Generative Evolutionary Strategy For Black-Box Optimizations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Many scientific and technological problems are related to optimization. Among them, black-box optimization in high-dimensional space is particularly challenging. Recent neural network-based black-box optimization studies have shown noteworthy achievements. However, their capability in high-dimensional search space is still limited. This study proposes a black-box optimization method based on evolution strategy and generative neural network model. We designed the algorithm so that the evolutionary strategy and the generative neural network model work cooperatively with each other. This hybrid model enables reliable training of surrogate networks; it optimizes multi-objective, high-dimensional, and stochastic black-box functions. In this experiment, our method outperforms baseline optimization methods, including evolution strategies, and a Bayesian optimization.

1 Introduction

Optimization is one of the most crucial issues in science and technology. Various simulations and experiments work as black-box functions, and there have also been innumerable optimization studies. Gradient-based optimization methods are easy choices for differentiable functions or simple convex functions. However, black-box functions are often non-differentiable and non-convex. Furthermore, they can be multi-objective and stochastic. A simple description of multi-objective black-box optimization is as follows

$$Optimize(f^1(X), \ldots, f^m(X))$$
$$X \in R^N$$

The search space is defined in real space R, where N is the dimension. f^i is a single-objective function, which can be stochastic.

The evaluation of electronic device designs is a practical problem of the black-box optimization. Since many device simulators have time-sequential input-output structures, it seems like they can be solved in reinforcement learning. However, if the observation cost is too high, it will be almost impossible to observe time-sequential data. Instead, the only information which we can observe is the final score. Therefore, the evaluation problem is defined as a black-box optimization, in this case.

For practical purposes, researchers have studied optimization methods in various ways. [4] [5] [6]. Typically, Bayesian optimization [1][2][3] and evolutionary strategies [7]-[20] are widely used. Notably, the Bayesian optimization is advantageous when the cost of the target function is high and the number of function calls is limited.

The estimating process of Bayesian optimization is very efficient when the number of search points is small. On the contrary, it becomes inefficient when the number of search points increases. Therefore,

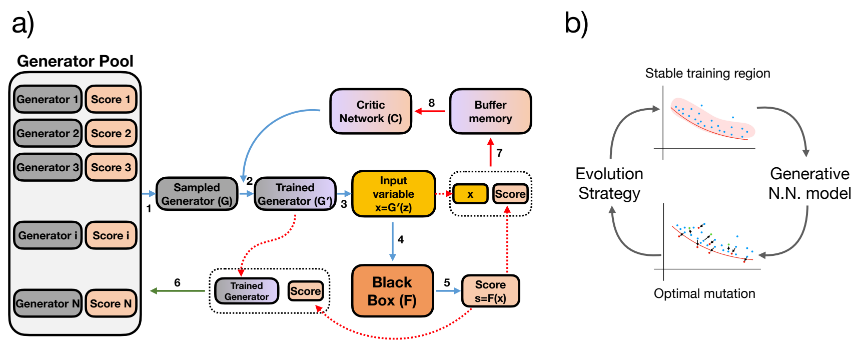

Figure 1: Schematic figures of GEO. a) GEO full algorithm. 1: Random sampling of generators. 2: Mutation (backpropagation). 3: Variables (x) generation. 4: Black-box run. 5: Data save and repeat 1-4. 6: Generator sorting with Pareto efficiency. 7: Storing search history in the buffer memory. 8: Critic network training. b) A simple description of the cooperative workflow

Bayesian optimization may not be appropriate in high-dimensional problems that require a large number of function calls.

Meanwhile, evolution strategies (ES) can be better choices for aforementioned cases. In most evolution strategies, the computational cost does not drastically increase according to the number of search points. Nevertheless, it does not mean that they can avoid the *curse of dimensionality*. The optimization performance of the evolutionary strategies also decreases rapidly as the dimension of the black-box increases. Although SEP-CMA[55], VD-CMA[56], and LM-CMA[57] have shown optimization capability in high-dimensional space of convex functions, optimization of high-dimensional non-convex problems still seems difficult.

Generative neural network-based models are recent approaches. They show noteworthy performance in test function optimizations, but their capability seems to be limited to single-objective functions and to 100-dimension [21][28]. We present GEO: Generative Evolutionary Optimization, a method for general black-box optimizations. It is designed to optimize stochastic, multi-objective, and high-dimensional black-box problems. We show that GEO outperforms baseline methods in finding Pareto fronts of Styblinski-Tang [42], Ackley [39], Rastrigin [36][37][38], Rosenbrock [40][41], ZDT1, ZDT2, and ZDT3 [43] test functions. Also, by converting Cartpole-V1 [44] to high-dimensional black-box problems, we show that GEO can be used in sequential problems. We also tested LeNet-5 [45] to see how it generates sub-manifold structures.

2 Related works

GEO is related to Evolutionary Generative Adversarial Networks (EGAN) [22], and Local Generative Surrogates Optimization (L-GSO) [21]. This section briefly introduces them.

2.1 L-GSO

L-GSO is a surrogate network model based black-box optimizer. It has a surrogate network and a generator network. The main idea of L-GSO is that the surrogate network only estimates a local shape of the objective function. Since the stabilization of the surrogate network is difficult, they suggest only to surrogate a local region. Also, the optimizer can be used in stochastic environments since it works in the neural network.

It is shown that L-GSO outperforms baseline optimizers in $dimension = 10$ and sub-manifold $dimension = 100$ problems. However, due to the limitation of the local sampling method, L-GSO applies only to a single-objective function.

Algorithm 1 GEO

Require: Initial generator pool $\{(G_1, s_1), (G_2, s_2), ..., (G_p, s_p)\}$, initial critic networks $\{C^1, C^2, \ldots, C^N\}$, a buffer memory $\{(x_1, s_1), (x_2, s_2), \ldots, (x_B, s_B)\}$, an input seed z (constant or random variable). The multi-objective function is defined as $(f^1, \ldots, f^N) = F$.

```
while iteration do
    while n < N do                                          ▷ Critic network training
        Sample C^n in {C^1, C^2, ..., C^N}
        g_c = ∇_θ (1/M_1) Σ_{j=0}^{M_1} ||C^n_θ(x_j) − f^n(x_j)||, x ∈ buffer
        C^n_θ ← Optimize(C^n_θ, g_c)
    end while
    while n < N do                        ▷ Generator network mutation (multi-objective)
        Sample C^n in {C^1, C^2, ..., C^N}
        while m < M_2 do                            ▷ M_2: the number of mutations
            RandomSample G_i in {(G_1, s_1), (G_2, s_2), ..., (G_p, s_p)}
            g_g = ∇_φ[±C^n(G_{i,φ}(z))]                   ▷ +: minimize, -:maximize
            G_i ← Optimize (G_i, g_g)
            x_i = G_i(z)
            s_i = F(x_i)
            buffer.append((x_i, s_i))
            pool.append((G_i, s_i))
        end while
    end while
    pool ← ParetoEfficiency(pool)                           ▷ Non-dominated sorting
    pool ← AgeEvolution(pool)                                          ▷ (optional)
end while
```

2.2 EGAN

Evolutionary Generative Adversarial Networks (EGAN) combines Generative Adversarial Network (GAN) [30]-[33] and evolution strategies. The core idea of EGAN is that the evolution strategy can assume a backpropagation as a mutation. It has one discriminator and multi generators. Generators of the evolution pool are mutated for each iteration, and they are sorted by fitness scores.

By comparing the mode collapsing results, the study shows that the evolution strategy efficiently complements the GAN algorithm. EGAN is not an optimizer. Nevertheless, we expected that a combination of an evolution strategy and a GAN could be adopted in our black-box optimization algorithm.

2.3 Other approaches

Global Topology Optimization network (GLOnet) [23] is a method for electromagnetic device designs. It is an advanced study of the previous research, adjoint-based topology-optimizer (ABTO) [24]-[27]. GLOnet increases optimization performance by adding generator networks on ABTO. GLOnet is not a black-box optimizer because the gradient is given directly from the target simulator. However, we can discover an essential role of the generator network for better optimization.

GNN-ES (Evolutionary Strategies with Generative Neural Networks) [28] is a combined method of bijective neural networks and evolutions. GNN-ES assumes latent space z and bijective Generative Network (GNN) $x = g(z)$, $z = h(x)$. It optimizes latent space and a bijective network. The update of latent space is carried out by evolution strategies. The study shows that GNN-ES can optimize test functions in $dimension = 10$. However, GNN-ES is not a surrogate model-based optimizer and it is restricted to bijective networks.

Conservative Objective Models (COMs) [29] is a surrogate model-based black-box optimizer. The key idea of COMs is regularizing the loss function of a surrogate model in training. Along with a standard supervised regression, it adds COMs-regularizers to prevent erroneously large predictions of the trained model.

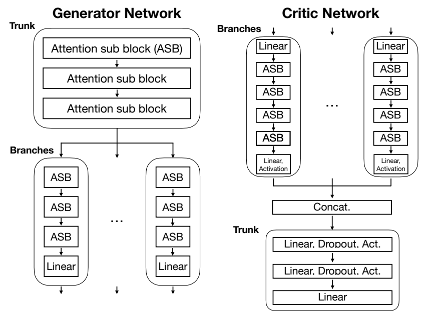

Figure 2: A trunk-branch network structure. The two-level structure is an ad hoc method to reduce memory overuse of attention networks.

Surrogate assisted evolution models are also related [58]. However, they do not guarantee $\mathcal{O}(n)$ computational complexity. This can be a weakness in high-dimensional problems.

3 Methods

GEO consists of two stages: the evolution and network training. The evolution pool maintains a certain number of generators on the basis of fitness scores. The evaluation and sorting of multi-objective scores is determined by Pareto-efficiency.

A generator training is also a mutation in the evolution. 1. A generator is randomly sampled from the pool. 2. The critic network trains the selected generator (using backpropagation, to increase or decrease a prediction of a critic). 3. The trained generator suggests a new variable $x{=}G(z)$. 4. Check $score{=}F(x)$. 5. Sort a new (G, $score$) pair in the pool.

(x, $score$) pairs are stored in a buffer memory and they are used to train critic networks. Each critic network is trained to surrogate a corresponding black-box object.

3.1 Generative model

GEO consists of a pool of generator networks and critic networks that create backpropagations. For each iteration step, generators are randomly sampled to make mutated generators. N-critic networks are prepared to make an N-objective surrogate model, each critic network corresponds to a single objective function. The mutated generators create variables $x = G_\phi(z)$, and N-objective scores $s^i = f^i(x)$ are measured. The set of scores (x, s), $s = (s^1, s^2, ..., s^N)$ is stored in the buffer memory. Training of the critic network is carried out using the buffer memory. After training, critic networks mutate the generator networks in the next iteration step. The backpropagation serves optimal mutations by increasing predictions of critic networks.

$$\mathcal{M}_G = \mathbb{E}[C^i(G_\phi(z))], C^i \in C^1, \dots, C^N$$

Usually, traditional GAN generators feed random latent variables z through the input layer, while some GAN algorithms separate latent variables from the input feeds [33]. Because GEO does not need inferences, we do not see z as a latent vector. We experimented with both random variables (Figure 5) and constants (Figure 4) as input feeds z.

Since each critic network has a corresponding objective, it must be trained separately using its corresponding objective function. We used L1 loss with a single objective function f^j and a critic network C^j. The loss function is defined as follows

$$\begin{gathered}\mathcal{L}_{C^j} = \mathbb{E}_{x \sim p_g} ||C^j_\theta(x) - f^j(x)|| \\ (f^1, \dots, f^N) = F\end{gathered}$$

The critic network learns variable x in a global region. Global training is essential for multi-dimensional Pareto front searches. (See 3.4)

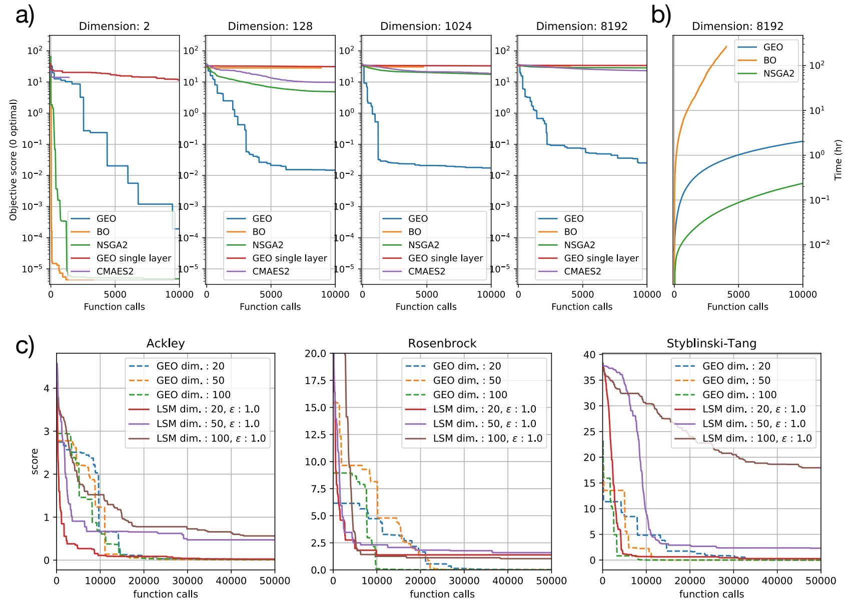

Figure 3: Performance comparisons of GEO and baseline optimizers in single-objective test functions. a) Optimization performances from 2 dimension to 8,192 dimension (Styblinski-Tang test function). b) Computational time in 8,192 dimension (real time). c) Optimization performances of GEO and LSM in single-objective functions. LSM is a modification of L-GSO.

3.2 Evolution strategy

Each generator is stored in the pool with a corresponding fitness score $s_j = (s_j^1, ...s_j^N) = F(x_j)$. Score data $\{(x_j, s_j)\dots\}$ are sorted by Pareto efficiency. The Pareto efficiency is defined as follows (for minimization cases)

$$\forall i \in 1, ..., N : f_i(x^*) \leq f_i(x), \exists j \in 1, ..., N : f_j(x^*) < f_j(x)$$

then, $x^* \in P$, where $x \in X$ and P is the Pareto efficiency. Pareto efficiency can be ranked in order P_1=Pareto(X), P_2=Pareto($X - P_1$), $\cdots$, they are calculated by non-dominated sorting methods. A more detailed explanation is provided in the supplement.

Optionally, age evolution can be added in the sorting part. The age evolution removes the oldest elements from the pool. Thereby, it prevents "the high score due to stochasticity" from surviving in the pool. A pool-refresh method is another option, but it makes the calculation time almost doubled.

In GEO, evolution strategy is not just an auxiliary tool. Without an evolution strategy, the training of networks can be unstable, which leads to the divergence. We discuss details in section 3.4.

3.3 Neural networks

Any kinds of neural networks, including Recurrent Neural Network (RNN) [47][48][49], Convolutional Neural Network (CNN) [46], and Full Connected (FC) network can be used as generator networks and critic networks. We chose a multi-head-self-attention network [50] for operational convenience. Figure 2 shows the self-attention network we used. The overall structure is modified from the original transformer model.

Because the attention network consumes gigantic memory size, it may cause GPU out-of-memory. It was a significant problem when we optimized high-dimensional functions. When using an NVIDIA Tesla V100 32G GPU in variable space of dimension $d > 2^{11}$, the total required memory exceeds the available memory size. We devised an ad hoc trunk-branch network structure to solve this problem.

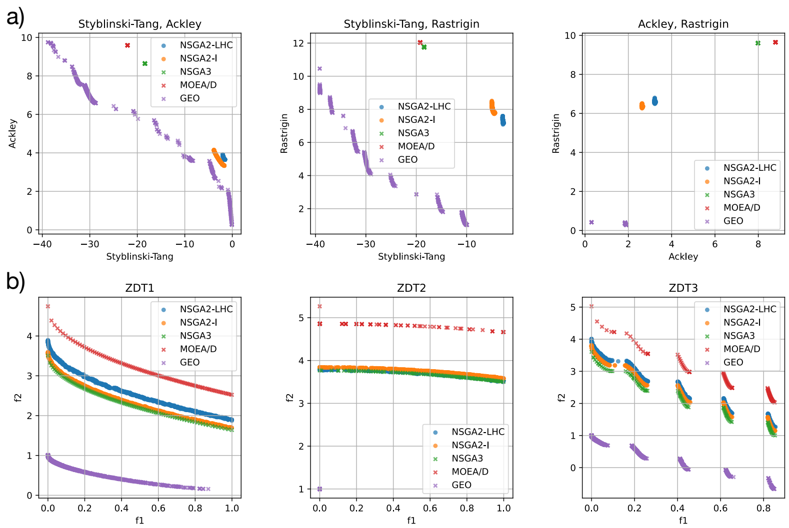

Figure 4: Optimization results of GEO, MOEA/D, NSGA-III, and NSGA-II in non-stochastic-multi-objective functions. Plots show optimization results in the $8,192$ dimension, after $100,000$ function calls (1,000 iterations). LHC indicates the latin hyper cube initial variable points, and I indicates the initial points which are obtained from GEO's initial points.

This structure has one trunk network and several branch networks, and each branch network extends from the trunk network. Branches have an identical structure with each other, and the length of the output tensor is defined by $n_{subvar} = n_{var}/n_{branches}$.

The split branch trick serves a memory-efficient structure, but it could be detrimental to optimization performance. Therefore, we implement the trunk-branch structure only for the sake of memory efficiency of GPU.

The baseline attention network structure includes dropout layers [51], and the dropout layers' randomness makes the generator stochastic. A random input feed z is also a source of stochastic behavior. We experimented with both stochastic generators (Figure 5) and non-stochastic generators (Figure 4). However, for critic networks, we maintained the dropout layers as non-zero.

3.4 A complementary strategy of generative network and evolution

Figure 1 shows the full algorithm of GEO. Training a surrogate model (critic network) is the essential part of a surrogate model-based optimizer, but it is also the trickiest part. The point is that the training data (the true data in GAN concept) is not prepared, and the data can be only acquired through on-the-fly searches. Without prepared data, training can be unstable since outbreaks of new data make the training region fluctuate. In this case, the algorithm diverges for the following reasons:

1. The generator suggests an input variable x in a wrong direction. 2. The critic network is trained with input variable x, but x has no information of a Pareto front. 3. The critic network trains the generator, but it does not give meaningful information.

In short, the divergence is a result of evil cycles of two networks.

The local sampling of L-GSO seems to be a simple stabilization method. In a case of N=1, L-GSO samples data in a local region, where the center is a current point. The current point is the Pareto-front, in this case.

However, it can be challenging in N-objective functions (in cases of N>1). Since the Pareto-front is R^{N-1} surface (not a single point), there will be a lot of centers of sampling. Then, it is not local anymore. The local sampling method cannot be used in multi-objective problems.

Therefore, we need to devise a stabilizing method for multi-objective functions. We suggest that the evolution strategy can be a good solution. The role of an evolution pool is to trap G and corresponding x near the Pareto-front (rank 1). At the same time, the data that is far from the

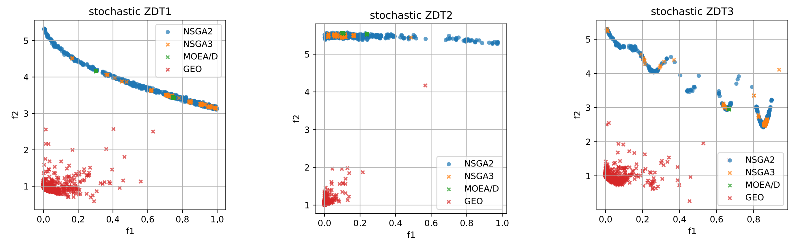

Figure 5: Optimization results of GEO, MOEA/D, NSGA-III, and NSGA-II in stochastic-multi-objective functions. Plots show optimization results in the $8,192$ dimension, after 100,000 function calls (1,000 iterations).

Pareto-front is discarded. Also, it slows down the fluctuation of the training data. As a result, the training data region is stabilized around the Pareto-front.

We expect the stabilization method to serve as an anchor, which prevents the training data from floating. At the same time, a properly trained critic network provides better mutation strategies. The interdependent cooperation of an evolution strategy and surrogate models is our main idea.

4 Experimental results

In this section, we compare the black-box optimization results of GEO with baseline optimizers. We tested single and multi-objective functions, stochastic and non-stochastic functions. The baseline optimizers are Bayesian optimization (Gaussian process), NSGA-II [7] (GA in 1-object), NSGA-III [15], MOEA/D [18], and CMA-ES [9] evolution algorithms.

4.1 Single objective functions

Figure 3a) shows performances of optimizers according to dimensions. For the single-objective test function, we used Styblinski-Tang function. At $dimension = 2$, Bayesian optimization shows the best performance. However, as the dimension increases, GEO outperforms baseline optimizers. At $dimension = 8,192$, baseline optimizers rarely finds the global minimum, while the GEO shows better performances.

To see how the depth of the generator network affects the performance, we also tested GEO with a single-layer generator. The single-layer generator appears to have little optimization capability. Even in the $dimension = 2$ problem, it shows a considerably slow optimization.

Bayesian optimization is a powerful method for high-cost black-box problems, but its performance can be weakened when the problem requires numerous function calls. The computational complexity of Bayesian optimization is known to be $\mathcal{O}(n^3)$ [3]. Figure 3b) shows the computational time of GEO, NSGA-II, and Bayesian optimization. Like most evolution strategies, GEO is designed to have a computational complexity of $\mathcal{O}(n)$.

Figure 3c) shows performance comparisons to a Local Surrogate Model (LSM), a modification of the L-GSO algorithm. LSM follows the general outline of L-GSO GAN implementation but adopts the self-attention network used in GEO. In the Styblinski-Tang function, LSM shows worse performance than in other functions. Also, the performance of LSM rapidly decreases as the dimension increases. On the other hand, GEO shows better optimization performance under various conditions.

4.2 Multi objective functions

In this section, we show optimization performance comparisons in multi-objective problems. We only compare GEO and evolution strategies because the high-dimensional problems require a lot of function calls.

Figure 4 shows optimization results in the non-stochastic-multi-objective functions. Each figure is a result after 100,000 function calls. For the multi-objective test functions, ZDT functions and

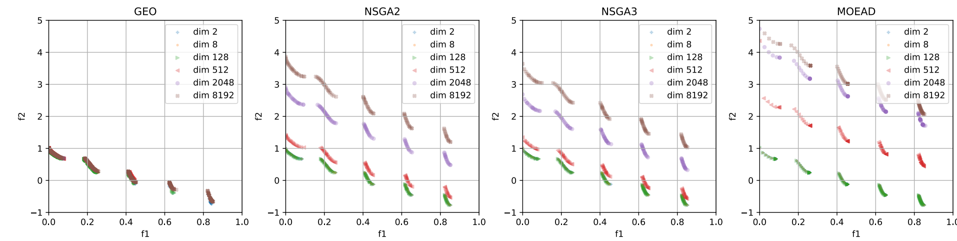

Figure 6: Optimization results of GEO, NSGA-II, NSGA-III and MOEA/D according to dimensions. (After 100,000 function calls with a ZDT3 test function.)

Table 1: Black-box optimization of Cartpole-V1. Scores are measured in relative scale (max score = 1.0, max steps = 500) after $50,000$ function calls.

Sequence length	256	512	1024	2048
GEO	0.598 ± 0.05	0.305 ± 0.03	0.310 ± 0.02	0.323 ± 0.03
CMAES	0.583 ± 0.09	0.292 ± 0.05	0.263 ± 0.02	0.280 ± 0.02
NSGA2	0.243 ± 0.01	0.120 ± 0.01	0.124 ± 0.01	0.121 ± 0.01

combined functions $F = [f^1, f^2]$ (f^i: Styblinski-Tang, Ackley and Rastrigin function) were used. A Latin Hyper Cube (LHC) [53] method is a good guess for initial states. However, GEO cannot implement LHC because it generates initial points x through the neural network. We gave two initial states in the baseline optimizer to control performance according to the initial state. $NSGA2\text{-}I$ uses GEO's initial distribution $x \in G(z)$ as its initial points, while $NSGA2\text{-}LHC$ uses LHC-initial points.

In the high-dimension, GEO outperforms baseline optimizers. The choice of initial points for NSGA-II rarely affects final results. A slightly different result appears in ZDT2. In ZDT2, GEO finds an optimal point, but it fails to find a global shape of the Pareto-front. Figure 6 shows the result of ZDT3 optimization according to dimensions. Classical ES algorithms significantly reduce performance in high-dimensional space, while GEO shows more robust performance in high-dimensional space.

Figure 5 shows optimization results in the stochastic-multi-objective functions. The optimization of stochastic functions is defined as follows

$$x^* = argmin_x \mathbb{E}[F(x)]$$
$$F(x) = (f_1, f_2)$$
$$f_i \leftarrow f_i + \mathcal{N}_i(\mu, \sigma)$$

$\mathcal{N}$ is a random normal distribution, and we set $\mu = 0.0$ and $\sigma = 1.0$. GEO outperforms baseline optimizers even in a stochastic environment, but it still has a single-point collapsing problem in the ZDT2 function.

4.3 Cartpole-v1

For the variety of test functions, we optimized the OpenAI [54] Cartpole-V1 by converting it into a black-box problem. It is also a simple toy model of time-sequential input-output (I/O) problems.

Cartpole-V1 is a test package that is mainly used in reinforcement learning [34][35]. Reinforcement learning requires a series of I/O structure. However, in the black-box problem, the entire input sequence is assumed as one large input, and only the final score is measured without observing the intermediate rewards and states. Since this experiment is a toy model of real-world problems which have stochastic environments, we kept the Cartpole-V1 stochastic.

The final score is measured in a relative score to the sequence length. (If the Cartpole is alive for m-length in n-sequences, the score is m/n.) Therefore, the maximum and minimum score set to 1.0 and 0.0. GEO outperforms others from 256 to 2048 dimensions in the experiment (Table 1).

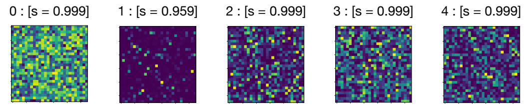

Figure 7: Black-box optimization of LeNet-5 (MNIST-trained). Each score corresponds to a prediction of LeNet-5 for a target number. After optimizations, scores get close to the maximum score (1.0). 0: 0.999, 1: 0.959, 2: 0.999, 3: 0.999, 4: 0.999.

Cartpole-V1 is difficult to solve with a black-box optimization approach due to the stochastic change of the initial state, but it shows a clear performance difference between the black-box optimizers.

4.4 LeNet-5

We also optimized LeNet-5, which is trained with the MNIST dataset. The optimization goal is to generate an image that makes the LeNet-5 predict a target number with a maximum score (maximum prediction score = 1.0). The LeNet-5 is regarded as a non-differentiable black-box. After $50,000$ function calls, the final scores of generated images reach very close to the maximum score (Figure 7).

In the related experiment, L-GSO, generative models appear to be better at finding the local optimum in sub-manifolds. For the same reason, we expected to see the sub-manifold structure in the generated image, but the generated image does not seem intuitive.

5 Discussion

Often, a neural network's learning mechanism is likened to learning manifolds in a high-dimensional space. Similarly, we guess that the critic network in GEO learns low-dimensional manifolds in the high-dimensional space. Therefore, we expect that finding global or local optima would be easy if the optima are in low-dimensional manifolds. Also, we consider that it is the reason why the depth of generators is important.

Meanwhile, we guess that the collapse problem of ZDT2 is caused by a concave shape of its Pareto-front. This is because, when the data is formed as a concave shape, a non-dominated sorting selects the edge state first. We have yet to find a clear solution to solve the collapse problem without compromising performance.

6 Conclusion

We have described a method for stochastic-multi-objective black-box optimization. GEO is an interdependent cooperation method of generative neural networks and the evolution strategy. The evolution strategy provides a stable training region for critic networks, and the critic networks provide efficient mutations to the evolution strategy. As our design intent, GEO seems to work appropriately in stochastic and high-dimensional multi-objective test functions.

Meanwhile, the Pareto-front collapsing problem, shown in ZDT2, is an important issue to be dealt with. Another limitation of GEO is the GPU memory consumption problem. The excessive memory consumption of attention networks limits its search space to around $10,000$ dimensions. In future researches, we can study other memory-efficient networks to solve this problem.

GEO is designed for optimization in extremely high-dimensions. However, the performance at lower dimensions is not guaranteed (see supplement). We think that [1,000 < d < 10,000] is the practical range of use of GEO, unless we improve the efficiency of network structures. In addition, the mutation of generators concentrates on an exploit, the explore strategy could be weak. In the next study, a strong exploit & explore strategy should be added to improve optimization performance.

References

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

[2] Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

[3] Lan, G., Tomczak, J. M., Roijers, D. M., & Eiben, A. E. (2022). Time efficiency in optimization with a bayesian-evolutionary algorithm. *Swarm and Evolutionary Computation, 69,* 100970.

[4] Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal, 7*(4), 308-313.

[5] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). IEEE.

[6] Hooke, R., & Jeeves, T. A. (1961). "Direct Search"Solution of Numerical and Statistical Problems. *Journal of the ACM (JACM), 8*(2), 212-229.

[7] Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation, 6*(2), 182-197.

[8] Deb, K., & Sundar, J. (2006, July). Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (pp. 635-642).

[9] Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation, 9*(2), 159-195.

[10] Hansen, N. (2006). The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation*, 75-102.

[11] Price, K., Storn, R. M., & Lampinen, J. A. (2006). *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media.

[12] Runarsson, T. P., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on evolutionary computation, 4*(3), 284-294.

[13] Runarsson, T. P., & Yao, X. (2005). Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 35*(2), 233-243.

[14] Deb, K., & Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE transactions on evolutionary computation, 18*(4), 577-601.

[15] Blank, J., Deb, K., & Roy, P. C. (2019, March). Investigating the normalization procedure of NSGA-III. In *International Conference on Evolutionary Multi-Criterion Optimization* (pp. 229-240). Springer, Cham.

[16] Seada, H., & Deb, K. (2015). A unified evolutionary optimization procedure for single, multiple, and many objectives. *IEEE Transactions on Evolutionary Computation, 20*(3), 358-369.

[17] Vesikar, Y., Deb, K., & Blank, J. (2018, November). Reference point based NSGA-III for preferred solutions. In *2018 IEEE symposium series on computational intelligence (SSCI)* (pp. 1587-1594). IEEE.

[18] Carvalho, R. D., Saldanha, R. R., Gomes, B. N., Lisboa, A. C., & Martins, A. X. (2012). A multi-objective evolutionary algorithm based on decomposition for optimal design of Yagi-Uda antennas. *IEEE Transactions on Magnetics, 48*(2), 803-806.

[19] Li, K., Chen, R., Fu, G., & Yao, X. (2018). Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation, 23*(2), 303-315.

[20] Panichella, A. (2019, July). An adaptive evolutionary algorithm based on non-Euclidean geometry for many-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 595-603).

[21] Shirobokov, S., Belavin, V., Kagan, M., Ustyuzhanin, A., & Baydin, A. G. (2020). Black-box optimization with local generative surrogates. *Advances in Neural Information Processing Systems, 33*, 14650-14662.

[22] Wang, C., Xu, C., Yao, X., & Tao, D. (2019). Evolutionary generative adversarial networks. *IEEE Transactions on Evolutionary Computation, 23*(6), 921-934.

[23] Jiang, J., & Fan, J. A. (2019). Global optimization of dielectric metasurfaces using a physics-driven neural network. *Nano letters, 19*(8), 5366-5372.

[24] Yang, J., Sell, D., & Fan, J. A. (2018). Freeform metagratings based on complex light scattering dynamics for extreme, high efficiency beam steering. *Annalen der Physik, 530*(1), 1700302.

[25] Hughes, T. W., Minkov, M., Williamson, I. A., & Fan, S. (2018). Adjoint method and inverse design for nonlinear nanophotonic devices. *ACS Photonics, 5*(12), 4781-4787.

[26] Jensen, J. S., & Sigmund, O. (2011). Topology optimization for nano-photonics. *Laser & Photonics Reviews, 5*(2), 308-321.

[27] Molesky, S., Lin, Z., Piggott, A. Y., Jin, W., Vucković, J., & Rodriguez, A. W. (2018). Inverse design in nanophotonics. *Nature Photonics, 12*(11), 659-670.

[28] Faury, L., Calauzenes, C., Fercoq, O., & Krichen, S. (2019). Improving evolutionary strategies with generative neural networks. *arXiv preprint arXiv:1901.11271*.

[29] Trabucco, B., Kumar, A., Geng, X., & Levine, S. (2021, July). Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning* (pp. 10358-10368). PMLR.

[30] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems, 27*.

[31] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

[32] Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4401-4410).

[33] Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

[34] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[35] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928-1937). PMLR.

[36] Rastrigin, L. A. (1974) Systems of extremal control. Mir, Moscow.

[37] Hoffmeister, F., & Bäck, T. (1990, October). Genetic algorithms and evolution strategies: Similarities and differences. In *International Conference on Parallel Problem Solving from Nature* (pp. 455-469). Springer, Berlin, Heidelberg.

[38] Mühlenbein, H., Schomisch, M., & Born, J. (1991). The parallel genetic algorithm as function optimizer. Parallel computing, 17(6-7), 619-632.

[39] Ackley, D. (2012). *A connectionist machine for genetic hillclimbing* (Vol. 28). Springer Science & Business Media.

[40] Rosenbrock, H. (1960). An automatic method for finding the greatest or least value of a function. *The Computer Journal, 3*(3), 175-184.

[41] Dixon, L. C. W., & Mills, D. J. (1994). Effect of rounding errors on the variable metric method. *Journal of Optimization Theory and Applications, 80*(1), 175-179.

[42] Styblinski, M. A., & Tang, T. S. (1990). Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. *Neural Networks, 3*(4), 467-483.

[43] Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2002, May). Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 1, pp. 825-830). IEEE.

[44] Kumar, S. (2020). Balancing a CartPole System with Reinforcement Learning–A Tutorial. *arXiv preprint arXiv:2006.04938*.

[45] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.

[46] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation, 1*(4), 541-551.

[47] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323*(6088), 533-536.

[48] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*(8), 1735-1780.

[49] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[50] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.

[51] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929-1958.

[52] Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access, 8*, 89497-89509.

[53] Iman, R. L., Davenport, J. M., & Zeigler, D. K. (1980). *Latin hypercube sampling (program user's guide).[LHC, in FORTRAN]* (No. SAND-79-1473). Sandia Labs., Albuquerque, NM (USA).

[54] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

[55] Ros, R., & Hansen, N. (2008, September). A simple modification in CMA-ES achieving linear time and space complexity. *In International conference on parallel problem solving from nature* (pp. 296-305). Springer, Berlin, Heidelberg.

[56] Akimoto, Y., Auger, A., & Hansen, N. (2014, July). Comparison-based natural gradient optimization in high dimension. *In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation* (pp. 373-380).

[57] Loshchilov, I. (2017). LM-CMA: An alternative to L-BFGS for large-scale black box optimization. *Evolutionary computation*, 25(1), 143-171.

[58] Blank, J., & Deb, K. (2022). pysamoo: Surrogate-Assisted Multi-Objective Optimization in Python. *arXiv preprint arXiv:2204.05855.*

[59] Tian, Y., Wang, H., Zhang, X., & Jin, Y. (2017). Effectiveness and efficiency of non-dominated sorting for evolutionary multi-and many-objective optimization. *Complex & Intelligent Systems*, 3(4), 247-263.

[60] Long, Q., Wu, X., & Wu, C. (2021). Non-dominated sorting methods for multi-objective optimization: review and numerical comparison. *Journal of Industrial & Management Optimization, 17*(2), 1001.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See discussion and conclusion
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary for implementation details
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary for implementation details
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See supplementary for experimental details
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary for experimental details
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See references
   (b) Did you mention the license of the assets? [Yes] See supplementary
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]