Available online at www.sciencedirect.com

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

Process optimization using a dynamic self-adaptive constraint handling technique coupled to a Differential Evolution algorithm

J. Cortez-González[a], A. Hernández-Aguirre[b], R. Murrieta-Dueñas[a], R. Gutiérrez-Guerra[c,*], S. Hernández[d], J.G. Segovia-Hernández[d]

[a] Tecnológico Nacional de México, Instituto Tecnológico Superior de Irapuato, Departamento de Ingeniería Química y Bioquímica, Carretera Irapuato-Silao km 12.5, C.P. 36821 Irapuato, Gto, Mexico
[b] Centro de Investigación en Matemáticas, A.C., Departamento de Ciencias Computacionales, Callejón de Jalisco s/n, C.P. 36240, Mineral de Valenciana, Guanajuato, Gto, Mexico
[c] Universidad Tecnológica de León, Blvd. Universidad Tecnológica 225, Col San Carlos, C.P. 37670 León, Gto, Mexico
[d] Universidad de Guanajuato, Campus Guanajuato, Departamento de Ingeniería Química, Noria Alta s/n, C.P. 36050 Guanajuato, Gto, Mexico

ARTICLE INFO

Article history:
Received 1 June 2022
Received in revised form
4 November 2022
Accepted 7 November 2022
Available online 10 November 2022

Keywords:
Optimization
Differential evolution
Self-adaptive dynamic constraint handling
Weighting constraint handling
Chemical processes

ABSTRACT

Nowadays, chemical processes are projected to obtain their best performance in energy and water consumption, pollutant emissions and total annual costs, while still meeting quality of products and good operational performance. These goals are accomplished through adequate optimization of the fitness function by manipulating the operational variables (decision variables) of the process. However, a successful optimization process depends completely on the constraint handling established in the modeling of the process. The weighted summation of constraint violations (weighting function technique, WF) is one of the most common approaches for handling constraints in optimization problems. Nevertheless, in spite of this technique yielding good results, in this work we show a novel self-adaptive constraint handling technique (SA) based on a self-adaptation dynamic threshold and self-adaptation (weight) factors. This technique deals with real and discrete variables and converts equality constraints into inequality constraints through a dynamic threshold. Both penalization techniques (WF and SA) were, respectively, coupled to a Differential Evolution (DE) algorithm to optimize some benchmark functions and chemical engineering optimization problems. In addition, the rigorous model of a distillation train was optimized in Aspen One for the first time with a self-adaptive constraint handling technique in chemical engineering. Although both penalization techniques were coupled to the same DE algorithm and both cases were run under the same conditions, the results show that the dynamic self-adaptive constraint handling technique coupled to DE (DE-SA) achieves considerably better best-solutions than the best-solutions obtained by the weighting function technique coupled to DE (DE-WF). In addition, DE-SA led to substantial reductions of numerical effort in relation to DE-WF. These conclusions are supported by statistical analysis of the results of 30 runs of the optimization process for each constraint handling technique, for a distillation train.

* Corresponding author.
E-mail address:
rogutierrez@utleon.edu.mx (R. Gutiérrez-Guerra).
https://doi.org/10.1016/j.cherd.2022.11.006

1. Introduction

Sustainable development demands chemical processes able to transform raw materials into final products using optimal amounts of energy and water, minimizing waste and reducing emissions of pollutants into the environment.

These goals can be tracked by implementing robust design and optimization strategies. However, due to the complex nature of the chemical processes, their mathematical models are nonlinear, multivariable and constrained. Thus, a key concern is the constraint handling technique used to keep chemical processes under optimal design and operational conditions. Constraint handling is not commonly emphasized when deterministic or evolutionary algorithms (EA) are implemented to optimize chemical processes.

For the evolutionary algorithms the constraint handling techniques has been classified intro five categories: methods based on penalty functions, methods based on special representations and operators, methods based on repair algorithms, methods based on separation of objectives and constraints, and other hybrid methods (Coello, 2002). The penalty function is the most common approach in the evolutionary algorithm to handle constraints. In this method, the constrained optimization problem is transforms into an unconstrained optimization problem that includes penalizing the infeasible points and the degree of constraints violation (Kheawhom, 2010). The methods based on special representations and operators, consist in reduces the dimension and simplifying the shape of the search space through of a change of representation in the problem (Dasgupta and Michalewicz, 1997). In the methods based on repair algorithms, attempts to move infeasible solutions into the feasible region (Riche and Haftka, 1997). Typically, heuristic rules are used to guide the repair process. In the other approaches separately handle objective functions and constraints (Separation of constraints and objectives). These approaches require several extra parameters and the constraints are can handled as additional objectives (Coello and Mezura-Montes 2002). Finally, the hybrid methods normally are coupled with a numerical optimization approach to handle constraints in evolutionary algorithm (Adeli and Cheng, 1994).

In fact, a weighting penalization function is the most applied approach to handle constraints during the optimization process (which triggers some difficulties in selection of a convenient penalty factor). The weighting penalization function can be traced back to the first numerical algorithms used to solve constrained optimization problems. They became the de factor approach due to their broad applicability. The penalization concept is simple to grasp: for every violated constraint, a penalty factor is applied to the optimization function; therefore, functions that do not violate the constraints have the best function value. The trick with this approach is to find the weight (factor) of each penalty such that larger penalizations should correspond to larger constraint violations.

Static penalties retain their weight value throughout the optimization process. Dynamic penalties change their value during the process. There are methods based on dynamic penalties in which the penalty increases over generations or that employ information from the search to adjust the penalization (Joines and Houck, 1994; Kazarlis and Petridis, 1998; Dadios and Ashraf, 2006). Other methods use a simulated annealing algorithm (Michalewicz and Attia, 1994) to adjust the weight factors. Several difficulties related to the search for adequate penalty factors have been experienced as they require parameters for the cooling model which is very sensitive to the performance of the algorithm. In addition, some methods eliminate equality constraints by reducing the dimension of the search space (Dasgupta and Michalewicz, 1997) or introduce random key encoding to maintain the feasibility of the solution (Asafuddoula et al., 2015). Another approach attempts to move infeasible solutions into feasible regions (Liepins and Vose, 1990). Unfortunately, this approach is problem-dependent and computationally expensive in cases of complex optimization problems.

On the other hand, if an infeasible solution can be easily fixed, repair algorithms are usually a better choice as they use non-dominance-based selection in which constraints are handled as additional objectives (Coello and Mezura-Montes, 2002; Mezura-Montes and Coello, 2011).

Other authors have incorporated a gradient-based repair algorithm with a dominance-based selection scheme (Kheawhom, 2010). Repair algorithms are usually a good choice if an infeasible solution can be easily fixed (Boukouvala and Ierapetritou, 2014; Yu et al., 2019).

Some stochastic global optimization algorithms have been used to solve constrained optimization problems by means of the feasibility approach (Chanthasuwannasin et al., 2017), or by finding a feasible point using the gradient of constraints at an infeasible point (Takahama and Sakai, 2006). Other authors have proposed re-formulating (Zhang and Rangaiah, 2012), hiding (Na et al., 2017) or eliminating (Yiqing et al., 2007) equality constraints to solve non-convex nonlinear programming (NLP)/mixed integer nonlinear programming (MINLP) problems using evolutionary algorithms such as Differential Evolution, Particle Swarm Optimization, Genetic Algorithm and self-adaptive algorithms.

Several authors in chemical engineering have used static penalty functions to solve optimization problems with inequality constraints, due to their effectiveness and ease of implementation. NLP and MINLP problems have been solved using a Differential Evolution algorithm and Tabu list (Srinivas and Rangaiah, 2007) coupled to the penalty function method. Improvements to the determination of penalty parameters (weight factors) in the weight functions have also been proposed (Sreepathi and Rangaiah, 2017). Finally, another approach consists of a constraint handling method utilizing process characteristics to formulate the weight function (Austbø et al., 2013; Austbø and Gundersen, 2016). In these implementations, the results show that the penalty function method with an adequate value for the penalty parameter provides better optimal solutions.

Other penalty approaches use several constraint handling techniques to enhance the convergence speed and population diversity. For engineering design problems, an implementation supported by the gradient repair method and ranking by constraint fitness has been analyzed (Zahara and Kao, 2009). One technique involves the assignment of a random sequence of constraints to the solution, the number of satisfied constraints and the violation measure (Asafuddoula et al., 2015). Another technique uses a repair algorithm based on the gradient information derived from the equality constraints to deal with infeasible solutions through dominance-based selection (Kheawhom, 2010). Recently, Yu et al. (2019) proposed a mechanism which transforms the equality constraints into inequality constraints,

combined with seven criteria to compare feasible and infeasible solutions. Their results showed that the constraint handling scheme is very important for identifying feasible solutions early and reducing computational cost.

Despite the sophistication of the constraint handling techniques reported in the literature, they have only been applied for continuous search spaces and explicit objective (fitness) functions (in other words, the objective function is a simplified mathematical function of the problem). In several research's, Aspen has been used as an evaluator of the fitness function in processes with constraints, for example intensified distillation systems (Vazquez–Castillo et al., 2009), a crude distillation system (More et al., 2010), a natural gas liquefaction processes (Austbø and Gundersen, 2016), an extractive distillation (Franke, 2019) and recently a multiple-interconnection process (Lyu et al., 2022). In these cases the optimization problem is subject a set of constraints.

In this paper we present a self-adaptive constraint handling technique coupled to a Differential Evolution algorithm. This approach computes the weight factors based on the magnitude of the violation and the number of constraints not satisfied for each individual during the optimization process. In other words, when the amount of violation is large, a strong penalization is applied, and when the violation is smaller, a smaller penalization is used. Additionally, for each individual, if the number of violated constraints is high, the penalization factor is strong, and it is weak when the number of violated constraints is small. In our proposal, the dynamic threshold searches for feasible individuals and it is only updated when the whole population accomplishes every constraint. This dynamic threshold approach requires the transformation of equality constraints into inequality constraints.

We present validation of the proposed technique with five benchmark problems and three typical chemical engineering problems that have solutions in continuous spaces.

The benchmark problems selected are representative of the nature of chemical engineering problems. The first problem corresponds to an NLP problem, this problem has five variables and three equality constraints and taken from Deb et al. (2000). The second problem is a NLP problem with four variables, two linear inequality constraints and three nonlinear equality constraints. This problem has been previously studied in Yiqing et al. (2007). The third problem is taken from Diwekar et al. (1992), this MINLP problem and it contains five variables, four inequality constraints and five equality constraints and represents two reactors to minimize the cost of producing a desired product. The fourth problem is a non-convex MINLP problem also studied in Summanwar et al. (2002), it has three binary and two continuous, in this problem the two equality constraints are the sources of nonconvexities for this example. The fifth problem considered a heat exchanger network synthesis problem requiring minimization of the total cost, has formulated it as a one variable problem with 12 constraints proposed by Summanwar et al. (2002). The typical chemical engineering problems selected are a heat exchanger network, a reactor network and an absorption process. These case studies were taken from Edgar et al. (2001). For heat exchanger network the total annual cost was minimize and is subject to a logarithmic mean temperature difference. In the reaction network the objective function was minimization of the total volume, subject to an overall conversion. Finally, in the absorption process, we optimize the flowrate of the absorber agent subject to the absorbent compositions. Furthermore, we present the implementation of the proposed technique for the rigorous optimization of a distillation train through evaluation of the objective function with Aspen One. This problem is non convex, non-linear, multivariable subject are equality and inequality constraints.

The paper is organized as follows. In Section 2, we introduce the new algorithm coupled to a Differential Evolution algorithm and implementation of the optimization process. In Section 3, we show the results of the benchmark functions, the explicit chemical engineering problems and, finally, the optimization and statistical analysis of the distillation train. In Section 4, we discuss the results and in Section 5 we present our conclusions.

2. Optimization algorithm and constraint handling

Constrained optimization problems are characterized by the fact that the solution must be found inside the feasible region which is delimited by the equality and/or inequality constraints. This means that the search for solutions favors those individuals that better meet the constraints and that also maximize or minimize the objective function.

Generally, a constrained nonlinear optimization problem is expressed as follows:

Find vector

$$Z = (z_1, \ldots, z_D), \quad Z \in R^D$$

To minimize fitness function

$$f(Z)$$

Subject to equality constraints

$$h_j \quad (Z) = 0 \quad j = 1, \ldots, \quad n$$

Subject to inequality constraints

$$g_k \quad (Z) \leq 0 \quad k = 1, \ldots, \quad m$$

And subject to boundary constraints

$$z_i^{(L)} \leq z_i \leq z_i^{(U)} \quad i = 1, \ldots, D$$

Thus, the optimization goal is to find a feasible vector Z, composed of D dimensions, to minimize the fitness function $f(Z)$ accomplishing the equality $h_k \quad (Z)$ and inequality constraints $g_j \quad (Z)$. Evolutionary algorithms need to be coupled to a constraint handling technique that allows them to find feasible solutions in a constrained search space. The chosen technique plays an important role in the quality of the solutions delivered and its computation time.

Techniques that favor the feasible solutions are named overpenalized and may lead to a fast convergence to a feasible solution but it may be suboptimal. Premature convergence means that the search space is not adequately explored during the initial optimization process (Ben, 2016). On the other hand, techniques that favor the infeasible solutions are named underpenalized and can lead to a slow convergence to the feasible zone or even no feasible solution at all.

A weighting function (WF) technique is the more usual way to handle constraints in chemical engineering. Nonetheless, finding the correct weights is an issue which we propose to overcome via self-adaptation. In this technique, $f(Z)$ is the fitness function penalized by the weighted sum of

constraint violations, w_j are the factors that we need to find and $f'(Z)$ is the new optimization function.

$$f'(Z) = f(Z) + \sum_{j=1}^{m} w_j \cdot \max(0, G_j \quad Z)$$

In this case, for the WF technique to handle the constraints we used two sets of penalties (Leboreiro and Acevedo, 2004). A major penalty was applied when the individual was infeasible in evaluator (non-convergence evaluator), while a successful simulation was just slightly penalized. One is used to penalize infeasible individuals ($w_1 = 1000$) and the other penalizes each time the constraints are not met ($w_2 = 100$). Otherwise, in both cases the weight factor value is 0. According to the experiments realized, these values perform well for most problems.

Larger or smaller values of w_j lead to the overpenalization or underpenalization described. In chemical engineering, some authors have proposed constraint handling techniques based on the WF approach that can be applied to problems in continuous spaces (Teh and Rangaiah, 2003; Babu and Angira, 2006; Kheawhom, 2010; Zhang and Rangaiah, 2012). For this reason, it is necessary to design a constraint handling technique capable of coupling with evolutionary algorithms to optimize complex problems with mixed variables and highly nonlinear models.

2.1. *Self-adaptive constraint handling technique*

Generally, the optimization of chemical engineering problems uses design constraints in discrete and continuous search spaces are multivariable and multimodal; in addition, the objective function is highly nonlinear and non-convex subject to nonlinear equality and inequality constraints. Therefore, robust constraint handling techniques, which consider the information of non-feasible and feasible individuals to favor the optimization process, are required.

Next, we present a constraint handling technique that can be coupled to evolutionary algorithms, in this case Differential Evolution (DE). This approach includes four different penalization criteria and transforms equality constraints into inequality constraints through a dynamic threshold involving self-adaptive factors. Fig. 1 shows the general self-adaptive (SA) penalization approach.

A. Infeasible individuals

In the first step, individuals are tested for feasibility and the death penalty applied for non-convergence in the evaluator ($viola^{convergence} = 1$). This penalty in only applied to non-feasible individuals, and the worst value (f^{max}) is assigned to the fitness function ($f^{fitness}$) and a poor value (s) is assigned at all equality constraints (h_j). Otherwise, the individual is evaluated to obtain f^{total} and the constraint values ($h_j^{reached}(z)$). This step is shown in lines 2–14 in Fig. 1.

B. Self-adaptive dynamic threshold

In the second step, the main objective is to search areas of the feasible region through relaxing and stiffening of the constraints, to determine the directions of movement through the dynamic threshold ε.

In other approaches, this ε uses as level comparison between the aptitudes of the parent and the child (Takahama and Sakai, 2010). In this proposal ε is used to relax-stiff constraints and is progressively reduced throughout the optimization process, additionally transforms equality constraints into inequality constraints, as shown below:

$$h_j^{target}(z) - \quad h_j^{reached}(z) = 0$$

is treated as:

$$|h_j^{target}(z) - \quad h_j^{reached}(z)| > \varepsilon$$

Where: h_j^{target} is the set point of the constraint (target value) and $h_j^{reached}(z)$ is the value of the constraint reached, for each individual. In our approach, the constraint is considered as accomplished when $|h_j^{target}(z) - h_j^{reached}(z)|$ is within the interval $(-\varepsilon; \varepsilon)$ and $viola^j = 0$. The threshold ε reduces when the whole population accomplishes all constraints, $viola^{total} = 0$, $f^{constraints} = 0$ and $f^{pen} = f^{total}$.

The progressive reduction of epsilon is defined by the following equation:

$$\varepsilon^{update} = c^*\varepsilon$$

Where: ε is the current value of the dynamic threshold, c is the progressive reduction factor and ε^{update} is the new value of the dynamic threshold. The factor c is adjusted between 0.5 and 0.9. For each problem, this value must be empirically adjusted. According to the experiments realized, a value of 0.8 performs well for most problems. In this implementation, the initial value of ε is 0.5.

An important aspect considered is when $viola^{total} > 0$; in this case, we calculated the self-adaptation of penalty factors that depend on the degree of deviation of the constraint between the reached value ($h_j^{reached}(z)$) and the target value ($h_j^{target}(z)$). For each individual, the penalty factors for each constraint are calculated according to the following expression:

$$w_j^{constraints} = b^*(h_j^{target}(z) - \quad h_j^{reached}(z))^2$$

Notice that the penalty factor is computed using the residual value ($h_j^{target}(z) - h_j^{reached}(z)$) raised to the power 2 multiplied by coefficient b, for each equality constraint, $w_j^{constraints}$. The power 2, guarantees positive values and multiplying by b enables the handling of similar orders of magnitude in order to establish significance in the penalty method. When the residuals are large, a strong penalization is applied. For a small violation, the penalization is weak (see Fig. 1, lines 15–28). This step allows the best fitness function routes to be powered and no promissory search zones to be segregated by the algorithm.

C. Self-adaptive penalties

In addition to the magnitude of the violation, our technique also involves counting the number of unmet constraints, $viola^{total}$, and computes f^{num_pen}. The penalization is carried out as a function of the number of constraint violations, $viola^{total}$; the fitness function is penalized by a factor proportional to the amount the constraint is violated, w^{num_pen}. For each individual, if the number of violations is small the penalization is weak, and it is strong when the number of violations is high (see Fig. 1, lines 29–36).

Static penalties and self-adaptive penalties are applied in our proposal (Homaifar et al., 1994; Coello, 1999). In this kind of penalization, the user defines several levels of violation and penalty coefficients (penalty factors) for each constraint. Thus, the penalty coefficient increases as higher levels of constraint violations are experienced.

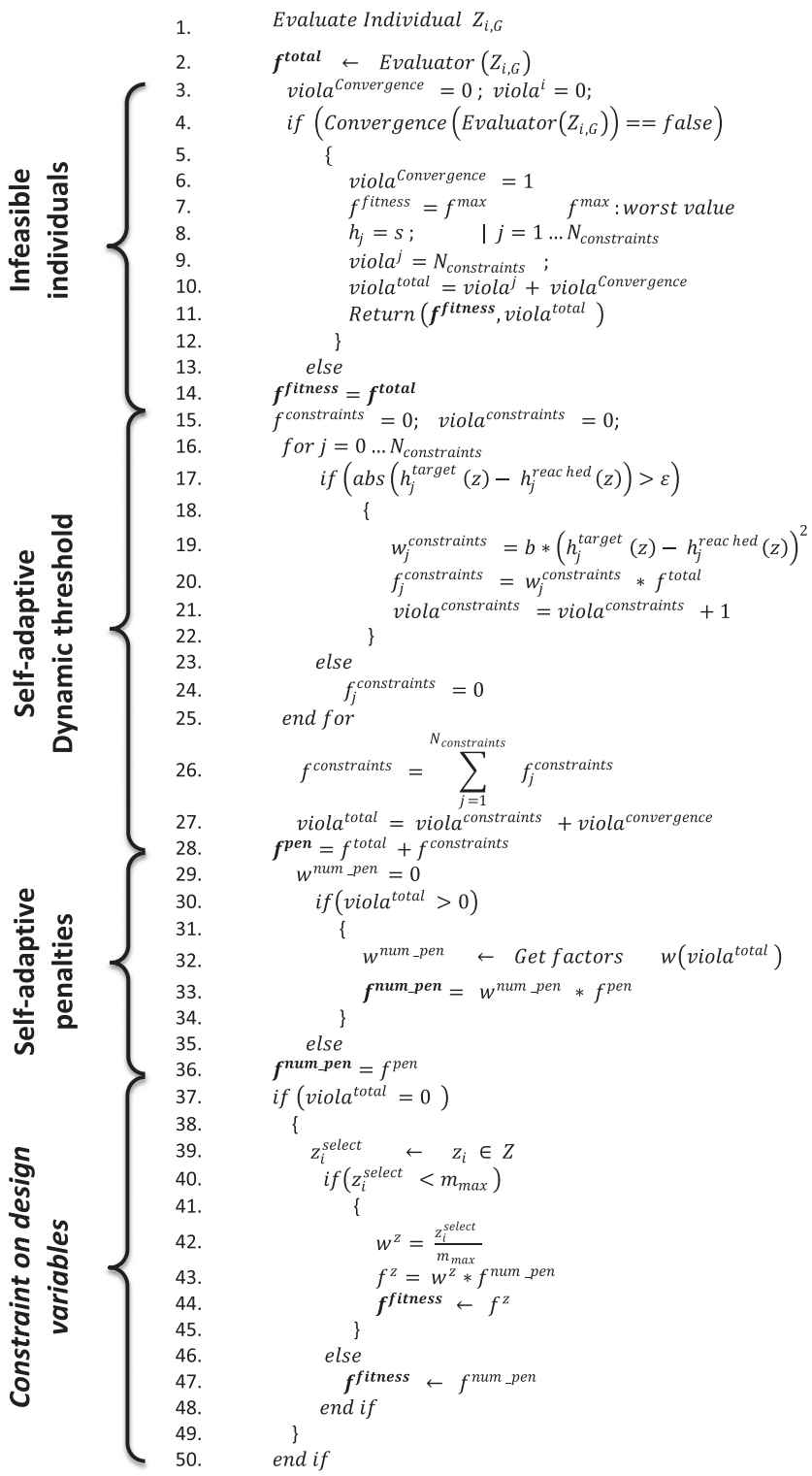

Fig. 1 – General self-adaptive penalization scheme.

These penalties are represented by integer factors, initialized to 0 and incremented by 1 for each constraint of the problem that is violated, regardless of the amount of violation. Notice that only the number of constraints violated is considered in the variable $viola^{total}$, but not the magnitude by which each constraint is violated. Through this combination of penalties, the evolutionary algorithm has enough information about how several constraints are violated, but also about the amount of constraints unaccomplished.

D. Constraint on design variables

In several problems, there are design variables that are bounded by a maximum value m_{max}, usually defined heuristically. This value is not a constraint but will be considered as an important part of the nature of the problem. We apply this type of penalty to guide the search towards solutions that find the appropriate values of this design variable, z_i^{select}.

This kind of penalization is applied depending on the nature of the optimization problem. For instance, if minimization of the fitness function is pursued, the penalty

method decreases the final fitness function value only if all restrictions lie within the dynamic threshold, ε. The fitness function using the penalty is f^{num_pen}. So, the reduction of the fitness function depends on the relationship between z_i^{select} and m_{max}. This favors intensification of the promising area, since this relationship is only between 0 and 1. In the last step, the evolutionary algorithm returns $f^{fitness}$ (see Fig. 1, lines 37–50). When any restriction is not met, this stage does not apply and $f^{fitness} = f^{num\_pen}$.

This technique allows to establish superiority over feasible points and to partially order them in order to guide the search towards individuals that decrease the objective function and that also implicitly minimize some of the dimensions (design variables) of the individual. To explain the implementation of this technique we will use as an example the case study of the distillation train. It is important to mention that in the optimization of distillation columns we know that a decrease in the thermal load, increases the number of stages of the column, which is why some authors have proposed to perform a multi-objective optimization with stochastic algorithms (Vázquez-Ojeda et al., 2013; Gómez - Castro et al., 2015; Medina - Herrera et al., 2017; Palma - Barrera et al., 2019; Contreras - Zarazúa et al., 2021).

Our implementation allows using a uniobjective algorithm, where the constraints can be additional objectives in the optimization process and it is possible to find zones that meet the decrease of both the heat duty and the number of total stages of the separation system.

In the case study of the distillation train, we assign as "constraint on desing variable" the total number of *NT* stages of each individual. *NT* is the sum of the number of stages of all the columns involved in the separation scheme. For this last stage, it is necessary to set a heuristic maximum value related to the feasibility of construction of the distillation column (mmax). The value of m_{max} is used in two stages, in the first one to compare it with *NT* and in the second one to calculate the penalty factor w_z. In the comparison between mmax with *NT*, when the value of *NT* is less than that of m_{max} then the fitness function should be penalized. The penalty factor is determined from the quotient between *NT* and m_{max}, and a value less than 1 is obtained. Therefore, when the fitness function is penalized, the numerical value will be further decreased. In a minimization problem, the individuals that have the highest probability of being chosen are those with a lower value of fitness function. So, implicitly, with this strategy the selection pressure increases for those individuals that minimize the objective function and also implicitly decrease the number of total stages of the distillation train. Thus, we define a superiority of feasible points, i.e., when the child faces the parent, the one with the lowest fitness function value and therefore the lowest number of total stages in the separation system will be chosen. This strategy intensifies the search for the feasible region where the objective function is minimized and also reduces the number of design stages of the separation scheme. Our proposal can include more than one design variable in the constraints. This allows to include some additional objective as a constraint in a uniobjective algorithm. Recently, we have implemented this technique to decrease the pressure and the number of total stages in oscillating pressure distillation systems, obtaining quite favorable results.

It is important to mention that in this penalty technique none of the design variables (dimensions) of the individual is altered, the only thing that is favored is the intensification in feasible regions that minimize the thermal load and also the total number of stages.

In the area of evolutionary computation, when developing constrained optimization algorithms it is necessary to design a constraint handling technique that allows the algorithm, during the optimization process, to find the feasible zone. In this sense, the techniques that have been recently developed consider sophisticated elements such as the implementation of as a repair algorithm based on the gradient information as rank-based selective pressure strategy (Fister et al., 2021) or basic concepts such as weight function. However, an important point of discussion in the area of evolutionary computation is the determination of additional parameters corresponding to the constraint handling technique. Traditionally, the tuning of these parameters depends on the nature of the optimization problems to be solved. That is, the technique is adapted to the characteristics of the problem, which limits its generalization, as expressed by Fister et al. (2021), this is because the algorithms are adapted to the type of functions to be optimized.

In particular, our technique uses 3 user-defined parameters: b, ε and c. These parameters depend on the nature of the problem to be solved.

- Parameter b is assigned a constant value during the whole optimization process, which guarantees that the order of magnitude of $f_{constraints}$ is equal to that of $f_{fitness}$. This parameter allows the penalty to be significant according to the degree of constraint violation. That is, if it violates a lot, it penalizes strongly, otherwise it penalizes softly.
- The parameter ε defined as dynamic threshold is dependent on the nature of the problem. The function of this parameter is to relaxed the constraints at the beginning of the optimization process, since it will accept as feasible individuals those that at that moment comply with the relaxed constraint. However, during the optimization process, the value of ε is updated, turning the constraints rigid, leading the search towards feasible zones.
- The parameter c is a constant value throughout the optimization process and corresponds to a progressive reduction factor of ε. Values of this parameter are between 0 and 1. Values very close to 1 cause epsilon to reduce slowly. Values close to 0 increase the rate of epsilon reduction. This parameter helps to control the selection pressure, avoiding premature convergence.

The weighting factors between f_{total} and $f_{constraints}$, which do not require initial values, are calculated from the parameters b and These parameters must be modified depending on the nature of the problem. According to our experience these proposed values obtain optimized values in more complex configurations, and it is not necessary to modify them. However, it is necessary to perform the tuning of the parameters of the proposed technique along with the chosen optimization algorithm.

It should be noted that the technique has no dependence on the optimization algorithm, so it can be coupled to any algorithm of the user-define considering the tuning of the parameters of both the optimization algorithm and the constraint handling technique.

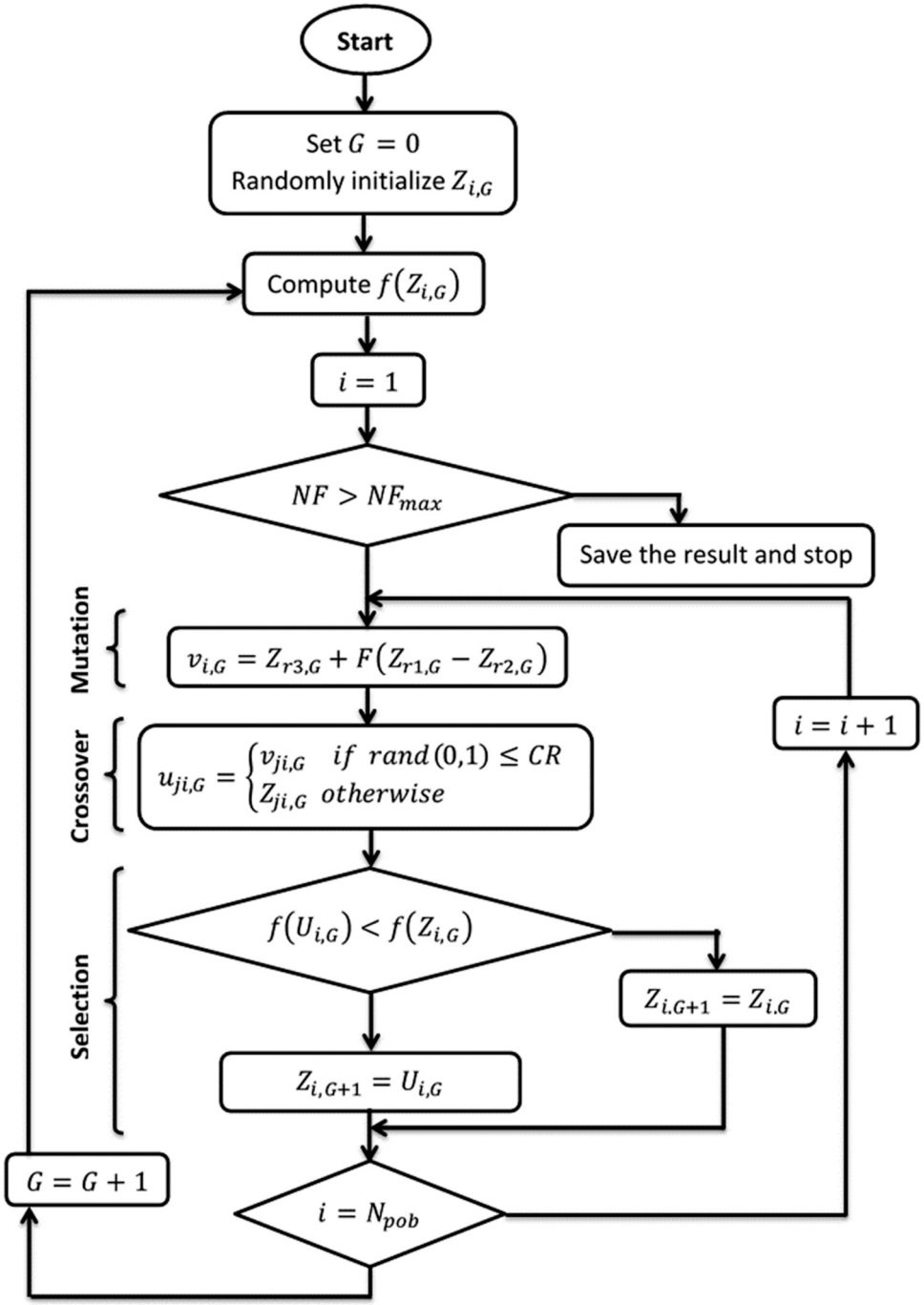

Fig. 2 – Flowsheet of Differential Evolution algorithm.

2.2. *Differential evolution algorithm*

The classic DE algorithm consists of four stages: initialization, mutation, crossover and selection (Storn and Price, 1997). As can be seen in Fig. 2, once the initial population is generated, the optimization loop is performed. Notice that the mutation, crossover and selection stages are carried out to generate the new individuals of each generation. Hereafter, these stages are performed continuously until the specified stopping criterion is satisfied. In this case, the number of function evaluations is established as the stopping criterion. This criterion is the most commonly used in evolutionary algorithm literature (Zielinski and Laur, 2008). In this case, we selected the number of iterations as stopping criteria, however, it is possible to implement a stop criterion for elitism, if after "n" number of generations the fitness function of elite individual remains constant. Other case is if the epsilon value decreases by more than 10–3 of all or part of the total population. Finally when some parameter of the evolutionary algorithm.

In DE, the initial population of individuals is randomly generated within the boundary constraint of each design variable (search space). Each individual i contains characteristic decision variables D and is called a decision vector $Z_{i,G}$. All individuals N_{pob} in each generation G are selected as parents and each of them is processed as described below: in general, in the mutation process, three individuals are randomly chosen as parents ($Z_{r1,G}$, $Z_{r2,G}$, $Z_{r3,G}$), except the parent in the processing ($Z_{i,G}$).

The base vector, $Z_{r3,G}$, is the first individual and the remaining two are used to obtain the residual $(Z_{r1,G} - Z_{r2,G})$ that is then scaled by the scale factor F and added to the base vector; this means that the resulting vector is recombined with the parent. The recombination probability is controlled by the crossover factor CR. The result of the crossover process produces a trial vector, $v_{i,G}$. Finally, in the survivor selection operator, the trial vector is accepted in the generation $G + 1$ if the trial vector is better than the parent (see Fig. 2).

The mutation process is performed based on the distribution of the solutions in the current population. In this way, search directions and possible step sizes depend on the location of the individuals selected to calculate the mutation values. The CR parameter controls the influence of the parent in the generation of the offspring. Higher values mean less influence of the parent. The F parameter scales the influence of the set of pairs of solutions selected to calculate the mutation value. The mutation and crossover steps together with the selection step constitute one generation or iteration of the DE algorithm. The procedure is repeated until the specified stopping criterion is satisfied.

2.3. *Constrained optimization procedure*

In the constrained optimization process, the DE algorithm was coupled to our self-adaptive constraint handling technique (DE-SA) and weighting penalty function (DE-WF), for comparison of results. In this optimization process, the DE algorithm requires the individuals to be physically feasible in all generations. An individual being physically feasible implies that it satisfies the problem and that it is possible to calculate the objective function through an evaluator.

For the first generation, in this work the population was randomly generated as follows:

$$z_{i,l,G+1} = \begin{cases} z_l^{(L)} + random_j \quad [0,1] \quad (z_l^{(U)} - z_l^{(L)}) \\ if \quad z_{i,l,G+1} < z_l^{(L)} \quad or \quad z_{i,l,G+1} > z_l^{(U)} \\ z_{i,l,G+1} = z_l^{(L)} + |z_{i,l,G+1} - z_l^{(L)}| \\ z_{i,l,G+1} = z_l^{(U)} - |z_{i,l,G+1} - z_l^{(U)}| \end{cases}$$

where:

$i = 1, \ldots, N_{pob} \quad l = 1, \ldots, D$

Through this approach, the set of variables that comprises each individual, i, is kept within the boundaries established. In this case, N represents the size of the population for each generation and $z_l^{(L)}$ and $z_l^{(U)}$ represent the lower and upper limits of the variables or dimensions of problem, respectively. This allows to replace the individuals with variables out boundaries for individuals that if accomplishing them.

Explicit and implicit fitness function problems are treated here. Explicit fitness functions are benchmark functions and typical chemical engineering problems. The distillation train case is an implicit fitness function. The constrained optimization procedure is different in each case, as described below:

- The constrained optimization process for the case of the benchmark functions selected for CEC 2006 and chemical engineering functions was implemented in MATLAB. The DE algorithm generates the population N_{pob} in each generation G and this is evaluated to obtain the objective function, f^{total}. The evaluated individuals are analyzed according to the constraints violated, for each constraint handling technique. This information is added to the penalized fitness function $f^{fitness}$ which is used by the algorithm to generate the new population $G + 1$. This process is repeated until the stopping criterion is reached.
- The distillation train optimization process requires the implementation of an interface between MATLAB, Excel and Aspen One©. The DE algorithm is coded in MATLAB, the SA constraint handling technique in Excel and the fitness function evaluator in Aspen One©. The process optimization consists of the DE algorithm generating the population, N_{pob}, for each generation, G. Each individual, $Z_{i,G}$, is sent to Aspen One to simulate the distillation train to obtain the objective function, f^{total}, and the purities and recoveries for each component x_i^{pur} and x_i^{rec}. This information is used to determine the fitness function penalization $f^{fitness}$. The fitness function value is returned to MATLAB to generate the new population, $G + 1$. This process is repeated until the stop criterion is met. A detailed description of the implementation of the SA constraint handling technique for the distillation train is in Appendix A. Notice that the constraints are not treated within the Aspen simulator, but in Excel where the implementation of the constraint handling technique was performed. In this case, our optimization proposal is based on the evaluation of the objective function through a sequential modular simulator Aspen Plus, so the optimization computing time considers three stages: the evaluation of the objective function (Aspen Plus), the evaluation of the constraint handling technique (Excel) and the optimization process of the DE algorithm (MatLab). Of these three, the most time-consuming stage is the one performed by Aspen Plus due to the time required to achieve convergence of the simulation and obtain the value of the objective function, as stated by Kiss et al. (2012). In our case optimization computing time corresponds to the evaluation time of each individual and is 4 s

3. Case studies and results

This work presents an SA technique based on a dynamic threshold and self-adaptation. This constraint handling technique has been coupled to a DE algorithm (DE-SA). We compared the performance of the SA technique with results obtained by weight function penalization (DE-WF) for the same problems. The DE algorithm is coupled to the two techniques. Thus, the performance of DE-SA versus DE-WF is particularly influenced by the penalization technique instead of the DE algorithm. The values of the parameters for the DE algorithm were: $CR = 0.8$ and $F = 0.85$. These are typical values used in the literature (Wong and Dong, 2005). The DE-SA and DE-WF algorithms were run 30 times for each problem. In the distillation train case, we performed a statistical comparison of the results.

We optimized five benchmark functions with constrained in continuous space, proposed in CEC 2006 and solved by Kheawhom (2010), to validate our SA technique. In addition, we analyzed three explicit chemical engineering problems proposed by Edgar et al. (2001). These case studies represent complex optimization problems, from constrained NLP to constrained non-convex MINLP.

Table 1 – Benchmark functions for constraint optimization.

Test 1	***Test 2***
$min \quad f_1 = e^{x_1 x_2 x_3 x_4 x_5}$ $st. \quad x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10$ $x_2 x_3 - 5x_4 x_5 = 0$ $x_1^2 + x_2^2 + 1 = 0$ $-2.3 \le x_i \le 2.3, i = 1,2$ $-3.2 \le x_i \le 3.2, i = 3,4,5$	$min \quad f_2 = 3x_1 - 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3$ $st. \quad 1000\ \sin(-x_3 - 0.25) + 1000\ \sin(-x_4 - 0.25)$ $+ 894.8 - x_1 = 0$ $1000\ \sin(x_3 - 0.25) + 1000\ \sin(x_3 - x_4 - 0.25)$ $+ 894.8 - x_2 = 0$ $1000\ \sin(x_4 - 0.25) + 1000\ \sin(x_4 - x_3 - 0.25)$ $+ 1294.8 = 0$ $x_1^2 + x_2^2 + 1 = 0$ $0 \le x_i \le 1200, \quad i = 1,2$ $-0.55 \le x_i \le 0.55, i = 3,4$
Test 3	***Test 4***
$f_3 = 7.5y_1 + 5.5y_2 + 7v_1 + 6v_2 + 5x$ $st. \quad y_1 + y_2 = 1$ $z_1 = 0.9[1 - e^{-0.5v_1}]x_1$ $z_2 = 0.8[1 - e^{-0.4v_2}]x_2$ $z_1 + z_2 = 10$ $x_1 + x_2 = x$ $v_1 \le 10y_1$ $v_2 \le 10y_2$ $x_1 \le 20y_1$ $x_2 \le 10y_2$ $x_i, y_i, z_i \ge 0 \quad i = 1,2$ $y_i \in \{0,1\} \quad i = 1,2$	$f_4 = 2x_1 + 3x_2 + 1.5y_1 + 2y_2 - 0.5y_3$ $x_1^2 + y_1 = 1.25$ $x_2^{1.5} + 1.5y_2 = 3$ $x_1 + y_1 \le 1.6$ $1.333x_2 + y_2 \le 3$ $-y_1 - y_2 + y_3 \le 0$ $x_i \ge 0 \quad i = 1,2$ $y_i \in \{0,1\} \quad i = 1,2,3$
Test 5	
$f_3 = 3.5y_1 + y_2 + 1.5y_3 + 7b_1 + b_2 + 1.2b_3 + 1.8a - 11c$ $b_2 - \ln(1 + a_2) = 0$ $b_3 - 1.2\ln(1 + a_3) = 0$ $c - 0.9b = 0$ $b_1 + b_2 + b_3 - b = 0$ $a - a_2 + a_3 = 0$	$b - 5y_1 \le 0$ $a_2 - 5y_2 \le 0$ $a_3 - 5y_3 \le 0$ $c - 1 \le 0$ $b_2 - 5 \le 0$ $a, a_i, b_i, c \ge 0 \quad i = 1,2,3$ $y_i \in \{0,1\} \quad i = 1,2,3$

Finally, the SA technique was implemented in a more complex search space. This technique was implemented to optimize a multicomponent distillation scheme based on MESH equations. This mathematical model is highly non-linear, non-convex and multimodal, with continuous and discrete variables subject to equality and inequality constraints. In this problem, the fitness functions were evaluated in Aspen One using a computer with the following characteristics Core i7 processor, 2.5MGz and 8 GB RAM.

3.1. *Optimization of benchmark functions*

In this case, the five benchmark functions shown in Table 1 were optimized. These benchmark functions have been widely used as optimization tests (Kheawhom, 2010). The first benchmark represents an NLP problem. The second benchmark is the modeling of an NLP problem with four variables, two linear inequality constraints and three non-linear equalities. The third benchmark is an MINLP problem. The fourth benchmark represents a non-convex MINLP

Table 2 – Optimization results.

Test	Function type	Fitness function ($f^{fitness}$)		
		Kheawhom (2010)	Weighting function (DE-WF)	Self-adaptive (DE-SA)
1	NLP	0.0539498	0.0646498	0.0539498
2	NLP	5126.5	5127.42	5126.5
3	MINLP	99.245209	99.5209	99.245209
4	Non-convex MINLP	7.66718	7.928	7.66718
5	MINLP	– 1.923098	– 1.9897	– 1.923098

problem, while the fifth case is formulated as an MINLP problem that contains three integer variables and five continuous variables and several equality and inequality constraints. In this case, 20,000 function evaluations and 100 individuals per generation were used in the optimization process. These problems are representative of the nature of some chemical engineering problems.

3.1.1. Results

Table 2 shows the optimization results for the five benchmark functions. The optimization based on dynamic self-adaptive constraint handling (DE-SA) obtained the same optimal values of $f^{fitness}$ reported in the literature (Kheawhom, 2010) for all cases evaluated. The values determined with the weight function penalization technique (DE-WF) are larger than the values reported, particularly in the last two problems that correspond to MINLP problems. SA and WF use the same stop criterion as Kheawhom (2010) to validate the results.

Kheawhom (2010) performed optimization of the benchmark functions using a DE algorithm coupled with a repair algorithm for constraint handling. Comparing the results obtained using DE-SA, DE-WF and those obtained by Kheawhom (2010), a large difference in the number of function evaluations used to optimize the benchmark functions was observed. In fact, the optimization performed by DE-SA required 20,000 function evaluations, which corresponds to using a population of 100 individuals and 200 generations. Comparatively, the optimization performed by Kheawhom (2010) required 2,000,000 function evaluations, which results from a population of 1000 individuals and 2000 generations.

On the other hand, the optimal value of the fitness function was obtained in generation 35 in the case of DE-SA whereas the optimal value obtained by Kheawhom (2010) was determined in generation 120. Therefore, a considerable reduction of numerical effort and computing time was required by DE-SA. This demonstrates the robustness and efficiency of the constraint handling technique to deal with problems with different levels of complexity, from NLP problems to highly nonlinear and multivariable optimization problems with strong interactions between the variables, such as non-convex MINLP with inequality and equality constraints.

3.2. Optimization of typical chemical engineering problems

The following case studies were optimized: a heat exchanger network, a reaction network and an absorption process. These case studies were taken from Edgar et al. (2001) and formulated as minimization problems. Minimization of the total annual cost was established as the fitness function in the heat exchanger network optimization problem, subject to a logarithmic mean temperature difference (*LMTD*). The fitness function for the reaction network optimization problem was minimization of the total volume (V_T), subject to an overall conversion of 0.9 for X_{A3}. The flowrate minimization of the absorber agent (solvent) was defined as the fitness function of the absorption process optimization problem subject to the absorbent compositions y_1 and y_2 being within the range of y_0 to y_3.

These case studies are composed of continuous search spaces and nonlinear fitness functions. In addition, these optimization problems are subject to equality and inequality constraints. Likewise, both case study 2 and case study 3 have an important dependence between the optimization variables due to reaction rates and equilibrium ratios, respectively. The formulation of the optimization problems is shown in Table 3. In this optimization, a population of 60 individuals per generation and 3000 function evaluations were used.

3.2.1. Results

Fig. 3 shows the optimization results for both cases, DE-SA (blue line) and DE-WF (red line), and they are compared with the optimal value determined by Edgar et al. (2001) using a deterministic method (green line).

According to the behavior depicted in Fig. 3, it was observed that DE-SA found the optimal solution in around 650 function evaluations, while DE-WF obtained the optimal value in about 1800 function evaluations. Thus, it is evident that this performance of DE-SA significantly reduces the computational effort by more than 60%. In fact, DE-SA obtains the best solution in between 1 and 3 min, while obtaining the optimal value with DE-WF takes between 3 and 5 min. So, at least 50% reductions of computation times are obtained.

Table 4 shows the optimization results determined using DE-SA and DE-WF and the corresponding results reported by Edgar et al. (2001). According to the results for the three case studies, the solutions obtained using the SA technique were very similar to the optimal values reported previously by Edgar et al. (2001), while those obtained using the WF technique experienced a larger deviation in the optimal value and greater computing effort was required.

The statistical analysis showed that the SA algorithm is robust and efficient since the same optimal value was determined in the 30 runs achieved. So, it is evident that the SA technique is a worthy tool since it is able to deal with optimization problems subject to equality and inequality constraints and interdependence on the optimization variables, as the separation schemes.

3.3. Distillation train

In this case study, a distillation process (Fig. 4) is used to split a mixture made of four lineal aliphatic hydrocarbons: n-butane, n-pentane, n-hexane and n-octane. A flow rate of 45.36 kmol/h is introduced in the first column as a saturated liquid. The feed composition is as follows: 0.45 for intermediate components and 0.05 for the other components. The design specifications for the purity and recovery for each component were established as 0.987 for A, 0.98 for intermediate components and 0.986 for the last component. The design pressure for the separation was chosen to ensure using

Table 3 – Typical chemical engineering case studies.

Heat exchanger network

$$CA = \left(\frac{350}{na}\right)\left[\frac{w_1 cp_1(T_{1S} - T_{1E})}{U * \Delta T_{ML,1}} + \frac{w_2 cp_2(T_{2S} - T_{2E})}{U * \Delta T_{ML,2}}\right]^{0.65} + \left[\frac{0.005 * 8500}{l_{vap}}\right]\left(w_1 cp_1(T_{1F} - T_{1S})\right)$$

$$\Delta T_{ML,2} = \frac{(T_{2E} - T_{1S}) - (T_{2S} - T_{1E})}{\ln\frac{(T_{2E} - T_{1S})}{(T_{2S} - T_{1E})}}$$

$$\Delta T_{ML,i} \geq 10; \qquad i = 1,2$$

$$T_{1E} > T_1$$

$$T_1 > T_{1F}$$

$$\frac{(T_{2E} - T_{1S})}{(T_{2S} - T_{1E})} \neq 0$$

Data	Value
T_{vap}	212 °F
λ_{vap}	1000 Btu/lb
T_{1E}	77 °F
T_{1F}	200 °F
Cp_1	0.9 Btu/lb °F
Cp_2	0.1 Btu/lb °F
U	200 Btu/h ft^2 °F

Absorption process

$$L_T = \frac{G_o(y_o - y_1)}{x_1} + \frac{G_o(y_1 - y_2)}{x_2} + \frac{G_o(y_2 - y_3)}{x_3}$$

$$y_i = \frac{x_i^2}{100}$$

$$y_0 > y_1$$

$$y_1 > y_2$$

$$y_2 > y_3$$

$$y_0 = 0.03\frac{lb\ sal}{lb\ solvent}$$

$$y_3 = 0.0001\frac{lb\ sal}{lb\ solvent}$$

$$y_0 > y_1, y_2 > y_3$$

Data	Value
Q	1000 l/h
C_{AF}	1 gmol/l
X_{AF}	0
K_{10}	3×10^7 min^{-1}
K_{20}	4×10^{19} min^{-1}
E_1	11,600 cal/gmol °K
E_2	29,500 cal/gmol °K

Reactor network

$$V_1 = \frac{Q\ C_{AF}(X_{A1} - X_{A2})}{-r_{A1}}$$

$$V_2 = \frac{Q\ C_{AF}(X_{A2} - X_{A3})}{-r_{A2}}$$

$$V_3 = \frac{Q\ C_{AF}(X_{A3} - X_{AF})}{-r_{A3}}$$

$$V_T = \sum_{i=1}^{3} V_i$$

$$V_T \neq 0$$

$$V_i \neq 0 \qquad i = 1,2,3$$

Irreversible 1st order

$$A \longleftrightarrow B$$

$$-r_{Ai} = C_{AF}K_{10}\left(Exp\frac{(-E_1)}{RT}\right) - C_{AF}X_{Ai}\left[K_{10}\left(Exp\frac{(-E_1)}{RT}\right) + K_{20}\left(Exp\frac{(-E_2)}{RT}\right)\right]$$

cooling water in the condensers. The phase equilibrium for the liquid phase was determined using the Chao–Seader model, since the system is a hydrocarbon mixture to be separated at low or medium pressure (Aspen Plus 13.0, 2007).

The optimization problem is the following:

Find vector

$$Z = (\mathbf{N}_{T,B1}, \mathbf{N}_{F,B1}, \mathbf{N}_{T,B2}, \mathbf{N}_{F,B2}, \mathbf{N}_{T,B3}, \mathbf{N}_{F,B3}, \mathbf{RR}_{B1}, \mathbf{RR}_{B2}, \mathbf{RR}_{B3})$$

To minimize fitness function

$$Q = f(Z)$$

Subject to constraint function

$$h_{1,j}\ (Z) = x_{target}^{pur} \quad j = 1, \ldots, \quad M_{constraints}$$

$$h_{2,j}\ (Z) = x_{target}^{rec} \quad j = 1, \ldots, \quad M_{constraints}$$

And subject to boundary constraints

$$z_i^{(L)} \leq \quad z_i \quad \leq z_i^{(U)} \quad i = 1, \ldots, D$$

$$2 \leq N_{F,Bl} \leq N_{T,Bl} l = 1, \ldots, B_{columns}$$

$$N_{S1}, \neq N_{S2} \neq N_F$$

$$x_i^{pur} = x_0^{pur}$$

$$x_i^{rec} = x_0^{rec}$$

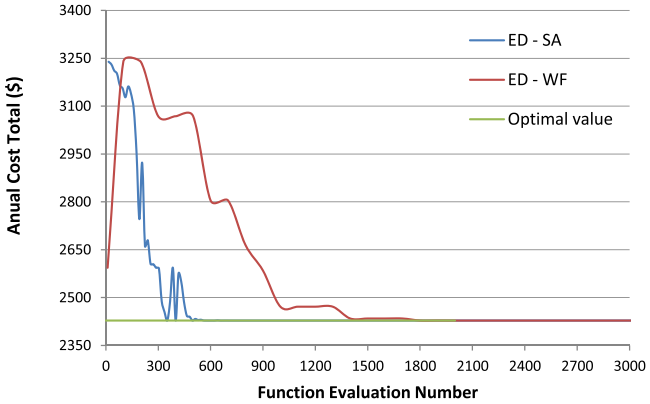

Fig. 3 – General performance of the optimization process in a typical chemical engineering case.

where: N_T is the total number of stages, N_F is the feed stage and RR is the reflux ratio. N_T and N_F represent discrete variables, while RR represents a continuous variable. Consequently, there are nine dimensions, which represent six discrete variables and three continuous variables.

Minimization of the total heat duty (Q) in the reboiler is the fitness function of the problem. Notice that Q is a function of the vector Z. The design specification for purity (purity constraint) is defined as x_{target}^{pur} and that for recovery (recovery constraint) is given by x_{target}^{rec}. As before, in order to conduct a comparative analysis, optimization of the problem was performed using DE-SA and DE-WF, respectively. In this case, a population of 100 individuals and 200 generations were used. The CPU time of the optimization process corresponds to the total time required to evaluate 20,000 individuals and is approximately 22 h. A high percentage of the time corresponds to the long convergence time required by the Aspen Plus simulator that ensures rigorous process simulation results (Vázquez-Ojeda et al., 2013).

3.3.1. Results

As shown in Fig. 5a), the results indicate that DE-WF has some difficulty in exploring effectively and determining optimal values in the feasible search zone. This behavior is produced due to the weight function penalization of feasible individuals (designs) having a similar order of magnitude as infeasible individuals. In other words, the weight penalizations were not enough to improve the optimization process. The wide dispersion of the results reflects this.

Besides that, it was determined that most individuals concentrate between 20 and 40 GW/y whereas the best fitness function values represent a heat duty of 20 GW/y. However, in spite of lots of individuals lying around this optimum value, most of them do not meet all constraints.

The results generated in the optimization using DE-SA are shown in Fig. 5b). In this case, it is evident that from the first generations (300 function evaluations) that the DE-SA algorithm obtains feasible solutions for a value of the threshold. These initial solutions were feasible as the threshold is relatively large initially. Nevertheless, notice that from about 300–6500 function evaluations, the penalty function is larger due to the threshold value (ε) reducing as optimization progresses. This means that the population must have larger purities and recoveries than the first individuals, which increases energy consumption. Also observe that as the threshold reduces, the DE-SA algorithm has difficulty in, first, finding a feasible zone and, second, achieving stabilization in the feasible zone, increasing the fitness function.

Nonetheless, notice that after 6500 function evaluations, the DE algorithm was capable of identifying and remaining in a feasible zone. Thus, the robustness of the algorithm led to the generation of feasible solutions that meet purity and recovery targets with a set of design variables that allow minimization of the total heat duty (fitness function). This behavior was determined in an experience of 30 experiments (runs).

On the other hand, Table 5 shows the best five solutions, from 20,000 total function evaluations, obtained by DE-WF. Notice that the best solution has a fitness function of 21.5846 GW/y and the worst individual has a fitness function of 32.6248 GW/y. In addition, the results in Table 5 evidence that all constraints are accomplished, but all purities and recoveries are above the target value. Therefore, in terms of design, high values of purity and recovery mean high energy consumption, which save concordance with the fitness function values.Table 6.

3.3.2. Comparative analysis

Based on the previous analysis, it is clear that the dynamic self-adaptive constraint handling presented in this paper improves the performance of the DE algorithm through a more intensive exploration of the search space. It led to better solutions, less numerical effort and a shorter computation time than the other constraint handling technique. These results are a consequence of the dynamic behavior of the threshold during the optimization process. It is

Table 4 – Objective function results for typical chemical engineering cases.

Test	Objective function	Edgar et al. (2001)	Our technique	
			Weighting function (DE-WF)	Self-adaptive (DE-SA)
1	$\min CAT = \left(\frac{350}{na}\right)\left[\frac{w_1 cp_1 (T_{1S}-T_{1E})}{U^* \Delta T_{ML,1}} + \frac{w_2 cp_2 (T_{2S}-T_{2E})}{U^* \Delta T_{ML,2}}\right]^{0.65} + \left[\frac{0.005 * 8500}{l_{vap}}\right](w_1 cp_1 (T_{1F}-T_{1S}))$	2428.022	2428.025	2428.019
2	$\min L_T = \frac{G_0(y_0-y_1)}{x_1} + \frac{G_0(y_1-y_2)}{x_2} + \frac{G_0(y_2-y_3)}{x_3}$	55.154	56.789	55.154
3	$\min V_T = \frac{QC_{AF}(X_{A1}-X_{A2})}{-r_{A1}} + \frac{QC_{AF}(X_{A2}-X_{A3})}{-r_{A2}}$	814.648	815.742	814.646

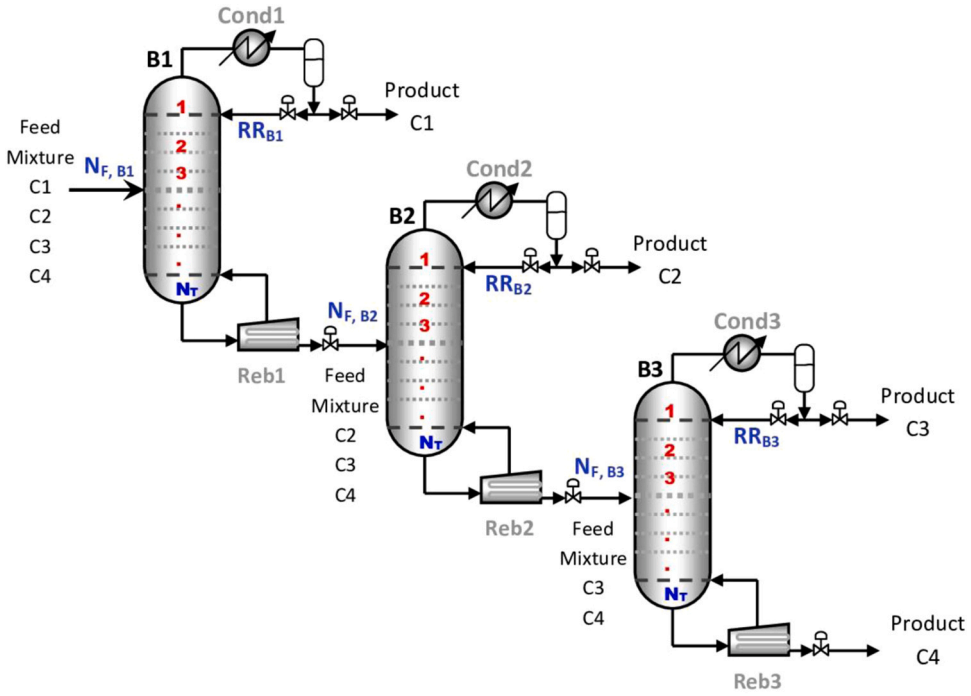

Fig. 4 – Distillation train.

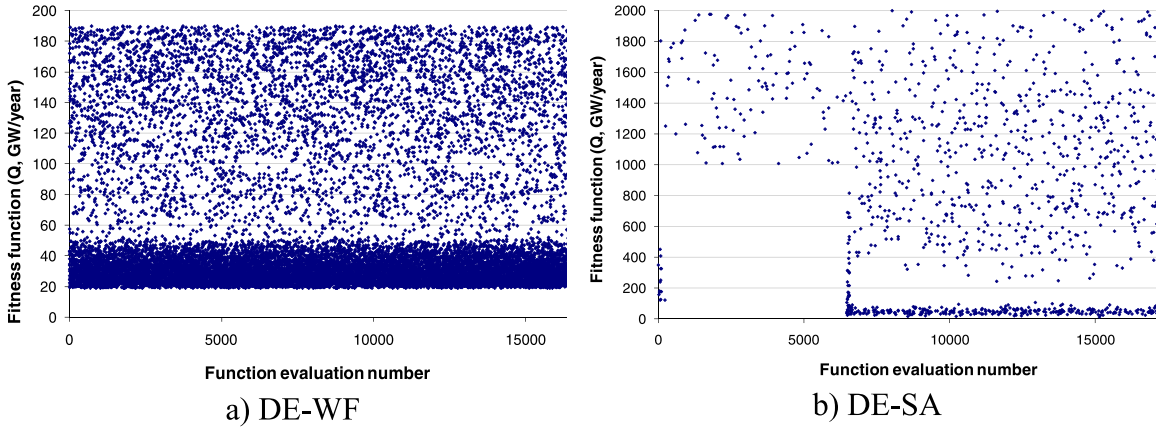

Fig. 5 – Optimization of distillation train, DE-WF vs DE-SA.

important to point out that the degree of relaxation and reduction of the threshold was determined using information of feasible and non-feasible individuals.

On the other hand, taking into account that the major compromise in a distillation column is given by the total heat duty and total number of stages, it is important to point out that the designs found by DE-SA are better for both N_T and Q than those determined by DE-WF. In fact, the best design obtained with DE-SA requires 56% less energy (9.4257 GW/y vs 21.5846 GW/y) and 26% fewer stages (82 stages vs 111 stages) than the best design determined by DE-WF. Hence, the best solution obtained by DE-SA provides the best compromise between energy consumption and total number of stages for this distillation train.

In addition, in order to conduct a more rigorous statistical analysis, DE-SA and DE-WF results were compared using a nonparametric statistical hypothesis test called the bootstrap signed-rank test (Hesterberg, 2011). This test is used when the average of the optimal fitness values can be assumed to be normally distributed. The null hypothesis in the bootstrap signed-rank test is that no significant difference exists between the performance of DE-SA and that of DE-WF. The alternative hypothesis is that there is a significant difference in the performance of DE-SA compared to its competitor. In the following expressions, H_0 represents the null hypothesis and H_1 is the alternative hypothesis.

$$H_0: \quad \mu_{DE-SA} = \mu_{DE-WF}$$

$$H_1: \quad \mu_{DE-SA} \neq \mu_{DE-WF}$$

where: μ_{DE-SA} and μ_{DE-WF} represent the average of the fitness function values for DE-SA and DE-WF, respectively. The bootstrap signed-rank test works by comparing two parameters, called α and p-value. α is defined as significance level and it is considered as the reference parameter in this approach. In the literature, a standard value of 5% (0.05) is assigned for α. Hence, the null hypothesis is rejected if the p-value is less than the significance level. Outcomes obtained

Table 5 – Best five designs found by DE-WF.

Design	1	2	3	4	5
Total heat duty, Q (GW/y)	21.5846	26.8670	30.4748	31.9111	32.62484
			Column 1, B1		
NT	49	41	44	27	50
NF	33	21	18	19	31
RR	15.9506	14.2624	19.6461	20	20
			Column 1, B2		
NT	23	28	44	28	31
NF	11	16	25	18	16
RR	8.6047	20	15.6566	10.7606	8.4996
			Column 1, B3		
NT	39	42	25	25	50
NF	8	22	7	14	49
RR	7.5152	1.9634	8.1922	13.6602	16.2600
			Purity		
x_A^{pur}	1.0000	0.9999	0.9999	0.9999	1.0000
x_B^{pur}	0.9920	0.9920	0.9920	0.9920	0.9920
x_C^{pur}	0.9921	0.9921	0.9843	0.9921	0.9891
x_D^{pur}	1	0.9999	0.9798	0.9999	0.9703
			Recovery		
x_A^{rec}	1.0000	0.9999	0.9999	0.9999	1.0000
x_B^{rec}	0.9920	0.9920	0.9920	0.9920	0.9920
x_C^{rec}	0.9921	0.9921	0.9843	0.9921	0.9891
x_D^{rec}	1	0.9999	0.9798	0.9999	0.9703

Table 6 – Best five designs found by DE-SA.

Total heat duty, Q (GW/y)	9.4257	10.4795	10.9875	11.5112	12.0309
			Column 1, B1		
NT	18	41	32	44	42
NF	11	12	11	14	25
RR	20	14.4178	16.9542	16.8805	17.4485
			Column 1, B2		
NT	34	47	38	27	36
NF	21	27	5	17	12
RR	3.9457	4.6335	5.1058	3.2294	5.0529
			Column 1, B3		
NT	30	35	38	15	21
NF	19	28	12	7	5
RR	1.4821	2.1354	1.9390	3.9878	2.7984
			Purity		
x_A^{pur}	0.9905	0.9990	0.9992	1.0000	1.0000
x_B^{pur}	0.9910	0.9920	0.9837	0.9909	0.9921
x_C^{pur}	0.9920	0.9916	0.9839	0.9904	0.9922
x_D^{pur}	0.9993	0.9946	1.0000	0.9943	0.9999
			Recovery		
x_A^{rec}	0.9905	0.9990	0.9992	1.0000	1.0000
x_B^{rec}	0.9910	0.9920	0.9837	0.9909	0.9921
x_C^{rec}	0.9920	0.9916	0.9839	0.9904	0.9922
x_D^{rec}	0.9993	0.9946	1.0000	0.9943	0.9999

through the bootstrap signed-rank test at $\alpha = 0.05$ confirm that the performance of DE-SA is better than that of DE-WF. This statement is also supported by the results obtained by applying other statistical tests such as the median and variance.

On the other hand, with the aim to support earlier results, a confidence interval was determined. This interval was obtained through a resampling with replacement of the best fitness function values. These fitness function values were determined by carrying out 30 experiments (runs) for each algorithm (30 runs for DE-SA and 30 runs for DE-WF). In this case, a resampling rate of 80% was applied. Thus, from 30 total individuals, 24 individuals were randomly chosen and the mean was computed. The resampling was performed

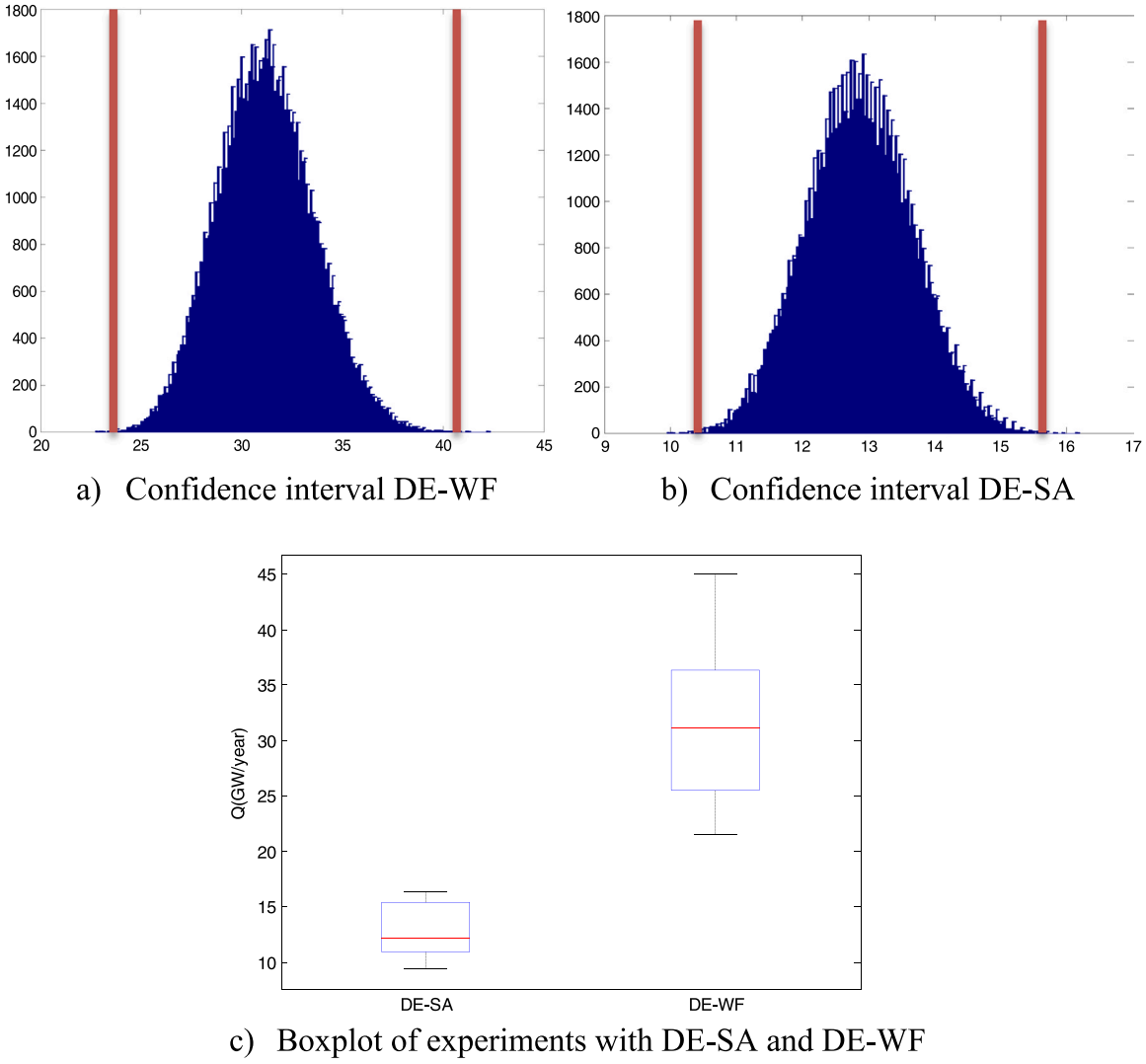

Fig. 6 – Statistical analysis comparing DE-SA and DE-WF.

50,000 times, obtaining the histogram shown in Fig. 6a) and b).

The results obtained by DE-SA showed that in about 95% of essays (runs) the fitness function values of the best individuals lie between 10.3422 and 15.6312 GW/y, with an average value of 12.8605 GW/y and a median of 12.8450 GW/y. Meanwhile, the range of the fitness function values for the best individuals determined using DE-WF varies between 23.9886 and 41.6991 GW/y, the average being 31.1890 GW/y and the median 31.1071 GW/y. These intervals are represented through the red bars shown in Fig. 6a) and b), where a normal distribution is evidenced. Through the behavior shown, it is clear than the optimization performance of DE-SA is better than that of DE-WF. In fact, the designs obtained using DE-SA show heat duty reductions of more than 50% compared with the best heat duty designs obtained by DE-WF.

On the other hand, with the aim to provide an alternative visualization of the results, a comparative analysis of DE-SA and DE-WF was carried out through the box diagram depicted in Fig. 6c). In this scheme it can be observed that the median for the DE-SA algorithm, 12.84 GW/y, is less than that for DE-WF, 23.9886 GW/y. Also notice that there is little dispersion in the experiments achieved using DE-SA in relation to the experiments carried out using DE-WF. This can be observed through the size of the box with respect to the y axis. In addition, observe that the DE-SA experiments show that even the solutions obtained in the last quartile are a substantial improvement on those obtained with DE-WF.

Regarding CPU time, DE-SA and DE-WF require approximately 22 h per run to obtain the optimal designs. There is no substantial difference between both techniques, since a high percentage corresponds to the evaluation of the fitness function in Aspen Plus. Authors such as Vazquez–Castillo et al. (2009) optimize Intensified distillation systems in quaternary mixtures through genetic algorithms, evaluating 12000 functions in a time between 8 and 10 h. Li et al., (2020) employ Simulated annealing for distillation process for separating benzene-isopropanol-water evaluating 81,000 in a time of 18 h. Recently, Lyu et al., (2021), report CPU time from 13 to 69 h to evaluate between 40,000 and 140,000 individuals, per run, in the optimization of three separation systems in ternary mixtures through dynamical DE. They propose the parelization of the evaluation of the objective function reducing substantially the time, however this implementation slightly affects the quality of the solution. They report that

the complexity of the case study will increase the time it takes for an Aspen simulation to converge. Some other authors omit to report the time and only establish as a point of comparison the number of function evaluations (Vázquez-Ojeda et al., 2013).

4. Discussion of results

As shown, the optimization performed using DE coupled with dynamic self-adaptive constraint handling successfully tackled all kinds of optimization problems presented, from NLP problems to multivariable MINLP problems with high dependence among variables and optimization of a distillation train. The results evidence the robustness of the optimization strategy composed of a DE algorithm and dynamic self-adaptive constraint handling supported by a self-adaptation dynamic threshold. In addition, the approach implemented demonstrated that the penalization strategy has a large influence on the trend of the results. This was established as the optimization was conducted using a DE algorithm in both cases, DE-SA and DE-WF, but the best performance was attributed to DE-SA. In general terms, the performance of DE-SA shows clear dominance over DE-WF in terms of the quality of the optimal values obtained (best fitness function) and numerical effort and computing time. It is assumed that this behavior is triggered by means of efficient exploration of the search space, which is powered by the constraint handling technique to an important degree. In addition, the structure of the penalization, composed of sequential self-adaptive and dynamic penalizations, led the population towards the best values of the objective function. Notice that the dynamic self-adaptive constraint handling was successfully applied to penalize in a proportional way according to the degree of deviation in relation to the target established. A key factor in this mechanism is the modulation of relaxing and stiffening of the dynamic threshold depending on the behavior of the fitness function of the population during the optimization process.

Thus, through the optimization supported by adequate constraint handling (DE-SA), the following benefits were obtained: optimal fitness functions were obtained using regular numbers of function evaluations, with neither large populations nor a large number of generations. For instance, compared with a repair algorithm, DE-SA required 20,000 function evaluations, while the repair algorithm performed 2000,000 function evaluations. In addition, the optimal individual for the repair algorithm was determined in evaluation 120, whereas the best individual for DE-SA was found in evaluation 35. On the other hand, compared with DE-WF, a roughly 50% reduction in computing time and better fitness functions were determined for DE-SA. In addition, optimization of the distillation train led to a 56% reduction in reboiler duty and 26% fewer stages.

The optimization results for the distillation train were corroborated by rigorous statistical analysis, using the bootstrap signed-rank test, confidence intervals and box diagrams. Hypothesis analysis using the bootstrap signed-rank test showed that effectively DE-SA has better performance that DE-WF. Similarly, the confidence intervals and box diagrams evidenced that the fitness function obtained with DE-SA is less dispersed than that computed with DE-WF. In fact, it was obvious that the worst fitness function obtained with DE-SA was better than the best one obtained with DE-WF.

Hence, based on the results obtained, it is clear that the constraint handling technique is a cornerstone for successfully dealing with optimization problems. Besides that, the experience showed that the dynamic self-adaptive constraint handling improves the performance of the DE algorithm. At the same time, it was demonstrated that the implementation of robust statistical tools is an important factor to support optimization results.

5. Conclusions

This work introduces a novel self-adaptive constraint handling technique coupled to a Differential Evolution algorithm (DE-SA) based on a self-adaptation dynamic threshold, self-adaptation factors (weights) and added penalization due to an interest in design variables. The optimization performance of DE-SA was compared with that using the constraint technique based on a weight function coupled to a Differential Evolution algorithm (DE-WF). In addition, to validate our proposal, the performance of DE-SA was also compared with that of a repair algorithm proposed by Kheawhom (2010) to optimize some benchmark functions, and with the optimal value determined by Edgar et al. (2001) who used a deterministic method to optimize typical chemical engineering problems. Finally, a distillation train optimization problem was solved with DE-SA and DE-WF. To compare results and perform statistical analysis, we carried out 30 runs under the same conditions, for each technique. The comparative behavior is essentially based on the constraint handling approaches, not on the optimization algorithm (DE) itself.

The optimization results for the benchmark functions indicate that the best performance is given by DE-SA. In fact, the repair algorithm and DE-SA led to the same fitness function optimal values but DE-SA used fewer function evaluations and less computing time than the repair algorithm. In addition, the optimal values obtained by DE-WF were larger than those reported by Kheawhom (2010). Similarly, optimization of the chemical engineering problems showed that the best fitness function values were obtained using DE-SA, using less computational time and numerical effort than the optimization process carried out by DE-WF to find the best solution reported by Edgar et al. (2001). In the distillation train problem, the best compromise (heat duty and total number of stages) for the distillation train was found using DE-SA. So, designs with both less energy consumption (heat duty) and a smaller total number of stages were determined, compared with DE-WF. These conclusions are supported by statistical analysis.

The results shown for DE-SA in all case studies, are attributed to the following features:

- Feasible individuals
  The feasibility of an individual determines whether that individual's information can be used to focus the search in feasible areas and avoid exploration in non-feasible areas. Thus, no promissory search zones are segregated by the algorithm.
- Dynamic threshold
  The dynamic threshold is applied to inequality constraints so equality constraints must be converted into their inequality form through a small tolerance value (epsilon). The main characteristic of the dynamic threshold is the relaxation/stiffness of the equality constraints through the

epsilon tolerance, allowing the population to follow the best search trajectories based on information from previous generations. Eventually, progressive reduction of the dynamic threshold enables the population to fulfill all the equality constraints.

- Self-adaptation factors based on dynamic threshold
  Through the self-adaptation of penalty factors based on the magnitude of constraint violations (according to the dynamic threshold value), the algorithm identifies and classifies individuals with the best characteristics and segregates those whose degree of deviation from a valid constraint is larger. When the residuals are large, a strong penalization is applied. For individuals with low constraint violation, the penalization is weak.
- Self-adaptation factors (weights) based on number of violated constraints
  This implementation allows identification of the number of constraints that are violated regardless of their magnitude, increasing the effect of the self-adaptation factors (see previous paragraph). Thus, for each individual, if the number of violated constraints is small the penalization is weak, and it becomes stronger with a larger number of violated constraints.
- Constraint of design variables

We apply this type of penalty to guide the search towards solutions to find the appropriate value of a specific design variable. The proposed penalty method decreases the final fitness function value only when all constraints lie within the dynamic threshold, for the minimization problem.

It is evident that the constraint handling technique proposed in this work has an effective selection pressure even when the total amount of constraint violation decreases. Additionally, it allows a suitable convergence to the feasible regions of the search space and leads the population towards the best values of the fitness function, in all case studies. DE-SA is a worthy tool since it is able to deal with optimization problems subject to equality and inequality constraints and interdependence on the optimization variables. Therefore, to further evaluate the proposed technique, our future work aims to couple the dynamic self-adaptive constraint handling technique to other evolutionary algorithms to optimize more complex chemical engineering processes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Constraint handling with self-adaptation for the distillation train problem

See. Fig. A1, Fig. A2, Fig. A3. and Fig. A4.

Implementation of the constraint handling technique for the case of the distillation train is detailed below. The self-adaptive constraint handling approach begins by verifying if the individual $Z_{i,G}$ is feasible. In this step $Q^{fitness}$ and $viola^{total}$ values are return to DE algorithm and shows the Figure A.1.

- An infeasible individual is when Aspen One non convergence ($viola^{convergence}$is1), and worst value at the fitness function, Q^{max} is assigned. The purities x_i^{pur} and recoveries x_i^{rec} are set to a "bad" value (0.01). In this case, for $viola^{pur}$ and $viola^{rec}$ the $N_{components}$ value are assigned.

$$
\begin{aligned}
&Evaluate\ Individual\ Z_{i,G}\\
&\quad Q^{total} \leftarrow Aspen\ One\ (Z_{i,G})\\
&\quad if\ \left(convergence\left(Aspen\ One\ (Z_{i,G})\right) == false\right)\\
&\quad\quad \{\\
&\quad\quad\quad viola^{convergence} = 1\\
&\quad\quad\quad Q^{fitness} = Q^{max}\\
&\quad\quad\quad x_i^{pur} = 0.01 \qquad x_i^{rec} = 0.01\\
&\quad\quad\quad w_i^{pur} = 0 \qquad w_i^{rec} = 0\\
&\quad\quad\quad viola^{pur} = N_{components} \qquad viola^{rec} = N_{components}\\
&\quad\quad\quad viola^{total} = viola^{pur} + viola^{rec} + viola^{convergence}\\
&\quad\quad\quad Return\ (Q^{fitness}, viola^{total})\\
&\quad\quad \}\\
&\quad else\\
&\quad\quad Q^{fitness} = Q^{total}
\end{aligned}
$$

Fig. A1 – Step 1 of the self-adaptive constraint handling technique.

$$
\begin{aligned}
&Penalize\ Q^{total}\ with\ the\ violated\ constraints\ of\ purity\ and\ recovery,\\
&\qquad for\ each\ component\\
&\quad Q^{pur} = 0 \qquad\qquad Q^{rec} = 0\\
&for\ i = 1\ to\ N_{components} \qquad for\ i = 1\ to\ N_{components}\\
&\quad if\left(abs\left(x_{target}^{pur} - x_i^{pur}\right) > \varepsilon\right) \qquad if\left(abs\left(x_{target}^{rec} - x_i^{rec}\right) > \varepsilon\right)\\
&\quad\quad \{ \qquad\qquad\quad \{\\
&\quad\quad w_i^{pur} = 10\left(x_{target}^{pur} - x_i^{pur}\right)^2 \qquad w_i^{pur} = 10\left(x_{target}^{rec} - x_i^{rec}\right)^2\\
&\quad\quad Q_i^{pur} = w_i^{pur} \cdot Q^{total} \qquad Q_i^{rec} = w_i^{rec} \cdot Q^{total}\\
&\quad\quad viola^{pur} = viola^{pur} + 1 \qquad viola^{rec} = viola^{rec} + 1\\
&\quad\quad \} \qquad\qquad\quad \}\\
&\quad else \qquad\qquad else\\
&\quad\quad Q_i^{pur} = 0 \qquad\qquad Q_i^{rec} = 0\\
&end\ for \qquad\qquad end\ for\\
&\quad Q^{pur} = \sum_{i=1}^{N_{components}} Q_i^{pur} \qquad Q^{rec} = \sum_{i=1}^{N_{components}} Q_i^{rec}\\
&\quad viola^{total} = viola^{pur} + viola^{rec} + viola^{convergence}\\
&\quad Q^{pen} = Q^{total} + Q^{pur} + Q^{rec}
\end{aligned}
$$

Fig. A2 – Step 2 of the self-adaptive constraint handling technique.

$$
\begin{aligned}
&Penalize\ Q^{pen}\ by\ the\ number\ of\ constraints\ violated\\
&\quad if(viola^{total} > 0)\\
&\quad\quad \{\\
&\quad\quad\quad w^{num\_pen} \leftarrow Get\ factors\ w(viola^{total})\\
&\quad\quad\quad Q^{num\_pen} = w^{num\_pen} \cdot Q^{pen}\\
&\quad\quad \}\\
&\quad else\\
&\quad\quad\quad Q^{num\_pen} = Q^{pen}
\end{aligned}
$$

Fig. A3 – Step 3 of the self-adaptive constraint handling technique.

$Penalize\ Q^{num\_pen}\ by\ the\ number\ stages\ and\ get\ Q^{fitness}$

$NT_{total} = NT_{B1} + \cdots + NT_{Bcolumns}$

$if(NT_{total} < NT_{max})$

{

$w_{stages} = \frac{NT_{total}}{NT_{max}}$

$Q^{pen\_stages} = w_{stages} Q^{num\_pen}$

$Q^{fitness} \leftarrow Q^{pen\_stages}$

}

$else$

$Q^{fitness} \leftarrow Q^{num\_pen}$

Fig. A4 – Step 4 of the constraint handling approach.

- For a feasible individual, the fitness function $Q^{fitness}$ is calculated using Aspen One (Q^{total}), $viola^{convergence}$ is 0, and the purities, x_i^{pur} and recoveries, x_i^{rec} for each component are reported.

For the feasible individual, we calculate the violation magnitude, for each component, accordance with dynamic threshold value, ε. The magnitude are obtained with the residual between the target value x_{target}^{pur} and the achieved value x_i^{pur} of purities. With residual, w_i^{pur} and Q^{pur} is calculated. For recoveries likewise is applied and obtain the w_i^{rec} and Q^{rec}. Finally, computes the Q^{pen} adding Q^{pur}, Q^{rec} and Q^{total}. When the individual meet whole constraints, Q^{pen} is not penalized and assigned at Q^{num_pen} (see Figure A.2). In addition, we determine the total number of no accomplished constraints $viola^{total}$ with the sum of $viola^{pur}$ and $viola^{rec}$ values (see Figure A.3). These values are obtained count the violations of purities and recoveries for each component for determines, w_{num_pen} and Q^{num_pen}. In this step Q^{num_pen} and $viola^{total}$ values are return to DE algorithm.

Finally, only individuals that accomplish whole constraints decrease the fitness function. This penalization favor the search in feasible spaces where the heat duty is minimized and also reduces the total number of stages of the distillation scheme. We determines the factor w_{stages} with the relationship between a design value heuristic, NT_{max} and total number of stages, NT_{total}. When $NT_{total} < NT_{max}$ obtain Q_{pen_stages}.

that and assign to $Q_{fitness}$, otherwise $Q_{fitness}$ is Q_{num_pen} (see Figure A.4).

References

Adeli, H., Cheng, N.T., 1994. Augmented Lagrangian genetic algorithm for structural optimization. J. Aerosp. Eng. 7 (1), 104–118.

Asafuddoula, M., Ray, T., Sarker, R., 2015. A differential evolution algorithm with constraint sequencing: An efficient approach for problems with inequality constraints. Appl. Soft Comput. 36, 101–113.

Aspen Plus 13.0, 2007. User Models. Aspen Technology, Inc.

Austbø, B., Gundersen, T., 2016. Impact of problem formulation on LNG process optimization. AIChE J. 62 (10), 3598–3610.

Austbø, B., Wahl, P.E., Gundersen, T., 2013. Constraint handling in stochastic optimization algorithms for natural gas liquefaction processes. Computer Aided Chem. Eng. 32, 445–450.

Babu, B.V., Angira, R., 2006. Modified differential evolution (MDE) for optimization of non-linear chemical processes. Comput. Chem. Eng. 30 (6–7), 989–1002.

Ben, Hamida, S., 2016. Extension of evolutionary algorithms to constrained optimization. Metaheuristics. Springer, Cham, pp. 329–356.

Boukouvala, F., Ierapetritou, M.G., 2014. Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function. AIChE J. 60 (7), 2462–2474.

Chanthasuwannasin, M., Kottititum, B., Srinophakun, T., 2017. A mixed coding scheme of a particle swarm optimization and a hybrid genetic algorithm with sequential quadratic programming for mixed integer nonlinear programming in common chemical engineering practice. Chem. Eng. Commun. 204 (8), 840–851.

Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Eng. 191 (11–12), 1245–1287.

Coello, Coello, C.A., 1999. Self-adaptive penalties for GA-based optimization. Evolutionary Computation (CEC 99). Proc. Congr. Evolut. Comput. 1, 573–580.

Coello Coello, C. A., Mezura Montes, E, 2002. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv. Eng. Inform. 16 (3), 193–203.

Contreras - Zarazúa, G., Jasso - Villegas, M.E., Ramírez - Márquez, C., Sánchez - Ramírez, E., Vázquez - Castillo, J.A., Segovia - Hernández, J.G., 2021. Design and intensification of distillation processes for furfural and Co-products purification considering economic, environmental, safety and control issues. Chem. Eng. Process. Process.Intensif. 159, 108218.

Dadios, E., Ashraf, J., 2006. Genetic algorithm with adaptive and dynamic penalty functions for the selection of cleaner production measures: a constrained optimization problem. Clean. Technol. Environ. Policy 8 (2), 85–95.

Dasgupta, D., Michalewicz, Z., 1997. Evolutionary algorithms—an overview. Evolutionary Algorithms in Engineering Applications. Springer, Berlin, Heidelberg, pp. 3–28.

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In International conference on parallel problem solving from nature 849–858. https://doi.org/10.1007/3-540-45356-3_83

Diwekar, U.M., Grossmann, I.E., Rubin, E.S., 1992. An MINLP process synthesizer for a sequential modular simulator. Ind. Eng. Chem. Res. 31 (1), 313–322.

Edgar, T.F., Himmelblau, D.M., Lasdon, L.S., 2001. Optimization of Chemical Processes. McGraw-Hill, Boston.

Fister, I., Brest, J., Iglesias, A., Galvez, A., Deb, S., 2021. On selection of a benchmark by determining the Algorithms' qualities. IEEE Access 9, 51166–51178. https://doi.org/10.1109/ACCESS.2021.3058285

Franke, M.B., 2019. Mixed-integer optimization of distillation sequences with Aspen Plus: a practical approach. Comput. Chem. Eng. 131, 106583.

Gómez – Castro, F.I., Segovia - Hernández, J.G., Hernández, S., Gutiérrez – Antonio, C., Briones – Ramírez, A., Gamiño – Arroyo, Z, 2015. Design of non-equilibrium stage separation systems by a stochastic optimization approach for a class of mixtures. Chem. Eng. Process.: Process.Intensif. 88, 58–69.

Hesterberg, T., 2011. Bootstrap. Wiley Interdiscip. Rev.: Comput. Stat. 3 (6), 497–526.

Homaifar, A., Qi, C.X., Lai, S.H., 1994. Constrained optimization via genetic algorithms. Simulation 62 (4), 242–253.

Joines, J., Houck, C., 1994. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. Proc. 1st IEEE Conf. Evolut. Comput., Orlando 579–584.

Kazarlis, S., Petridis, V., 1998. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. International Conference on Parallel Problem

Solving from Nature. Springer,, Berlin, Heidelberg, pp. 211–220.
Kheawhom, S., 2010. Efficient constraint handling scheme for differential evolutionary algorithm in solving chemical engineering optimization problem. J. Ind. Eng. Chem. 16 (4), 620–628.
Kiss, A.A., Segovia-Hernández, J.G., Bildea, C.S., Miranda-Galindo, E.Y., Hernández, S., 2012. Reactive DWC leading the way to FAME and fortune. Fuel 95, 352–359 In press.
Leboreiro, J., Acevedo, J., 2004. Processes synthesis and design of distillation sequences using modular simulators: a genetic algorithm framework. Comput. Chem. Eng. 28 (8), 1223–1236.
Liepins, G., Vose, M., 1990. Representational issues in genetic optimization. J. Exp. Theor. Artif. Intell. 2 (2), 101–115.
Lyu, H., Cui, C., Zhang, X., Sun, J., 2021. Population-distributed stochastic optimization for distillation processes: implementation and distribution strategy. Chem. Eng. Res. Des. 168, 357–368.
Lyu, H., Zhang, X., Cui, C., Sun, J., 2022. Adaptive superstructure for multiple-interconnection process synthesis: Eliminate unnecessary flowsheet predetermination to reduce complexity. Chem. Eng. Process. -Process. Intensif. 171, 108731.
Medina - Herrera, N., Tuttuti - Ávila, S., Jiménez - Gutiérrez, A., Segovia - Hernández, J.G., 2017. Optimal design of a multiproduct reactive distillation system for silanes production. Comput. Chem. Eng. 105, 132–141.
Mezura Montes, E., Coello Coello, C. A, 2011. Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm Evolut. Comput. 1 (4), 173–194.
Michalewicz, Z., Attia, N., 1994. Evolutionary optimization of constrained problems. Proc. 3rd Annu. Conf. Evolut. Program. Singap. 98–108.
More, R.K., Bulasara, V.K., Uppaluri, R., Banjara, V.R., 2010. Optimization of crude distillation system using aspen plus: Effect of binary feed selection on grass-root design. Chem. Eng. Res. Des. 88 (2), 121–134.
Na, J., Lim, Y., Han, C., 2017. A modified DIRECT algorithm for hidden constraints in an LNG process optimization. Energy 126, 488–500.
Palma - Barrera, J.P., Sánchez - Ramírez, E., Ramírez - Márquez, C., Cervantes - Jauregui, J.A., Segovia - Hernández, J.G., 2019. Reactive distillation column design for tetraethoxysilane (TEOS) production. Part II: dynamic properties and inherent safety. Ind. Eng. Chem. Res. 58, 259–275.
Riche, R.L., Haftka, R.T., 1997. Evolutionary optimization of composite structures. Evolutionary Algorithms in Engineering Applications. Springer, Berlin, Heidelberg, pp. 87–102.
Sreepathi, B.K., Rangaiah, G.P., 2017. Optimization of heat exchanger network retrofitting: comparison of penalty function and feasibility approach for handling constraints. Multi Object. Optim. Tech. Appl. Chem. Eng. 501–532.
Srinivas, M., Rangaiah, G.P., 2007. Differential evolution with tabu list for solving nonlinear and mixed-integer nonlinear programming problems. Ind. Eng. Chem. Res. 46 (22), 7126–7135.
Storn, R.M., Price, K.V., 1997. Differential evolutions a simple and efficient heuristics for global optimization. J. Glob. Optim. 11, 341–359.
Summanwar, V.S., Jayaraman, V.K., Kulkarni, B.D., Kusumakar, H.S., Gupta, K., Rajesh, J., 2002. Solution of constrained optimization problems by multi-objective genetic algorithm. Comput. Chem. Eng. 26 (10), 1481–1492.
Takahama, T., Sakai, S., 2006. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. IEEE Int. Conf. Evolut. Comput. Vanc. 1–8.
Takahama, T., Sakai, S., 2010. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. IEEE Congr. Evolut. Comput. 1–9.
Teh, Y.S., Rangaiah, G.P., 2003. Tabu search for global optimization of continuous functions with application to phase equilibrium calculations. Comput. Chem. Eng. 27 (11), 1665–1679.
Vazquez–Castillo, J.A., Venegas–Sánchez, J.A., Segovia–Hernández, J.G., Hernández-Escoto, H., Hernandez, S., Gutiérrez–Antonio, C., Briones–Ramírez, A., 2009. Design and optimization, using genetic algorithms, of intensified distillation systems for a class of quaternary mixtures. Comput. Chem. Eng. 33 (11), 1841–1850.
Vázquez-Ojeda, M., Segovia - Hernández, J.G., Hernández, S., Hernández – Aguirre, A., Kiss, A.A, 2013. Design and optimization of an ethanol dehydration process using stochastic methods. Sep. Purif. Technol. 105, 90–97.
Wong, K.P., Dong, Z.Y., 2005. Differential evolution, an alternative approach to evolutionary algorithm. Proc. 13th Int. Conf. Intell. Syst. Appl. Power Syst. 73–83.
Yiqing, L., Xigang, Y., Yongjian, L., 2007. An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. Comput. Chem. Eng. 31 (3), 153–162.
Yu, X., Lu, Y., Wang, X., Luo, X., Cai, M., 2019. An effective improved differential evolution algorithm to solve constrained optimization problems. Soft Comput. 23 (7), 2409–2427.
Zahara, E., Kao, Y.T., 2009. Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst. Appl. 36 (2), 3880–3886.
Zhang, H., Rangaiah, G.P., 2012. An efficient constraint handling method with integrated differential evolution for numerical and engineering optimization. Comput. Chem. Eng. 37, 74–88.
Zielinski, K., Laur, R., 2008. Stopping criteria for differential evolution in constrained single-objective optimization. Advances in Differential Evolution. Springer, Berlin, Heidelberg, pp. 111–138.