Journal of Korea Multimedia Society Vol. 23, No. 7, July 2020(pp. 830-839)
https://doi.org/10.9717/kmms.2020.23.7.830

마르코프 결정 과정에서 시뮬레이션 기반 정책 개선의 효율성 향상을 위한 시뮬레이션 샘플 누적 방법 연구

황시랑†, 최선한††

A Simulation Sample Accumulation Method for Efficient Simulation-based Policy Improvement in Markov Decision Process

Xi-Lang Huang†, Seon Han Choi††

ABSTRACT

As a popular mathematical framework for modeling decision making, Markov decision process (MDP) has been widely used to solve problem in many engineering fields. MDP consists of a set of discrete states, a finite set of actions, and rewards received after reaching a new state by taking action from the previous state. The objective of MDP is to find an optimal policy, that is, to find the best action to be taken in each state to maximize the expected discounted reward of policy (EDR). In practice, MDP is typically unknown, so simulation-based policy improvement (SBPI), which improves a given base policy sequentially by selecting the best action in each state depending on rewards observed via simulation, can be a practical way to find the optimal policy. However, the efficiency of SBPI is still a concern since many simulation samples are required to precisely estimate EDR for each action in each state. In this paper, we propose a method to select the best action accurately in each state using a small number of simulation samples, thereby improving the efficiency of SBPI. The proposed method accumulates the simulation samples observed in the previous states, so it is possible to precisely estimate EDR even with a small number of samples in the current state. The results of comparative experiments on the existing method demonstrate that the proposed method can improve the efficiency of SBPI.

Key words: Markov Decision Process, Simulation-based Policy Improvement, Sample Accumulation Method, Efficiency Improvement

1. 서 론

마르코프 결정 과정(MDP: Markov decision process)은 의사 결정 과정을 모델링하기 위한 잘 알려진 수학적 도구로서 무선 센서 네트워크[1], 영상 인식[2], 배송 로봇[3], 모바일 클라우드 컴퓨팅[4], 에이전트 학습[5] 등의 다양한 분야에 활용된다. MDP는 이산 상태(State) 집합과 한정된 개수의 동작(Action) 집합으로 구성되며, 각 상태에서 수행해야 할 동작을 정의한 것을 정책(Policy) π라고 한다. 에이전트(Agent)로 표현되는 의사 결정자가 특정 시점 t에서 어떤 상태 s에서 정책에 따른 동작 a(즉, $\pi(s)$)를 수행하였을 때, 다음 시점 $t+1$에서 상태는 s'으로 변경되며, 동작에 따른 결과로 즉각적인 보상 $r(s')$을 받는

※ Corresponding Author : Seon Han Choi, Address: (48513) 45 Yongso-ro, Nam-gu, Busan, Korea, TEL : +82-51-629-6240, FAX : +82-51-629-6240, E-mail : shchoi@pknu.ac.kr
Receipt date : Jun. 17, 2020, Approval date : Jul. 7, 2020

† Dept. of Artificial Intelligence Convergence, Graduate School, Pukyong National University
(E-mail : huangxl901@pukyong.ac.kr)
†† Dept. of IT Convergence and Application Eng., School of Engineering, Pukyong National University
※ This work was supported by the Pukyong National University Research Fund in 2019(C-D-2019-1038).

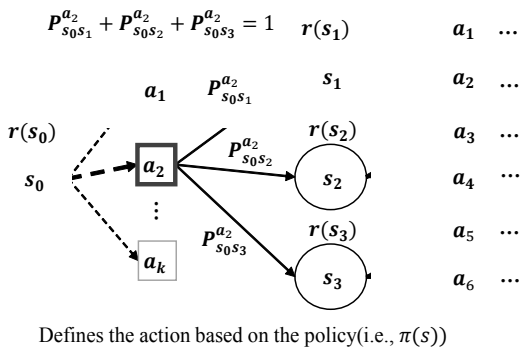

Fig. 1. A simple example of MDP.

다. 다음 상태 s'은 확률적으로 결정이 되며, s에서 a에 따라 s'에 도달할 전이 확률(State transition probability)은 $P^a_{ss'}$로 정의된다. 전이 확률로 인하여 s에서 같은 동작 a를 수행하여도 도달하는 s'에 따라서 다른 보상을 받을 수 있으며, s'에 포함되는 모든 상태에 대해서 전이 확률의 합은 1이다(즉 $\sum_{\forall s'} P^a_{ss'} = 1$)[6]. Fig. 1은 간단한 MDP의 예를 나타낸다.

MDP의 주된 목적은 미래에 얻게 될 할인된 보상의 기댓값(EDR: Expected discounted reward)을 최대화하는 최적의 정책을 찾는 것이다(EDR에 대해서는 다음 장에서 자세히 정의한다). 예를 들어서 최단 경로로 미로를 빠져나가기 위해서 각 갈림길에서 이동해야 할 방향을 찾는 것이라고 할 수 있다. 상태 집합과 전이 확률이 명확하게 알려진 경우에 최적의 정책은 동적 계획법(Dynamic programming)[7]으로 찾을 수 있다. 하지만 실제 시스템을 모델링하는 MDP의 경우 일반적으로 상태 집합이 매우 크며, 상태 집합과 전이 확률이 알려지지 않은 경우가 대부분이다. 이러한 경우 강화학습(Reinforcement learning)으로 알려진 Q-Learning 방법[8] 또는 학습 오토마타(Learning automata) 방법[9]으로 찾을 수 있다. 또 다른 방법으로는 시뮬레이션 기반 정책 개선(SBPI: Simulation-based policy improvement) 방법[10]이 활용될 수 있으며, 이 방법은 주어진 기본 정책을 기반으로 각 상태에서 EDR을 최대화 하는 최선의 동작을 선택하므로 순차적으로 기본 정책을 개선하는 방법이다. 실제 시스템의 경우 기본 매뉴얼 등으로 제시되는 기본 정책을 가지고 있는 경우가 많으므로, 이 기본정책을 바탕으로 개선된 정책을 찾는 SBPI는 상대적으로 실용적인 방법이라고 할 수 있다. 따라서 SBPI는 풍력 발전 계획 최적화[11], 빌딩 에너지 분산 최적화[12], 대피 정책 최적화[13] 등에 적용되었다.

SBPI에서 시뮬레이션은 어떤 상태로부터 주어진 기본 정책에 따라서 일련의 동작들을 취하면서 얻어지는 보상들의 할인된 합, 즉 EDR에 대한 샘플을 관찰하는 것을 의미한다. 전이 확률로 인하여 매 시뮬레이션 마다 얻어지는 샘플 값은 다를 수 있다. SBPI를 통해 기본 정책을 개선하기 위해서는 한 상태에서 수행 가능한 동작들에 대해서, 각 동작을 취하고 이후에 기본 정책에 따라서 일련의 동작들을 수행함에 따라 얻게 될 EDR을 최대화하는 최선의 동작을 선택해야 한다. 알려져 있지 않은 전이 확률로 인하여 EDR은 시뮬레이션을 통해 얻어진 샘플들의 평균을 통해 점추정 될 수 있다. 하지만, 만약 각 상태에서 수행 가능한 동작들의 개수가 많고, 각 동작을 수행함에 따라서 도달할 수 있는 상태가 많을 경우, 수집된 EDR에 대한 각 샘플은 큰 분산을 가지게 된다. 따라서 각 상태에서 최선의 동작을 정확하게 선택하기 위해서는 각 동작에 대한 정밀한 EDR 추정치를 얻기 위해 많은 독립적 반복 시뮬레이션이 필요하며, 이는 효율성 문제를 야기할 수 있다[14].

순위 및 선택(R&S: Ranking and selection) 방법은 이러한 효율성 문제를 해결할 수 있는 적절한 해답이 될 수 있다[15]. 왜냐하면 R&S는 한정된 시뮬레이션 반복 횟수를 시뮬레이션 결과의 통계적 추론에 기반 하여 지능적으로 각 동작에 할당하므로 최선의 동작을 효율적으로 선택할 수 있도록 하기 때문이다. 따라서 각 동작에 대해서 한정된 반복 횟수를 동일하게 할당하는 것에 비해서 보다 정확하게 최선의 동작을 선택할 수 있다. Jia[16]는 다양한 R&S 알고리즘들 중에서 최적 컴퓨팅 자원 할당(OCBA: Optimal computing budget allocation) 방법을 SBPI에 그대로 적용하여서 각 상태에서 최선의 동작을 효율적으로 찾도록 하였다. 또한 Wu et al.[17]는 각 동작에 대한 EDR의 샘플을 추출할 때, 첫 동작 이후 수행되는 일련의 동작들은 모두 기본 정책에 따라 진행된다는 것을 바탕으로 공통되는 EDR 샘플들을 바탕으로 선택의 정확도를 향상시키는 방법을 제시하였다. 이전 방법들과의 비교 실험 결과는 이 방법을 적용하므로 효율성이 향상된다는 것을 입증한다.

SBPI를 통해서 기본 정책을 바탕으로 최적의 정

책을 도출하기 위해서는 각 상태를 순회하면서 기본 정책의 EDR이 더 이상 개선되지 않을 때 까지 각 상태에서 기본 정책의 동작을 개선해야 한다. 하지만 앞서 언급한 기존의 효율성 개선을 위한 연구들은 한 상태에서 최선의 동작을 선택할 때의 효율성 개선에만 집중하였다. 이는 SBPI의 효율성 개선에 분명히 기여하는 바가 있지만, 동작 개선을 반복하는 SBPI의 특성상 그 기여도는 제한적일 수밖에 없다. 본 논문에서는 각 상태에서 최선의 동작을 선택하기 위해 수행된 시뮬레이션 샘플들을 공유하고 누적하므로 SBPI의 전체 효율성을 개선시킬 수 있는 방법을 제시한다. 각 상태에서의 시뮬레이션 샘플을 공유하고 누적하기 위해서, 상태와 동작에 따른 전이 확률을 추정하기 위한 표를 제시한다. 상태를 순회하며 시뮬레이션이 진행 될수록 누적된 결과는 전이 확률을 보다 정밀하게 추정할 수 있도록 하며, 이를 바탕으로 각 동작에 대한 EDR을 보다 정확하게 계산할 수 있다. 누적된 시뮬레이션 결과로 계산된 EDR은 기존의 방법들이 각 상태에 한정된 시뮬레이션 횟수로 추정한 EDR 보다 상대적으로 더 정확하게 최선의 동작을 선택할 수 있도록 하며, 이는 SBPI를 위한 전체 시뮬레이션 횟수가 한정되어 있는 상황에서 보다 최적의 정책을 도출하도록 한다. 본 방법만 활용되었을 경우에도, 단일 상태에 한정되는 기존의 방법에 비하여 SBPI의 전체 효율성을 높일 수 있지만, 본 방법은 Jia가 제시한 OCBA를 적용하는 방법과 동시에 활용될 수 있으며, 이 경우 SBPI의 효율성을 보다 개선할 수 있다. 예제 문제에 대한 실험 결과는 이 효율성 개선을 입증한다.

본 논문의 구성은 다음과 같다. 2장에서 문제를 정의하고, 3장에서 시뮬레이션 샘플을 누적하므로 SBPI의 효율성을 개선하기 위한 방법을 제시한다. 4장에서는 예제 문제를 통하여 제안하는 방법이 효율성을 개선할 수 있음을 기존 연구와의 비교 실험을 통하여 입증한다. 마지막으로 5장에서 결론을 맺는다.

2. 문제 정의

한 상태 s에서 동작 a를 수행하였을 때 도달할 수 있는 다음 상태가 s_1, s_2, s_3가 있다고 하면, 각 상태에 도달함에 따라 받는 즉각적인 보상이 $r(s_1)$, $r(s_2)$, $r(s_3)$으로 정의된다. 이 경우 s에서 a를 수행함에 따라서 받을 수 있는 보상 $f(s,a)$는 다음과 같이 정의될 수 있다.

$$f(s,a)\in\{r(s_1),r(s_2),r(s_3)\} \tag{1}$$

각 상태에 대한 전이 확률을 $P^a_{ss_1}$, $P^a_{ss_2}$, $P^a_{ss_3}$라고 할 때, 보상의 기댓값은 다음과 같이 계산될 수 있다.

$$E[f(s,a)]=r(s_1)\cdot P^a_{ss_1}+r(s_2)\cdot P^a_{ss_2}+r(s_3)\cdot P^a_{ss_3} \tag{2}$$

s에서 a를 수행하여 도달하는 상태에서 주어진 기본 정책 π에 따라서 일련의 동작들을 순차적으로 수행할 때, 각 동작을 수행함에 따른 보상을 얻게 된다. 예를 들어서 시점 t에 s에서 a를 수행하여 s_1에 도달하고, $t+1$에 s_1에서 π에 따라 정의된 a_1을 수행하여 s_3에 도달하고, $t+2$에 s_3에서 π에 따라 정의된 a_2을 수행하여 s_5에 도달했다고 할 때, $f(s,a)$, $f(s_1,a_1)$, $f(s_3,a_2)$을 보상으로 얻을 수 있다. 이 경우 $f(s,a)=r(s_1)$, $f(s_1,a_1)=r(s_3)$, $f(s_3,a_2)=r(s_5)$이 되며 전이확률에 따라서 다른 결과를 얻을 수 있다. s_5이후 계속해서 동작을 수행해서 얻어진 보상들을 그대로 더하게 될 경우 보상의 누적합은 발산하게 된다. 이를 방지하고자 현재 시점 $t=0$을 기준으로 미래에 얻어질 보상의 경우(즉, $t+1$, $t+2$ 등의 시점에서 얻어지는 보상) 할인 인자 $\gamma\in[0,1]$를 통하여 감하게 된다. 따라서 s에서 a를 수행하고, 그 이후에 기본 정책에 따라서 일련의 동작들을 순차적으로 수행할 때 얻어지는 미래에 얻게 될 할인된 보상의 기댓값 EDR은 다음과 같이 정의된다.

$$Q_\pi(s,a)=\lim_{T\to\infty}\left\{E[f(s,a)]+E\left[\sum_{t=1}^{T-1}\gamma^t f(s_t,\pi(s_t))\,|\,s,a\right]\right\} \tag{3}$$

여기서 s_t는 t시점의 상태를 나타내며, s_{t-1}에서 $\pi(s_{t-1})$의 동작을 수행하였을 때 도달하는 상태이다. 단, s_1의 경우 s에서 a를 수행하였을 때 도달하는 상태이다. 또한 식 (3)의 두 번째 항에서 기댓값에 s,a의 조건이 있는 이유는 $f(s_t,\pi(s_t))$의 기댓값을 계산할 때 사용되는 전이 확률이 s,a를 기반으로 계산되기 때문이다. 이는 다음 장에서 더 자세히 다룬다(식 (12)를 참고하라).

한 상태 s에서 수행할 수 있는 동작 $a_1,\ldots,a_k$이 있을 때, 기본 정책을 개선하기 위해서는 k개의 동작 중에서 $Q_\pi(s,a_i)$를 최대화하는 최선의 동작 a_b를 선택해야 하며, 이는 아래와 같이 정의된다.

$$\pi_{PI}(s)=a_b=\arg\max_{a_i\in\{a_1,\ldots,a_k\}}Q_\pi(s,a_i) \tag{4}$$

여기서 π_{PI}는 기본 정책 π로부터 한 단계(One-step), 즉 한 상태에서의 동작 개선이 이루어진 정책을 의미한다. 하지만 실제로 알려지지 않은 전이 확률로 인하여 $Q_\pi(s,a_i)$를 계산할 수 없으므로, SBPI는 시뮬레이션을 통해서 얻은 $Q_\pi(s,a_i)$의 샘플을 바탕으로 추정된 값을 사용하여 π_{PI}를 도출한다. 한 번의 시뮬레이션을 통해 얻은 $Q_\pi(s,a_i)$의 샘플은 아래와 같이 정의되며, 이 때 식 (3)에서와 같이 무한 시점까지 동작을 수행할 수 없으므로 T는 한정된 숫자로 제한되며, 이 제한된 수를 에폭(Epoch)이라고 한다.

$$\hat{Q}_\pi^T(s,a_i)=f(s,a_i)+\sum_{t=1}^{T-1}\gamma^t f(s_t,\pi(s_t)) \tag{5}$$

샘플들을 바탕으로 계산되는 $Q_\pi(s,a_i)$의 추정값은 아래와 같이 계산될 수 있다.

$$\overline{Q}_\pi^T(s,a_i)=\frac{1}{n_i}\sum_{l=1}^{n_i}\hat{Q}_\pi^{T,l}(s,a_i) \tag{6}$$

$\overline{Q}_\pi^T(s,a_i)$은 중심 극한 정리(Central limit theorem)에 따라서 n_i가 커질수록 $Q_\pi(s,a_i)$을 평균(Mean)으로 하는 정규 분포(Normal distribution)를 따르게 된다. SBPI를 통해서 도출된 한 단계 개선된 정책 π_{SBPI}는 다음과 같이 정의된다.

$$\pi_{SBPI}(s)=a_e=\arg\max_{a_i\in\{a_1,\ldots,a_k\}}\overline{Q}_\pi^T(s,a_i) \tag{7}$$

$\overline{Q}_\pi^T(s,a_i)$를 바탕으로 선택된 최선의 동작 a_e는 $\overline{Q}_\pi^T(s,a_i)$가 가지고 있는 확률적 노이즈로 인하여 $Q_\pi(s,a_i)$를 바탕으로 선택된 실제 최선의 동작 a_b와 다를 수 있다. a_e와 a_b가 같을 확률을 나타내는 선택의 정확도($P\{\mathrm{CS}\}$: Probability of correct selection)는 아래와 같이 정의되며, Bonferroni 부등식과 베이지안 사후 확률(Bayesian posterior probability)을 바탕으로 $P\{\mathrm{CS}\}$의 하한을 계산할 수 있다[15].

$$P\{\mathrm{CS}\}=P\{a_b=a_e\}\geq 1-\sum_{i=1,i\neq e}^{k}P\left\{\tilde{Q}_\pi^T(s,a_e)<\tilde{Q}_\pi^T(s,a_i)\right\} \tag{8}$$

여기서 $\tilde{Q}_\pi^T(s,a_i)$는 $Q_\pi(s,a_i)$에 대한 관측된 샘플들을 바탕으로 도출된 베이지안 사후 분포를 의미하며, $Q_\pi(s,a_i)$에 대한 사전 정보가 없다고 할 때 $\tilde{Q}_\pi^T(s,a_i)$는 평균이 $\overline{Q}_\pi^T(s,a_i)$인 정규 분포를 따른다[15]. 각 동작에 대하여 n_i을 증가시켜서 $\overline{Q}_\pi^T(s,a_i)$의 정밀도를 높일수록 $P\{\mathrm{CS}\}$를 증가시킬 수 있으나, kn개의 많은 시뮬레이션 반복 횟수가 소모되어 효율성 문제를 야기한다. 앞서 언급한 기존 연구에서는 이 효율성을 개선하기 위하여 한정된 반복 횟수 N이 있을 때, OCBA를 활용하여 이를 k개의 동작에 적절하게 배분하여서 $P\{\mathrm{CS}\}$를 최대화 하도록 하며 이러한 문제는 아래와 같이 정의될 수 있다.

$$\arg\max_{n_1,\ldots n_k}P\{\mathrm{CS}\}\quad \text{s.t.}\sum_{i=1}^{k}n_i=N \text{and} n_i\geq 0 \tag{9}$$

즉 OCBA를 활용하여 $\tilde{Q}_\pi^T(s,a_i)$를 바탕으로 실제 최선의 동작 a_b를 더 적은 반복 횟수로 정확하게 찾으므로 효율성을 높인다. Algorithm 1은 이를 위한 OCBA 절차를 나타낸다.

시작 상태를 s_{init}라고 가정할 때(만약 시작 상태가 존재하지 않는 경우 아무 상태를 사용하여도 무방하다), 식 (3)을 바탕으로 정책 π에 대한 EDR $\eta_\pi(s_{init})$은 아래와 같이 정의될 수 있다.

$$\eta_\pi(s_{init})=\lim_{T\to\infty}E\left[\sum_{t=0}^{T-1}\gamma^t f(s_t,\pi(s_t))|s_0=s_{init}\right] \tag{10}$$

SBPI의 최종 목적은 상태를 순회하며 각 상태에서 EDR을 최대화하는 최선의 동작을 선택하여 주어진 기본 정책 π를 순차적으로 개선해나가는 것, 즉 $\eta_\pi(s_{init})$를 최대화하는 것이다. 일반적으로 활용 가능한 시뮬레이션 반복 횟수는 시간 또는 컴퓨팅 자원의 형태로 제한되어 있는 경우가 많다. 이 제한된 총 반복 횟수를 B라고 할 때, $\eta_\pi(s_{init})$를 최대화 하는 문제는 다음과 같이 정의된다.

$$\max\eta_\pi(s_{init})\quad \text{s.t.}\sum_{i=1}^{m}N=B \tag{11}$$

여기서 N은 각 상태에서 최선의 동작을 선택하기 위해 사용되는 반복 횟수이며, m은 상태 순회 횟수(중복 포함) 이다.

상태 순회 방식이 고정되어 있을 때, 순회하는 각 상태에서 최선의 동작을 매번 정확하게(즉 $P\{\mathrm{CS}\}\cong 1$) 선택할 수 있다면), 최소의 순회 횟수로 $\eta_\pi(s_{init})$를 최대화 할 수 있다. 하지만 매번 정확한 선택을 위해서는 (비록 OCBA를 적용하여 필요한 반복 횟수를 크게 줄였지만) 각 상태에서 많은 반복 횟수 N이 요구되고, 이는 식 (11)에서 보듯이 m을 감소시켜 $\eta_\pi(s_{init})$가 최댓값에 수렴하지 못할 수도 있다. 따라서 SBPI의 효율성을 높이기 위해서는 N을 증가시키지 않고도 각 상태에서 최선의 동작을 정확하게 선택

Algorithm 1. Optimal Computing Budget Allocation Procedure for Selecting the Best Action Out of k Alternatives [15]

Input: $\Theta=\{a_1,...,a_k\}$ (a set of k action alternatives), N (a limited number of simulation replications)

Control Param.: n_0 (the initial number of simulation replications), Δ (the one-time increment, i.e., the number of replications additionally allocated per iteration)

Output: a_e (the estimated best action)

Procedures

set the iteration number $l\leftarrow 0$

collect initial n_0 samples of $\hat{Q}_\pi(s,a_i)$ (i.e., simulate n_0 times) **for each** a_i, $i\in\{1,...,k\}$

set $n_1^l=n_2^l=\cdots=n_k^l\leftarrow n_0$

calculate $\overline{Q}_\pi^T(s,a_i)$ and the sample variance s_i^2 **for each** a_i

select $a_e\leftarrow \arg\max_{a_i\in\{a_1,...,a_k\}}\overline{Q}_\pi^T(s,a_i)$

while $\sum_{i=1}^{k}N_i^l<N$ **do**

  set $S^{l+1}\leftarrow\sum_{i=1}^{k}n_i^l+\min\left(\Delta,N-\sum_{i=1}^{k}n_i^l\right)$

  calculate n_i^* **for each** a_i using the below equations:

$$\frac{n_i^*}{n_j^*}=\left[\frac{s_i/\left(\overline{Q}_\pi(s,a_e)-\overline{Q}_\pi(s,a_i)\right)}{s_j/\left(\overline{Q}_\pi(s,a_e)-\overline{Q}_\pi(s,a_j)\right)}\right]^2,\quad i,j\in\{1,...,k\},\text{ and } i\neq j\neq e,$$

$$n_e^*=s_e\sqrt{\sum_{i=1,i\neq e}^{k}\left(\frac{n_i^*}{s_i}\right)^2},\text{ and }\sum_{i=1}^{k}n_i^*=S^{l+1}$$

  collect additional $\max(n_i^*-n_i^l,0)$ samples of $\hat{Q}_\pi(s,a_i)$ **for each** a_i

  set $n_1^{l+1}\leftarrow n_1^l+\max(n_i^*-n_i^l,0)$ **for each** a_i

  update $\overline{Q}_\pi^T(s,a_i)$ and s_i^2 **for each** a_i using the additionally collected samples

  select $a_e\leftarrow \arg\max_{a_i\in\{a_1,...,a_k\}}\overline{Q}_\pi^T(s,a_i)$

  set $l\leftarrow l+1$

end while

return a_e

할 수 있도록 해야 한다. 본 논문에서는 이를 위해서 각 상태에서 수행한 시뮬레이션의 결과를 공유하고 누적하므로, 순회가 진행될수록 누적된 데이터를 바탕으로 최선의 동작을 정확하게 선택할 수 있도록 하는 방법을 제시한다.

3. 제안한 방법

식 (3)의 EDR이 기댓값의 형태로 정의되는 것은 각 동작에 따른 전이 확률이 존재하기 때문이다. 이 전이 확률을 사용하여서 식 (3)은 식 (2)와 같이 아래 식으로 계산될 수 있다.

$$Q_\pi(s,a)=\sum_{s_1\in g(s,a)}P_{ss_1}^a r(s_1)+\gamma\sum_{s_1\in g(s,a)}P_{ss_1}^a\left[\sum_{s_2\in g(s_1,\pi(s_1))}P_{s_1s_2}^{\pi(s_1)}r(s_2)\right]+\gamma^2\sum_{s_1\in g(s,a)}P_{ss_1}^a\left[\sum_{s_2\in g(s_1,\pi(s_1))}P_{s_1s_2}^{\pi(s_1)}\left[\sum_{s_3\in g(s_2,\pi(s_2))}P_{s_2s_3}^{\pi(s_2)}r(s_3)\right]\right]+\cdots \quad (12)$$

여기서 $g(s,a)$는 s에서 a를 수행하므로 도달할 수 있는 다음 상태의 집합을 의미한다. 식 (12)가 Σ의 곱으로 정의되는 것은 식 (3)의 두 번째 항 기댓값 식에 존재하는 s,a의 조건으로 인하여 s_t에 도달했을 때 받을 수 있는 보상 $r(s_t)$에 대한 확률 값은 s에서부터 a와 π에 따른 일련의 동작을 수행하여 s_t에 도달

할 확률 $P_{ss_t}^{a,\ldots,\pi(s_{t-1})}$로 정의되기 때문이다. $P_{ss_t}^{a,\ldots,\pi(s_{t-1})}$는 s에서 s_t에 도달하기까지 거치는 각 상태에 대한 전이확률 $P_{s_{t-1}s_t}^{\pi(s_{t-1})}$의 곱으로 계산된다. 식 (12)를 그대로 컴퓨터로 계산하는 것은 많은 메모리 사용 및 반복으로 인하여 적합하지 않다. 이는 아래와 같이 재귀 형식으로 다시 정리될 수 있다.

$$Q_\pi(s,a)=\sum_{s_1\in g(s,a)} P_{ss_1}^{a}\left[r(s_1)+\gamma Q_\pi(s_1,\pi(s_1))\right] \tag{13}$$

식 (5)에서 나타나듯이, 한 번의 시뮬레이션을 수행하게 되면, s를 포함한 T개의 상태에 대해서 a를 포함한 T개의 동작을 각각 수행하였을 때 도달하는 상태에 대한 샘플을 수집할 수 있다. 즉 상태와 동작의 조합에 대해서 그 동작을 그 상태에서 수행하였을 때의 도달하는 상태에 대한 빈도를 수집할 수 있으며, 이는 Fig. 2의 표 형태로 정리될 수 있다.

각 행은 상태와 동작의 조합에 대해서 이 동작을 총 수행한 횟수와, 동작을 수행하였을 때 도달한 상태의 빈도를 나타낸다. 이 동작을 수행하였을 때 도달할 수 있는 모든 상태 집합을 알고 있지 않기 때문에, 새로운 상태에 도달하는 경우 우측에 추가된다. 상태와 동작의 새로운 조합이 수행될 경우 새로운 행에 그 데이터가 추가된다.

본 논문에서 제안하는 방법은 위의 표를 바탕으로 각 상태에 대한 시뮬레이션 샘플을 공유하고 누적하는 것이다. 각 상태를 순회하며 최선의 동작을 선택하기 위해 시뮬레이션 반복 횟수가 증가할수록 더 많은 데이터가 표에 누적되게 되며, 이를 바탕으로 아래와 같이 전이 확률에 대한 불편(unbiased) 추정치를 계산할 수 있다.

	# of state-action pairs	Next state s'			
(s,a_1)	$\sum_{s'\in g(s,a_1)} N_{ss'}^{a_1}$	state: s_1 reward: $r(s_1)$ $N_{ss_1}^{a_1}$	state: s_2 reward: $r(s_2)$ $N_{ss_2}^{a_1}$	state: s_3 reward: $r(s_3)$ $N_{ss_3}^{a_1}$	…
(s,a_2)	$\sum_{s'\in g(s,a_2)} N_{ss'}^{a_2}$	state: s_4 reward: $r(s_4)$ $N_{ss_4}^{a_2}$	state: s_5 reward: $r(s_5)$ $N_{ss_5}^{a_2}$	… …	… …
(s_1,a_3)	$\sum_{s'\in g(s_1,a_3)} N_{s_1s'}^{a_3}$	state: s_6 reward: $r(s_6)$ $N_{ss_6}^{a_3}$	… …	… …	… …
⋮	⋮	⋮	⋮	⋮	

Fig. 2. Simulation sample accumulation table.

$$\overline{P}_{ss'}^{a}=\frac{N_{ss'}^{a}}{\sum_{s'\in g(s,a)} N_{ss'}^{a}} \tag{14}$$

그리고 이 추정치와, 각 상태에 대한 관찰된 즉각적인 보상 r을 사용하여, 식 (13)을 기반으로 아래와 같이 EDR의 추정치를 도출할 수 있다.

$$\overline{Q}_\pi(s,a)=\sum_{s_1\in g(s,a)} \overline{P}_{ss_1}^{a}\left[r(s_1)+\gamma\overline{Q}_\pi(s_1,\pi(s_1))\right] \tag{15}$$

기존의 방법의 경우 OCBA를 통해서 효율적으로 반복 횟수를 할당하더라도, 식 (6)을 통해 EDR의 추정치를 도출한다. 각 상태의 최선의 동작을 선택하기 위해 할당되는 반복 횟수가 N이라고 할 때, 각 동작에 대해서 식 (6)을 통해 계산하기 위해 사용되는 샘플의 개수는 최대 N일 수 밖에 없다. 하지만, 제안하는 방법은 시뮬레이션 결과가 누적된 Fig. 2의 표를 바탕으로 식 (15)를 통해 추정하므로 이 N의 한계를 넘어설 수 있다. 기존의 방법의 경우 순회가 계속되더라도, 각 상태에서 최선의 동작을 정확하게 선택할 확률은 서로 독립적인 반면에, 제안하는 방법은 순회가 계속될수록 더 누적된 데이터를 바탕으로 최선의 동작을 정확하게 선택할 확률을 높일 수 있다. SBPI의 경우 기본 정책을 바탕으로 각 상태의 동작을 순차적으로 개선해 나가기 때문에, 각 상태에서 선택하는 최선의 동작은 현재 주어진 기본 정책 하에서 최선으로 간주되는 동작이다. 기본 정책은 최적의 정책이 아니기 때문에, 기본 정책이 변경되는 경우 같은 상태에서도 최선의 동작은 달라질 수 있다. 따라서 SBPI를 통해서 $\eta_\pi(s_{init})$를 최대화하는 최적의 정책을 찾기 위해서 같은 상태라도 정책 개선 이후에 다시 순회해야 할 필요성이 있다. 이러한 SBPI의 특성상 각 상태에서 정확한 선택을 위해서 N을 높이는 것은 중요하지만 비효율적이다. 제안하는 방법은 각 상태의 시뮬레이션 샘플을 계속해서 누적시키기 때문에, N이 작은 상황에서도 데이터가 누적될수록 정확하게 최선의 정책을 선택할 수 있게 된다. 이는 순회 초기에 선택한 최선의 동작이 나중에 가서 변경될 수도 있는 SBPI의 특성을 고려할 때, 제안하는 방법은 SBPI의 전체 효율성 개선에 효과적인 방법이 된다.

4. 실 험

제안된 방법의 효율성 향상을 입증하기 위해서 본 장에서는 기존의 방법들과의 비교실험을 수행한 결

Table 1. Summary of comparison methods

Name	Description	Allocation way of given N in each state	Applying the proposed method (simulation sample accumulation method)	Calculating $\overline{Q}_{\pi}(s,a)$
EA	Standard SBPI	Equal allocation: $n_i = N/k$	X	Equation (6)
OCBA[16]	Applying OCBA for efficient allocation of N	OCBA (Algorithm 1)	X	Equation (6)
OCBA-S [17]	Improving OCBA[16] method with the sample path sharing	OCBA (Algorithm 1)	X	Based on equation (6) (see [17] for detail)
Proposed	Applying the proposed method to the standard SBPI	Equal allocation: $n_i = N/k$	O	Equation (15)
Proposed ×OCBA	Applying the proposed method to OCBA for further improving the efficiency of SBPI	OCBA (Algorithm 1)	O	Equation (15)

과를 제시한다. Table 1은 비교실험에 사용된 방법들을 요약한다. EA는 대조군으로 활용되는 기본 SBPI를 나타내며, 각 상태에서 모든 동작에 동일한 횟수의 시뮬레이션 반복을 수행하여 얻은 샘플들로 식 (6)을 바탕으로 EDR의 추정치를 도출하여 최선의 동작을 선택한다. 예를 들어서 5개의 동작이 있고, 50개의 반복 횟수가 주어지는 경우, 각 동작마다 10개의 $\hat{Q}_{\pi}^{T}(s,a_i)$를 수집하고, 10개 샘플의 평균으로 $\overline{Q}_{\pi}(s,a)$를 계산한다. OCBA는 Jia[16]가 제시한 방법으로 Algorithm 1의 OCBA를 활용하여 주어진 반복 횟수를 k개의 동작에 할당한 뒤, 각 동작에 대해 식 (6)을 통해 $\overline{Q}_{\pi}(s,a)$를 계산하고, 이를 바탕으로 최선의 동작을 선택한다. OCBA-s는 Wu et al.[17]이 제시한 OCBA의 개선안으로, 동일하게 Algorithm 1의 OCBA를 활용하여 주어진 반복 횟수를 할당하지만, 공통되는 EDR 샘플들을 활용하기 위해 식 (6)을 기반으로 수정된 식을 바탕으로 $\overline{Q}_{\pi}(s,a)$를 계산한다(자세한 사항은 [17]를 참고하라). Proposed의 경우 제안하는 방법의 효율성 향상을 입증하기 위해서 기본 SBPI에 제안하는 방법을 적용한 것이다. 또한 앞서 언급하였듯이 제안하는 방법은 OCBA 방법과 함께 적용되어 효율성을 추가로 향상시킬 수 있음을 입증하기 위해 Proposed×OCBA를 추가하였다.

Fig. 3은 본 비교 실험에 사용된 MDP를 나타낸다. 본 MDP는 OCBA 방법을 검증하기 위해 활용된 모델[16]에 상태를 추가하여 개선한 버전이다. MDP는 2개의 상태를 가지고 있으며, 각 상태에서 20개의 동작을 수행할 수 있다. s_1에서 한 동작 α_i를 수행하는 경우 전이 확률에 따라서 s_1또는 s_2에 도달하게 되고, 이때 전이 확률 $P_{s_1s_1}^{a_i}$는 $0.05(i-1)$로 정의되며, $P_{s_1s_2}^{a_i}$는 $1-P_{s_1s_1}^{a_i}=1-0.05(i-1)$로 정의된다. 마찬가지로 s_2에서 한 동작 β_i를 수행하는 경우 전이 확률에 따라서 s_2또는 s_1에 도달하게 되고, 이때 전이 확률 $P_{s_2s_2}^{\beta_i}$는 $0.05(i-1)$로 정의되며, $P_{s_2s_1}^{\beta_i}$는 $1-P_{s_2s_2}^{\beta_i}=1-0.05(i-1)$로 정의된다. 그리고 각 상태에 도달했을 때 받는 즉

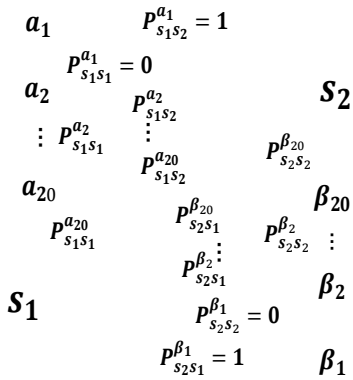

Fig. 3. A two-state MDP model.

각적인 보상 $r(s_1)$은 -1이며, $r(s_2)$는 0이다. 본 MDP의 최적 정책은 s_1에서 α_1을, s_2에서 β_{20}을 수행하는 것이며, 이 경우 EDR은 0이다.

하지만 최적의 정책을 찾아야 하는 에이전트는 본 MDP의 상태가 몇 개 인지, 각 동작에 따른 전이 확률은 어떻게 되는지, 각 상태에서 받는 즉각적인 보상은 무엇인지 어떠한 정보도 가지고 있지 않다. 에이전트에 주어진 정보는 시작 상태 s_1과 기본 정책 $\pi=\{s_1:\alpha_{11}, s_2:\beta_{11}\}$이다. 에이전트는 SBPI를 활용하여 시뮬레이션을 통해 동작에 따른 상태 천이와, 도달하는 상태, 그리고 보상을 관찰하고 이 결과를 바탕으로 각 상태에서 최선의 동작을 선택하므로 기본 정책을 개선해 나간다. 본 실험에서는 이 상황에서 SBPI의 효율성을 향상시키기 위해 Table 1의 방법들을 적용하였다. 각 상태에서 할당되는 시뮬레이션 반복 횟수, 즉 추출 가능한 샘플 개수 N은 60이며, OCBA를 활용하는 경우 Algorithm 1에 표시된 n_0와 Δ는 각각 2로 설정되었으며, 시뮬레이션 에폭 T는 10으로 설정되었다. 모든 방법은 동일한 상태 순회 방식을 따르며 s_1에서 시작해서 s_1과 s_2를 반복한다. Table 1의 각 방법에 대해서 주어진 기본 정책으로부터 상태 순회 방식을 따라 매 번 최선의 동작을 선택하여 최적의 정책에 도달하기 까지 (즉, 더 이상 EDR이 개선되지 않을 때 까지) 순회한 상태의 개수를 측정하였으며, Fig. 4는 그 결과를 나타낸다. 그리고 Fig. 5는 순회하는 각 상태에서 선택의 정확도 $P\{CS\}$를 나타낸다. Fig. 4와 Fig. 5의 결과는 각 방법에 대해서 5,000번의 독립적인 반복 실험을 통해서 얻어진 평균치이다.

실험 결과에서 보듯이, 제안하는 샘플 누적 방법을 적용한 Proposed가 기존의 방법인 EA, OCBA, OCBA-S에 비해서 빠르게 수렴하는 것을 확인할 수 있다. 즉 Proposed는 적은 순회 횟수만으로도 최적의 정책에 도달하므로 SBPI의 효율성을 높인다. 특히 제안하는 방법과 OCBA를 함께 적용한 Proposed×OCBA의 경우 기본 SBPI에 제안하는 방법을 적용한 Proposed에 비해서 더 빠르게 수렴하므로 SBPI의 효율성을 더 향상시킬 수 있음을 보여준다.

기본 SBPI인 EA의 경우, Fig. 5의 $P\{CS\}$가 굉장히 낮으며, 이는 부정확한 값의 $\overline{Q}_\pi(s,a)$로 인하여 최선의 동작을 제대로 선택하지 못함을 나타낸다. 즉 EA의 경우 최선의 동작을 정확하게 선택하기 위해서는 각 동작에 대한 정밀한 $\overline{Q}_\pi(s,a)$을 얻기 위한 더 큰 값의 N이 필요하다. EA는 낮은 선택의 정확도로 인해서 Fig. 4에서 보듯이 상태 순회를 계속 하여도 최적의 정책에 수렴하지 못하는 것을 확인할 수 있다. 반면에 OCBA와 OCBA-S의 경우 Algorithm 1의 OCBA로 주어진 N을 효과적으로 할당하여 각 상태에서 최선의 동작을 정확하게 찾는 것을 Fig. 5에서 확인할 수 있다. 따라서 Fig. 4에서 보듯이 EA에 비해서 빠르게 수렴하므로 SBPI의 효율성을 높인다. EA, OCBA, OCBA-S 모두 각 상태에서의 시뮬레이션 결과를 각 상태에서만 활용하는 것에 비해서, 제안하는 샘플 누적 방법을 적용한 Proposed와 Pro-

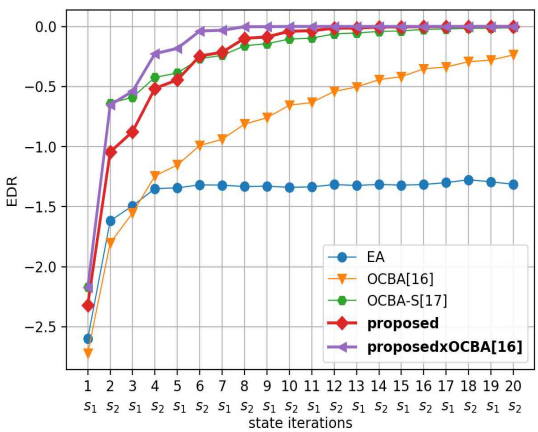

Fig. 4. Evaluated EDR of the improved policy in each state.

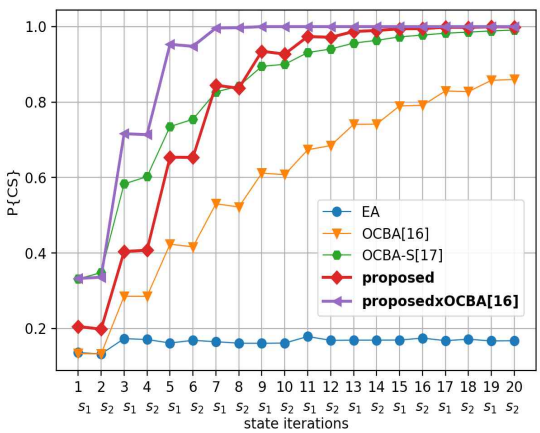

Fig. 5. Estimated $P\{CS\}$ (i.e., probability for correctly selecting the true best action) in each state.

posed×OCBA는 이전 상태의 시뮬레이션 결과를 Fig. 2의 표를 통하여 누적하여 상태 전이 확률에 대한 정밀한 추정을 가능하게 한다. Fig. 5에서 보듯이 순회가 진행되어 데이터가 쌓일수록 식 (15)를 통해 정밀하게 추정된 $\overline{Q}_{\pi}(s,a)$를 바탕으로 선택의 정확도가 높아지는 것을 확인할 수 있다. 이를 바탕으로 다른 방법에 비해서 빠르게 최적의 정책으로 수렴하므로, 상태 순회 횟수를 줄여 보다 더 효율적으로 SBPI을 활용할 수 있도록 한다. 또한 Proposed에 비해서 OCBA를 결합하는 경우 주어진 N을 비효율적으로 각 동작에 동일하게 할당하기보다 Algorithm 1의 OCBA를 활용하여 효율적으로 할당하므로 SBPI의 효율성을 보다 개선할 수 있다.

5. 결 론

본 논문에서는 SBPI의 효율성을 향상시키기 위한 시뮬레이션 샘플 누적 방법을 제시하였다. SBPI의 효율성을 향상시키기 위한 기존의 연구는 OCBA를 적용하여 한 상태에서 최선의 동작을 선택하기 위해 필요한 시뮬레이션 반복 횟수를 감소시키는데 집중하였다. 하지만 제안하는 방법은 이전 상태에서 최선의 동작을 선택하기 위해 수행된 시뮬레이션 샘플을 누적하므로 더 적은 반복 횟수로도 최선의 동작을 정확하게 선택하도록 하여 SBPI의 효율성을 보다 향상시킨다. 제안하는 방법은 시뮬레이션 샘플을 바탕으로 상태와 동작의 조합에 대하여, 그 동작을 그 상태에서 수행하였을 때의 도달하는 상태에 대한 빈도를 수집한다. 그리고 시뮬레이션이 반복됨에 따라 누적된 빈도수를 바탕으로 각 조합에 대한 전이 확률을 추정하고, 추정한 전이 확률을 바탕으로 동작에 대한 EDR의 추정치를 계산한다. 상태를 순회하여 시뮬레이션 샘플이 누적될수록 전이 확률이 보다 정확하게 추정되며, 따라서 EDR의 추정치도 더 정밀한 값을 가지게 된다. 따라서 각 상태에 할당하는 샘플의 개수가 작더라도 순회가 진행될수록 제안하는 방법은 정밀한 EDR의 추정치를 도출하므로 최적의 정책으로 효율적으로 수렴할 수 있도록 한다. 하지만 기존의 방법의 경우 각 상태에 할당하는 샘플로만 EDR의 추정치를 계산하기 때문에, 샘플의 개수가 작아지면 부정확한 추정치로 인해 최선의 동작을 정확하게 선택하지 못한다. 순회가 반복되더라도 각 상태간의 시뮬레이션 결과가 공유되지 않기 때문에 선택의 정확도가 개선되지 않으며, 이는 최적의 정책으로 수렴하지 못하게 한다. 기존 방법과의 비교 실험 결과는 이를 입증한다. 또한 제안하는 방법은 기존의 방법인 OCBA와 함께 적용되었을 때 SBPI의 효율성을 크게 향상시킬 수 있다.

추후 연구로는 본 논문에서 상세히 다루지 않은 시뮬레이션 에폭에 대해서 연구한다. 에폭의 크기가 클수록 무한대의 에폭이 요구되는 EDR을 정확하게 도출할 수 있지만, 한 번의 시뮬레이션에 그만큼 많은 동작을 수행하여 보상을 관찰해야하기 때문에 비효율적일 수 있다. 두 개의 동작이 있을 때, 각 동작에 대해서 작은 수의 에폭으로 계산되는 EDR의 대소관계가 많은 수의 에폭으로 계산되는 EDR의 대소관계와 동일하다면, 굳이 에폭을 크게 설정할 필요가 없다. 즉 적절한 에폭값을 설정하면 제안하는 방법이 적용되었을 때 SBPI의 효율성을 추가로 향상시킬 수 있으며, 본 논문에서는 이를 추후 연구로 남겨둔다.

REFERENCE

[1] A. Munir and A.G. Ross, “An MDP-based Application Oriented Optimal Policy for Wireless Sensor Networks,” *Proceeding of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, pp. 183-192, 2009.

[2] A. Oner, M.S. Albayrak, F. Guner, and I.M. Atakli, “An Activity Recognition Application Based on Markov Decision Process Through Fish Eye Camera,” *Proceeding of 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering,* pp. 251-258, 2019.

[3] Y. Wang, Z. Hu, and Y. Wang, “The Application of Markov Decision Process with Penalty Function in Restaurant Delivery Robot,” *Proceeding of American Institute of Physics Conference,* pp. 1-4, 2017.

[4] M.B. Terefe, H. Lee, N. Heo, G.C. Fox, and S. Oh, “Energy-efficient Multisite Offloading Policy Using Markov Decision Process for Mobile Cloud Computing,” *Pervasive and Mobile Computing,* Vol. 27, pp. 75-89, 2016.

[5] W. Song, K.H. Cho, and K.E. Um, "Multiple Behavior s Learning and Prediction in Unknown Environment," *Journal of Korea Multimedia Society*, Vol. 13, No. 12, pp. 1820-1831, 2010.

[6] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley and Sons, Inc., New York, USA, 2014.

[7] B. Richard, "The Theory of Dynamic Programming," *Proceeding of National Academy of Sciences*, pp. 503-715, 1952.

[8] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction,* MIT Press, Cambridge, USA, 2017.

[9] J. Zhang, C. Wang, D. Zang, and M.C. Zhou, "Incorporation of Optimal Computing Budget Allocation for Ordinal Optimization into Learning Automata," *IEEE Transactions on Automation Science and Engineering,* Vol. 13, No. 2, pp. 1008-1017, 2016.

[10] D.P. Bertsekas and D.A. Castan, "Rollout Algorithms for Stochastic Scheduling Problems," *Journal of Heuristics,* Vol. 5, No. 1, pp. 89-108, 1999.

[11] Q. Huang, Q.S. Jia, Z. Qiu, X. Guan, and G. Deconinck, "Matching EV Charging Load with Uncertain Wind: A Simulation-based Policy Improvement Approach," *IEEE Transactions on Smart Grid*, Vol. 6, No. 3, pp. 1425-1433, 2015.

[12] Y. Zhang and Q.S. Jia, "A Simulation Based Policy Improvement Method for Joint-operation of Building Microgrids with Distributed Solar Power and Battery," *IEEE Transactions on Smart Grid*, Vol. 9, No. 6, pp. 6242-6252, 2018.

[13] Q.S. Jia and Y. Guo, "Event-based Evacuation in Outdoor Environment," *Proceeding of the 24th Chinese Control and Decision Conference*, pp. 23-25, 2012.

[14] S.H. Choi and T.G. Kim, "Efficient Ranking and Selection for Stochastic Simulation Model Based on Hypothesis Test," I*EEE Transactions on Systems, Man, and Cybernetics: Systems,* Vol. 48, No. 9, pp. 1555-1565, 2018.

[15] C.H. Chen and L.H. Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation,* World Scientific, Singapore, 2011.

[16] 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, 2012.

[17] D. Wu, Q.S. Jia, and C.H. Chen, "Sample Path Sharing in Simulation-based Policy Improvement," *Proceeding of the 2014 IEEE International Conference on Robotics and Automation*, pp. 3291-3296, 2014.

황 시 랑

2016년 광동해양대학 촌금학원 전기및전자공학과 학사
2018년 부산대학교 전기전자컴퓨터공학과 석사
2019년~현재 부경대학교 IT융합응용공학과 박사과정

관심분야 : 시스템 모델링 시뮬레이션, 강화학습, 최적화

최 선 한

2012년 한국과학기술원(KAIST) 전기 및 전자공학과 학사
2014년 한국과학기술원(KAIST) 전기 및 전자공학과 석사
2018년 한국과학기술원(KAIST) 전기 및 전자공학부 박사

2018년 한국생산기술연구원(KITECH) 선임연구원
2019년~현재 부경대학교 IT융합응용공학과 조교수
관심분야 : 시스템 모델링 시뮬레이션, 인공지능, 최적화, 데이터 모델링