Emergence of Strategies for Univariate Estimation-of-Distribution Algorithms with Evolved Neural Networks

Olivier Goudet, Adrien Goëffon, Frédéric Saubion, Sébastien Verel

Emergence of Strategies for Univariate Estimation-of-Distribution Algorithms with Evolved Neural Networks

Olivier Goudet
LERIA, Université d'Angers
Angers, France
olivier.goudet@univ-angers.com

Adrien Goëffon
LERIA, Université d'Angers
Angers, France
adrien.goeffon@univ-angers.com

Frédéric Saubion
LERIA, Université d'Angers
Angers, France
frederic.saubion@univ-angers.com

Sébastien Verel
LISIC, Univ. Littoral Côte d'Opale
Calais, France
verel@univ-littoral.fr

ABSTRACT

Our study focuses on the development of new Estimation of Distribution Algorithms (EDAs) with neuro-evolution for pseudo-Boolean optimization problems. We define a strategy for updating the frequency vector at each generation using a neural network, trained by an evolutionary algorithm. To evaluate the effectiveness of our approach, we conducted experiments using QUBO instances of different sizes, density matrices, and variable importance. The algorithm automatically discovered demonstrates its competitiveness with existing EDAs in the literature.

ACM Reference Format:
Olivier Goudet, Adrien Goëffon, Frédéric Saubion, and Sébastien Verel. 2024. Emergence of Strategies for Univariate Estimation-of-Distribution Algorithms with Evolved Neural Networks. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3638530.3654319

1 INTRODUCTION

The aim of Estimation-of-Distribution Algorithms (EDAs) [12, 13, 15] is to solve combinatorial optimization problems characterized by a search space $\mathcal{X}$ and an objective function $f : \mathcal{X} \to \mathbb{R}$. EDAs achieve this by building a probabilistic model on $\mathcal{X}$ through an evolutionary process.

This paper, as in the related literature we consider on EDAs, focuses on binary problems, where $\mathcal{X} = \{0, 1\}^n$ for a given size n. A simple but effective method in EDAs is to use a univariate model. More precisely, each variable x_i, where $1 \leq i \leq n$, in a solution $x \in \mathcal{X}$ is associated with a Bernoulli distribution parameterized by p_i. At each iteration, the probability vector $p = (p_1, \ldots, p_n)$ is updated to converge toward the best possible generative distribution of solutions.

From the simple initial framework presented in [17], various adaptations of EDAs have been explored, and general algorithms have been proposed to offer a comprehensive understanding of their fundamental principles and study their properties [2, 3]. Broadly speaking, an EDA can be conceptualized as the iterative generation of a set of sampled solutions based on an estimated distribution vector and the subsequent adjustment of this vector using a specified heuristic. It is worth noting that EDAs have shown good performance compared with other methods in many scenarios [14, 18].

Various univariate EDAs, with increasingly complex updating mechanisms, have been proposed in the literature. Introduced by [17], the univariate marginal distribution algorithm (**UMDA**) begins by generating λ solutions denoted $x[1], \ldots, x[\lambda]$ using a sampling process based on a p frequency vector. These solutions are then ranked in order of their fitness values to give $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$. Next, **UMDA** selects the μ best solutions and calculates the univariate marginal frequencies considering the $\tilde{x}[1], \ldots, \tilde{x}[\mu]$ individuals to update the frequency vector: $p(t) = \max(\frac{1}{n}, \min(1 - \frac{1}{n}, \frac{1}{\mu}\sum_{i=1}^{\mu} \tilde{x}[i]))$. **PBIL** [1] uses an additional intertia parameter $\rho \in [0, 1]$. Then, the only difference with **UMDA** concerns the update policy of the frequency vector, replaced by: $p(t) = \max(\frac{1}{n}, \min(1 - \frac{1}{n}, (1-\rho)\cdot p(t-1) + \frac{\rho}{\mu}\sum_{i=1}^{\mu} \tilde{x}[i]))$. In [2], the authors propose a general EDA algorithm, which we call **Linear-EDA**. It generalizes algorithms **UMDA**, or **PBIL** mentioned above. At the end of each generation t, **Linear-EDA** adjusts the frequency vector $p(t)$ by a linear recombination involving the prior probability value $p(t-1)$ and and all the λ individuals sampled at that time step, $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$: $p(t) = \max(\theta_\epsilon, \min(1 - \theta_\epsilon, \theta_0 \cdot p(t-1) + \sum_{i=1}^{\lambda} \theta_i \tilde{x}[i]))$, with $\theta_\epsilon \in [0, \frac{1}{2}[$ a calibrated frequency bound parameter. **Linear-EDA** is parameterized by a vector of parameters $\theta = (\theta_\epsilon, \theta_0, \theta_1, \ldots, \theta_\lambda)$ of size $\lambda + 2$.

Such frequency vector update relies on a linear recombination of the individuals in the population which can pose limitations, especially for complex pseudo-boolean problems. The main idea of this paper is to replace the linear update process with a non-linear update modeled by a neural network.

To train this neural network, we adopt a methodology aligned with neuro-evolution, a proven successful approach in various learning scenarios [10], especially in reinforcement learning problems featuring non-Markovian rewards [4], which exhibit similarities

Algorithm 1 Neuro-EDA with parameter $\lambda \in \mathbb{N}$ and neural network g_θ parameterized by a vector of parameters θ.

Input: an instance $(\mathcal{X}, f)$, a number of iteration T
$p(0) = (\frac{1}{2}, \ldots, \frac{1}{2}) \in [0,1]^n$
for $t = 1, 2, \ldots, T$ **do**
  for $i = 1, 2, \ldots, \lambda$ **do**
    $x[i] \sim Sample(p(t-1))$
  end for
  Sort the individuals into $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$ (in decreasing fitness order).
  %% Update the probability vector
  for $j = 1, 2, \ldots, n$ **do**
    $p(t)_j = \Phi(g_\theta([p(t-1)_j, \tilde{x}[1]_j, \tilde{x}[2]_j, \ldots, \tilde{x}[\lambda]_j]))$
  end for
end for
Output: the best solution found x^*

to our context. Utilizing the CMA-ES algorithm [8], recognized for its effectiveness in black-box continuous optimization problems, we harness its capabilities for parameter tuning [11] within our framework. To comprehensively evaluate the performance of our algorithm empirically, we employ PUBO_i [19], a versatile generator of unconstrained quadratic binary optimization problem (QUBO) instances.

2 DISCOVERING NEW ESTIMATION-OF-DISTRIBUTION ALGORITHMS WITH NEURO-EVOLUTION

In this section we first present a new EDA algorithm, called **Neuro-EDA**, using a neural network to update the frequency vector at each generation. Then, we describe the learning process we have chosen to train this neural network.

2.1 Neural network Estimation-of-Distribution algorithms

The proposed EDA, described in Algorithm 1, is a univariate EDA, which uses a feed-forward neural network, multi-layer perceptron (MLP), $g_\theta : \mathbb{R}^{\lambda+1} \to \mathbb{R}$ to update the frequency vector. This neural network is parametrized by a vector θ (neural network weights). $\Phi : \mathbb{R} \to]0,1[$ corresponds the distribution function of the centered reduced normal distribution employed in probit models. Utilizing the probit activation function, bounded between 0 and 1, offers the advantage of never entirely attaining probability values of 0 or 1. Conversely, it enables us to approach these boundaries as closely as needed.

Note that in this algorithm, the same neural network g_θ is applied to compute $p(t)_j$ for each variable j, taking as input $p(t-1)_j$, as well as the values $\tilde{x}[1]_j, \tilde{x}[2]_j, \ldots, \tilde{x}[\lambda]_j$. Therefore, this algorithm is invariant to permutation of the n variables of the instance to be solved, like in the classical EDAs presented in Section 1.

2.2 EDA strategy Learning for a Set of Instances

The **Linear-EDA** of [2] and **Neuro-EDA** parameterized by θ, can be considered as stochastic functions π_θ. For each instance $(\mathcal{X}, f)$ drawn from $PUBO_i(n, m, d, \alpha)$, π_θ will be launched during T iterations. At the end of this search, it returns the solution $x^* = \pi_\theta(\mathcal{X}, f, T)$ with the maximum fitness value $x^* \in \mathcal{X}$ encountered during the sampling process. An algorithm π_θ can be viewed as a sampling policy in $\mathcal{X}$.

To assess the performance of the stochastic policy π_θ on random instances generated as $(\mathcal{X}, f) = PUBO_i(n, m, d, \alpha)$ and solved with a budget of T iterations (corresponding to $T \times \lambda$ calls to the objective function f), we propose to estimate the quantity:

$F(\pi_\theta, \mathcal{X}, f, T) := \mathbb{E}_{(\mathcal{X},f)\sim PUBO_i(n,m,d,\alpha)}[f(\pi_\theta(\mathcal{X}, f, T))]$.

The judicious choice of the values of θ is paramount in defining an efficient EDA algorithm. Given an EDA π_θ, the generator of instances $PUBO_i(n, m, d, \alpha)$ and a budget of T generations, the optimization goal is to find the vector of parameters θ in real-valued $\mathbb{R}^{|\theta|}$ search space that maximize an estimated score $\bar{F}(\pi_\theta, \mathcal{X}, f, T)$ of $F(\pi_\theta, \mathcal{X}, f, T)$, computed as an average score for a finite set of instances drawn from the generator. Learning this set of parameters with gradient descent techniques is impossible in this case due to the nonderivable sampling of the binary variables in the solutions $x^* \in \mathcal{X}$ produced by the EDA π_θ. Therefore, to solve this problem, we propose to use black-box optimization evolutionary algorithm, such as the covariance matrix adaptation evolution strategy (CMA-ES) [7, 9], which was already successfully applied to learn neural networks for episodic reinforcement learning [10].

3 EXPERIMENTS

In this section, we first seek to evaluate the efficiency of the proposed **Neuro-EDA** strategy. This evaluation involves a comparative analysis against the following existing EDAs: **UMDA** [17], **PBIL** [1] and **Linear-EDA** [2]. The assessment is conducted across QUBO instances generated by the PUBO_i generator, each configured with three distinct parameters. The second objective is to study the emerging strategy discovered by **Neuro-EDA** at the end of the evolutionary process.

3.1 Experimental settings

In this section, we first describe the generation of independent QUBO instances. We depict the EDAs' configurations and training parameters.

3.1.1 QUBO datasets generation . In this study, the algorithms are evaluated on instances of the unconstrained quadratic binary optimization problem (QUBO), which is a single-objective pseudo-Boolean optimization problem with quadratic interactions between binary variables. QUBO problems are general, as many NP-hard and NP-complete combinatorial optimization problems can be easily translated into this model [5, 16].

The objective function $f : \{0,1\}^n \to \mathbb{R}$ to maximize is defined by: $f(x) = x^t Q x$ where Q is a real matrix of dimension $n \times n$ and x^t is the transposed vector of x. We assume that Q is unknown and that we are in a black-box optimization scenario.

In our study the QUBO instances are generated with the PUBO_i generator [19] able to bring QUBO with different properties. We consider instances of size $n \in \{32, 64, 128\}$ for the training, validation, and test phases. The number of sub-functions m tunes the density of the matrix (16% and 43% for uniform instances, respectively, for the two values of m). Three types of interaction mechanisms are

used between variables. The instances I_{uni} have no specific important variables, *i.e.* I_{uni} instances are similar to QUBO problems with a random matrix. The instances I_{imp} have important variables: the marginal probability of having important variables is equal to $d = 10$ times the probability of non-important ones. Additionally, for the I_{ic} instances, the value of the co-appearance parameter is high, the selection of important variables is not independent, and the selection of important variables is concentrated.

For each tuple of parameter values, we generate a training set $\mathcal{D}_{train}$ including 1,000 independent instances, a validation set $\mathcal{D}_{valid}$ containing 100 instances, and an additional test set $\mathcal{D}_{test}$ with 100 instances.

3.1.2 EDA configurations. All EDAs presented in this paper generate $\lambda = 20$ solutions at each iteration. A sensitivity analysis regarding this critical parameter (which is not presented here due to space limitations) has been conducted. It reveals that increasing the value of λ beyond 20 does not significantly improve results for all EDAs and does not change the ranking of the different strategies among themselves for the instances of size up to $n = 128$ considered in this paper.

Concerning **Neuro-EDA**, we consider neural networks with one hidden layer of $h = 20$ neurons and a sigmoid activation function. We deliberately chose a fairly small neural network size to limit its training time.

3.2 EDA training and validation phase

For each of the 18 configurations of the PUBO_i generator described in Section 3.1.1, we run two different training processes:

(1) For the baseline EDAs, namely **UMDA** and **PBIL**, we performed a grid search to seek the best set of parameter values on each type of training set, with parameters $\mu \in \{1, \ldots, \lambda\}$ and $\rho \in \{0, 0.1, 0.2, \ldots, 1\}$.

(2) For the **Linear-EDA** and **Neuro-EDA** variants, we run 10 independent training processes with CMA-ES, as described in Section 2.2. During this training phase, at each generation of CMA-ES, a batch of 100 training instances is randomly sampled without replacement in $\mathcal{D}_{train}$ and each configuration of the vector of parameters θ (which are indeed the individuals of the CMA-ES population) are evaluated on these same 100 instances. At each generation, the EDA configuration obtaining the best score on average on these 100 training instances is evaluated on the 100 instances of the validation set $\mathcal{D}_{valid}$.

3.3 Test phase

In this phase, we perform evaluations to assess if the best strategies selected for each configuration of the validation set $\mathcal{D}_{valid}$ generated by PUBO_i, are still performing well on the corresponding test set $\mathcal{D}_{test}$, which has been independently sampled with the same PUBO_i configuration.

Table 1 summarizes the average scores obtained by the different EDAs' strategies on the test sets $\mathcal{D}_{test}$ with 100 independent restarts for each instance of the same type (10,000 runs), whose scores are of the same order of magnitude. We also report the result obtained by a classic Tabu search algorithm (**TS**) with aspiration criterion [6]. In **TS**, after each flip, a move is set tabu during $\Delta = \beta + R$ iterations where R is a random integer in $[1, 10]$ and β is a hyperparameter calibrated in the range $\{1, \ldots, n\}$ for each specific distribution of instances. We give **TS** the same budget of $T \times \lambda$ calls to the objective fitness function, as for all the EDAs.

Significantly better results obtained by an algorithm compared to the others are underlined. The significance test is a Student t-test with p-value 0.001 and Bonferroni correction for multiple comparisons. The normality condition required for this test was first confirmed using a Shapiro-Wilk statistical test on the empirical distribution of the 10,000 scores obtained by each strategy.

In Table 1, we observe that **Neuro-EDA** performs better than all other versions for datasets of size $n = 64$ and $n = 128$, with $I = I_{imp}$ and $I = I_{ic}$, corresponding to instances with more complex interactions between variables, while **Linear-EDA** is more robust for the set of instances with simpler fitness landscapes (*i.e.* corresponding to $I = I_{uni}$).

We also observe that the best proposed EDA strategies are better than **TS** for all types of instances, except for the set $(128, 0.2, I_{uni})$ that exhibits less rough fitness landscapes. This highlights that these EDA strategies can be competitive with classic local search algorithms such as **TS**, solving QUBO as a black-box problem.

3.4 Neural network EDA emerging strategy

To interpret the strategy employed by **Neuro-EDA** (Algorithm 1), we calculate the partial derivatives of the neural network g_θ with respect to the probability vector $p(t-1)$ and the λ values associated with individuals sorted by decreasing fitness, all evaluated at the point $(0.5, 0.5, \ldots, 0.5)$. This analysis is conducted across all the top-performing neural networks utilized within **Neuro-EDA** and trained on various instance types as previously outlined. The distributions of the partial derivatives are depicted in Figure 1. The initial blue boxplot illustrates sensitivity to $p(t-1)$, which often manifests as a negative value, suggesting a tendency to promote diversity by slightly deviating from the previous generation's variable sampling. The subsequent red boxplot reflects sensitivity to the variables of the population's best individual, typically showing a notably positive value, indicating a strong focus on the most promising individual. Given that negative gradient values are associated with the worst individuals, it can be inferred that a relevant strategy involves avoiding the same variable assignments as the worst individuals in the preceding iteration $t-1$ when sampling new individuals in iteration t.

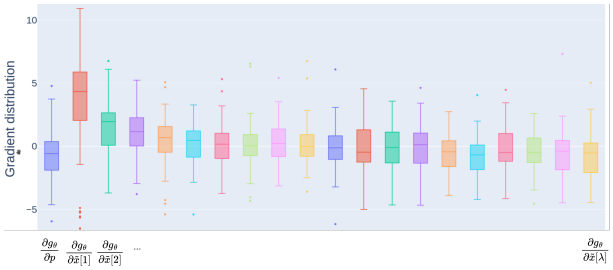

Figure 1: Distribution of gradients of the learned neural networks g_θ used by Neuro-EDA with respect to its inputs.

Instances			Methods				
n	m	I	**TS**	**UMDA**	**PBIL**	**Linear-EDA**	**Neuro-EDA**
32	0.05	I_{uni}	65.47	64.57	64.58	**66.50**	66.49
32	0.05	I_{imp}	46.71	46.38	46.41	**46.82**	**46.82**
32	0.05	I_{ic}	42.02	41.91	41.84	42.06	**42.07**
32	0.20	I_{uni}	145.84	143.11	143.12	147.45	**147.46**
32	0.20	I_{imp}	107.51	106.13	106.39	108.15	**108.23**
32	0.20	I_{ic}	99.64	98.38	98.53	**100.79**	100.78
64	0.05	I_{uni}	208.61	202.37	202.37	**209.63**	**209.63**
64	0.05	I_{imp}	148.44	146.44	146.36	152.01	**152.06**
64	0.05	I_{ic}	140.18	140.31	140.26	144.87	**145.17**
64	0.20	I_{uni}	446.72	428.73	429.68	**444.64**	443.38
64	0.20	I_{imp}	333.97	324.48	326.52	339.72	**340.77**
64	0.20	I_{ic}	318.87	315.74	315.94	325.39	**328.22**
128	0.05	I_{uni}	624.86	605.48	605.42	**624.90**	621.59
128	0.05	I_{imp}	458.19	447.86	448.00	467.06	**470.22**
128	0.05	I_{ic}	433.05	427.48	428.40	440.50	**445.72**
128	0.20	I_{uni}	**1282.02**	1239.41	1239.30	1278.41	1271.81
128	0.20	I_{imp}	964.81	938.39	939.30	979.52	**991.69**
128	0.20	I_{ic}	950.41	937.26	936.68	962.64	**978.00**

Table 1: Average score (fitness values) obtained by different EDAs on test sets. The best scores are in bold. Underlined values correspond to significant better results (t-test with p-value 0.001 and Bonferroni correction for multiple comparisons).

4 CONCLUSION

We proposed a new framework to discover univariate EDAs for pseudo-boolean optimization problems. We highlighted that a neural network strategy can be competitive with existing EDAs of the literature for different types of problem instances. In future work, we would like to investigate the impact of more complex neural network architectures for the design of new univariate strategies, such as convolutional graph neural networks, which could potentially take into account the interaction graph between variables induced by the Q matrix in white-box scenarios.

ACKNOWLEDGMENT

This work was granted access to the HPC resources of IDRIS (Grant No. AD010611887R1) from GENCI. The authors would like to thank the Pays de la Loire region for its financial support for the Deep Meta project (Etoiles Montantes en Pays de la Loire). The authors also acknowledge ANR – FRANCE (French National Research Agency) for its financial support of the COMBO project (PRC - AAPG 2023 - Axe E.2 - CE23). We are grateful to the reviewers for their comments.

REFERENCES

[1] Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning (1994)
[2] Doerr, B., Dufay, M.: General univariate estimation-of-distribution algorithms. In: International Conference on Parallel Problem Solving from Nature. pp. 470–484. Springer (2022)
[3] Doerr, B., Krejca, M.S.: The univariate marginal distribution algorithm copes well with deception and epistasis. In: Paquete, L., Zarges, C. (eds.) Evolutionary Computation in Combinatorial Optimization - 20th European Conference, EvoCOP 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12102, pp. 51–66. Springer (2020)
[4] Gaon, M., Brafman, R.: Reinforcement learning with non-markovian rewards. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 3980–3987 (2020)
[5] Glover, F., Kochenberger, G., Hennig, R., Du, Y.: Quantum bridge analytics I: a tutorial on formulating and using QUBO models. Annals of Operations Research pp. 1–43 (2022)
[6] Glover, F., Laguna, M.: Tabu Search, pp. 2093–2229. Springer US, Boston, MA (1998)
[7] Hansen, N., Arnold, D.V., Auger, A.: Evolution strategies. Springer handbook of computational intelligence pp. 871–898 (2015)
[8] Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. **11**(1), 1–18 (2003)
[9] Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2), 159–195 (2001)
[10] Heidrich-Meisner, V., Igel, C.: Neuroevolution strategies for episodic reinforcement learning. J. Algorithms **64**(4), 152–168 (2009)
[11] Jedrzejewski-Szmek, Z., Abrahao, K.P., Jedrzejewska-Szmek, J., Lovinger, D.M., Blackwell, K.T.: Parameter optimization using covariance matrix adaptation - evolutionary strategy (cma-es), an approach to investigate differences in channel properties between neuron subtypes. Frontiers Neuroinformatics **12**, 47 (2018)
[12] Juels, A., Baluja, S., Sinclair, A.: The equilibrium genetic algorithm and the role of crossover. Unpublished manuscript (1993)
[13] Krejca, M.S., Witt, C.: Theory of estimation-of-distribution algorithms. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation - Recent Developments in Discrete Optimization, pp. 405–442. Natural Computing Series, Springer (2020)
[14] Larranaga, P., Bielza, C.: Estimation of distribution algorithms in machine learning: A survey. IEEE Transactions on Evolutionary Computation (2023)
[15] Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. Genetic Algorithms and Evolutionary Computation, Springer (2002)
[16] Lodewijks, B.: Mapping NP-hard and NP-complete optimisation problems to quadratic unconstrained binary optimisation problems. arXiv preprint arXiv:1911.08043 (2019)
[17] Mühlenbein, H., Paass, G.: From recombination of genes to the estimation of distributions i. binary parameters. In: Voigt, H., Ebeling, W., Rechenberg, I., Schwefel, H. (eds.) Parallel Problem Solving from Nature - PPSN IV, International Conference on Evolutionary Computation. The 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26, 1996, Proceedings. Lecture Notes in Computer Science, vol. 1141, pp. 178–187. Springer (1996)
[18] Pelikan, M., Hauschild, M., Lobo, F.G.: Estimation of distribution algorithms. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 899–928. Springer Handbooks, Springer (2015)
[19] Tari, S., Verel, S., Omidvar, M.: Pubo$_i$: A tunable benchmark with variable importance. In: Cáceres, L.P., Verel, S. (eds.) Evolutionary Computation in Combinatorial Optimization - 22nd European Conference, EvoCOP 2022, Held as Part of EvoStar 2022, Madrid, Spain, April 20-22, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13222, pp. 175–190. Springer (2022)