An improved Estimation of Distribution Algorithm for Solving Constrained Mixed-Integer Nonlinear Programming Problems

1st Daniel Molina Pérez
CIDETEC
Instituto Politécnico Nacional
Ciudad de México, México
dmolinap1800@alumno.ipn.mx

2nd Edgar Alfredo Portilla-Flores
UPIIT
Instituto Politécnico Nacional
Tlaxcala, México
aportilla@ipn.mx

3rd Efrén Mezura-Montes
Artificial Intelligence Research Center
University of Veracruz
Veracruz, México
emezura@uv.mx

4th Eduardo Vega-Alvarado
CIDETEC
Instituto Politécnico Nacional
Ciudad de México, México
evega@ipn.mx

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870338

***Abstract*—In a mixed-integer nonlinear programming problem, integer restrictions divide the feasible region into discontinuous feasible parts with different sizes. Evolutionary Algorithms (EAs) are usually vulnerable to being trapped in larger discontinuous feasible parts. In this work, an improved version of an Estimation of Distribution Algorithm (EDA) is developed, where two new operations are proposed. The first one establishes a link between the learning-based histogram model and the ε-constrained method. Here, the constraint violation level of the ε-constrained method is used to explore the smaller discontinuous parts and form a better statistical model. The second operation is the hybridization of the EDA with a mutation operator to generate offspring from both the global distribution information and the parent information. A benchmark is used to test the performance of the improved proposal. The results indicated that the proposed approach shows a better performance against other tested EAs. This new proposal solves to a great extent the influence of the larger discontinuous feasible parts, and improve the local refinement of the real variables.**

***Index Terms*—estimation of distribution algorithm, evolutionary algorithms, integer restriction handling, mixed integer nonlinear programming.**

I. Introduction

A large number of engineering optimization problems fall into the category of Mixed-Integer Nonlinear Programming (MINLP) [1], that addresses nonlinear problems including continuous, integer and/or discrete variables. The mathematical model of a MINLP problem can be defined by (1):

$$\begin{aligned} \text{minimize } & f(\mathbf{x},\mathbf{y}) \\ \text{subject to: } & g_i(\mathbf{x},\mathbf{y}) \leq 0, && i = 1,...,n \\ & h_j(\mathbf{x},\mathbf{y}) = 0, && j = 1,...,m \\ & x_k^{lb} \leq x_k \leq x_k^{ub}, && k = 1,...,n_1 \\ & y_q^{lb} \leq y_q \leq y_q^{ub}: && \text{integer, } q = 1,...,m_1 \\ & [\mathbf{x},\mathbf{y}] \in \Omega \end{aligned} \tag{1}$$

where $f(\mathbf{x},\mathbf{y})$ is the objective function, $\mathbf{x}$ is a vector of continuous variables, $\mathbf{y}$ is a vector of integer variables, x_k^{lb} and x_k^{ub} are the lower and upper bounds of x_k, respectively, y_q^{lb} and y_q^{ub} are the lower and upper bounds of y_q, respectively, Ω is the decision variable space, $g_i(\mathbf{x},\mathbf{y})$ is the ith inequality constraint, and $h_j(\mathbf{x},\mathbf{y})$ is the jth equality constraint.

In a MINLP problem, the constraints define the feasible region, whereas the integer restrictions divide the feasible region into discontinuous feasible parts with different sizes [2]. This fact is shown in Fig. 1, where x is a continuous variable, and y is an integer variable. The shaded area is the feasible region defined by the constraints, and the red lines are the discontinuous feasible parts that also satisfy the integer restrictions.

Diverse classical methods (branch and bound, cutting planes, outer approximation, among others) have been widely used to solve MINLP problems [3]. However, in the case of non-convex problems, these techniques may cut-off the global optima. Even for large-scale problems the generated tree may be arbitrarily large and the solution time particularly high. [4]. Consequently, stochastic methods such as Evolutionary Algorithms (EAs) have been modified to solve MINLP problems. For this purpose, several integer-handling strategies have been implemented in EAs [5]–[10]. However, these extended algorithms are usually vulnerable to the size of the discontinuous feasible parts. That is, the population tends to enter into the larger discontinuous parts, because the algorithm finds feasible solutions more easily in such regions [2]. As a consequence, the small parts remain poorly explored or ignored. Therefore, if the best known solution is located in a smaller part, the algorithm may converge to the wrong part and get trapped in a local optimal solution.

Estimation of Distribution Algorithms (EDAs) are stochastic optimization techniques that explore the search space by

978-1-6654-6708-7/22/$31.00 ©2022 IEEE

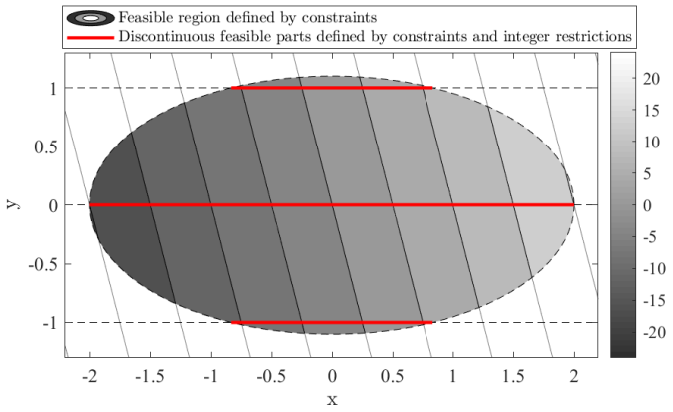

Fig. 1. The shaded area represents the feasible region defined by constraints, the red lines are the discontinuous feasible parts that also satisfy the integer restrictions.

sampling probabilistic models. The new offspring is generated from the most promising regions detected by the model. EDAs have been proposed for solving MINLP problems [11], [12]. According to [13], EDAs are considered EAs paradigms, because they do not use crossover and mutation operators as in traditional EAs. In [10], [14], EDA operators have been used in the Particle Swarm Optimization algorithm (PSO) and a surrogate model to solve MINLP problems.

However, the operators used by EDAs to handle discrete variables may also be vulnerable to being trapped in the larger discontinuous feasible parts, as is demonstrated in this work. On the other hand, it is important to note that the offspring generated by the crossover or mutation operators may be close to the parents, whereas EDAs use global statistical information and cannot directly control the similarity between offspring and parents. For such reason, EDAs are very effective in finding promising areas, but not in the local refinement of real values. In order to cover this deficiency, many real-valued EDAs have been combined with classical EAs, local search algorithms, and even with crossover and mutation operators [13], [15].

EDA_{mvn} was initially proposed to solve a Mixed-Variable Newsvendor problem [12]. This work presents an improved version of EDA_{mvn}, namely EDA_{mv}. The objective is to achieve a better performance when solving MINLP problems, considering the quality of the solution and the computational cost of the algorithm. Two modifications of the original algorithm are proposed, aimed at solving (i) the influence of the larger discontinuous feasible parts in the algorithm (ii) the drawbacks of EDAs in the local refinement of the real values. A benchmark was used to compare the performance between the improved proposal and two other EAs reported in the literature (including the original algorithm). The results show that the performance of EDA_{mv} is better than the other tested EAs. This new proposal solves to a great extent the influence of the larger discontinuous feasible parts in algorithm behavior, and the drawbacks of EDA_{mvn} in the local refinement in the continuous search space.

II. Estimation of Distribution Algorithm

EDA_{mvn} uses an Adaptive-Width Histogram (AWH) model for handling continuous variables, and a Learning-Based Histogram (LBH) model for handling discrete variables. New variable values are generated from the statistical population (statistical sample). The best N solutions selected from the offspring and the parent solutions compose the population of the next iteration. Despite the fact that in [12] the authors propose a strategy to generate the feasible initial population, in this work both EDA_{mvn} and the proposed EDA_{mv} generate the initial population randomly between the bounded ranges. Discrete variables begin with random discrete values.

A. Adaptive-width histogram model

The AWH model promotes promising areas by assigning them high probabilities, while the rest of them are assigned very low probability to prevent premature convergence. One AWH is developed for each decision variable independently (univariate models). The search space $[a_i, b_i]$ of the ith variable x_i is divided into $(W + 2)$ bins (regions), to built the distribution model Pr_i^c for the AWH model. Points $[p_{i,0}, p_{i,1}, ..., p_{i,w+1}, p_{i,w+2}]$ are the limits of the bins (Fig. 2), where $p_{i,0} = a_i$ and $p_{i,w+2} = b_i$ (x_i lower and upper bounds, respectively). The total number of bins is $(W+2)$ but the input parameter for EDA_{mvn} is W. However, the algorithm creates two more bins: one between the lower boundary a_i and the point $p_{i,1}$, and another between the point $p_{i,w+1}$ and the upper boundary b_i, to avoid premature convergence. Considering that $x^1_{i,min}$ and $x^2_{i,min}$ are the smallest and the second smallest existing values of variable x_i, respectively, and $x^1_{i,max}$ and $x^2_{i,max}$ are the highest and the second highest existing values of variable x_i, respectively, points $p_{i,1}$ and $p_{i,w+1}$ are defined in (2) and (3):

$$p_{i,1} = \max\left\{x^1_{i,min} - 0.5(x^2_{i,min} - x^1_{i,min}), p_{i,0}\right\} \quad (2)$$

$$p_{i,w+1} = \min\left\{x^1_{i,max} + 0.5(x^1_{i,max} - x^2_{i,max}), p_{i,w+2}\right\} \quad (3)$$

The W bins of the promising areas are located in the range $[p_{i,1}, p_{i,w+1}]$, and have the same width b, given by (4):

$$b = \frac{(p_{i,w+1} - p_{i,1})}{W} \quad (4)$$

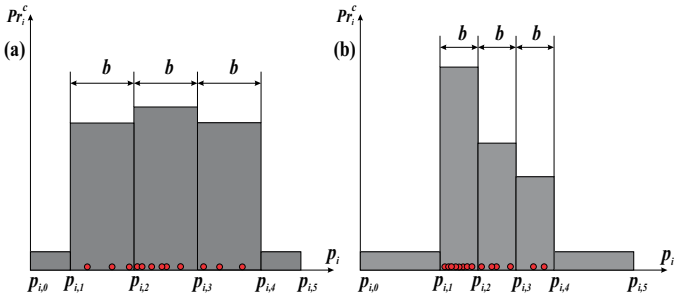

Fig. 2. Progression of the AWH model for W = 3. (a) first generations, (b) later generations.

Let $A_{i,j}$ be the count of individuals for the ith variable located in the jth bin. As can be seen in Fig. 2, the end bins do not contain points (unpromising areas), then the values $A_{i,1} = A_{i,W+2} = 0$. However, a small value will be assigned through the input parameter e_b to avoid premature convergence. Therefore, $A_{i,j}$ is obtained from (5):

$$A_{i,j} = \begin{cases} A_{i,j}, & \text{if } 2 \leq j \leq (W+1) \\ e_b, & \text{if } j = 1, (W+2), \text{ and } p_{i,j} > p_{i,j-1} \\ 0, & \text{if } j = 1, (W+2), \text{ and } p_{i,j} = p_{i,j-1} \end{cases} \quad (5)$$

Note that the first case in (5) is the bins count of the promising areas $[p_{i,1}, p_{i,w+1}]$. The second case assigns the value e_b to the end bins. The third case assigns zero to the end bins with empty range. Finally, the probability of the ith variable in the jth bin, could be approximated by (6):

$$Pr^c_{i,j} = \frac{A_{i,j}}{\sum_{k=1}^{W+2} A_{i,k}} \quad (6)$$

Fig. 2 shows how the promising area becomes very small in advanced stages of the search process. In that case, the algorithm focuses on exploitation. However, the end bins guarantee that the rest of the search space maintains an exploration behavior, which helps to avoiding premature convergence.

B. Learning-based histogram model

In [12], the LBH model is proposed for handling integer variables. The variable x_m has v available integer values, with $v \in \{lb_m, lb_m + 1, lb_m + 2, ..., ub_m\}$. Regarding discrete values the same process is applied, but encoding the variables indexes instead of their values. In this model, the probability of the nth available value of v is defined as $Pr^d_{m,v}$ (Fig. 3). The probability in the first iteration is set by (7) to ensure that each v value can be selected from the beginning:

$$Pr^d_{m,v}(0) = \frac{1}{(ub_m - lb_m) + 1} \quad (7)$$

The probability is updated in subsequent generations by (8):

$$Pr^d_{m,v}(t) = (1 - \gamma) \cdot Pr^d_{m,v}(t-1) + \gamma \cdot \frac{Count_v}{N} \quad (8)$$

where N is the population size, t is the current generation, γ is the population learning rate, and $Count_v$ is the number of individuals with value equal to v. Let t_{max} be the maximum number of generations, then the parameter γ is set in (9):

$$\gamma = \frac{t}{t_{max}} \quad (9)$$

Therefore, as the number of generations increases, γ gradually increases as well, which implies that the model uses more information from individuals in those promising areas.

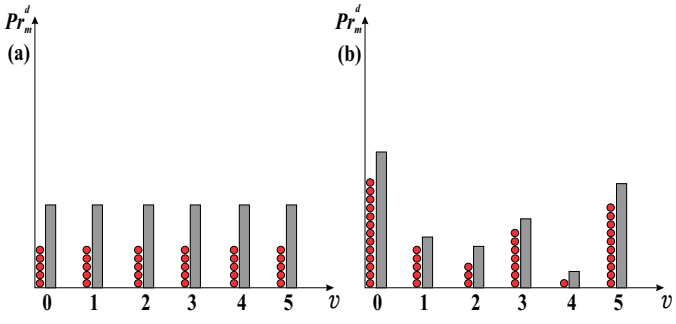

Fig. 3. Progression of the LBH model for W = 3. (a) first generations, (b) later generations.

C. Sampling

After the histograms have been developed, the offspring is obtained by sampling from the models. In case of a continuous variable x_i, a bin j is firstly selected according to a randomly generated probability, then x_i is uniformly sampled from the points that limit the bin selected $[p_{i,j-1}, p_{i,j})$. For a discrete variable x_m, an available value of $v \in \{lb_m, ub_m\}$ is selected by a randomly generated probability.

D. Constraint handling

EDA_{mvn} uses a penalty method as a constraint-handling technique. The fitness values of each individual is divided in two parts: (i) objective function value U and (ii) sum of constraint violation CV. This fitness value is defined in (10):

$$f(\mathbf{x}, \mathbf{y}) = U(\mathbf{x}, \mathbf{y}) + CV(\mathbf{x}, \mathbf{y}) \quad (10)$$

The constraint violation is obtained from (11):

$$CV(\mathbf{x}, \mathbf{y}) = k_1 \sum_{i=1}^{n} \max(g_i(\mathbf{x}, \mathbf{y}), 0) + k_2 \sum_{j=1}^{m} \max(|h_j(\mathbf{x}, \mathbf{y})|, E) \quad (11)$$

where E is the tolerance parameter for equality constraints, k_1 and k_2 are the penalty factors for inequality constraints and equality constraints, respectively.

III. Vulnerability to the Size of Discontinuous Feasible Parts

In [2], the authors showed the influence of the larger discontinuous feasible parts in the truncation and rounding techniques in MINLP problems. Unlike previous techniques, EDA_{mvn} uses a direct integer-restriction-handling technique, since the LBH model directly generates integer values. In this section, we disclose the influence of those discontinuous parts on the LBH model. For that, the example in (12) is presented.

$$\begin{aligned} \text{minimize } & f(x, y) = 2(x-1)^2 + (y-3)^2 \\ \text{subject to: } & g(x, y) = x^2 + y^2 - 4 \leq 0 \\ & x \in [-3, 3] \\ & y \in \{-3, 3\} \end{aligned} \quad (12)$$

In the MINLP problem in (12), x is a continuous variable and y is a integer variable, both in the range between -3 and 3. The optimal solution is $f(0, 2) = 3$, as shown in Fig. 4(a). The shaded area represents the feasible region defined by the constraints, and the red zones are the discontinuous feasible parts that also satisfy the integer restrictions. Note in Fig. 4(b), the global optimal solution is located in a small discontinuous

feasible part (upper red point). Local optimal solutions can be found in the larger discontinuous feasible parts.

The MINLP problem in (12) was solved by EDA_{mvn} and PSO for mixed-variable (PSO_{mv}), to illustrate the vulnerability of the LBH model to being trapped in larger discontinuous feasible parts. PSO_{mv} [10] is a hybrid approach that uses the LBH model as integer restrictions handler. The parameter values used by both algorithms were obtained by the calibration process explained in Section V-C.

Fig. 5 shows the execution of EDA_{mvn}. In the 2nd generation the population is in the exploration phase. However, in the 5th generation the entire population is located in the largest discontinuous feasible parts, as shown in Fig. 5(b). Therefore, the probability of the LBH model for the available value $y = 2$ is zero. This inhibits any possibility of exploring the global optimum feasible region. In the later generations, the population tends to the best local optimum in the large discontinuous feasible parts, as shown in Figs. 5(c), 5(d). In EDA_{mvn}, the search in small discontinuous feasible parts improves when the penalty factor values decrease, since in regions close to feasibility parts low $CV(\mathbf{x}, \mathbf{y})$ are obtained. However, since the precision to the feasibility is affected by the penalization values, in many cases the algorithm does not get a real feasible solution.

PSO_{mv} (see Fig. 6) also starts with a randomly generated population. It can be observed in Fig. 6(c) that, even in the 10th generation there are individuals exploring the region close to the global optimum. However, the population tends towards the best local optimal solution of the larger discontinuous feasible parts, as shown in Fig. 6(d).

In summary, the LBH model finds good candidates in large areas with relative ease, but it only finds infeasible points around small areas. This behavior increases the probability of available values in larger parts, and quickly discards small parts that remain unexplored or poorly explored. Once the entire population enters the wrong part, the loss of diversity causes the population to be trapped in a local optimum.

IV. Improvement Proposal

In this section, two modifications for the original EDA_{mvn} are proposed. The new variant EDA_{mv} is aimed at solving

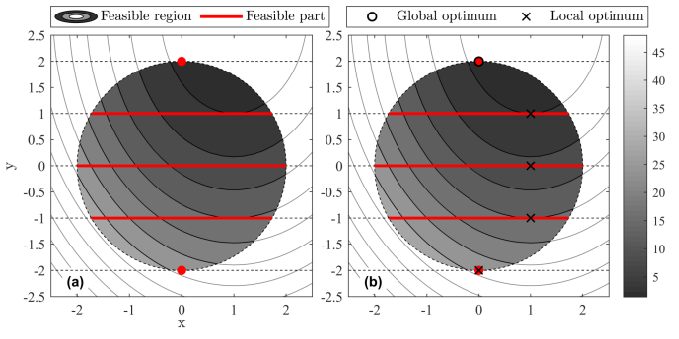

Fig. 4. MINLP problem in Eq. (12) (a) Feasible region and feasible parts, (b) global optimal solution and local optimal solutions.

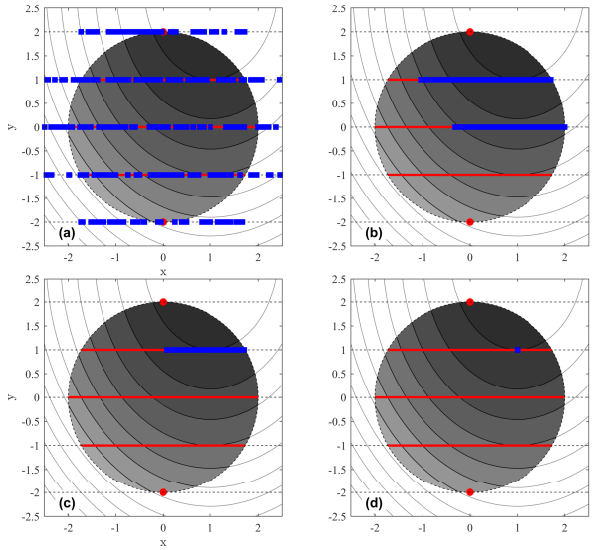

Fig. 5. EDA_{mvn} execution for the MINLP problem in (12): (a) 2nd generation; (b) 5th generation; (c) 10th generation; (d) 60th generation.

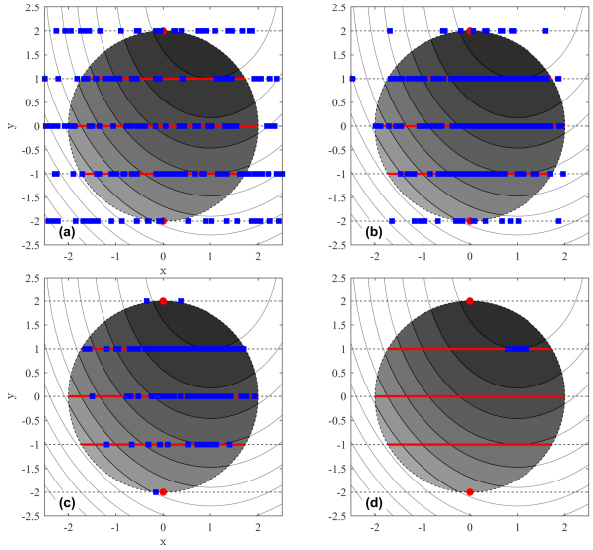

Fig. 6. PSO_{mv} execution for the MINLP problem in (12): (a) 2nd generation; (b) 5th generation; (c) 10th generation; (d) 60th generation.

(i) the influence of the larger discontinuous feasible parts in the algorithm and (ii) the drawbacks of EDAs in the local refinement of the real values. In the first modification, the ε-constrained method is implemented in a special link with the LBH model, for exploring from non-feasible regions, and then improve the searching in small feasible parts. The second modification is the hybridization of EDA with a mutation operator to generate offspring from both the global distribution information and the parent information.

A. ε-constrained method

The ε-constrained method was proposed by Takahama and Sakai [16] as a constraint-handling technique. Given two function values $f(x_1)$, $f(x_2)$, and two constraint violations

$\phi(x_1)$, $\phi(x_2)$ for two points x_1 and x_2, the ε-constrained method uses the ε-level comparisons described in (13):

$$(f_1, \phi_1) \leq_\varepsilon (f_2, \phi_2) = \begin{cases} f_1 \leq f_2, & \text{if } \phi_1, \phi_2 \leq \varepsilon \\ f_1 \leq f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \quad (13)$$

where ε-level comparisons are defined as an order relation on a pair of objective function and constraint violation values $(f(x), \phi(x))$.This means that the candidates with violation sum lower than ε are considered as feasible solutions and are ordered according to their fitness function values. In the case of $\varepsilon = 0$, $\phi(x)$ always precedes $f(x)$. Therefore, this method favors the approach to the feasible region by keeping slightly infeasible solutions with promising objective function value.

The ε-level decreases at each iteration G until the predefined iteration number Tc is reached, after that $\varepsilon = 0$, see (14):

$$\varepsilon(0) = \phi(x_\theta)$$
$$\varepsilon(G) = \begin{cases} \varepsilon(0)(1 - \frac{G}{T_c})^{cp}, & \text{if } 0 < G < T_c \\ 0, & \text{otherwise} \end{cases} \quad (14)$$

where cp is a parameter to control the speed of constraint relaxation, $\varepsilon(0)$ is the initial value of ε, and x_θ is the top θth in an array sorted by total constraint violation ($\theta = 0.2N$).

B. Link between the LBH model and the ε-constrained method

The first modification is to use the ε-constrained method as a constraint-handling technique. Then, the selection mechanism to get the next population is carried out through parent-offspring competition using the ε-constrained method. The main idea is to improve exploration in small discontinuous feasible parts, approaching from the infeasible contours, i.e., infeasible candidates are not necessarily discarded, with the aim to reach feasible parts by using such solutions.

This modification fulfills the goal only for some runs, as will be shown in the results section. In other runs, the feasible region could not be reached. The problem occurs mainly when the LBH model converges while the ε value is still high. If the entire population covers a region with constraint violation lower than the ε-level, the algorithm may be trapped in such infeasible region due to loss of diversity.

To overcome this drawback, a link between the LBH model and the ε-constrained method is established. The LBH model maintains equal probability for all available integer values until ε reaches a predefined value ε_p. In this point, the LBH model begins to operate considering the statistical information (learning). If the ε-constrained method has been effective, for values of ε sufficiently small, the solutions must be close to those parts of the feasible region with promising objective function values. Therefore, if the histogram begins the learning process at that point, it has a better chance of converging to good solutions.

The AWH and LBH histogram models for the new proposal EDA_{mv} are shown in Algorithms 1, 2, respectively. The AWH is unchanged from the original EDA_{mvn}. However, the LBH model (now called LBHε) works linked with the ε-level, as shown in line 2 of Algorithm 2.

Algorithm 1 $PR^c \leftarrow \text{AWH}(p, W, e_b)$

Input: the population p, the number of bins W, end bins parameter e_b.
Output: the probability model PR^c

1: **for** $i \leftarrow 1$ **to** $Nvar^c$ **do**
2: Get $x^1_{i,min}, x^2_{i,min}, x^1_{i,max}, x^2_{i,max}$
3: Calculate the boundaries of the promising area $p_{i,1}$ and $p_{i,w-1}$ by (2) and (3)
4: Calculate the bin width b by (4)
5: **for** $j \leftarrow 1$ **to** $W + 2$ **do**
6: Count the candidates that fall into the jth bin and set $A_{i,j}$ by (5)
7: Update the probability model $PR^c_{i,j}$ by (6)
8: **end for**
9: **end for**

Algorithm 2 $PR^d \leftarrow \text{LBH}\varepsilon(p, t, \varepsilon, \varepsilon_p)$

Input: the population p, the current generation t, level ε, link parameter ε_p.
Output: the probability model PR^d

1: **for** $m \leftarrow 1$ **to** $Nvar^d$ **do**
2: **if** t=1 **or** $\varepsilon > \varepsilon_p$ **then**
3: Update $PR^d_{m,v}$ by (7)
4: **else**
5: **for** $v \leftarrow lb_m$ **to** ub_m **do**
6: Count the individuals with values x_m equals to v and set as $count_v$
7: Update $PR^d_{m,v}$ by (8)
8: **end for**
9: **end if**
10: **end for**

C. Hybridization with a mutation operator

The second modification is aimed at improving the search process in the space of real variables. EDAs work with probabilistic models, since the global distribution of the population is used to generate new solutions. In contrast, in conventional EAs, crossover and mutation operators are used to generate new solutions. Both types have their pros and cons, the solutions generated by crossover and mutation operators may be close to the parents, but far from other promising areas, while the EDAs cannot directly control the similarities between the new solutions and the parents. In this work, the mutation operation is added to generate the real variables. This operation is alternated with the original sampling operation taking into account the predefined mutation probability r_M. In this way, if this probability is satisfied for a solution vector, its real variables are computed as shown in (15) and (16):

$$x^{g+1}_{i,j} = x^g_{best,j} + \beta \cdot (x^g_{best,j} - x^g_{i,j}) \quad (15)$$

$$\beta = \beta_{min} + rand_{i,j} \cdot (\beta_{max} - \beta_{min}) \quad (16)$$

where i, j are the index of the current solution vector and current variable, respectively, g is the current generation,

$x^g_{best,j}$ is the jth variable of the best solution vector found so far, $rand_{i,j}$ is a random number between 0 and 1, and β_{min} and β_{max} are the lower and upper bounds of β predefined by the user, with values between 0 and 1.

Consequently, the real variables of solution vectors could be generated by either sampling or mutation. Algorithm 3 shows the generation of real variables in both ways (lines 1-12), as well as the sampling of the integer variables (lines 13-16) without any change from the original algorithm. The overall procedure of EDA_{mv} is presented in Algorithm 4.

Algorithm 3 Offs ← S/M($PR^c, PR^d, r_M, \beta_{min}, \beta_{max}, x_{best}$)

Input: the probability models PR^c and PR^d; mutation parameters $r_M, \beta_{min}, \beta_{max}$
Output: a new candidate offspring $[x_i, x_m]$

1: $r_1 \leftarrow$ Uniform(0,1)
2: **if** $r_1 \leq r_M$ **then**
3: **for** $i \leftarrow 1$ **to** $Nvar^c$ **do**
4: calculate variable x_i by (15)
5: **end for**
6: **else**
7: **for** $i \leftarrow 1$ **to** $Nvar^c$ **do**
8: $r_2 \leftarrow$ Uniform(0,1)
9: Select a bin j according PR^c and r_2
10: $x_i \leftarrow$ Uniform$[p_{i,j-1}, p_{i,j})$
11: **end for**
12: **end if**
13: **for** $m \leftarrow 1$ **to** $Nvar^d$ **do**
14: $r_3 \leftarrow$ Uniform(0,1)
15: Select a value v from $\{l_b, u_b\}$ according PR^d and r_3
16: **end for**

Algorithm 4 EDA_{mv} framework

Input: $N, W, e_b, G, r_M, T_c, cp, \beta_{min}, \beta_{max}, \varepsilon_p$
Output: The best solution

1: Initialize population p
2: **for** $t \leftarrow 1$ **to** G **do**
3: Get the best solution so far x_{best}
4: temporal $\leftarrow \varnothing$
5: $PR^c \leftarrow$ AWH(p, W, e_b)
6: $PR^d \leftarrow$ LBH$\varepsilon(p, t, \varepsilon, \varepsilon_p)$
7: **for** $i \leftarrow 1$ **to** N **do**
8: Offs ← S/M($PR^c, PR^d, r_M, \beta_{min}, \beta_{max}, x_{best}$)
9: temporal ← temporal ∪ Offs
10: **end for**
11: temporal ← temporal ∪ p; and clear p
12: Select the best N individuals from temporal by the ε-constrained method and set to p
13: **end for**

V. Experimentation and Results

A. How EDA_{mv} works?

The problem described in (12) is solved by EDA_{mv}. The parameters used will be described in the Section V-C. The progress of the algorithm along generations is observed in Fig. 7. In the early stages, the ε-level comparison allows the algorithm to explore smaller parts from the infeasible contours, as shown in Figs. 7(a), 7(b). At that moment, the LBHε model maintains a uniform probability for all available integer values in order to avoid premature convergence towards infeasible parts. This procedure contributes to the placement of solutions in various discontinuous parts and, as a consequence, the exploration of each part by means of the sampling/mutation operation of real variables operators. Fig. 7(c) shows how the solutions are located very close to the optimal solutions of those discontinuous parts. Once the value of ε reaches ε_p, the LBHε begins the learning process. At that point, the statistical information is comprised of solutions located in feasible or almost feasible regions, with promising values of the objective function. As a consequence, the LBHε model has a better chance of converging to good solutions, as shown in Fig. 7(d).

B. Benchmark problems

Twelve MINLP problems (F1-F12) proposed in [2] were used to evaluate and compare the performance of the proposed EDA_{mv}. Problems F1-F4 have many discontinuous feasible parts with different sizes and the optimal solution is located in a feasible part with a small size. Both F5 and F7 are problems with equality constraints. F8-F12 are problems with at least eight decision variables. Details of F1-F12 are presented in the Supplementary File in [2].

For the experimentation, the maximum number of the objective function evaluations was set at 1.8E+05, and 25 independent runs were carried out for each problem. The tolerance value E for the equality constraints was set at 1.0E-04. A run is considered as successful if: $|f(x_{best}) - f(x^*)|$ ≤1.0E-4, where x^* is the best known solution and x_{best} is the best feasible solution provided by the tested algorithm.

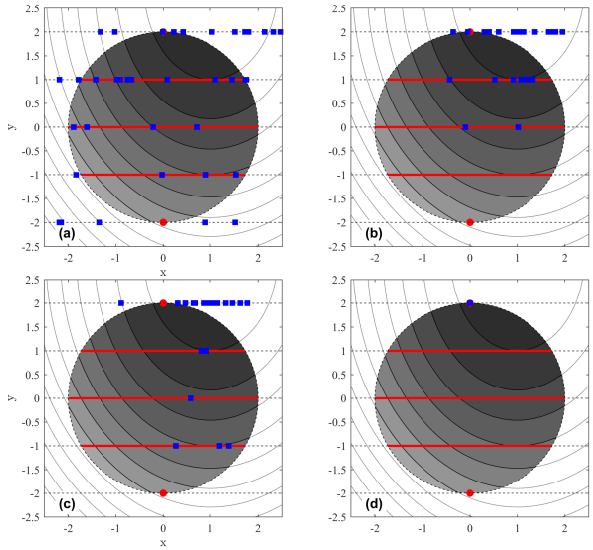

Fig. 7. EDA_{mv} execution for the MINLP problem in (12): (a) 2nd generation; (b) 5th generation; (c) 10th generation; (d) 60th generation.

TABLE I
PSO_{mv}, EDA_{mvn}, $EDA\text{-}I$, AND $EDA\text{-}II$ RESULTS OVER 25 INDEPENDENT RUNS.

Problem	Status	PSO_{mv}	EDA_{mvn}	$EDA\text{-}I$	$EDA\text{-}II$	EDA_{mv}
F1	FR	100	20	52	64	**100**
	SR	0	16	48	64	**100**
	Ave ± Std Desv	17.000±0.000 -	NA -	NA -	NA -	**13.000±0.000**
F2	FR	**100**	**100**	0	100	**100**
	SR	**100**	**100**	0	88	**100**
	Ave ± Std Desv	**1.000±0.000** ≈	**1.000±0.000** ≈	NA -	1.120±0.332 ≈	**1.000±0.000**
F3	FR	100	100	0	100	**100**
	SR	28	4	0	72	**100**
	Ave ± Std Desv	-3.839±0.237 -	-3.817±0.246 -	NA -	-3.879±0.221 -	**-4.000±0.000**
F4	FR	**100**	100	0	100	**100**
	SR	**100**	16	0	32	**100**
	Ave ± Std Desv	**-6.000±0.000** ≈	-5.994±0.009 -	NA -	-5.995±0.008 -	**-6.000±0.000**
F5	FR	100	88	88	100	**100**
	SR	8	4	80	52	**100**
	Ave ± Std Desv	1.161±0.274 -	NA -	NA -	0.725±0.505 -	**0.250±0.000**
F6	FR	**100**	100	0	100	**100**
	SR	**100**	0	0	0	**100**
	Ave ± Std Desv	**-6,783.582±0.000** ≈	-6,657.585±347.491 -	NA -	-6,783.420±0.226 -	**-6,783.582±0.000**
F7	FR	96	8	4	0	**100**
	SR	0	0	0	0	**24**
	Ave ± Std Desv	NA -	NA -	NA -	NA -	**0.800±0.338**
F8	FR	**100**	88	0	92	92
	SR	**0**	0	0	0	0
	Ave ± Std Desv	**7,218.753±87.347** +	NA ≈	NA ≈	NA ≈	NA
F9	FR	**100**	56	0	100	80
	SR	**12**	0	0	0	0
	Ave ± Std Desv	**7,224.667±244.750** +	NA ≈	NA ≈	10,370.614±1,240.634 +	NA
F10	FR	**100**	28	0	100	80
	SR	**60**	0	0	0	0
	Ave ± Std Desv	**7,359.999±283.334** +	NA ≈	NA ≈	10,372.112±2,165.086 +	NA
F11	FR	100	100	100	100	**100**
	SR	0	0	0	0	**0**
	Ave ± Std Desv	46.164±7.067 ≈	133.562±20.661 -	94.065±20.778 -	151.547±22.654 -	**43.518±7.263**
F12	FR	100	100	100	100	**100**
	SR	0	0	0	0	**0**
	Ave ± Std Desv	90.709±20.437 -	136.633±44.896 -	485.797±254.204 -	140.123±28.073 -	**65.745±26.771**
[-/=/+]		[5/4/3]	[8/4/0]	[9/3/0]	[8/2/2]	—

C. Algorithms

The test problems were solved by the PSO_{mv}, EDA_{mvn}, and EDA_{mv} algorithms. However, to evaluate the individual contribution of each modification, the variants $EDA\text{-}I$ (only with the ε-constrained method), and $EDA\text{-}II$ (with a link between the ε-constrained and the LBH model) are also included. The algorithms were tuned using the iRace parameter tuning tool [17]. The instances used for tuning were the F1-F12 problems. For each algorithm, 2000 experiments were carried out in iRace, using the Friedman test for the selection of parameter sets.

PSO_{mv} used a swarm size $N = 300$, acceleration coefficient $c = 1.5299$, learning rate $\alpha = 0.0125$. EDA_{mvn} used $N = 300$, numbers of bins $W = 10$, end bins parameter $e_b = 5.9058$, penalty parameters $k_1 = 500,000$ and $k_2 = 1,000$. $EDA\text{-}I$ used $N = 50$, $W = 2$, $e_b = 2.0192$, control generation $T_c = 3,000$ and control speed parameter $cp = 5$. $EDA\text{-}II$ used $N = 50$, $W = 2$, $e_b = 2.0192$, $T_c = 3,000$, $cp = 5$, link parameter $\varepsilon_p = 0.01$. EDA_{mv} used $N = 50$, $W = 4$, $e_b = 2.3959$, $T_c = 3000$, $cp = 8$, mutation parameters: $r_M = 0.6$, $\varepsilon_p = 0.2399$, $\beta_{min} = 0.3$, $\beta_{max} = 0.9$.

D. Analysis of results

Table I summarizes the results of the test problems solved by PSO_{mv}, EDA_{mvn}, $EDA\text{-}I$, $EDA\text{-}II$, and EDA_{mv}. The results are assessed in terms of the Feasible Rate (FR), Successful Rate (SR), Average (Ave), and Standard Deviation (Std Dev), over 25 independent runs. "NA" means that an algorithm cannot achieve 100% FR.

Firstly, the focus is on the progressive advancement of the original EDA_{mvn}. The ε-constrained method implementation ($EDA\text{-}I$) aims to explore the smallest discontinuous feasible parts. However, it gave the worst results for the test problems. As can be seen in Table I, $EDA\text{-}I$ improved few results, since in many cases no feasible solutions were found. As mentioned above, this drawback mainly occurs when the LBH model

converges while the ε value is still high. At that point, the whole population enters into a permissible infeasible region, and the algorithm can be trapped due to loss of diversity.

In $EDA\text{-}II$, a link between the LBH model and the ε-constrained method was established (LBHε). The learning process of LBHε begins when $\varepsilon \leq 0.01$, then statistical information is comprised by better solutions. $EDA\text{-}II$ is better than EDA_{mvn} in seven problems, F1, F3, F4, F5, F6, F9 and F10. The FR and SR indicators are substantially improved over previous proposals. However, it was detected that the failed solutions were often very close to the best known solution.

In the final proposal EDA_{mv}, the mutation operator is also added to improve the search process for real variables. The Wilcoxon's rank-sum test at a 0.05 significance level was carried out between EDA_{mv} and the other four algorithms. In Table I, $[-]$, $[\approx]$ and $[+]$ means that the performance of the algorithm under test is worse, equal, or better than the corresponding to EDA_{mv}, respectively.

As can be seen in the last row of Table I, the results of EDA_{mv} are significantly better than the original EDA_{mvn} in eight problems: F1, F3, F4-F7, F11, F12, and similar in four other problems: F2, F8, F9, F10. Therefore, in no case EDA_{mvn} improves the results of the new proposal. Significantly better EDA_{mv} results can also be observed with regard to the $EDA\text{-}I$ and $EDA\text{-}II$ algorithms, indicating the effectiveness of the proposed modifications.

Finally, EDA_{mv} is compared with the hybrid PSO_{mv}. From Table I, EDA_{mv} is significantly better than PSO_{mv} in five test problems: F1, F3, F5, F7, F12, no difference in four problems: F2, F4, F6, F11, while PSO_{mv} outperformed EDA_{mv} in three problems: F8, F9, F10. For problems F9 and F10 the mutation operator of EDA_{mv} produces premature convergence, which causes the population to be trapped in infeasible zones for some runs. It is notable that the original EDA_{mvn} shows a worse performance than PSO_{mv}. In contrast, after the proposed modifications EDA_{mv} provided a better performance than those of the four compared algorithms.

VI. Conclusions

EDA_{mvn} was initially proposed to solve a mixed-variable newsvendor problem. In this work, the vulnerability of EDA_{mvn} of being trapped in larger discontinuous feasible parts has been proven, as well as its drawbacks in the local refinement of real variables. To overcome such disadvantages, EDA_{mv} was proposed with two modifications to the original algorithm (i) a link between the LBH model and the ε-constrained method, and (ii) the hybridization with a mutation operator. This new proposal solves to a great extent the influence of the larger discontinuous feasible parts, and the disadvantages of EDAs in the local refinement of the real values.

A benchmark from the related literature was used to test and compare the performance of the improved proposal against the original EDA_{mvn} and hybrid PSO_{mv} algorithms. According to the Wilcoxon's rank-sum tests applied to the results obtained in the experimental stage, the performance of EDA_{mv} was better than those of the other algorithms tested. Different instances were also executed considering each proposed modification separately, and their effectiveness was verified.

Acknowledgment

The first author acknowledges support from the Mexican National Council of Science and Technology (CONACyT) through a scholarship to pursue graduate studies at the CIDETEC-IPN. Second and fourth author acknowledge support from SIP-IPN through project No. 20221928 and No. 20221960, respectively.

References

[1] H. Peng, Y. Han, C. Deng, J. Wang, and Z. Wu, "Multi-strategy co-evolutionary differential evolution for mixed-variable optimization," *Knowledge-Based Systems*, vol. 229, p. 107366, 2021.

[2] J. Liu, Y. Wang, B. Xin, and L. Wang, "A biobjective perspective for mixed-integer programming," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[3] C. D'Ambrosio and A. Lodi, "Mixed integer nonlinear programming tools: an updated practical overview," *Annals of Operations Research*, vol. 204, no. 1, pp. 301–320, 2013.

[4] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, 2013.

[5] Y. Lin, Y. Liu, W.-N. Chen, and J. Zhang, "A hybrid differential evolution algorithm for mixed-variable optimization problems," *Information Sciences*, vol. 466, pp. 170–188, 2018.

[6] A. Ponsich, C. Azzaro-Pantel, S. Domenech, and L. Pibouleau, "Mixed-integer nonlinear programming optimization strategies for batch plant design problems," *Industrial & engineering chemistry research*, vol. 46, no. 3, pp. 854–863, 2007.

[7] A. Ponsich and C. C. Coello, "Differential evolution performances for the solution of mixed-integer constrained process engineering problems," *Applied Soft Computing*, vol. 11, no. 1, pp. 399–409, 2011.

[8] MathWorks, "Mixed integer ga optimization: Optimization toolbox for use with matlab®." Available at https://www.mathworks.com/help/gads/mixed-integer-optimization.html (2021/12/23).

[9] L. Wang, X. Fu, M. I. Menhas, and M. Fei, "A modified binary differential evolution algorithm," in *Life System Modeling and Intelligent Computing*, pp. 49–57, Springer, 2010.

[10] F. Wang, H. Zhang, and A. Zhou, "A particle swarm optimization algorithm for mixed-variable optimization problems," *Swarm and Evolutionary Computation*, vol. 60, p. 100808, 2021.

[11] W. Shi, W.-N. Chen, Y. Lin, T. Gu, S. Kwong, and J. Zhang, "An adaptive estimation of distribution algorithm for multipolicy insurance investment planning," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 1–14, 2017.

[12] F. Wang, Y. Li, A. Zhou, and K. Tang, "An estimation of distribution algorithm for mixed-variable newsvendor problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 3, pp. 479–493, 2019.

[13] A. Zhou, J. Sun, and Q. Zhang, "An estimation of distribution algorithm with cheap and expensive local search methods," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 807–822, 2015.

[14] J.-Y. Li, Z.-H. Zhan, J. Xu, S. Kwong, and J. Zhang, "Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[15] Q. Zhang, J. Sun, and E. Tsang, "Combinations of estimation of distribution algorithms and other techniques," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 273–280, 2007.

[16] T. Takahama and S. Sakai, "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites," in *2006 IEEE international conference on evolutionary computation*, pp. 1–8, IEEE, 2006.

[17] M. López-Ibáñez, L. P. Cáceres, J. Dubois-Lacoste, T. G. Stützle, and M. Birattari, *The irace package: User guide*. IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en …, 2016.