. Hirche, "A bi-level approach for the design of event-triggered control systems over a shared network," *Discrete Event Dynamic Systems - Theory and Applications*, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10626-012-0156-9