Replacing The Acquisition Function in Bayesian Optimization by a Neural Network

How effectively do meta-learned acquisition functions in Bayesian optimization perform when optimizing for control variates of unknown functions, as compared to BO with standard acquisition functions

Shayan Ramezani[1]

Supervisors: Matthijs Spaan[1]**, Joery de Vries**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Shayan Ramezani
Final project course: CSE3000 Research Project
Thesis committee: Matthijs Spaan, Joery de vries, Christoph Lofi

Abstract

Bayesian Optimization (BO) has demonstrated significant utility across numerous applications. However, due to it being designed as a universal optimizer, its performance can often be suboptimal in specialized environments. To overcome this issue, research has been conducted into the application of transfer learning for enhancing BO performance in these specialized contexts. This paper describes the research done into evaluating the MetaBO algorithm in some specific environments. MetaBO innovates by substituting the acquisition function component in BO with a neural network that serves as an acquisition function, trained via a reinforcement learning framework. Although the results indicate that the algorithm's performance is not optimal in the environments tested, these limitations are ascribed to elements of the implementation rather than the concept of the algorithm itself. Consequently, further research is necessary to refine the implementation process and fully exploit the potential of the MetaBO algorithm.

1 Introduction

Numerous real-world problems involve optimizing control variates of complex systems without a solid understanding of the underlying dynamics. Such complex systems can be conceived as encompassing an objective function subject to a limited budget, whether it is in terms of time or money, for function evaluation. The array of problems[1], with significant real-world implications, vary from drug design [2, 3] to machine design [4, 5], from robotics [6] to automatic machine learning and hyper-parameter optimization [7, 8], and beyond.

The research community has made substantial progress in applying Bayesian optimization (BO) methods [1, 9–15] to optimize such systems. This optimization process frequently relies on numerical, derivative free, optimization methods also known as black-box optimization. BO utilizes a surrogate model, typically a Gaussian process (GP), to generalize over information from individual data points by incorporating a measure of uncertainty and uses this surrogate to find the next data point to evaluate by optimizing "an acquisition function (AF) that balances exploration and exploitation" [1].

Due to being designed as universal optimizers, the performance of current Bayesian optimizers are suboptimal, e.g. have slow convergence issues, in specialized scenarios [1, 13], and thus research effort has been directed towards reducing the emphasis on universality by incorporating transfer learning into BO for improved performance. In numerous practical applications, optimization is often repeated under similar settings, which means information garnered from optimizing previous tasks could prove beneficial for optimizing an instance of a comparable task. In such cases, transfer learning can help to leverage knowledge from previous tasks; and even in the face of drastic changes in the settings to optimize in, there might still be some correlation between past tasks and the current task at hand [1].

This paper explores the potential of using reinforcement learning to replace standard AFs in BO by a neural network that functions as the AF and uses the algorithm proposed by Volpp et al.[13], called MetaBO, as inspiration and the main source.

The primary goal of this research is to address the following question: *How effectively do meta-learned acquisition functions in Bayesian optimization perform when optimizing for control variates of unknown functions, as compared to BO with standard acquisition functions?*

The major contributions of this paper are centered around answering this question through the testing of the MetaBO algorithm on different objective functions, as provided by the BBOx library[2].

The structure of the paper is as follows: it begins with a discussion of some related work. followed by the presentation of some essential background knowledge to aid reader comprehension. The implementation steps of the algorithm are detailed in section 4 and this implementation is put to the test in section 5. Section 6 offers a brief digression into responsible research practices and how they have been applied in the context of this project. The paper then concludes with the conclusion and suggestions for potential future works.

2 Related Work

Transfer learning can be incorporated into the different components of BO, which are the surrogate component, acquisition function component, initialization component and search space design component [1]. More specifically for the AF component, transfer learning can be applied by using a multi-task BO function, ensemble GPs-based function, or a reinforcement learning based function [1].

Currently, various methods have been developed to enhance the surrogate model through transfer learning. One of the common methods involves utilizing transfer learning to improve the surrogate component in situations where data from previous optimization runs is used to warm-start a new run. Feurer et al. [16] developed an RPGE ensemble model for BO that can accomplish this, "while avoiding the poor scaling that is associated with incorporating all the results into a single GP model". Another solution for the same problem is offered by Golovin et al. [17] where the idea is to construct a stack of GP models, where each model is associated with an optimization problem it was used in, and these optimizations are done in a linear fashion where the model from one run is trained on the residuals relative to the model before it. In addition, there are numerous other examples of employing transfer learning to improve the surrogate component, e.g. [18–20].

As may be clear from the introduction, applying transfer learning to improve the AF component of BO is more important to this research. In the literature, one can find different ways of achieving this. Wistuba et al. [21] proposes the idea

[1] For a thorough discussion see [1]

[2] This library is not yet publicly available but will be in the future: https://github.com/joeryjoery/BBOx

to use a common AF to do the transfer learning. Specifically, the AF can be replaced by the weighted average of the expected improvement[3] on the data from the current problem and the predicted improvement on all the data from the previous problems, to learn a mapping between the data and the improvement.

As pointed out in the previous section, the algorithm proposed by Volpp et al. [13] is the main inspiration for this paper and proposes another way of applying transfer learning to replace AFs. The contributions of that paper to the fields of BO are[13]:

- Transfer learning structural knowledge about related objective functions by replacing the AF of BO with a learned neural AF with the goal of increasing data-efficiencies on new comparable tasks;
- Meta-learning in such a way that the procedure is fully compatible with the black-box optimization setting, i.e., not requiring objective function gradients
- Demonstration of the efficiency and practical application of MetaBO

An important part for the transfer-learning in this case is the reinforcement learning algorithm which is used to train the neural network replacing the AF. More specifically, a type of Proximal Policy Optimization algorithm is used for reinforcement learning, which was first proposed by Schulman et al. [22]; this, in theory, has the benefits of trust region policy optimization[4] while being "simpler to implement, more general, and having better sample complexity (empirically)".

In addition, there is also research that has been done into the initialization component, e.g. [23–25], and into the search space design component, e.g. [26, 27].

For a more thorough discussion of works on transfer learning, refer to the work by Bai et al. [1].

3 Preliminaries

Bayesian Optimization

As mentioned in the introduction, Bayesian optimization is useful for optimizing functions that are complex to evaluate, typically due to time or cost of evaluation. Its utility has been demonstrated in continuous domains of less than ± 20 dimensions and it tolerates stochastic noise when evaluating functions [28]. The four components of BO are the search space design component, the initialization component, the surrogate model component, and the acquisition function component.

Bayesian optimization starts with first initializing its surrogate model, typically a Gaussian process. This model is intended to statistically model the objective function. To initialize it, often random points, distributed uniformly in the search space, are evaluated for the objective function; keeping in mind that this will cost part of the evaluation budget. The GP model is then fitted on these evaluated points.

[3]The expected improvement indicates what the expected magnitude of improvement of evaluating for a next data point compared to the best data point so far.

[4]https://arxiv.org/abs/1502.05477

To optimize for the objective function, the acquisition function is updated with the latest GP model by using the mean and the standard deviation, at different points in the domain, that can be obtained from the GP model. Afterwards, the maximum of the AF indicates what point to evaluate next for the objective function. A snapshot of a BO process is given in Figure 1.

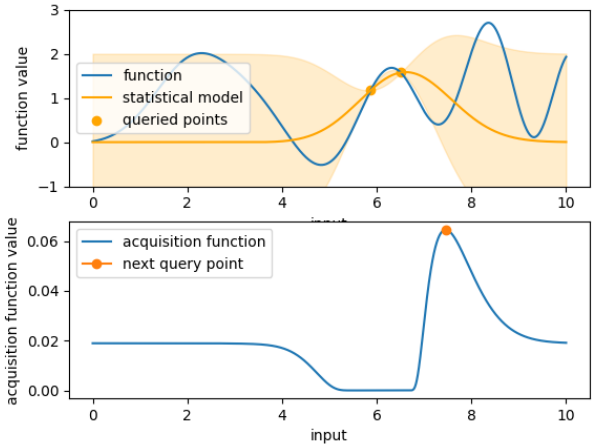

Figure 1: Snapshot of the BO process: The upper plot is showing the objective function, the GP model with 2 times the standard deviation around it and the points the GP is fitted on. The lower plot shows the acquisition functions that needs to be maximized, so the next point of evaluation is around 7.5 as marked.

Reinforcement Learning

Reinforcement learning is designed to teach an agent appropriate behaviour within a specific environment by letting the agent take actions and rewarding/punishing the agent accordingly. The environment can be formally defined as a 5-tuple Markov Decision Process [29], $\langle S, A, R, P, \rho_0 \rangle$, where:

- S is the set of all valid states;
- A is the set of all valid actions, which in this study is related to the input of the objective function;
- $R : S \times A \times S \to \mathbb{R}$ is the reward function, with $r_t = R(s_t, a_t, s_{t+1})$;
- $P : S \times A \to \mathcal{P}(S)$ is the transition probability function, with $P(s'|s, a)$ being the probability of transitioning into state s' if you start in state s and take action a;
- ρ_0 us the starting state distribution.

A critical aspect in this context is the policy, which is a part of the agent that stipulates the rule the agent adheres to when deciding what actions to take. Here, a balance between exploration and exploitation is necessary, where exploration is meant to prevent the policy from getting stuck in local maxima and exploitation is used to exploit what the system has learned.

In this paper, PPO is the policy optimization method used within the RL algorithm, where the key idea is to increase the probability of selecting actions that give a higher return and to decrease the probability of choosing actions that lead to lower returns [29]. PPO aims to identify the most significant improvement step for the policy using the collected data, while at the same time imposing a limit on the improvement step to maintain training stability.

4 Implementation

The algorithm implemented in this paper comprises three main parts:

- The agent that should learn to replace the AF component of the BO, called the neural AF as it uses neural networks internally;
- The environment, constructed around the objective function, within which the agent trains and evaluates;
- The training/evaluation module which uses the other two components to conduct the training/evaluation

This section will go into the details of the implementation and the training, which is heavily inspired by the MetaBO algorithm [13], while the subsequent section will delve into the conducted evaluations.

Agent

The agent is meant to be used in a Proximal Policy Optimization framework, drawing inspiration from the actor-critic architecture, where the actor controls how the agent behaves and the critic estimates the expected cumulative reward. The actor is subdivided into two components: the first being a network that takes as inputs a potential action (x), the mean ($\mu(x)$) and standard deviation ($\sigma(x)$) corresponding to that action as inferred from the GP, the current step (t), and the budget (T) [5].The network outputs a value, intended to be larger for more promising actions. This can be represented as:

$$\alpha_t(x) = \alpha_t(\mu(x), \sigma(x), x, t, T) \tag{1}$$

The second component, the action selector, interprets outputs of the network, which simulates Equation 1, at different action points as the log probabilities of a categorical distribution and outputs the next action that should be taken. This forms the agent's policy and should lead to the meta-algorithm updating its actor network such as to output higher values for more promising actions.

The critic, consisting solely of a neural network, aims to learn a value function to predict the expected cumulative reward from a state. Volpp et al. [13] suggest using the current step and budget as inputs for the critic network, arguing these to be reliable indicators of future regrets. Despite any reservations on its logical correctness, their method, as validated on multiple objective functions by them, serves as a strong starting point.

[5]Together these five values for all possible actions form the true state of the environment, which is explained in the Environment subsection

Environment

The environment, where the agent takes incremental steps and adjusts itself based on received rewards, first and foremost encapsulates the objective function. It also estimates the optimum of its contained objective function, enabling reward computation during training. The reward is formulated as the negative simple regret, as used by Volpp et al.[13].

In addition, the environment houses the budget, current step, action space, observation space, the GP model that it fits on evaluated points, and keeps a record of the actions taken and the best reward achieved. The state of the environment is seen as the composite of the updated GP model, the step and the budget. Although the step and budget are easily used as input for the neural AF, the GP model needs to be approximated by discretizing the continuous action space.

For this, looking back at what the agent does is helpful. For the actor part of the agent to be able to select the optimal action at step t, it needs to have the outputs of its network for the state (which consists out of infinitely many actions), this way enabling the action selector to make the right choice. Volpp et al.[13] suggests one way of evaluating the actor network at step t, which is to start with a coarse static Sobol grid of actions, which produces a discretized representation of the state, ζ_{global}. After evaluating the actor network for this set, local Sobol grids around the k actions corresponding to the highest outputs of the network can be created, giving $\zeta_{local,t}$, which further refines the state approximation. The static grid is useful for exploration while the adaptive grid encourages exploitation. The union of these sets, ζ_t, represents a discretized version of the state that in reality contains a continuous action space.

The GP model connects the action space to the state, allowing for the state of the environment to be defined first by evaluating the GP model for a static discrete set of actions, and then expanding this set to encompass the top k most promising actions.

The state hence becomes a set of actions, their corresponding mean and standard deviation (from the GP model), the current step, and the budget. Notably, the network part of the actor is also a part of the environment as it is used to select the k most promising actions for the local grids.

Training

The training part can be divided into making observations in the environment and taking actions, and updating the policy for this gathered data. At the start, an environment with a random objective function belonging to a specific group is selected. Afterwards, a set of states, actions taken in each state and rewards for each action are obtained by iteratively feeding ζ_t into the agent's policy to select an action, take a step in the environment with that action, and subsequently update the GP model and hence ζ_t. Here, the rewards granted after each step in the environment are recorded for policy updates. This data collection continues until some batch size is reached. To complete the batch, depending on whether the environment is to be reused or not, it may be reset after reaching the budget, or a new environment with a different random function from the same group is created.

Once a batch of data has been accumulated, the policy is updated according to the PPO algorithm [6]. This process continues until a pre-determined number of steps has been executed.

Worth mentioning is the fact that before arriving at the implementation described here, considerable amount of time was spent exploring alternative implementations, as outlined in the following subsection.

Alternate implementation

There have been different attempts of implementing the algorithm to have the agent be able to learn well. One approach mirrored the main features of the implementation outlined in the previous subsection but altered the distribution of tasks between the agent and the environment. In addition, a slightly different PPO implementation was contemplated.

After no satisfactory results, the original MetaBO implementation by Volpp et al. [13] was considered. This required introducing the objective function as an environment[7]. Afterwards, the training algorithm was run on a constant budget but the learning showed no improvement in the rewards, see Figure 2. Importantly, it has not been possible to utilize the original implementation with a fluctuating budget upon function reset, a feature crucial for some flexibility in the learned knowledge.

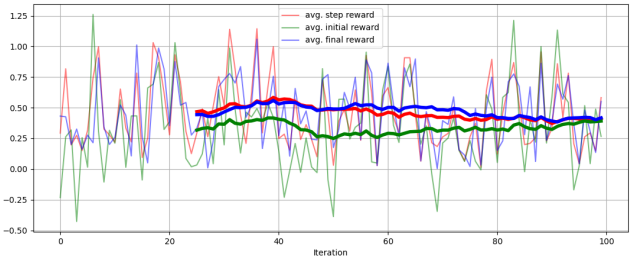

Figure 2: The evolution of the rewards when training with the original implementation of MetaBO by Volpp et al. [13]

.

Training performance

Before diving into the evaluations performed, it is informative to examine a representative instance of the agent's learning performance during training. Figure 3 and Figure 4 demonstrate the evolution of important loss functions during the training. Based on these, it can be inferred that the agent is not learning to approximate a desired AF, as the losses do not appear to converge at all.

Seeing the learning capability of the agent in this implementation and alternative implementations has led to the conclusion that the correct implementation of the MetaBO algorithm is challenging, especially when one has limited knowledge of RL. It also underscores the complexity involved in implementing MetaBO. Despite the availability of numerous plug-and-play reinforcement learning algorithms requiring only the design of an environment, these algorithms appear to underperform due to the intricate nature of the MetaBO environment.

[6]For detailed discussion of how PPO is used in RL and thus in this paper, see the original paper [22].

[7]To be specific, the objective function was added as one of the hard-coded functions in the MetaBO class.

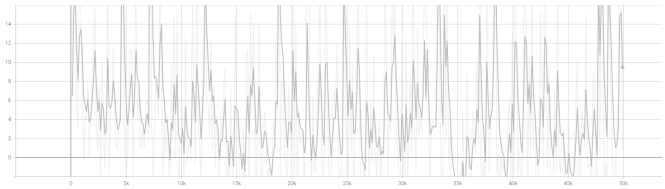

Figure 3: Policy loss during one of the training runs with 50000 timestep

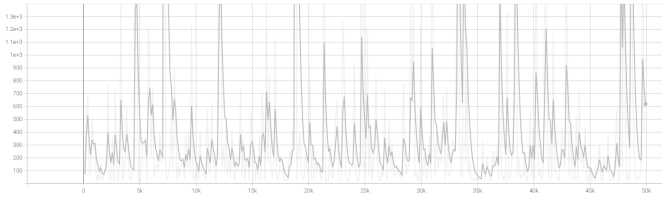

Figure 4: Value loss during one of the training runs with 50000 timestep

5 Evaluation

The primary objective of the evaluation process was to measure the efficacy of the implemented algorithm, that is, how effectively the agent had learned an acquisition function. This was achieved by running Bayesian Optimization (BO) with different acquisition functions (AFs). The first was the trained agent itself, serving as the AF. Additionally, the well-known AFs Probability of Improvement (POI), Expected Improvement (EI), and Upper Confidence Bound (UCB) were employed. In order to establish a benchmark, a random action selector was also utilized as an AF. The subsequent subsection elucidates the specific setup of the evaluation algorithm.

Setup

In order to deploy the agent as an alternative for the AF, a similar environment was set up as the one used during the training process, containing an objective function from the group it was trained for. For the evaluation in this paper, two different groups were used. The first group consisted of a simple convex function, while the second was a more complex Gaussian process function.

Once the environment and the objective function within it were established, the agent was iteratively tasked with proposing an action to be executed. Following this, the environment was updated with the action, and the agent was updated with the new state of the environment. This process was repeated until the budget was exhausted.

Moreover, the objective function of the environment was incorporated into the other BO algorithms[8] with the different AFs mentioned, as well as the random action selector. The

[8]The BO implementation by Nogueira was employed in this paper [30]

BO algorithm was subsequently run until the budget was depleted.

Results

As suggested in the preceding subsection, two distinct groups of objective functions were employed to evaluate the BO algorithms. The performance of the different AFs was measured using the metric of simple regret, essentially reflecting how close the algorithm was able to get to the optimum value of the objective function.

Due to randomness of budget and noise when generating the objective functions, the evaluation was executed multiple times. However, due to suboptimal performance, a more comprehensive evaluation was deemed unnecessary and deferred to a point in time when a more effective implementation has been accomplished.

Figure 5 depicts a few runs of the evaluation when assessing the algorithms for the simple convex functions, in which we can see the performance of the MetaBO algorithm to be comparable to the other ones. Figure 6 shows a few runs when applying them for the Gaussian process function, which shows the lack of performance of the MetaBO algorithm.

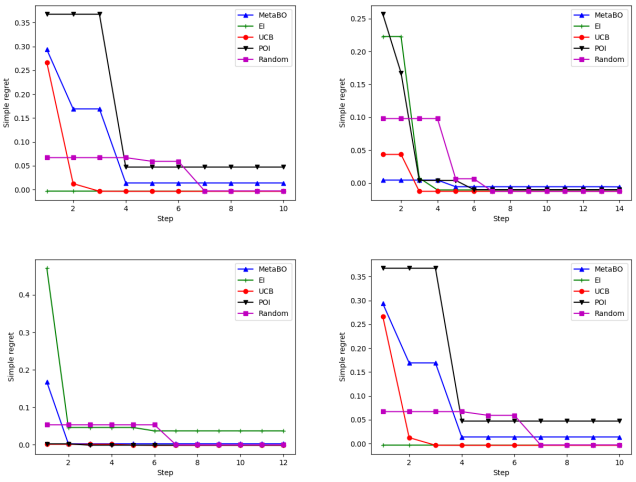

Figure 5: The performance of different AFs when optimizing for a simple convex objective function with domain $[-1, 1]$. These results suggest that the performance of the MetaBO algorithm is akin to standard BO and the random action selector (which although taking more steps, still manage to make other algorithms appear weaker). Note: the negative simple regret is due to the optimum value of the objective function being an approximation. Thus, the smaller the value of the simple regret, the better.

Based on these figures, it can be inferred that the MetaBO algorithm does not surpass the performance of standard BO under any circumstances. This can primarily be attributed to the training process failing to demonstrate any learning capability, as detailed in the subsection on training performance in the preceding section.

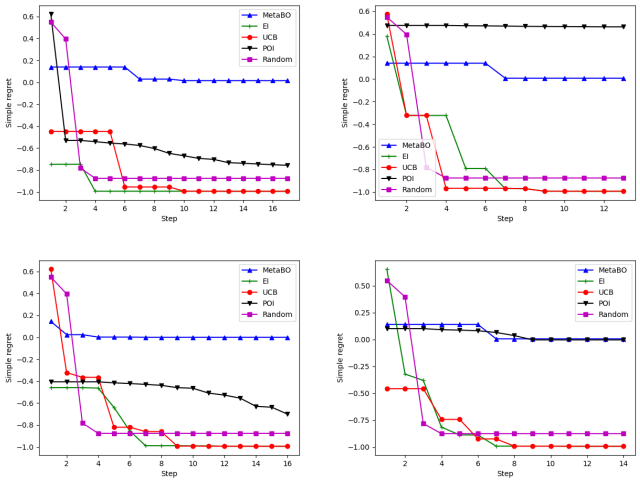

Figure 6: The performance of different AFs when optimizing for a simple convex objective function with domain $[-1, 1]$. These results highlight the poor performance of the MetaBO algorithm in comparison to standard BO and the random action selector. Note: the negative simple regret is due to the optimum value of the objective function being an approximation. Thus, the smaller the value of the simple regret, the better.

6 Responsible Research

Research without integrity is of no value. Therefore, The Netherlands Code of Conduct for Research Integrity report [31] defines principles that can be the basis of integrity in research. These principles are honesty, scrupulousness, transparency, independence, and responsibility. These principles are important to this paper in the following ways:

Honesty This paper explicitly mentions the fact that the writer is not fully confident about the result. section 5 shortly discusses why the results are not acceptable and what the reasons are.

Scrupulousness This principle has been the hardest to fully comply to. The main reasons were the lack of knowledge of reinforcement learning combined with underestimation of how much knowledge was required plus the fact that the project had only a duration of ten weeks. So, designing and implementing the algorithm has not been done as effectively as possible. However, the reporting of the research has been done in a thorough manner as to help clarify mistakes.

Transparency This paper describes the exact implementation of the final algorithm and describes the evaluations done step by step as to help reproducing the research later on. In addition, the code can be shared with parties interested in improving directly on the implementation already available instead of starting from nothing.

Independence The research done for this paper is fully independent of any involved party's interest, which is actually only the researcher himself, the responsible professor, and the supervisor. This means that the paper states all the observation as they were and these can be verified by replication.

Responsibility The responsibility in this paper is mainly present in connection to the fact that a research has been conducted that is scientifically relevant. By improving Bayesian optimization, different fields that need to optimize for complex objective function as part of their work, can carry out this work more efficiently.

Ethical issues for the application

As a researcher, it is worth pondering upon what the ethical issues are with the research one is involved with. Although the research done for this paper has had no ethical issues involved during implementation, the application of the solution offered varies a lot. And although as researcher there is no full control on these application, a part of the responsibility is to discuss these applications and be aware of them.

As indicated in the introduction, BO has many different applications but due to it being an abstract concept and more often than not being a part of a larger system, it usually will not be directly used for unethical applications. It can however be part of an application that can be used unethically, e.g. hackers could try to explore a system's vulnerabilities with BO, or it can be used in finding optimal ways of maximizing user engagement for the spread of misinformation.

Despite these application being a possibility, there are many useful and valuable applications like helping with drug design as mentioned in the introduction. Altogether, it seems that the further improvement of BO does more good than harm, certainly when taking into account that the unethical applications mentioned have not been researched in any way and will be much harder to implement than the valuable applications.

Looking at this research specifically, it is fair to say that the ideas mentioned will not be a catalyst for unethical applications as BO in general is a fairly well-developed concept and can already be applied in many ways.

7 Conclusions and Future Work

This study explored the possibility of replacing the acquisition function component in Bayesian Optimization with a neural network trained through reinforcement learning, termed as the MetaBO algorithm. The overarching aim was to ascertain the performance of meta-learning acquisition functions in BO when optimizing control variates of unknown functions, in comparison to BO equipped with standard acquisition functions.

The detailed implementation of the algorithm was presented, and the ensuing examination of the training performance and evaluation results underscored the subpar performance of the MetaBO algorithm. One contributing factor to this is the limited expertise of the author in reinforcement learning, which raises questions about potential enhancements in the implementation. Nonetheless, it also underscores the complexity involved in implementing MetaBO. Despite the availability of numerous plug-and-play reinforcement learning algorithms requiring only the design of an environment, these algorithms appear to underperform due to the intricate nature of the MetaBO environment.

Despite these disappointing results, this work aims to provide clarity on some of the steps involved in implementing the MetaBO algorithm and testing it on objective functions supplied in the BBOx library.

One possible approach for enhancing the training process could involve modeling the objective function on which the algorithm will be trained, as opposed to using the objective functions themselves. The rationale for this is that objective functions are often too complex for training purposes in practical applications, and must therefore be simplified by models that can still facilitate learning for the algorithm. Another potential improvement could involve modifying the input provided to the critic, as briefly discussed in section 4.

Moreover, future research could examine the application of transfer learning to multiple components of BO concurrently. This could potentially enhance the algorithm's performance by capitalizing on previously learned features and applying them across multiple components.

References

[1] T. Bai, Y. Li, Y. Shen, X. Zhang, W. Zhang, and B. Cui, *Transfer learning for bayesian optimization: A survey*, 2023. arXiv: 2302.05927 [cs.LG].

[2] M. Imani and S. F. Ghoreishi, "Bayesian optimization objective-based experimental design," in *2020 American Control Conference (ACC)*, 2020, pp. 3405–3411. DOI: 10.23919/ACC45564.2020.9147824.

[3] E. O. Pyzer-Knapp, "Bayesian optimization for accelerated drug discovery," *IBM Journal of Research and Development*, vol. 62, no. 6, 2:1–2:7, 2018. DOI: 10.1147/JRD.2018.2881731.

[4] J. Shigley, C. Mischke, and T. Brown, *Standard Handbook of Machine Design* (McGraw-Hill standard handbooks). Mcgraw-hill, 2004, ISBN: 9780071441643. [Online]. Available: https://books.google.nl/books?id=Mafom8J9sqYC.

[5] R. Gupta, *A Textbook of Machine Design*. Eurasia Publishing House, 2005, ISBN: 9788121925372. [Online]. Available: https://books.google.nl/books?id=6FZ9UvDgBoMC.

[6] W. Burgard, O. Brock, and C. Stachniss, "Active policy learning for robot planning and exploration under uncertainty," in *Robotics: Science and Systems III*. 2008, pp. 321–328.

[7] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf.

[8] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24, Curran Associates, Inc., 2011. [Online]. Available: https://

proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[9] B. Hsieh, P. Hsieh, and X. Liu, "Reinforced few-shot acquisition function learning for bayesian optimization," *CoRR*, vol. abs/2106.04335, 2021. arXiv: 2106.04335. [Online]. Available: https://arxiv.org/abs/2106.04335.

[10] S. Jiang, D. R. Jiang, M. Balandat, B. Karrer, J. R. Gardner, and R. Garnett, "Efficient nonmyopic bayesian optimization via one-shot multi-step trees," *CoRR*, vol. abs/2006.15779, 2020. arXiv: 2006.15779. [Online]. Available: https://arxiv.org/abs/2006.15779.

[11] B. Letham, R. Calandra, A. Rai, and E. Bakshy, *Re-examining linear embeddings for high-dimensional bayesian optimization*, 2020. arXiv: 2001 . 11659 [stat.ML].

[12] A. Shah and Z. Ghahramani, "Parallel predictive entropy search for batch global optimization of expensive objective functions," *CoRR*, vol. abs/1511.07130, 2015. arXiv: 1511 . 07130. [Online]. Available: http://arxiv.org/abs/1511.07130.

[13] M. Volpp *et al.*, *Meta-learning acquisition functions for transfer learning in bayesian optimization*, 2020. arXiv: 1904.02642 [stat.ML].

[14] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier, *Parallel bayesian global optimization of expensive functions*, 2019. arXiv: 1602.05149 [stat.ML].

[15] J. Wu and P. I. Frazier, *The parallel knowledge gradient method for batch bayesian optimization*, 2018. arXiv: 1606.04414 [stat.ML].

[16] M. Feurer, B. Letham, and E. Bakshy, "Scalable meta-learning for bayesian optimization using ranking-weighted gaussian process ensembles," 2018.

[17] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. E. Karro, and D. Sculley, Eds., *Google Vizier: A Service for Black-Box Optimization*, 2017, pp. 1487–1495. [Online]. Available: http://www.kdd.org/kdd2017/papers/view/google-vizier-a-service-for-black-box-optimization.

[18] T. Theckel Joy, S. Rana, S. Gupta, and S. Venkatesh, "A flexible transfer learning framework for bayesian optimization with convergence guarantee," *Expert Systems with Applications*, vol. 115, Aug. 2018. DOI: 10.1016/j.eswa.2018.08.023.

[19] H. C. L. Law, P. Zhao, L. Chan, J. Huang, and D. Sejdinovic, *Hyperparameter learning via distributional transfer*, 2019. arXiv: 1810.06305 [stat.ML].

[20] M. Poloczek, J. Wang, and P. I. Frazier, *Warm starting bayesian optimization*, 2016. arXiv: 1608.03585 [stat.ML].

[21] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Scalable gaussian process-based transfer surrogates for hyperparameter optimization," *Machine Learning*, vol. 107, pp. 43–78, 2017.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[23] J. Kim, S. Kim, and S. Choi, *Learning to warm-start bayesian hyperparameter optimization*, 2018. arXiv: 1710.06219 [stat.ML].

[24] M. Feurer, T. Springenberg, and F. Hutter, "Initializing bayesian hyperparameter optimization via meta-learning," *Proceedings of the Twenty-ninth AAAI Conference on Artificial Intelligence*, vol. 29, pp. 1128–1135, Feb. 2015. DOI: 10.1609/aaai.v29i1.9354.

[25] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Learning hyperparameter optimization initializations," in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–10. DOI: 10.1109/DSAA.2015.7344817.

[26] V. Perrone, H. Shen, M. Seeger, C. Archambeau, and R. Jenatton, *Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning*, 2019. arXiv: 1909.12552 [stat.ML].

[27] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Hyperparameter search space pruning - a new component for sequential model-based hyperparameter optimization," Sep. 2015, pp. 104–119, ISBN: 978-3-319-23524-0. DOI: 10.1007/978-3-319-23525-7_7.

[28] P. I. Frazier, *A tutorial on bayesian optimization*, 2018. arXiv: 1807.02811 [stat.ML].

[29] J. Achiam. "Spinning Up in Deep Reinforcement Learning." (2018), [Online]. Available: https://spinningup.openai.com/. (Accessed on: 13-06-2023).

[30] F. Nogueira, *Bayesian Optimization: Open source constrained global optimization tool for Python*, 2014– . [Online]. Available: https://github.com/fmfn/BayesianOptimization, (Accessed: 18.06.2023).

[31] KNAW, NFU, NWO, TO2-federatie, N. A. of Universities of Applied Sciences, and VSNU, "Netherlands code of conduct for research integrity," 2018. [Online]. Available: https://doi.org/10.17026/dans-2cj-nvwu.