Reinforcement Learning based Multi-Step Look-Ahead Bayesian Optimization

Mujin Cheon*, Haeun Byun*, Jay H. Lee*

** Department of Chemical and Biomolecular Engineering, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Republic of Korea (e-mail: jayhlee@kaist.ac.kr).*

Abstract: This paper considers the situation where data-based optimization is to be performed but data sampling is limited due to high cost and time. Such situations demand highly efficient data-sampling and utilization and Bayesian optimization (BO) is the most commonly used method as it allows users to balance between exploration and exploitation in deciding where to sample next in the design space. However, the standard acquisition functions used in Bayesian optimization such as the expected improvement have been criticized for being greedy and myopic in many situations. To address the limitation of the standard acquisition functions of BO due to its near-sighted nature, this paper suggests a novel reinforcement learning based method which enables multi-step lookahead Bayesian optimization. Several benchmark functions are tested to compare the performance of the RL based method against the traditional BO methods using expected improvement and its rollout-based extensions. The proposed method outperformed popular Bayesian optimization methods in the case study.

Keywords: Black box optimization, Bayesian optimization, surrogate modeling, acquisition function, reinforcement learning, dynamic programming, sequential decision making

1. INTRODUCTION

Black-box optimization using a data-based model is one of the core tasks in engineering and process control as first principles are unknown or too complex to be used for optimization. Therefore, data driven modeling and optimization has become a popular research theme in many engineering domains. Problems such as materials design, protein folding, and process control are some of the well-known cases where data driven modeling/optimization has been popular or is getting traction. Black-box optimization can be challenged by many issues when the underlying function is non-convex and is expensive to sample (e.g., polymer property, cell function). In such cases, high data-efficiency of the optimization method is critical.

Bayesian optimization (BO) is an optimal sequential decision-making strategy through iteration between search and evaluation. It has recently gained great popularity owing to its high efficiency in terms of data requirements (J. Snoek et al, 2012; D. R. Jones, 2001). The utility of BO has been examined in solving diverse experiment design problems, including those for materials discovery, reaction design, and control (Schmidt J. et al, 2019; Green hill et al, 2020; Shalloo et al, 2020).

The efficiency of BO is brought by controlling the balance between exploration and exploitation through a so-called acquisition function, which is optimized in deciding the next sample to try. In BO, standard acquisition functions (e.g. expected improvement, probability of improvement, and upper confidence bound) are for a single step, which means that they consider just the immediate improvement at the next step, and do not optimize the long-term gain obtained through many rounds of future evaluations. Such limitation of BO has been recognized by several researches in the past (Wu et al, 2019; Lam et al, 2018).

However, few real-world decision-making problems can be solved in just a single step of iteration. Real-world problems often require decision making over multiple iterations of sampling and evaluation starting from the initial knowledge state. In theory, to obtain an optimal solution to the multi-step lookahead BO problem, a stochastic dynamic programming (SDP) problem should be solved, which is computationally intractable in almost all cases. Thus, several approximate methods for solving the multi-step lookahead BO problem have been suggested (Wu et al, 2019, Lam et al, 2016); however, they are either computationally very expensive to implement or restricted to two-step lookahead.

In this work, we propose a reinforcement learning (RL) based BO architecture for multi-step lookahead decision making in an unknown environment. RL is used to approximately solve the stochastic DP problem, in optimal or near-optimal ways in many cases, to enable improved multi-step lookahead decision making. To incorporate RL into the BO, the BO problem has to be translated into a Markov Decision Process (MDP) based on which RL methods can readily be applied. Unlike games or robotics where the power of RL has been successfully demonstrated thus far, proper definitions of the knowledge state in BO are not clear-cut. One contribution of this paper is to suggest a novel way to define an MDP that addresses the multi-step lookahead BO problem. The key idea is to latticize the search space and define the mean and standard deviation at the lattice points as the state of the MDP.

2. BAYESIAN OPTIMIZATION

The ultimate goal of Bayesian optimization (BO) is to solve the following problem:

$$\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x}\in\Omega} f(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ is a d-dimensional vector inside the d-dimensional search space $\boldsymbol{\Omega} \subseteq \mathbb{R}^d$, and $\mathrm{f}: \boldsymbol{\Omega} \to \mathbb{R}$ is the black box function which is "expensive" to evaluate. Therefore, finding $\boldsymbol{x}^*$ which corresponds to a minimum value of $f(\boldsymbol{x})$ should be searched in a data-efficient way, i.e., through fewest iterations possible. To achieve this goal, objective function $f(\boldsymbol{x})$ of BO is often modeled with a Gaussian process (GP) from the collected data set $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^k$. GP prior $f \sim \mathcal{GP}(\mu, K)$ is defined by a mean function $\mu: \Omega \to \mathbb{R}$ and a kernel function $K: \Omega \times \Omega \to \mathbb{R}$. A kernel function, which defines how neighboring points are related with each other (i.e., the smoothness of the GP model), is selected based on available prior knowledge on the system. Thus, the choice of kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ and its hyperparameters should be carefully selected. Most common choices of kernel functions are the radial basis function (RBF) kernel and the Matérn kernel.

The BO algorithm starts with the initial data $\mathcal{D}_1 = (x_1, y_1)$, with $y_i = f(x_i)$. GP prior is constructed based on the initial data. On top of the GP prior, new data is added after each iteration/experiment, and the GP prior is updated using Bayes' rule to obtain the GP posterior distribution. Therefore, when the data at the current time step $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^k$ become available, the posterior mean $\bar{\mu}_k(x)$ and the posterior variance $\bar{\sigma}_k^2(x)$ of the GP are evaluated as:

$$\bar{\mu}_k(x) = K(X_k, x)^\top [K(X_k, X_k) + \lambda I]^{-1} Y_k$$
$$\bar{\sigma}_k^2(x) = \kappa(x, x) - K(X_k, x)^\top [K(X_k, X_k) + \lambda I]^{-1} K(X_k, x).$$

where $K(X_k, X_k)$ is the $k \times k$ matrix whose ij^{th} entry is $\kappa(x_i, x_j)$, $K(X_k, x)$ (respectively Y_k) is the $k \times 1$ vector whose i^{th} entry is $\kappa(x_i, x)$ (respectively y_i), and λ is the noise variance. Overall, the function value at a location x is represented by the normal distribution $\mathcal{N}\left(\bar{\mu}_k(x; \mathcal{D}_k), \bar{\sigma}_k^2(x, x; \mathcal{D}_k)\right)$.

Based on the GP model, the next evaluation point x_{k+1} is selected by maximizing the acquisition function $\Lambda(x \mid \mathcal{D}_k)$: $x_{k+1} = \underset{\Omega}{\text{argmax}} \Lambda(x \mid \mathcal{D}_k)$. The most popular choice for the acquisition function is the expected improvement (EI) (D. R. Jones et al, 1998). For the minimization problem, the EI can be represented mathematically as:

$$u(x) = \max(f^{best} - f(x),\ 0), \tag{2}$$

$$\Lambda_{\text{EI}}(x) = \mathbb{E}[u(x) \mid x, \mathcal{D}_k]. \tag{3}$$

From equations (2) and (3), it is clear that the EI based BO only considers the next time-step's decision (i.e., there is no consideration of $k + 1^{\text{th}}$ or $k + n^{\text{th}}$ time-step's decision at the k^{th} time step's decision). This can lead to suboptimal results as most real-world problems cannot be solved after just one iteration. Generally, iterations would repeat many times, and data acquired at $k + 1^{\text{th}}$ time step (x_{k+1}, y_{k+1}), and so on, would be utilized for subsequent decisions. Therefore, the whole decision-making process that BO is intended to address would be more precisely formulated as a stochastic dynamic programming (DP) problem that considers all future decisions.

3. MULTISTEP LOOKAHEAD BAYESIAN OPTIMIZATION

As mentioned in the previous section, the entire process of BO can be viewed as a multi-stage stochastic dynamic programming (DP) problem cast over the information state. Previous works on using the DP approach for BO (Lam et al, 2016) have formulated the system dynamics on $k\text{-}th$ time step in the following way:

$$\forall k \in \{1, \cdots, N\}, \forall (z_k, u_k, w_k) \in \mathcal{Z}_k \times \mathcal{U}_k \times \mathcal{W}_k,$$
$$z_{k+1} = \mathcal{F}_k(z_k, u_k, w_k), \tag{4}$$

where N refers to the total number of stage, z_k refers to the state $z_k \in \mathcal{Z}_k$, u_k refers to the control which is a function of the state $u_k \in \mathcal{U}_k(z_k)$, and w_k is a random disturbance.

Under such dynamics, the goal is to find a policy $\pi = \{\pi_k, \cdots, \pi_N\}$, which is a function of the state, i.e., $\pi_k: \mathcal{Z}_k \mapsto \mathcal{U}_k$, that maximizes the expected total reward. The reward function can be defined according to the user's preference, but one of the most choices is a measure of the improvement. For the minimization problem, a stage-reward function $R_k: \mathcal{Z}_{k+1} \mapsto \mathbb{R}$, which is a function of the state, control, and disturbance $R_k: \mathcal{Z}_k \times \mathcal{U}_k \times \mathcal{W}_k \mapsto \mathbb{R}$, can be defined as:

$$R_k = \max(y_k^* - y_{k+1}, 0), \tag{5}$$

where y_{k+1} refers to the observed result of $k + 1^{\text{th}}$ time step, and y_k^* refers to the smallest y value in the dataset $\mathcal{D}_k$. Thus, the expected total reward when following policy π from the current time step until the end of the time horizon N could be expressed as:

$$J_\pi(z_k) = \mathbb{E}\left[\sum_{i=k}^{N} R_i(z_i, \pi_i(z_i), w_i)\right], \tag{6}$$

The expectation is taken given the random disturbance term w_i, resulting in a probability distribution of the total reward value. An optimal policy π^* that maximizes the expected total reward could be expressed as:

$$J^*(z_k) = J_{\pi^*}(z_k) = \max_{\pi\in\Pi} J_\pi(z_k), \tag{7}$$

where Π is the set of all feasible control policies. However, solving the above problem through DP to find an optimal policy π^* can quickly become computationally intractable as the state dimension grows. Therefore, ways to solve dynamic programming in an approximate manner have been introduced by researchers in many fields, including machine learning, operations research, and control. Among the suggested

methods, one of the most popular method the rollout method suggested by Lam et al. (2016).

The rollout-based BO method interacts with GP posterior established based on $\mathcal{D}_k$ at time k. It explores all the possible action time k and receives a virtual output sampled from a normal distribution $y_{t+1} \sim \mathcal{N}\left(\bar{\mu}_k(x_{k+1}; \mathcal{D}_k), \bar{\sigma}_k^2(x_{k+1}, x_{k+1}; \mathcal{D}_k)\right)$. When the virtual result y_{t+1} is sampled, it is added to the dataset $\mathcal{D}_k$ and forms the hypothetical $\mathcal{D}_{k+1}$. From $k+1^{\text{th}}$ time step to N^{th} time step, the rollout-based BO method conducts a rollout using the expected improvement (EI) as their base policy. After the roll out, the algorithm finds the action for the current step $\boldsymbol{\pi}^*$ that maximizes the expected total reward of the rollout.

The rollout method is definitely a far-sighted approach compared to the traditional EI based BO method. However, there are still some gaps to be filled. After the 1st decision (assuming that $\mathcal{D}_k$ was initial data-in-hand), the rollout based BO method assumes that the EI based BO is in place for all future decisions, to approximately calculate the value function. Also, they defined the state space as an observation history $\mathcal{Z}_k = (\mathcal{X} \times \mathbb{R})^k$. Therefore, the dimension of the state space keeps increasing as the iteration procedes. To address the suboptimality caused by such a gap, this paper proposes to use a more general approach of reinforcement learning (RL) in addressing the multi-step Bayesian optimization problem.

4. REINFORCEMENT LEARNING BASED BAYESIAN OPTIMIZATION

Reinforcement learning (RL) is a machine learning method that can be used to find optimal or near-optimal solutions of dynamic programming (DP) problems. In RL, the agent interacts with the environment (i.e., at each time, the agent takes a certain action given a state, and receives information on the reward at the next state) to learn the optimal policy $\boldsymbol{\pi}^*$over the state space. This research proposes to use the RL approach as a way to solve the stochastic DP problem associated with multi-step BO.

To train the RL agent, the system should satisfy the Markov property, which in essence stipulates that the state dynamics be memoryless (i.e., $\mathrm{P}[\,future \mid present, past\,] = \mathrm{P}[\,future \mid present\,]$). The state should be defined such that the Markov property holds. For training the RL agent, the problem needs to be described as a Markov decision process (MDP).

An MDP is defined as a tuple: $\{T, \mathbb{S}, \mathbb{A}, P, R\}$ (Puterman, 2014), where $T = \{0,1,\dots,h-1\}$, $h < \infty$ is the set of decision epochs, assumed finite for our problem. State space $\mathbb{S}$ is the set of all states, which should be defined to contain all the information at time $t \in T$ that the agent can use to make a proper decision after observing it. Action space $\mathbb{A}$ is the set of actions (for clarity of exposition, the 1st decision of the agent is noted as a_0). $P : \mathbb{S} \times \mathbb{A} \mapsto (\mathbb{S} \mapsto R)$ is the state transition probability. When action $a \in \mathbb{A}$ is taken at state $s \in \mathbb{S}$, $P(s' \mid s, a)$ is the probability of the next state being s'. Last but not least, $R : \mathbb{S} \times \mathbb{A} \mapsto \mathbb{R}$ is the reward function. When action $a \in \mathbb{A}$ is taken at state $s \in \mathbb{S}$, the reward $R(s, a, s')$ is given to the agent and the state is changed to $s' \in \mathbb{S}$,

For the RL based BO approach, it is obvious how to define four out of the five components of the tuple, $\{T, \mathbb{A}, P, R\}$. T, the set of decision epochs can be considered user-chosen parameter (e.g. when a user wants to make 3-step look-ahead RL based BO, T would be defined as $T = \{0,1,2\}$). Also, $\mathbb{A}$ would be defined as the search space for the decision, P would be defined by the GP model, and R can be defined according to the user's preference, just as in the rollout based BO approach. However, how to define the state space $\mathbb{S}$ is not obvious as it should summarize all the information gathered up to a time point. One can always choose the data-observation-history as the state but such defined state would cause its dimension to change after each iteration. Therefore, this paper suggests the GP means and variances over the latticized input space as the components of the state space $\mathbb{S}$. The role of the RL agent is to interact with the environment given by the defined MDP to find the optimal policy π^*(Figure 1).

A decision policy, $\pi_t : \mathbb{S} \to \mathbb{A}$, takes the state as input and returns an action as output at time t. A policy $\pi = (\pi_0, \pi_1, \dots, \pi_{h-1})$ is implemented to the MDP System at each time step. Under the policy π, a starting state s_0, and a look-ahead horizon h, the expected total reward $V_h^\pi(s_0)$ can be defined as:

$$V_h^\pi(s_0) = \mathbb{E}\left[\sum_{t=0}^{h-1} R(s_t, \pi_t(s_t), s_{t+1})\right], \tag{8}$$

which is fundamentally the same expression as equation (7) when $N - k = h$. As the BO process has been expressed as an MDP, our goal is to find the optimal policy π^* that maximizes the expected total reward.

Speaking of $\{P\}$, when RL is applied in games or robotics, the agent interacts and learns from the real or simulated

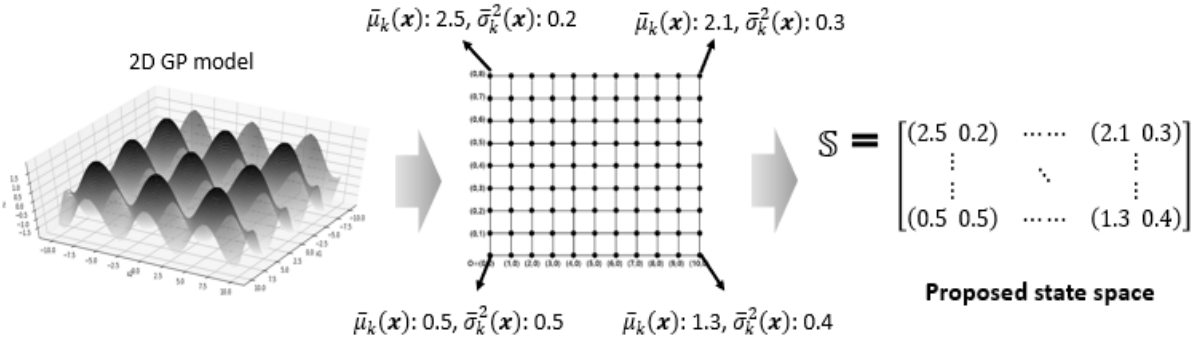

Figure 1. Proposed state space $\mathbb{S}$ of MDP tuple for RL based BO

environment to iteratively update their policy π . However, in the case of the RL for BO, which is needed when the action is expensive and time-consuming, one does not have the luxury to interact with the real environment while training. Therefore, the agent interacts with a simulated environment. At time *t*, the Gaussian Process model created from the initial data $\mathcal{D}_t$ with the mean $\mu^{(t)}$ and kernel $K^{(t)}$ acts as the simulated environment in this RL training. When the agent makes an action $\mathbf{x}_{t+1}$, the simulated data y_{t+1} is created by the normal distribution following:

$$y_{t+1} \sim \mathcal{N}\left(\mu^{(t)}(\mathbf{x}_{t+1}; \mathcal{D}_t), K^{(t)}(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}; \mathcal{D}_t)\right) \tag{9}$$

Such virtually generated data would be added to the dataset to form a new data set augmented with the virtual data (we use the notation $\widehat{\mathcal{D}}_{t+1}$ to denote this in order to distinghish it from the set with real data only) and the new GP based on updated dataset is drawn. So, based on $\mathcal{D}_t$, the RL agent would freely conduct a large number of virtual experiments and add the virtual data on its database to form $\widehat{\mathcal{D}}_{t+1}, \widehat{\mathcal{D}}_{t+2}$, etc. until the end of the user-defined decision epoch is reached. While doing so, the agent would receive rewards for its own education. The reward for the RL agent can be calculated in the same way as in the rollout based BO, expressed as:

$$\begin{aligned} R(\mathcal{D}_t, \mathbf{x}_{t+1}, \mathcal{D}_{t+1}) &= (y_t^* - y_{t+1})^+ \\ &\equiv \max(y_t^* - y_{t+1}, 0). \end{aligned} \tag{10}$$

The ultimate goal for the RL agent to achieve is to maximize the expected reward through the entire episode. Thus, when the agent is looking ahead h-step starting with dataset $\mathcal{D}_t$ at time t, the expected total reward can be expressed as:

$$\begin{aligned} V_h^{\pi}(\mathcal{D}_t) &= \mathbb{E}\left[\sum_{k=t}^{t+h-1} R\left(\widehat{\mathcal{D}}_k, \pi_k\left(\widehat{\mathcal{D}}_k\right), \widehat{\mathcal{D}}_{k+1}\right)\right] \\ &= \mathbb{E}\left[\sum_{t=k}^{k+h-1} (y_k^* - y_{k+1})^+\right]. \end{aligned} \tag{11}$$

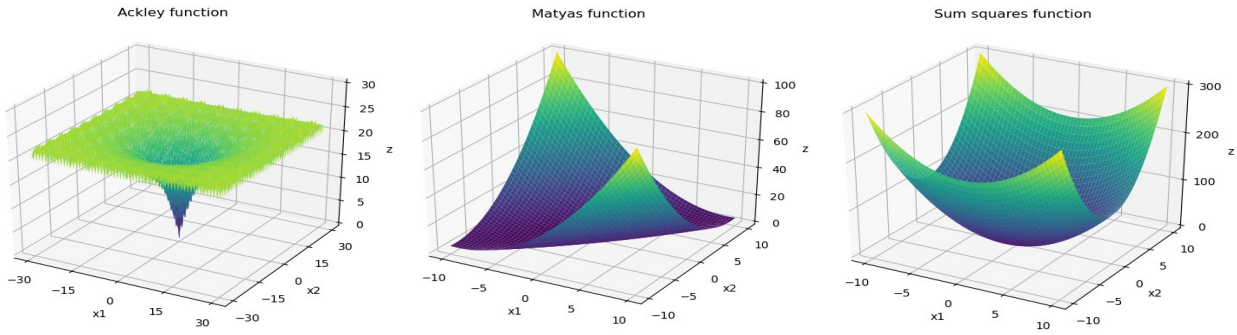

Figure 2. Graphical illustration of Ackley function (left), Matyas function (middle), and Sum squares function (right)

For the training of the RL agent, the proximal policy optimization (PPO) algorithm is employed in this work to find π^* under a given MDP. The PPO algorithm is simple to use and has shown exceptional computational efficiency in many case problems (Schulman J. et al, 2017). All in all, the proposed RL based BO approach works in the following way:

Step 0. At time step *t*, set *k*=*t* and $\widehat{\mathcal{D}}_k = \mathcal{D}_t$

Step 1-1. Based on given $\widehat{\mathcal{D}}_k$, a GP model is constructed.

Step 1-2. The RL agent makes a virtual action $\hat{x}_{k+1}$ based on the observation of GP model from step1-1 combined with its policy π. As a consequence, virtual data $\hat{y}_{k+1}$ is created, $\widehat{\mathcal{D}}_k$ is updated to $\widehat{\mathcal{D}}_{k+1}$, and the agent receives reward $\hat{R}_k$.

Step 1-3. Based on $\widehat{\mathcal{D}}_{k+1}$, a new GP model is constructed and the same process as step 1-1, 1-2 happens for h steps $(\widehat{\mathcal{D}}_{k+1}, \widehat{\mathcal{D}}_{k+2}, \dots, \widehat{\mathcal{D}}_{k+h})$. When the virtual h-step experiments are over (i.e., one episode is over), *histories* of states and rewards are saved.

Step 2. step 1-1 ~ 1-3 repeats for user-defined number of times (i.e., user-defined number of virtual episodes are conducted) and *histories* of episodes are saved. RL agent updates its policy π based on gathered *histories*.

Step 3. step 1~2 is repeated until the policy π reaches stopping criterion. When the updating of policy stops, we obtain the optimal policy π^*.

Step 4. Real experiment is conducted by action *a* suggested by π_t^*, the resulting optimal policy at the time step t. As consequence, a new real data point is sampled.

Step 5. Based on the new real data, dataset $\mathcal{D}_t$ is updated to $\mathcal{D}_{t+1}$. Set *t*=*t*+*1*. Go back to Step 0.

5. CASE STUDY

Data sampling efficiency of the RL based BO method is compared with the rollout -based BO method and the EI-based BO method under three different categories of benchmark functions. EI-based BO is known for being more explorative than PI (probability of improvement)-based and UCB (upper confidence bound)-based (Berk et al, 2018; De Ath et al, 2021). Therefore, EI based BO was selected as a comparison basis. Also, rewards for rollout-based BO and RL based BO were set as Equation (5), (10) for the same reason. Thus, all three method aims to maximize expected improvement in their own way.

The three selected benchmark functions are as follows (Figure 2):

$$f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}\right) - \exp\left(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)\right) + 20 + \exp(1) \quad (12)$$

$$f(\mathbf{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \quad (13)$$

$$f(\mathbf{x}) = \sum_{i=1}^{2} ix_i^2 \quad (14)$$

Equation (12) is the Ackley function which is known to have multiple local minima. It was evaluated on the 2D plane $x_i \in [-32.768, 32.768]$, for $i = 1, 2$. Equation (13) is the Matyas function which is known to be plate-shaped. It was evaluated on the 2D plane $x_i \in [-10, 10]$, for $i = 1, 2$. Equation (14) is the Sum Squares function which is known to be a bowl-shaped function. It was evaluated on the 2D plane $x_i \in [-5.12, 5.12]$, for $i = 1, 2$ (Surjanovic, S. et al, 2013).

For the GP, the RBF kernel was used and its hyper parameters were estimated by the maximum likelihood estimation method at each time step of iteration (Williams C. K. 2006). For the RL based BO, the GP hyper parameters were kept the same during the virtual lookahead training, and were updated only when the real data was observed.

To compare the data efficiency of each BO algorithm with the three different benchmark functions on a fair basis, random initial data (data size = 50) selected from each benchmark function's input space were given to each BO algorithm. Based on the initial data, BO was conducted with the three different BO algorithm without sharing information about newly acquired data during the BO process for 20 time-steps. For each time step t of iteration, regret ($= y_{opt} - y_t^*$), was recorded for the data efficiency index. Regret indicates the difference between the global optimum value and the best data-in-hand at the t-th iteration point. However, random initial data given to the BO algorithms could be locally biased to a certain area of the search space. Locally biased input data could affect the data efficiency of the BO algorithms. To address this issue, 10 different sets of initial data which contain 20 input-output relationships were given to each BO algorithm. Regret values recorded throughout the BO process for each initial dataset were averaged to measure the general data efficiency of each BO algorithm.

As a result, the proposed RL-based BO algorithm has shown lower average regret values in each iteration compared to the conventional EI based BO and the rollout-based BO at most of the iteration steps. This means that the RL-based BO has found a better optimum faster than the conventional BO and the rollout-based BO. The data efficiency was significantly better for the RL based BO algorithm compared to other BO algorithms for the Ackley function and the Sum Squares function. However, for the Matyas function, the data efficiency of the RL-based BO was not significantly better compared to the other BO algorithms. This is because the Maytas function is a plate-shaped benchmark function; due to the flatness of the function, there is not much room to optimize. Figures 3, 4, 5 show the average regret value of each iteration step for each benchmark function.

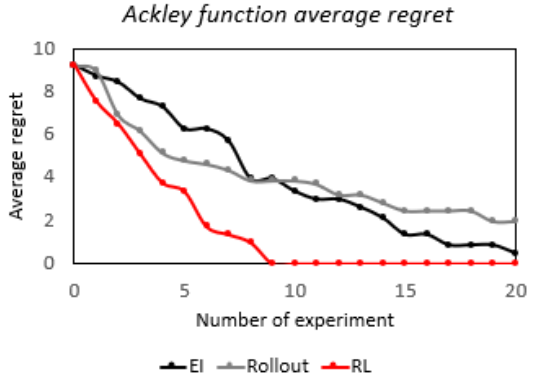

Figure 3. Average regret of each BO algorithm on each time step for the Ackley function

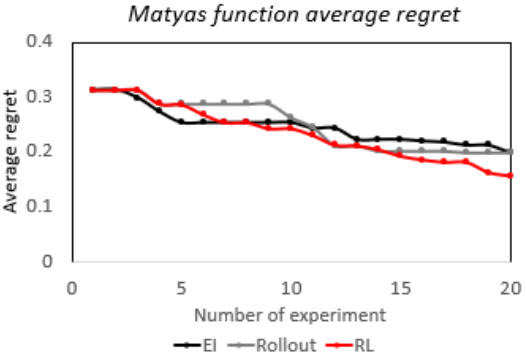

Figure 4. Average regret of each BO algorithm on each time step for the Matyas function

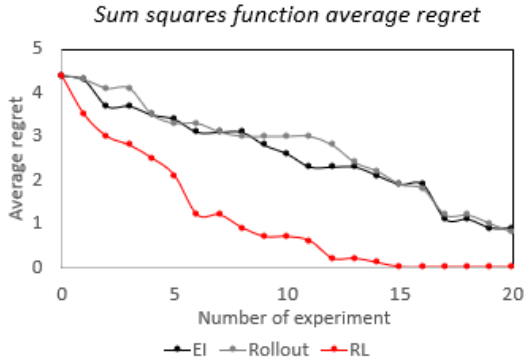

Figure 5. Average regret of each BO algorithm on each time step for the Sum Squares function

6. CONCLUSION AND FUTURE WORKS

This work suggested a novel reinforcement learning based Bayesian optimization method to solve the multi-step BO problem. To apply the RL approach, the multi-step BO problem was formulated as an MDP with a newly defined state space $\mathbb{S}$ base on the GP mean and variance over the latticized input space. The proposed BO method has been empirically

shown to give higher data efficiency based on testing with several types of benchmark functions and this can be attributed to its ability to make far-sighted sequential decisions by solving *h*-step look-ahead stochastic dynamic programming in a near optimal way. The suggested BO method can be applied to a variety of sequential decision-making problems cast in an unknown environment to accelerate the finding of the global optimum or an improved optimal point. For future work, we believe the RL based BO method should be compared with other existing BO methods such as PI- and UCB-based BO methods for various types of systems. Also, besides data efficiency, properties like stability and scalability need to be studied as training a RL agent can be very sensitive and time-consuming depending on the problem.

7. ACKNOWLEDGEMENT

This research was supported by the National Research Foundation (NRF) grant funded by the Korea government(MSIT) (No. 2021R1A2C200608311)

REFERENCES

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, *25*.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, *21*(4), 345-383.

Schmidt, J., Marques, M. R., Botti, S., & Marques, M. A. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, *5*(1), 1-36.

Greenhill, S., Rana, S., Gupta, S., Vellanki, P., & Venkatesh, S. (2020). Bayesian optimization for adaptive experimental design: A review. *IEEE access*, *8*, 13937-13948.

Shalloo, R. J., Dann, S. J. D., Gruse, J. N., Underwood, C. I. D., Antoine, A. F., Arran, C., ... & Streeter, M. J. V. (2020). Automation and control of laser wakefield accelerators using Bayesian optimization. *Nature communications*, *11*(1), 1-8.

Wu, J., & Frazier, P. (2019). Practical two-step lookahead Bayesian optimization. *Advances in neural information processing systems*, *32*.

Lam, R., Poloczek, M., Frazier, P., & Willcox, K. E. (2018). Advances in bayesian optimization with applications in aerospace engineering. In *2018 AIAA Non-Deterministic Approaches Conference* (p. 1656).

Lam, R., Willcox, K., & Wolpert, D. H. (2016). Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, *29*.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455-492.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Berk, J., Nguyen, V., Gupta, S., Rana, S., & Venkatesh, S. (2018, September). Exploration enhanced expected improvement for bayesian optimization. In *joint european conference on machine learning and knowledge discovery in databases* (pp. 621-637). Springer, Cham.

De Ath, G., Everson, R. M., Rahat, A. A., & Fieldsend, J. E. (2021). Greed is good: Exploration and exploitation trade-offs in Bayesian optimisation. *ACM Transactions on Evolutionary Learning and Optimization*, *1*(1), 1-22.

Surjanovic, S. & Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. Retrieved November 22, 2021, from http://www.sfu.ca/~ssurjano.

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning* (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.