Introducing Generative adversarial networks on Estimation of distribution algorithm to solve permutation-based problems

Sami Lemtenneche
Kasdi Merbah University
Ouargla, Algeria
lemtenneche.sami@univ-ouargla.dz

Abdelhakim Cheriet
Kasdi Merbah University
Ouargla, Algeria
abdelhakim.cheriet@univ-ouargla.dz

Abdallah Bensayah
Kasdi Merbah University
Ouargla, Algeria
bensayah.abdallah@univ-ouargla.dz

2022 International Symposium on iNnovative Informatics of Biskra (ISNIB) | 979-8-3503-2065-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISNIB57382.2022.10075720

***Abstract*—As a subclass of evolutionary algorithms, estimation of distribution algorithms (EDAs) have found widespread use in a variety of optimization problems with impressive performance. In each generation, they build a probabilistic model representing the promising individuals, and the next generation constructs by sampling this model. Constructing and sampling the probabilistic model is a real challenge in developing such algorithms. Generative adversarial networks (GANs) are a popular kind of generative model that has been widely adopted due to their ability to generate new samples that closely match the distribution of training data. However, research on how GANs handle permutation spaces is lacking. We suggest a novel Estimation of Distribution Algorithm (EDA) that uses GANs as a probabilistic model estimator to advance this subject. in order to preserve the information captured from the selected individuals, those promising individuals were represented by a one-hot matrix, then used to train GANs. The proposed algorithm is tested to solve two permutation problems: The Travelling Salesman Problem (TSP) and Permutation Flowshop Scheduling Problem (PFSP). The Experimental results show that the proposed algorithm can obtain the optimal solution in some instances and a near-optimal in others.**

***Index Terms*—Deep Generative Models, Permutation, Travelling Salesman Problem, Estimation of distribution algorithms.**

I. Introduction

Estimation of Distribution Algorithms (EDAs) [1] [2]is model-based evolutionary algorithms (MBEAs). EDA, Probabilistic Model Building Genetic Algorithms (PMBGAs) is another name for this algorithm [3], is regarded as a genetic algorithm (GAs) adaptation [4]. EDAs use two main steps in every generation to investigate the search space. Building a probabilistic model of best solutions is the first phase, and sampling this model to create the next generation is the second. Various optimization problems were effectively solved with EDAs. [5]. Lately, much research has been done on how to use EDAs to solve optimization problems where the solution is written as a permutation.

Furthermore, generative adversarial networks (GANs) [6] are very successful in learning the probability distribution and producing new data that closely resemble the training data distribution. That encouraged researchers to propose many variants of the original GAN [7] [8]. In this work, we propose to use GANs inside of the estimation of distribution algorithms in such a way as to replace the probabilistic model building and sampling phases. Also, an alternative method of individual representation is used in this paper, in which a permutation is transformed into a one-hot matrix.

The remainder of this paper is structured as follows: the second section introduces the main concepts of the Estimation of Distribution Algorithms and Generative Adversarial Networks. In section III, we cite some related works. Sections IV and V describe our proposal work and discussion about experimental results.

II. Background

A. Estimation of distribution algorithms

In 1996, Muhlenbein et al. [1] introduce the Estimation of Distribution Algorithm (EDA) as a new kind of Evolutionary Algorithms (EAs). Then they enriched after that by Larranaga et al. [9] and Pelikan [5]. The EDAs use neither crossover nor mutation operations. Instead, they try to learn the distribution of variables and features by constructing an explicit probabilistic model of fittest solutions for each generation. After the probabilistic model has been created, the sampling step will create the next generation and guide the algorithm to look in those directions for more feasible solutions. The evolutionary process is repeated until a criterion stop being met. The Algorithm 1 shows up the general scheme of EDAs.

Algorithm 1: The general outline of EDAs

$P_0 \leftarrow$ create M individuals at random;
repeat
  Selected set $\leftarrow$ Select $N < M$ solutions;
  $D \leftarrow$ Learn the probability distribution of selected set;
  Offspring $\leftarrow$ Sampling D to get a new offspring;
  $P_{i+1} \leftarrow$ Update P_i;
until *Termination criterion is met*;

B. Generative Adversarial Networks

in 2014, a new type of generative model, called generative adversarial networks (GANs) [6] was first proposed by Ian Goodfellow et al. A generative adversarial network is a model that typically uses two deep neural networks together, **discriminator** D and a **generator** G.

New data is produced by the generator using a random vector as input. Contrarily, the discriminator takes a sample from data training (labeled as genuine) or a constructed sample from the generator (labeled as false) and estimates the probability that this input originated from training data, as opposed to a generator, see Fig1.

The word "adversarial" refers to a situation in which the discriminator and the generator compete against one another to attain their respective objectives.

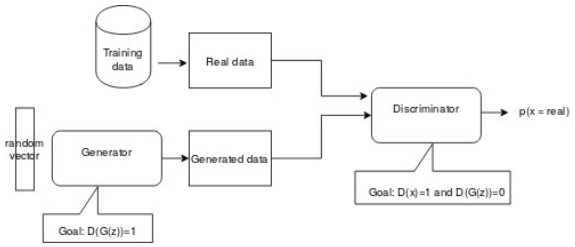

Fig. 1. GANs architecture

The Discriminator learn to maximize objective function such that $D(x)$ is near to 1 (real) and $D(G(z))$ is near to 0 (fake)

$$\min_G \max_D V(D,G) = E_{xp_{data}(x)}[\log(D(x))] + E_{zp_z(z)}[\log(1 - D(G(x)))] \quad (1)$$

The goal of the generator is to reduce the objective in such a way that the value of $D(G(z))$ is relatively near to 1. If the discriminator is confused, and the two players (D and G) arrive at a state of Nash equilibrium, the GAN model is converged.

C. permutation based problems

The permutation-based problems are a kind of NP-hard [10] [11] optimization problems in which the permutation represents the solution.

In the combinatorial optimization community, the term "permutation" refers to a set of n natural numbers (1,..., n) with exclusivity mutual occurrence for each number. Usually the permutation denoted by σ,π or τ, i.e. we use $\sigma(i) = j$ to say that element j is in position i.

1) Travelling Salesman Problem (TSP): The Travelling Salesman Problem (TSP) is the problem of determining the shortest path to travel n various cities, visiting each just once, then circling back to the starting point. Every possible combination of cities that may be visited is represented by a permutation, $sigma$, where the value $sigma(i) = j$ indicates that city j is visited during the $i - th$ step.

For example, in a problem of 4 cities, the solution $\sigma_1 = \{3, 2, 4, 1\}$ described the tour departing from city 3, then going to 2, 4, and 1, and finally coming back to 3. Formally, given a matrix $D = [d_{i,j}]$ with the distances between all the pairs of cities, The objective function, denoted by f, represents the accumulation of all of the distances that separate every two consecutive cities in the sequence indicated by σ:

$$f(\sigma) = \sum_{i=2}^{n} d_{\sigma(i-1),\sigma(i)}$$

Since we are working under the assumption that the starting point of the tour is not predetermined, the Traveling Salesman Problem (TSP) is a problem with symmetric solutions. This is due to the fact that $2n$ distinct permutations can represent each tour for symmetric instances, whereas n distinct permutations can represent each tour for asymmetric instances. For example, $\sigma_1 = \{1, 3, 2, 4, 5\}$ represents the same solution of $\sigma_2 = \{5, 4, 2, 3, 1\}$, because $f(\sigma_1) = f(\sigma_2)$.

2) Permutation Flowshop Scheduling Problem (PFSP): The PFSP is a problem of scheduling n tasks on a set of m machines. Each task is composed of m operations, the j^{th} operation must be executed by machine j. A job can take her turn on the machine j, if the machine j is available, and its $j - 1^{th}$ operation has been completely processed by machine $j - 1$. The objective of the optimization of this problem is to find a permutation of jobs that minimizes the performing time of all n jobs, it means the solution is represented by a permutation of length n, and this order is the same in all machines. The minimization of the makespan objective function can be formulated as follows:

$$F(\sigma) = C_{\sigma_n,m}$$

In equation above, $C_{\sigma_n,m}$ signifies the finishing time of last task σ_n on last machine *m*, which depends on processing time of previous jobs $\sigma_1, ..., \sigma_{n-1}$. The other objective function is Total Flow Time (TFT), it can formulated as:

$$F(\sigma) = \sum_{i=1}^{n} C_{\sigma_i,m}$$

III. RELATED WORK

In the literature, we come across certain EDAs that are intended especially to solve problems in which the solution is encoded by a permutation. In order to tackle permutation-based problems, the authors in [12]–[14] introduce the probability models on rankings such as Mallows and Plackett-Luce models in EDAs. The Generalized Mallows-EDA [13], and PlackettLuce-EDA [14] obtain good results for PFSP and LOP, respectively. Other algorithms include Edge Histogram Based Sampling Algorithm (EHBSA) [3]and Node Histogram Based Sampling Algorithm (NHBSA) [15], both with template and without template sampling, which also proves their good performance in solving permutation problems.

On another side, a few papers introduce GANs in EDAs. In [16], the author proposes to use GANs in EDA and test it to solve several combinatorial problems (no permutation problems). According to the author, their proposed algorithms

don't achieve the best results. Another attempt to use GAN in EDA is proposed in [17] by combining GAN with a genetic algorithm (GA) and testing it on PFSP. In their work, the GA remains the algorithm's primary component, while the GAN is utilized to assist the GA to avoid the local optimum and increase population variety by injecting produced samples into the population. The author states that the hybrid GAN-GA is more effective than conventional GA.

IV. Proposal

In this part, we'll go over the entire process of our suggested algorithm, starting with the individual representation and ending with the offspring production method.

A. Representation of individuals

The natural representation of a permutation is a vector of discrete numbers with mutual exclusivity, as defined in the above section. Since the generator produces real values as its outputs, it is obvious that a permutation is not appropriate for training GANs. so we need to transform these outputs into data suitable to train our models and, at the same time, conserve the information of the population distribution. In our proposed algorithm, we use a one-hot matrix of $(n * n)$ to represent the individuals (permutation) in such a way the rows represent the positions in the permutation (solution), and the columns represent the number in these positions. For example the permutation $\pi(2, 3, 1, 4)$ represents with the following matrix:

0	1	0	0
0	0	1	0
1	0	0	0
0	0	0	1

This representation allows us to avoid the problems of the generation of the permutation directly, as discussed above.

B. Proposed algorithm

The fundamental outline of the suggested method is shown in figure 2. The initial population and GANs parameters are initialized randomly. Then a set of individuals are selected according to their fitness and transformed into a matrix as the individual representation explained above. The selected individuals are labeled as real data and used to train the GANs with the generated data from the generator (fake data). The trained generator is then used to generate the next generation after a certain number of epochs.

During the training of the models, the generator can learn and estimate the distribution of the selected individuals (the real data), this phase replaces the model building in the standard EDAs, in fact, the GANs learn to build an implicit probabilistic model of real data. During the sampling phase, a set of matrices that imitate the distribution of the training ones are produced using the trained generator. then we apply the opposite operation to get the new individuals from the generated matrices.

Algorithm 2: The proposed Algorithm

$P_0 \leftarrow$ create M individuals at random;
for $gen \leftarrow 0$ **to** $maxgeneration$ **do**
  Selected set $\leftarrow$ Select $N < M$ best solutions;
  Real Data $\leftarrow$ Transform Selected set to matrices;
  Train the GAN with Real Data (Matrices);
  Offspring $\leftarrow$ Generate M new offspring by updated Generator;
  $P_{i+1} \leftarrow$ Update P_i;

C. Offspring reproduction strategy

In our proposed algorithm, the offspring are generated exclusively by the generator. The generator transform a random input z sampled from normal distribution to a matrix $L \in R^{N*N}$, each row of L represents a vector of probabilities to get a number in each position, for example, the first row of the generated matrix is: $(0.05, 0.1, 0.8, 0.2)$ will be $(0, 0, 1, 0)$ in the one-hot matrix. Next, we apply the inverse operation of individual representation to get offspring. Then a 2-opt algorithm is applied to the reproduced solutions to enhance the obtained results. The 2-opt local search method stop the search at the first improvement, and return the enhanced generated individual.

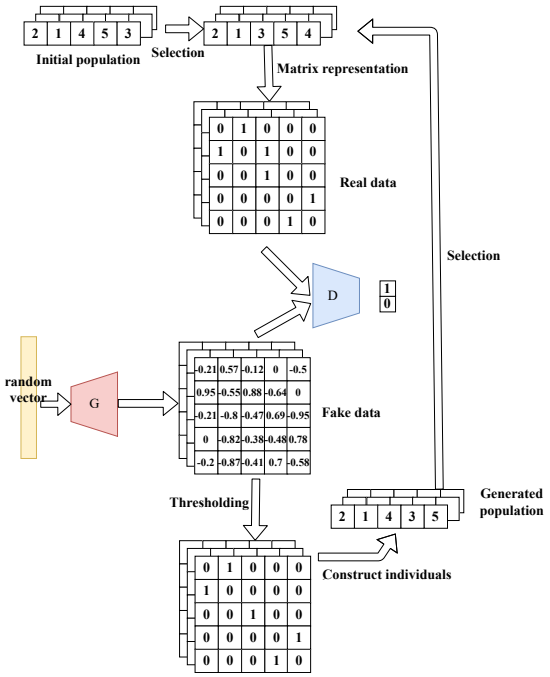

Fig. 2. General scheme of the proposed algorithm

V. Experiments and Results

A. Experiments

In order to test our proposed algorithm, we carried out a set of experiments using small instances of two well-known

permutation problems, the TSP instances from TSPLIB [18] and the PFSP from the Eric Taillard's benchmark's [19].

Problem	Used instances
TSP	fri26, bays29, gr17, gr21, gr24
PFSP	20 x 5 , 20 x 10, two first instances for both

The GANs models are implemented using the Pytorch library. Table I presents the experimentally best-found parameters of the GANs.

Parameter	Value
epochs	50
D hidden layers	1
G hidden layers	2
Learning rate	1e-3
G output activation function	Tanh
D output activation function	Sigmoid
Hidden layers activation function	Relu
Batch size	32

TABLE I
THE ARCHITECTURE AND PARAMETERS OF MODELS

B. Results and discussion

This part demonstrates the experimental results of the proposed algorithm. The II presents the Min, Max, and Mean after 10 runs of the proposed method for each chosen instance. The results marked in bold represent the best know solution.

Problem	Instance	Minimum	Maximum	Mean
TSP	fri26	**937**	1019	977
	bays29	2034	2104	2065
	gr17	2090	2142	2101.1
	gr21	**2707**	2801	2755
	gr24	**1272**	1346	1305.6
PFSP	tai20-5-0	1297	1307	1297.9
	tai20-5-1	1367	1395	1380.9
	tai20-10-0	1654	1709	1683.1
	tai20-10-1	1740	1781	1762

TABLE II
THE RESULTS OF EXPERIMENTAL STUDY

The suggested method is more effective on the TSP when the optimal solution is found in 3 used benchmarks from 5, and a near-optimal for the others. For the PFSP the results are less competitive but it is still acceptable. Table III shows the results obtained with our proposed algorithms with the one-hot matrix representation compared with our previous work [20] with two variants with 2-opt and without 2-opt. We can observe that the results obtained by our proposed algorithm were better with the all used TSP instances.

From the comparison of the results, we can say that the matrix representation enhances the information captured by the GANs from the population distribution.
The major disadvantage encountered is the high computational time caused by the number of parameters trained in the GANs.

Instances	Matrix representation	Random key representation	
		With 2-opt	Without 2-opt
fri26	**937**	1110	1334.6
bays29	2034	2806.4	4386.8
gr17	2090	2128.4	2808.6
gr21	**2707**	3555.5	5491.6
gr24	**1272**	/	/

TABLE III
COMPARISON OF THE OBTAINED RESULTS WITH THE RANDOM KEY REPRESENTATION AND MATRIX REPRESENTATION

VI. CONCLUSION

In this study, a novel EDA based on generative adversarial networks was developed for solving permutation-based problems. Designing an estimation of distribution algorithm depends on modeling the probability distribution of the selected solutions in each generation and then sampling from this model to generate a new population. Encouraged by the capability of a generative adversarial networks to estimate the distribution in the continuous space, we used it to learn the distribution of the best-selected individuals and to produce the next generation. However, no studies have examined how well GANs perform in the permutation space. Our novel method trains generative adversarial networks using a set of matrices that represents the current population. After the training, the updated GAN is used to generate a new solution that directs the algorithm during the optimization phase. The goal of the experiments was to find out how closely the GAN could produce samples that shared the same characteristics as the inputs (permutations). Our experimental results are encouraging, but further research is required to figure out how to make the GAN produce a more diversified population.
In further work, we want to test our suggested algorithm to large instance sets and investigate other GANs variants such as Wasserstein-GANs. We would also like to employ a Variational Autoencoder (VAE) instead of GAN.

REFERENCES

[1] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Parallel Problem Solving from Nature — PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 178–187.

[2] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[3] S. Tsutsui, M. Pelikan, and D. Goldberg, "Using edge histogram models to solve permutation problems with probabilistic model-building genetic algorithms," *IlliGAL Report*, vol. 2003022, no. 2003022, 2003.

[4] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[5] M. P. Mark Hauschild, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, 2011.

[6] I. J. Goodfellow, J. Pouget-abadie, M. Mirza, B. Xu, and D. Warde-farley, "Generative Adversarial Nets," pp. 1–9, 2014.

[7] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[9] Pedro Larranaga Jose A. Lozano, *ESTIMATION OF DISTRIBUTION ALGORITHMS: A New Tool for Evolutionary Computation*. Springer Sciencc+Business Media New York, 2002.
[10] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, ser. Mathematical Sciences Series. W. H. Freeman, 1979.
[11] J. V. a. Bernhard Korte, *Combinatorial Optimization: Theory and Algorithms*, ser. Algorithms and Combinatorics 21. Springer Berlin Heidelberg, 2008.
[12] J. Ceberio, A. Mendiburu, and J. Lozano, "Introducing the mallows model on estimation of distribution algorithms," vol. 7063, 11 2011, pp. 461–470.
[13] ——, "Kernels of mallows models for solving permutation-based problems," 07 2015, pp. 505–512.
[14] J. Ceberio, "The plackett-luce ranking model on permutation-based optimization problems," 06 2013.
[15] S. Tsutsui, "Node Histogram vs. Edge Histogram: A Comparison of Probabilistic Model-Building Genetic Algorithms in Permutation Domains," *2006 IEEE International Conference on Evolutionary Computation*, no. 2006009, pp. 1939–1946, 2006.
[16] M. Probst, "Generative adversarial networks in estimation of distribution algorithms for combinatorial optimization," 09 2015.
[17] M. Chen, R. Yu, S. Xu, Y. Luo, and Z. Yu, "An improved algorithm for solving scheduling problems by combining generative adversarial network with evolutionary algorithms," 10 2019, pp. 1–7.
[18] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library," *INFORMS Journal on Computing*, vol. 3, no. 4, pp. 376–384, November 1991. [Online]. Available: https://ideas.repec.org/a/inm/orijoc/v3y1991i4p376-384.html
[19] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993, project Management anf Scheduling. [Online]. Available: https://www.sciencedirect.com/science/article/pii/037722179390182M
[20] S. Lemtenneche, A. Cheriet, and B. Abdellah, "Permutation-based optimization using a generative adversarial network," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 159–160.