Towards Improving Mutations in Neuroevolution

Stav Bar-Sheshet
School of Mechanical Engineering
Tel-Aviv University
Tel-Aviv, Israel
stavb1@mail.tau.ac.il

Amiram Moshaiov
School of Mechanical Engineering
Tel-Aviv University
Tel-Aviv, Israel
moshaiov@tauex.tau.ac.il

Adham Salih
Mechanical Engineering Department
Braude College of Engineering
Karmiel, Israel
adhamsalih@braude.ac.il

Abstract— **The additive Gaussian noise mutation approach is commonly used in state-of-the-art neuro-evolution algorithms to optimize/train the weights of networks with fixed topology. In this paper, a novel modification to this mutation approach is proposed and studied. The proposed modification follows a theoretical analysis of mutations by the additive Gaussian noise. The well-known neuro-evolutionary algorithm of the Uber AI labs, which is based on additive Gaussian noise, is applied here as a reference algorithm. First, we modified the reference algorithm by replacing its mutation mechanism with the proposed mutation technique. Next, a comparison study is reported between the modified algorithm and the reference one. The comparison study is conducted on several well-known continuous control benchmark problems. The theoretical and computational studies indicate that the proposed modifications are not subjected to the constraints of the additive Gaussian noise approach, which appears to provide significantly better results with increasing problem dimensions.**

Keywords— Neuroevolution, Mutation Techniques, Parameter Space, Weight Distribution, Neuro-control, Continuous Control Benchmarks

I. INTRODUCTION

In the last decade, there has been a growth of interest in gradient-free methods for the weight-optimization/training of artificial Neural-Networks (NNs) [1]. Among such methods is the popular Neuro-Evolution (NE), which applies evolutionary computation techniques, e.g., in [2] and [3].

As appears from the literature, most NE studies generate offspring using mutation alone [1]. Moreover, recent advancements have demonstrated the successful application of NE to the fine-tuning of large-scale foundation models, specifically Large Language Models (LLMs) [4]. However, this scalability highlights a critical efficiency bottleneck, where immense computational resources were required for such tasks. Following such studies, the goal of this paper is to investigate and improve mutation techniques for NE without a major change from the computational efficiency viewpoint. This goal is accomplished by a combination of a theoretical approach and a computational study.

Our theoretical analysis concerns NN weight distribution. The significance of initializing weight exploration from a proper distribution is well-known, e.g., [5][6]. Yet, the significance of weight distribution is not restricted to the initialization stage of the search. When using mutation as the main technique to produce offspring, then the applied technique should be explored with respect to its ability to create weight distributions that are able to search the entire parameter space.

In the current investigation, we focus on one of the main mutation techniques in NE, i.e., additive Gaussian noise mutation as applied in [2]. This mutation mechanism is analyzed from the perspective of its resulting NN weight distribution. The key feature of the investigated mutation mechanism is to sample noise from a normal distribution. This raises the question of how the addition of noise affects the weight distribution along the evolutionary process.

The features of the additive Gaussian noise mutation approach inherently constrain it from producing offspring in terms of the mean and the variance of the weight distribution within each individual offspring. This suggests that algorithms such as in [2] have an implicit limited ability to explore the NNs parameter space. In the following, we refer to the constraining features of the additive Gaussian noise mutation approach as constraints.

The necessity of rigorous investigation into the weight distribution resulting from mutation operators is further supported by the recent work in [7]. The authors proposed a self-adaptive mutation operator based on the additive Gaussian noise approach, applied to both the weights and topology of the NN, which achieved results superior to the well-known NEAT algorithm [8]. However, while their approach introduces adaptivity, it only partially addresses the inherent constraints that we identify and aim to resolve in this work.

Furthermore, precedent for a negative mean value within the weight distribution is established by [9]. The authors demonstrate that initializing neural network weights with a negatively shifted mean is advantageous for training, thereby supporting our rationale for allowing the weight distribution mean to shift.

Based on the above observation, our study aims to propose an alternative mutation approach and to numerically investigate its potential as compared with that of the additive Gaussian noise mutation approach.

In addition to the additive Gaussian noise mutation approach, there is another well-known mutation method that is based on the covariance matrix, e.g., in [10] and [11]. It should be noted that the covariance approach is not subjected to the constraints that are investigated here. Yet, the covariance approach is known to be limited by its scalability and by several aspects of its computational efficiency, including complexity, the required memory, and communication bandwidth [12]. Therefore, we do not include comparisons to CMA-ES and NAS algorithms in our study.

The main contributions of this study are as follows:

1. We present a theoretical analysis of the additive Gaussian noise mutation approach to highlight its constraining characteristics, which suggest an implicit limitation on the exploration ability of the parameter space.
2. We provide a demonstration of the constraints of the additive Gaussian noise mutation approach.
3. We suggest a novel mutation technique that modifies the constraints of the additive Gaussian noise mutation without any major change to the computational efficiency.
4. We discuss the operational aspect of the proposed approach.
5. We test and compare our approach with the additive Gaussian noise using continuous control tasks of the Mujoco physics simulator [13]. The reference algorithm, which is used in the comparison study, is the neuro-evolutionary algorithm that was developed by the Uber AI labs [2].

The rest of this paper is organized as follows. Section II deals with the theoretical analysis of the additive Gaussian noise mutation approach and provides an associated numerical demonstration. Section III outlines the proposed mutation approach and its operational implications. Following a brief description of the compared algorithms, Section IV provides details of the computational investigation and results. Finally, Section V concludes this study.

II. The Theoretical Analysis

The additive Gaussian noise mutation method is mainly used in NE algorithms for weight optimization/training of NNs of fixed topology. It should be noted that in NE algorithms, such as in [2], [3] and [4], the additive Gaussian noise is sampled independently from the standard normal distribution, $\boldsymbol{\eta}\sim N(\mathbf{0}, \boldsymbol{I})$. The noise sampling is used to create offspring from a shared noise block for the entire evolutionary process. In such implementations, the weights are directly encoded as a vector $\boldsymbol{\theta} \in \mathbb{R}^n$. At each generation, any offspring, $\boldsymbol{\theta'}$, is created by adding a vector of Gaussian noise, $\boldsymbol{\eta}$, scaled by a factor σ to its parent vector of weights, $\boldsymbol{\theta}$, as follows:

$$\boldsymbol{\theta'} = \boldsymbol{\theta} + \sigma\boldsymbol{\eta} \tag{1}$$

The following provides a theoretical analysis of the additive Gaussian noise method using a statistical viewpoint. The focus is on the weight distribution characteristics as resulted by the considered mutation method. It is important to note that our statistical analysis concerns the statistics of the components of the vectors rather than the statistics of the vectors. Namely, the set of components of each of the weight and the noise vectors is considered as a random sample. Following [14], the statistical measures for a weight vector $\boldsymbol{\theta}$ are calculated as its sample mean and sample variance, i.e.:

$$\bar{\theta} = \frac{1}{n}\sum_{i=1}^{n}\theta_i \tag{2}$$

$$s_{\boldsymbol{\theta}}^2 = \frac{1}{n}\sum_{i=1}^{n}(\theta_i - \bar{\theta})^2 \tag{3}$$

where θ_i is the i^{th} component of $\boldsymbol{\theta}$. In the following, the mean and the variance of the noise vector are denoted as $\bar{\eta}$ and $s_{\boldsymbol{\eta}}^2$ respectively. These are calculated similarly to (2) and (3).

Here we are interested in the Pearson correlation between the associated components of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$. Following this approach, the sample correlation coefficient is calculated between pairs of components of the same index from each vector as follows [14]:

$$r_{\boldsymbol{\theta},\boldsymbol{\eta}} = \frac{\sum_{i=1}^{n}(\theta_i-\bar{\theta})(\eta_i-\bar{\eta})}{\sqrt{\sum_{i=1}^{n}(\theta_i-\bar{\theta})^2\sum_{i=1}^{n}(\eta_i-\bar{\eta})^2}} \tag{4}$$

This correlation describes the linear dependency between the samples. The values of this measure span between -1 to $+1$. It should be noted that the correlation is zero for independent samples while the limit values indicate full correlation in the negative and positive directions, respectively. The correlation can be described using a 2×2 sample correlation matrix $\boldsymbol{R}$ as follows:

$$\boldsymbol{R} = \begin{bmatrix} r_{\boldsymbol{\theta},\boldsymbol{\theta}} & r_{\boldsymbol{\theta},\boldsymbol{\eta}} \\ r_{\boldsymbol{\eta},\boldsymbol{\theta}} & r_{\boldsymbol{\eta},\boldsymbol{\eta}} \end{bmatrix} = \begin{bmatrix} 1 & r_{\boldsymbol{\theta},\boldsymbol{\eta}} \\ r_{\boldsymbol{\eta},\boldsymbol{\theta}} & 1 \end{bmatrix} \tag{5}$$

This matrix is symmetric and semi-positive definite.

In the following, the mean and variance of the calculated sum in (1) are analyzed for both cases of addition and subtraction of the noise vector. According to (2) and (3), the characteristics of the resulting weight vector, when the noise is either added or subtracted, are as follows:

$$\bar{\theta}' = \overline{\theta \pm \sigma\eta} = \bar{\theta} \pm \sigma\bar{\eta} \tag{6}$$

$$s_{\boldsymbol{\theta}'}^2 = s_{\boldsymbol{\theta}\pm\sigma\boldsymbol{\eta}}^2 = s_{\boldsymbol{\theta}}^2 + \sigma^2 s_{\boldsymbol{\eta}}^2 \pm 2\sigma r_{\boldsymbol{\theta},\boldsymbol{\eta}} s_{\boldsymbol{\theta}} s_{\boldsymbol{\eta}} \tag{7}$$

These equations are based on the theorem concerning the sum of random variables [15].

Observing (6) and (7) raises several issues for searching the parameter space using the described mutation method of (1). As noted above, the additive noise is commonly sampled independently from the standard normal distribution, $\boldsymbol{\eta}\sim N(\mathbf{0}, \boldsymbol{I})$. Following (6), since $\bar{\eta} = 0$, the resulting mean of the offspring does not change as compared with that of the parent, i.e., $\overline{\theta \pm \sigma\eta} = \bar{\theta}$. This means that the common use of the considered mutation method limits the search to NNs with the same mean for their offspring weight vectors as that of their parent vectors. We hereby refer to this constraint as the *"static mean"* constraint. Second, since the noise sample is independent with respect to the weight vector, their correlation $r_{\boldsymbol{\theta},\boldsymbol{\eta}}$ is zero. Therefore, following (7), the additive method results in a monotonic increase in the variance of its weight vector for both addition and subtraction, i.e., $s_{\boldsymbol{\theta}\pm\sigma\boldsymbol{\eta}}^2 = s_{\boldsymbol{\theta}}^2 + \sigma^2 s_{\boldsymbol{\eta}}^2$. This attribute limits the search to NNs with an increasing variance for their offspring weight vectors as compared to their parent vectors. We hereby refer to this constraint as the *" monotonically increasing variance "* constraint.

In summary, using the additive Gaussian noise mutation method, as applied in [2], implies that the search has two main constraints including static mean and monotonically increasing variance. To demonstrate these features when using additive Gaussian noise mutation method over the generations, we apply a Monte-Carlo simulation. The simulation starts with a weight vector that its weights are initialized using Kaiming initialization [6]. The applied NN

has a similar architecture to the one that we used in the case of the Swimmer-v4 (see Table I in Section IV). This means that the dimension of the considered weight vector is $n = 69{,}63\,\mathrm{K}$. From reasons of computational efficiency, we utilized the noise sampling method as applied in [2] which was inspired by [12]. The method samples the Gaussian noise from a pre-initialized block along the evolutionary process. Next, we used a Monte-Carlo simulation to mutate the weight vector 1000 times using Gaussian noise sampled from the initialized block.

The results of the simulation are presented in Fig. 1 in which the colors indicate the number of generations according to the color-bar. The figure includes a presentation of the resulting weight distributions along the generations, as well as graphs of the mean, variance and correlation along the generations. As shown in the figure, the mean values of the weight distribution remain constant around zero, while the variance curve appears to increase monotonically along the process. This demonstrates the constraints which are highlighted by the theoretical analysis. It should be noted that the resulting correlation between the sampled noise and the weight vector is distributed around zero. This phenomenon is a result of numerical reasons, i.e., it is due to the sampling from the noise block. Nevertheless, the obtained correlation values indicate that the noise and the parameter vectors are practically independent.

As noted in the introduction, the constraints suggests that algorithms such as in [2] have an implicit limited ability to explore the NNs parameter space. With this respect, it should be noted that the implicit limitation is about the exploration abilities of each offspring. Assuming that all individuals are created under these constraints then the constraints are relevant to the entire population. This remark is relevant here since this study is based on the algorithm of [2], which applies a population and a truncated selection approach while producing each of the individuals under the constraints of static mean and monotonically increasing variance.

III. The Suggested Mutation Method

Based on the analysis in Section II, this section provides a description of two modifications that are suggested to the additive Gaussian noise mutation. The first modification, which is described in Subsection A, aims at changing the static mean constraint. The second modification, which is presented in Subsection B, aims at changing the monotonically increasing variance constraint. In Subsection C, a combination of these two modifications is suggested including a demonstration to illustrate the effect on the resulting distribution. Finally, a discussion on the operational aspects of the combined approach is provided in Subsection D.

A. Modifying the Mean

To allow for a weight distribution with a non-zero mean, which can change over the generations, two features are added to the additive Gaussian mutation approach. First, a shared noise block of samples from a normal distribution is created with a non-zero mean. This non-zero mean value pre-defines $\bar{\eta} > 0$. Next, a sign operator is randomly sampled from a discrete uniform distribution with equal probability, $\iota \sim U\{-1,1\}$. The mean modification to the mutation approach results in:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \iota \cdot \sigma\boldsymbol{\eta} \,|\, \bar{\eta} > 0, \iota \sim U\{-1,1\} \tag{8}$$

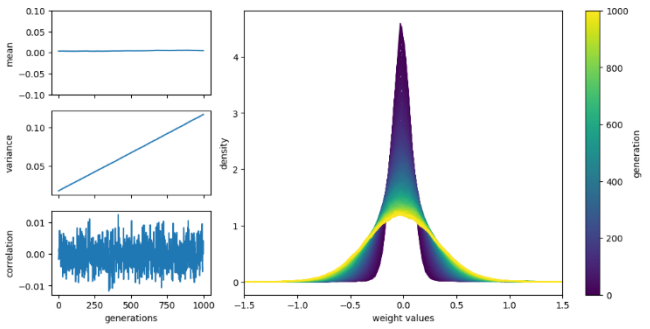

Fig. 1 Simulation of the additive Gaussian noise mutation

Using this approach, the sampled population in each generation spans different mean values. Since better-performing individuals are selected in each generation from the evolved population, the mean of the weight distribution changes stochastically both in terms of its value and direction according to (6) and (8). We call this feature *Swap* (Sw).

B. Modifying the Variance

To modify the variance of the mutated weights, we suggest utilizing the correlation-dependent component in (7). In particular, subtraction of that term can be used to reduce the variance.

The method in [16] is adopted to generate correlated normal noise, $\boldsymbol{\eta_c}$, from the independently sampled noise, $\boldsymbol{\eta}$. First, both the weight and noise vectors are standardized as follows:

$$\tilde{\theta}_i = \frac{\theta_i - \bar{\theta}}{s_{\boldsymbol{\theta}}} \sim N(0,1)\ ;\ \tilde{\eta}_i = \frac{\eta_i - \bar{\eta}}{s_{\boldsymbol{\eta}}} \sim N(0,1) \tag{9}$$

A correlation target value, c_t, is pre-defined (once) to create the following target correlation matrix $\boldsymbol{R}$:

$$\boldsymbol{R} = \begin{bmatrix} 1 & c_t \\ c_t & 1 \end{bmatrix} \tag{10}$$

$\boldsymbol{R}$ is symmetric and semi-positive definite. Hence, it can be uniquely decomposed using the Cholesky decomposition. This results in a product of a lower triangular matrix and its conjugate transpose, $\boldsymbol{R} = \boldsymbol{L}\boldsymbol{L}^T$. Next, a matrix $\boldsymbol{X} = [\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\eta}}] \in \mathbb{R}^{n\times 2}$ is defined. Then, $\boldsymbol{X}$ is transformed into $\boldsymbol{Y} = [\tilde{\boldsymbol{\theta}}, \tilde{\boldsymbol{\eta}}_{\boldsymbol{c}}] = \boldsymbol{X}\boldsymbol{L}^T$ in which the column vectors $\tilde{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{\eta}}_{\boldsymbol{c}}$ are correlated. Note that the first column does not change due to the transformation. The obtained vectors are then restored to their original mean and variance by the inverse functions of (9). We call this feature *Cholesky* (Ch) and formulate it as follows:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \iota \cdot \sigma\boldsymbol{\eta_c} \,|\, r_{\boldsymbol{\theta},\boldsymbol{\eta_c}} = c_t, \iota \sim U\{-1,1\} \tag{11}$$

C. The Combined Method

Both modifications are combined into a method that is hereby termed the *Cholesky Swap* mutation method. The formulation of the method simply combines (8) and (11). This results in:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} + \iota \cdot \sigma\boldsymbol{\eta_c} \,|\, \bar{\eta} > 0, r_{\boldsymbol{\theta},\boldsymbol{\eta_c}} = c_t, \iota \sim U\{-1,1\} \tag{12}$$

The results of a Monte-Carlo simulation of the combined approach are presented in Fig. 2. These are obtained using

simulation as in Section II, except for $\bar{\eta} = 0.1$ and $c_t = 0.3$. To support visualization, the sign operator is selected with a probability that slightly prefers subtraction over addition.

The results in Fig. 2 demonstrate that the proposed *Cholesky Swap* mutation method does not exhibit the constraints of the additive Gaussian noise mutation technique. Moreover, the correlation graph in Fig. 2 fluctuates around 0.3 as expected. This confirms the utility of the method in [16] to sample noise that is correlated with the weight vector.

In summary, the proposed mutation approach extends the original additive Gaussian noise mutation approach. Namely, it may provide NNs with weight distributions that either obey or disobey the constraints of the original approach. Hence, it is expected to provide alternative solutions that cannot be found by the original approach.

D. Discussion on the Operational Aspects

When NE is applied to large NNs, one should examine the search mechanism not just from a performance viewpoint but also from the operational aspects. The focus here is on adapting the parallel noise sampling method as described in [2] to the proposed mutation approach.

From a computational efficiency perspective, our approach allows reconstructing the correlated noise vector in each generation without the need to communicate it between workers. The required change to the communication approach of [2] is only the addition of the sign regarding $\iota \sim U\{-1,1\}$ in (12). This results in a method that is independent of the size of the network in the sense of the communication bandwidth, similar to [2].

Moreover, in the era of hardware-accelerated NE, where tensorized operations can yield orders-of-magnitude speedups on GPUs as demonstrated in [17], we emphasize that the operations required for the proposed method are fully compatible with such tensorized frameworks. This ensures that the benefits of our method do not come at the expense of the parallelism required for modern large-scale experiments.

IV. Experimental Study

This section concerns a numerical study to demonstrate our mutation approach and to compare its results with those obtained by the additive Gaussian noise approach. Subsection A describes the applied benchmarks. Subsection B briefly describes the algorithms that are used in this study including a reference algorithm and the proposed algorithm. Subsection C presents the comparison approach, whereas Subsection D provides details on the experimental setup. Finally, Subsection E presents the results.

TABLE I. Problem Description

Problem	Inputs	Outputs	NN Size	Difficulty
Swimmer-v4	8	2	69,63 K	Easy
Hopper-v4	11	3	70,66 K	Easy
HalfCheetah-v4	17	6	72,97 K	Moderate
Walker2d-v4	17	6	72,97 K	Moderate
Ant-v4	27	8	76,04 K	Hard
Humanoid-v4	376	17	167,70 K	Very Hard

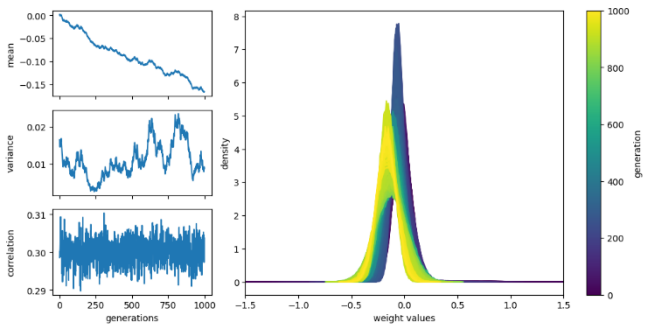

Fig. 2 Simulation of the suggested combined mutation approach

A. The Benchmark Problems

This study is based on six well-known benchmark problems as provided in Mujoco physics simulator [13]. The applied problems include Swimmer-v4, Hopper-v4, HalfCheetah-v4, Walker2d-v4, Ant-v4, and Humanoid-v4. These problems concern various control tasks in continuous state and action domains. The number of inputs, outputs, and the applied network size, n, are shown in Table I for each of the problems. Presumably, with increased numbers the search difficulty is increased. Hence, we suggest dividing the problems into four categories as presented in Table I.

B. The Compared Algorithms

In this subsection, the compared algorithms are briefly described including the reference algorithm and the proposed algorithm. As a reference, we use the neuro-evolutionary algorithm, which was developed at the Uber AI labs [2]. This reference algorithm, which is hereby referred to as Uber-GA, is an evolutionary algorithm that is based on the Gaussian additive noise mutation for the offspring generation. As such it is subjected to the constraints of that technique as described in Section II. In this reference algorithm, the population undergoes *truncation selection,* which is used to select a pre-defined number of the best individuals to pass to the next generation.

The proposed algorithm, which is hereby termed "*ChSw*," is a version of Uber-GA in which the mutation is replaced with the suggested Cholesky-Swap mutation method, as described in Section III.

C. The Comparison Approach

In order to present a fair comparison between the algorithms, the hyper-parameters were fixed over the experiments, and the number of evaluations was kept the same using the same number of generations and population size. Moreover, we fixed the noise scale factor which functions as a learning rate between the different algorithms. This allows testing which of the mutation methods results in a better-performing solution.

In addition, we fixed all the different random processes in the evaluation of the algorithms using the same seed no. '0'. This means that all experiments use the same initial population of NNs and the same initial states. The random noise block is also generated using seed no. '0'.

To statistically compare the algorithms, we applied a total of 13 runs per each of the benchmark problems. Each run was applied using a different seed.

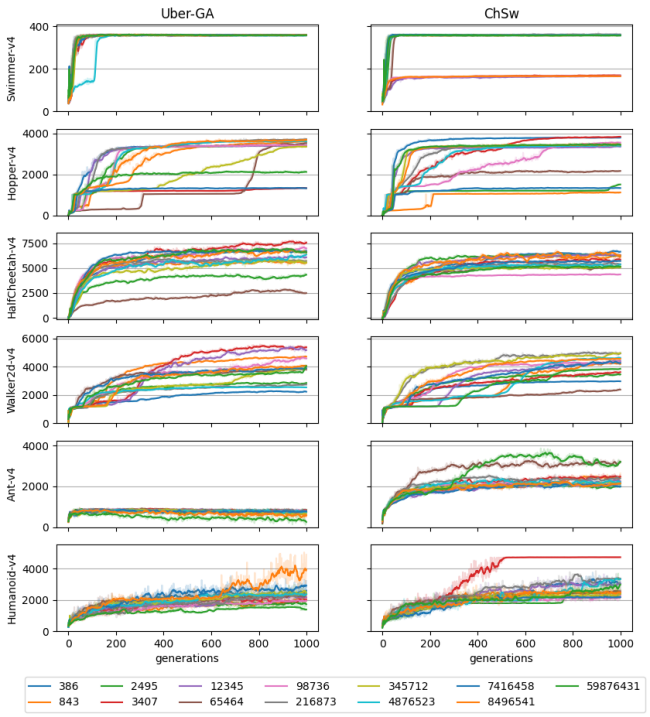

Fig. 3 Performance curves per seed for the different control problems

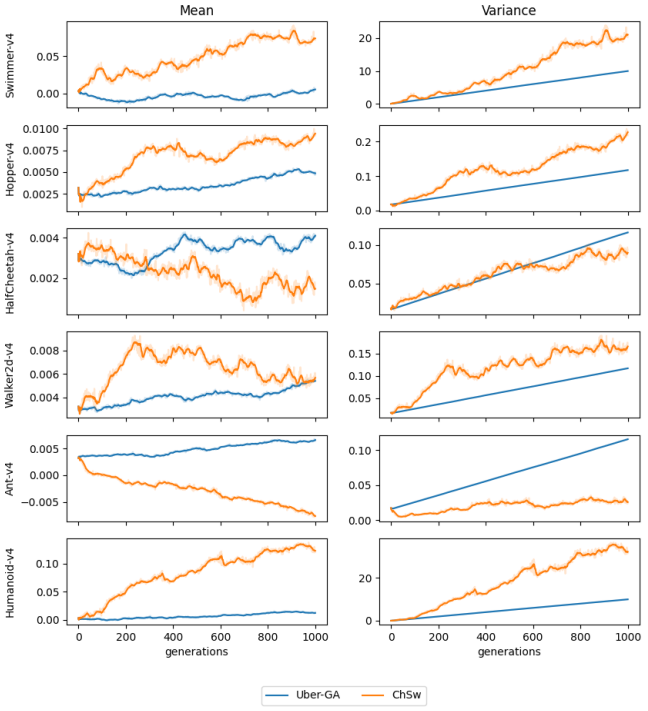

Fig. 4 The characteristics of the weight distribution along the evolutionary process of the best controller found using the same seed of "843" for the different algorithms and control problems.

D. *Experimental Setup*

As commonly done in studies that involve the considered benchmarks, e.g. in [2] and [12], all of our experiments involved a NN that includes two hidden layers with 256-unit, which is based on layer-normalization and Relu activation function. For the output units, tanh was used as the activation function. The network weights were initialized using Kaiming normal initialization [6] where all bias weights were initialized to zero. The evolution was based on a population of 96 individuals over the course of 1000 generations. Normalized inputs were used as in [2]. The shared noise block was sampled from a standard normal distribution for Uber-GA, while for ChSw a mean of 0.01 was used. The correlation coefficient target for the ChSw method was set to 0.3. These hyper-parameters were chosen to induce small, measured weight changes using the proposed methods. While we did not apply rigorous hyper-parameter tuning, we found the current values to be sufficient. The experiments were evaluated on a 48-core cluster.

The compared algorithms use a Learning Rate (LR) of 0.01, except for the Swimmer-v4 and Humanoid-v4 problems. In the latter two cases, an LR of 0.1 was used. For the Swimmer problem, we found that an LR of 0.1 quickly solved the problem while 0.01 did not. It should be noted that in [2], a population of 12.5K individuals was used over 1500 generations to achieve performance of 3000 for Humanoid-v1 with the same network architecture as used here. In contrast, as described above, our study applied a much smaller population of 96 individuals for 1000 generations. This was done due to the available computational resources for our study. Therefore, we increased the LR to apply a much faster search with less convergence stability. The code, seeds, and the full list of hyper-parameters are published for reproducibility purposes[1].

E. *Experimental Results*

In this subsection, the results of the experimental study are shown. The 2nd and the 3rd columns of Table II summarize the medians and the Interquartile Ranges (IQRs) of the performances as obtained by the compared algorithms. The best median results are marked using bold letters. The Wilcoxon rank-sum test is used to make statistical inferences with a significance level of 5%. The last column of Table II provides the p-values as obtained by applying this test for each comparison. Underlined results are those in which the null hypothesis is rejected. It can be concluded from the table that Uber-GA produced significantly better results only for the case of the Swimmer benchmark, which is considered as the easiest problem. However, for the most complex benchmarks, i.e., Ant and Humanoid, ChSw is found to be statistically better than Uber-GA. For all other problems, the performances were found to be statistically equivalent.

The plots in Fig. 3 present, at each generation, the best performances as obtained by the two algorithms. In particular, the left and right sides show the results by Uber-GA and by ChSw, respectively. Each color in the plot presents a different

TABLE II. PERFORMANCE RESULTS SUMMARY

Problem	GA	ChSw	p-value
Swimmer-v4	**363.5(±0.5)**	362.7(±191.2)	0.018
Hopper-v4	**3495.9(±553.5)**	3475.7(±1515.3)	0.837
HalfCheetah-v4	**6521.6(±1058.1)**	5797(±1012.7)	0.081
Walker2d-v4	3975.9(±1858.5)	**4455.6(±886.6)**	0.411
Ant-v4	925.9(±12.4)	**2381.5(±327.7)**	0.00002
Humanoid-v4	2776.7(±642.8)	**3127.3(±787.5)**	0.027

[1] https://github.com/stavbarsh/neuroevolution-mutations

random seed. It should be noted that the graphs were smoothed using the running average of the last 10 generations. Based on the presented plots, it can be observed that the obtained results from both algorithms are sensitive to the chosen seed. This phenomenon highlights the need for multiple runs with statistical tests when comparing such algorithms, as done here.

Typical results of the characteristics of the weight distribution, including the mean and the variance along the process, are shown in Fig. 4, which is based on seed no. "843." First, it should be noted that the mean curves in Fig. 4 appears to be non-static not just for ChSw but also for Uber-GA. The latter result appears to be in contrast with the theoretical findings. However, it is expected that the fluctuations of the mean of Uber-GA are due to sampling a subset of noise from the entire block which is not accurately distributed around zero. In contrast, the mean and the variance curves for ChSw are not restricted to the constraints of the additive Gaussian noise mutation and are controlled by the ChSw algorithm.

Finally, it should be noted that the presented results are restricted to the effect of the combination of the proposed two modifications. An additional study was conducted by the authors, which is not reported here, indicated that each of the modifications is relevant for the success of the proposed approach.

V. Conclusions and Future Work

This paper deals with the inherent constraints of the additive Gaussian noise mutation approach. This approach is commonly used in state-of-the-art neuro-evolution algorithms for its ability to deal with large networks as applied to parallel computing. Following an analysis and demonstration of these constraints, a novel modification is proposed and studied, which alleviates these constraints, while keeping the ability to be implemented using parallel processing. The proposed approach may provide NNs with weight distributions that either obey or disobey the constraints of the original approach.

To investigate the proposed modification as compared with the original mutation approach, the well-known neuro-evolutionary algorithm of the Uber AI labs is applied as a reference algorithm. The results of the reference algorithm are compared with a modified version in which the mutation mechanism is replaced with the proposed mutation technique. A comparison study is conducted on several well-known continuous control benchmark problems. The numerical results demonstrate the ability of the modified algorithm to search for networks without the constraints of the additive Gaussian noise approach. The statistical comparison of the algorithm is based on a set of runs using different seeds.

The conclusion from the comparison is that the modified approach provides significantly better results with increasing problem dimensions. This conclusion should be further investigated by increasing the size of the samples and using a larger set of benchmarks and types of problems.

Future studies might add the use of adaptative/dynamic parameters of the mutation, such as the mean and correlation factors. This should aim at improving exploration while also providing exploitation capabilities.

References

[1] E. Galván and P. Mooney, "Neuroevolution in deep neural networks: Current trends and future challenges," IEEE Transactions on Artificial Intelligence 2, no. 6, pp. 476-493, 2021.

[2] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," arXiv preprint, arXiv:1712.06567, 2017.

[3] S. Risi and K. O. Stanley, "Deep neuroevolution of recurrent and discrete world models," In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 456-462, 2019.

[4] Q. Xin, Y. Gan, C. F. Hayes, Q. Liang, E. Meyerson, B. Hodjat, and R. Miikkulainen. "Evolution strategies at scale: Llm fine-tuning beyond reinforcement learning," arXiv preprint arXiv:2509.24372, 2025.

[5] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, pp. 249-256, 2010.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," In Proceedings of the IEEE International Conference on Computer Vision, pp. 1026-1034, 2015.

[7] H. Motoaki, M. Komura, A. Miyamoto, D. Morimoto, and K. Ohkura. "Improving the performance of mutation-based evolving artificial neural networks with self-adaptive mutations," Plos one 19, no. 7, 2024.

[8] S. Kenneth O. and R. Miikkulainen. "Evolving neural networks through augmenting topologies." Evolutionary computation 10, no. 2, pp. 99-127, 2002.

[9] A. Yilmaz and R. Poli. "Successfully and efficiently training deep multi-layer perceptrons with logistic activation function simply requires initializing the weights with an appropriate negative mean." Neural Networks 153 pp. 87-103, 2022.

[10] N. Hansen, "The CMA evolution strategy: a comparing review," Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms, pp. 75-102, 2006.

[11] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters and J. Schmidhuber, "Natural evolution strategies." The Journal of Machine Learning Research 15, no. 1, pp. 949-980, 2014.

[12] T. Salimans, J. Ho, X. Chen, S. Sidor and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," arXiv preprint, arXiv:1703.03864, 2017.

[13] E. Todorov, T. Erez and Y. Tassa, "Mujoco: A physics engine for model-based control." In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026-5033, 2012.

[14] S. M. Ross, Introductory statistics, Academic Press, 2017.

[15] D. S. Lemons, An introduction to stochastic processes in physics, JHU Press, 2003.

[16] E. M. Scheuer and D. S. Stoller, "On the generation of normal random vectors," Technometrics 4, no. 2, pp. 278-281, 1962.

[17] W. Lishuang, M. Zhao, E. Liu, K. Sun, and R. Cheng. "Tensorized NeuroEvolution of augmenting topologies for GPU acceleration," In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1156-1164, 2024.