Constraint-Handling Techniques within Differential Evolution for Solving Process Engineering Problems

Victor H. Cantú[a], Catherine Azzaro-Pantel[a,*], Antonin Ponsich[b]

[a] *Laboratoire de Génie Chimique, UMR 5503 CNRS/INP/UPS, Université de Toulouse, Toulouse, France*
[b] *Departamento de Sistemas, Universidad Autónoma Metropolitana Azcapotzalco, Mexico City, Mexico*

Abstract

Chemical engineering optimization problems are typically considered as difficult optimization problems as non-convexities and discontinuities arise, and as they usually present a high number of constraints to be fulfilled. Differential Evolution algorithm (DE) has proven to be robust for the solution of highly non-convex and mixed-integer problems; nevertheless, its performance greatly depends on the constraint-handling technique used. In this study, numerical comparisons of some state-of-the-art constraint-handling techniques are performed: static penalty function, stochastic ranking, feasibility rules, ε constrained method and gradient-based repair. The obtained results show that the gradient-based repair technique deserves a special attention when solving highly constrained problems. This technique enables to efficiently satisfy both inequality and equality constraints, which makes it particularly adapted for the solution of process engineering optimization problems.

1. Introduction

Complex optimization problems are ubiquitous in chemical engineering, and more generally, in process engineering (PE). Examples of optimization problems related to this area encompass batch process design, phase equilibrium (Dowling and Biegler, 2015), distillation sequencing (Zhu et al., 2016), heat exchanger networks (Ayala et al., 2016), reactor network design (Kaiser et al., 2016), supply chain design (Almaraz et al., 2015; Woo et al., 2016), among others. All these real-world problems are typically represented by a mathematical model containing both binary and continuous variables, and a set of linear and non-linear constraints, i.e., leading to a mixed-integer non-linear programming (MINLP) approach. A single-objective formulation of these optimization problems can be stated as follows:

$$\begin{aligned} \text{minimize} \;\; & f(\mathbf{x}), && (1)\\ \text{subject to} \;\; & g_i(\mathbf{x}) \leq 0, \quad i = \{1,\dots,m\} \\ & h_j(\mathbf{x}) = 0, \quad j = \{1,\dots,q\} \\ & \mathrm{l}_i \leq x_i \leq \mathrm{u}_i, \quad i = \{1,\dots,n\} \end{aligned}$$

where $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ is a n-dimensional vector of decision variables (either discrete or continuous), $f(\mathbf{x})$ is the objective function to be minimized[1], m is the number of inequality constraints and q is the number of equality constraints. The functions g_i and h_j may be linear or non-linear, continuous or not, real-valued functions. Each variable x_i has upper and lower bounds, u_i and l_i, respectively, which define the search space $\mathcal{S} \subseteq \mathbb{R}^n$. Inequality and equality constraints define the feasible region $\mathcal{F} \subseteq \mathcal{S}$.

Throughout the years, mathematical programming techniques have been used to address the solution of these problems. Branch and Bound methods (BB), decomposition algorithms such as Generalized Benders Decomposition (GBD) and Outer-Approximation (OA) have been proposed to solve to global optimality these problems (Tawarmalani et al., 2002; Floudas and Gounaris, 2009). It should be noted, however, that all these Newton's methods rely on the initial given solution and on convexity assumptions of the non-linear functions to guarantee the global optimum solution. That is, the problem needs to satisfy some specific mathematical characteristics (e.g., convexity, derivability) so that a valid reformulation can be generated, otherwise, the reformulated convex problem might miss the original global optimum and converge to a local optimum. Furthermore, the application of these deterministic techniques can be computationally expensive to obtain the rigorous global solution of large-scale problems.

In that context, metaheuristics, and especially evolutionary algorithms techniques (EAs) have been proposed to solve highly non-convex optimization problems. Some examples of EAs are Evolutionary Strategies (ES), Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE), among others (Bozorg-Haddad et al., 2017). Even if these techniques lack of any theoretical convergence proof, it is well recognized that good quality solutions can be obtained and several advan-

*Corresponding author
Email address: catherine.azzaropantel@toulouse-inp.fr (Catherine Azzaro-Pantel)

[1] for the case of maximization, a negative sign is just added to the objective function

tages can be highlighted : (i) these population-based algorithms do not require any mathematical property for the treated problem as they only use the evaluation of the objective function; (ii) due to their population-based nature, EAs are particularly suited to tackle problems in which more than one objective is to be optimized. This property is important in PE for which several kinds of objectives (for instance, environmental or social impacts) need to be minimized simultaneously with an economic criterion.

However, as EAs have been conceived as directed search engines to work over unconstrained spaces, their main drawback appears when tackling constrained problems and thus, their applicability has been limited by a deficiency of general techniques to manage constraints. Consequently, much effort has been made to efficiently incorporate constraint-handling techniques into EAs, beginning, as in mathematical programming, by penalty functions. Nevertheless, though penalty functions may work well in some problems, they involve the tuning of parameters that may affect significantly the final solutions found. For this reason, evolutionary computation researchers have proposed many other sophisticated approaches for handling constraints, e.g., stochastic ranking(Runarsson and Yao, 2000), feasibility rules(Deb, 2000), ε constrained method (Takahama and Sakai, 2005), gradient-based repair(Chootinan and Chen, 2006), etc. Surprisingly enough, even if EAs have been widely used in literature for the solution of process engineering optimization problems, the same constraint-handling techniques are usually considered: static penalty function or feasibility rules, whereas the performance of more recent methods has not been investigated in the PE framework. Further, in order to improve the performance of these EAs in constrained problems, a reformulation of the problem is often done to reduce the number of decision variables and to remove the equality constraints. However, despite its efficiency, this approach may not work properly for large-scale real-world problems that usually contain many constraints.

The scientific objective of this work is therefore to explore the performance of some state-of-the art constraint-handling techniques for the solution of these particularly difficult optimization problems. A self-adaptive variant of DE is considered as the search engine, both because of its simplicity (it does not introduce any sophisticated operator nor any non-uniform probability distribution) and its claimed superiority over GAs and PSO in terms of its computational efficiency (Vesterstrom and Thomsen, 2004; Ponsich and Coello, 2011). The literature review led us to consider and compare five constraint-handling techniques, i.e., static penalty function, stochastic ranking, feasibility rules, ε constrained method and gradient-based repair. The aforementioned constraint-handling strategies have thus been embedded within DE for the solution of 12 well-known chemical engineering problems.

The remainder of this article is organized as follows. In the next section, DE is briefly described. Then, some state-of-the-art constraint-handling techniques in EAs are reviewed. After that, the methodology developed for the computational experiments is presented along with some information regarding the test problems considered. Then, the results are presented and discussed. The last section highlights the conclusions and perspectives of this work.

2. Differential Evolution

Differential Evolution (DE), proposed by Storn and Price (Storn and Price, 1997), is a simple yet efficient population-based search engine. As typical evolutionary algorithms, it relies on the Darwinian principle of *survival of the fittest* to obtain good quality solutions through the *reproduction*, *mutation*, *competition* and *selection* processes. DE has been successfully applied to optimization problems including non-linear, non-differentiable, non-convex and multi-modal functions. It has been shown that DE is fast and robust for solving these kinds of functions (Storn and Price, 1997).

In DE, an initial population is randomly generated within the search space. Each individual in the population represents a solution, i.e., a vector of n decision variables. At each generation, all individuals are selected as parents to generate new solutions. The reproduction and mutation processes are carried out as follows: a mutant vector $\mathbf{v}$ is produced choosing randomly $1+2n_d$ individuals from the population excluding the current target parent. The first individual is a base vector, whereas the n_d subsequent individuals are paired to create difference vectors. The difference vectors are scaled by factor F and then added to the base vector. The most widely used mutation process only considers one difference term and involves random vectors, as shown in Eq.(2):

$$\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot (\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \quad (2)$$

where $r_1 \neq r_2 \neq r_3 \neq i \in \{1, \dots, NP\}$ are random indexes. The resulting vector is then recombined with the parent with a probability CR, the crossover rate factor. This crossover process produces a trial vector $\mathbf{u}$ according to:

$$u_{i,j} = \begin{cases} v_{i,j} & \text{if } w \sim U(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j} & \text{otherwise} \end{cases} \quad (3)$$

where $j \in \{1, \dots, n\}$ and j_{rand} is a randomly chosen index $\in \{1, \dots, n\}$ ensuring that $\mathbf{u}_i$ gets at least one variables from the mutant $\mathbf{v}_i$. Finally, the trial vector is compared with its parent, that obtaining the best fitness is selected and inserted into the next population.

Depending on the mutation and crossover scheme considered, several variants of DE exist. They can be denoted as $DE/x/y/z$, where x specifies the base vector which can be *rand* (randomly chosen) or *best* (the vector with better fitness from the current population), y denotes the number of difference vectors used, and z denotes the crossover scheme that can be *bin* (independent binomial experiments) or *exp* (exponential crossover, similar to 1-point crossover in GAs). In this work, the most popular version of DE is used, that is, $DE/rand/1/bin$ which considers one randomly difference from the population and performs binomial crossover, as represented in Eqs.(2) and (3), respectively.

Furthermore, some self-adaptive versions of DE were introduced in the literature, in which the control parameters F and CR are self-adapted throughout the evolutionary process, e.g. SaDE (Qin and Suganthan, 2005), JADE (Zhang and Sanderson, 2009), jDE(Brest et al., 2006), SHADE (Tanabe and Fukunaga, 2013). In this work, the jDE algorithm because of its low computational complexity. jDE has shown good performance solving numerical benchmark problems and has proven to be robust.

Also, it must be mentioned that DE is a technique devoted to continuous optimization, although it can easily be extended for the treatment of mixed-integer problems. In this study, binary variables are encoded as continuous variables, and their values are rounded to the next integer only for the evaluation of the fitness function.

3. Constraint-handling techniques

In this section, the most popular techniques for handling constraints in evolutionary computation are presented, namely, penalty functions, Deb's feasibility rules, stochastic ranking, ε constrained method and gradient-based repair. Several other techniques exist in the specialized literature (Mezura-Montes and Coello, 2011; Coello Coello, 2016), however there is no evidence that they significantly outperform the techniques tackled here. In addition, these more sophisticated techniques usually involve the setting of several parameters(Mezura-Montes and Coello, 2005; Mallipeddi and Suganthan, 2010; Padhye et al., 2015).

3.1. Penalty functions

Historically, the most common approach to incorporate constraints (both in evolutionary algorithms and in mathematical programming) involves penalty functions, which were originally proposed in the 1940s and later expanded in many research studies mainly because of their simplicity and efficiency. With this method, the fitness landscape is modified as some penalty value is added to the objective value of each infeasible individual. In their general form, penalty functions can be represented as:

$$\psi(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} r_i \cdot \max\{0, g_i(\mathbf{x})\}^p + \sum_{j=1}^{q} c_j \cdot |h_j(\mathbf{x})|^p \tag{4}$$

where $\psi(\mathbf{x})$ is the new fitness function to be minimized, r_i and c_j are positive constants called *penalty factors*, and p normally takes values of 1 or 2.

As can be noted, this implementation, though quite simple, requires the use of a number of parameters to be tuned (equals to the number of constraints) which might be impractical in highly constrained problems. For this reason, the static penalty function, the simplest form of penalty function, has remained as the most popular one:

$$\psi(\mathbf{x}) = f(\mathbf{x}) + r \cdot \phi(\mathbf{x}) \tag{5}$$

where r is the penalty coefficient and $\phi(\mathbf{x})$ is the overall constraint violation:

$$\phi(\mathbf{x}) = \sum_{i=1}^{m} \max\{0, g_i(\mathbf{x})\}^p + \sum_{j=1}^{q} |h_j(\mathbf{x})|^p \tag{6}$$

Although the static penalty function only needs the tuning of one parameter, its value is not straightforward to set. On the one hand, if r is too low, the search will be directed towards regions where the objective function is minimized, but the final obtained solutions are very likely to be infeasible. On the other hand, if r is too high, the minimization of the overall constraint violation will be prioritized, obtaining a feasible solution in early generations with the disadvantage that, if the search space is disjointed or highly constrained, it will be very difficult to escape from the first feasible region found, the process being thus stuck in a local optimum. Ideally, the penalty should be kept as low as possible, just above the limit where the found solutions are infeasible, that is called the *minimum penalty rule*.

Furthermore, dynamic penalty functions, in which the coefficient r varies throughout the evolutionary process, have been proposed (Coello, 2000; Nanakorn and Meesomklin, 2001; Tessema and Yen, 2006). The constant idea in dynamic penalty functions is that allowing low values of r at early generations enables to explore the regions where the objective function is minimized, whereas a high value of r is desired at final generations in order to push the search towards the feasible region. Such an idea would work well for problems for which the unconstrained global optimum is close to its constrained global optimum, but there is no guarantee that this strategy will be efficient in all cases. Besides, the additional parameters needed to define the penalty coefficient schedule make this method less attractive than the simple static penalty function.

Finally, since a good choice of the penalty coefficient is necessary to enable a good balance between the objective function and the overall constraint violation minimizations, adaptive strategies have been suggested where information gathered from the search process is used to control the amount of penalty added to infeasible individuals. Adaptive penalty functions are not difficult to implement and they usually do not require user-defined parameters. Nevertheless, the results found in literature are not very encouraging as adaptive penalty methods usually need a lot of iterations to find the optimal solution as illustrated in (Tessema and Yen, 2006).

3.2. Stochastic ranking

Stochastic ranking (SR) has been proposed by Runarsson and Yao (Runarsson and Yao, 2000) as an attempt to balance the relative weights of the objective and the constraint violation that occurs in penalty functions. In this method, the population is sorted following a probabilistic procedure: two individuals are compared according to their objective function with a probability P_f, otherwise, the overall constraint violation is used for the comparison as indicated in the pseudo-code presented in Fig. 1. Once the population has been sorted by SR, a part of the population assigned with highest rank is selected for recombination, thus sharing its characteristics to the next generation. In this way, the search is directed by the minimization of the objective function and by feasibility concepts at the same time.

```
for i = 1 to NP
    for j = 1 to NP − 1
        sample u ~ U(0,1)
        if φ(I_j) = φ(I_{j+1}) = 0 or u < P_f
            if f(I_j) > f(I_{j+1})
                swap (I_j, I_{j+1})
            end
        else
            if φ(I_j) > φ(I_{j+1})
                swap (I_j, I_{j+1})
            end
        end
    end
    if no swap done then break end
end
```

Figure 1: Stochastic ranking procedure. I is an individual in the population, P_f is the probability of using only the objective function for comparisons. The initial ranking is always generated randomly.

It is worth noting that stochastic ranking was originally designed to work with Evolutionary Strategies (ES), which indeed requires a ranking process in its replacement mechanism, however, its implementation within other search paradigms is not straightforward, even if some studies have extended its use to other EAs (Zhang et al., 2008; Fan et al., 2009; Ali et al., 2012). Considering DE, SR could be used in two different ways: for selecting a part of the population that would participate in the mutation process, or for selecting the individuals that would survive to the next generation (after the mutation and crossover processes). In (Fan et al., 2009), the authors proposed to rank the population according to SR procedure before the mutation process: they divide the population into two parts (that they call higher and lower parts), the upper part containing the *better* individuals, i.e., the individuals ranked higher after SR. Then, for every trial vector, $\mathbf{v}_i$, the upper part contributes with two *good* individuals, while the lower part provides only one *less good* individual. This procedure was initially considered in this study, but since the obtained results were not satisfactory, SR was implemented within the selection process as follows: the new population is generated normally by DE operator, i.e., using the entire population, and then both populations (parents and offspring) are ranked according to SR. Finally, each new individual is compared with his parent, and the one ranked higher survives to the next generation.

3.3. Feasibility rules

This constraint handling technique establishes the superiority of feasible solutions over infeasible ones, that is, as opposite to the penalty functions, feasibility rules do not merge both information from constraint violation and objective function, but consider them separately. Proposed by Deb in (Deb, 2000), feasibility rules (also called lexicographical order) consist in a binary tournament selection according to the following criteria:

1. Any feasible solution is preferred to any infeasible solution.
2. Among two feasible solutions, that with better objective function value is preferred.
3. Among two infeasible solutions, that with smaller constraint violation is preferred.

Deb's feasibility rules represent an easy-to-implement, parameter-free technique to handle constraints. Further, due to its simplicity and its overall good performance, feasibility rules are usually the first constraint-handling technique tested for treating a given problem with EAs. However, one of the main drawbacks of this method appears when dealing with problems with a reduced and disconnected feasible region (e.g., problems with one or several equality constraints). Because any feasible solution is preferred over an infeasible one, once the algorithm has converged to some feasible region, it may be very difficult to escape from there in order to explore other regions, i.e., once the constraints are fulfilled, the algorithm is very likely to get trapped prematurely in some subregion of the search space. Moreover, considering that there are high probabilities that the optimum lies close to the feasibility boundary, slightly infeasible solutions might be more useful to the search process than solutions wide inside the feasible region. However, the feasibility rules would prefer the latter solution to the former one. In fact, feasibility rules can be seen as a limiting case of static penalty function when the penalty value takes a very high value, in this way, when comparing two solutions, that with a lesser amount of overall constraint violation will be always preferred.

3.4. ε constrained method

In order to tackle the above-mentioned issues related to feasibility rules in severely constrained problems, the ε constrained method for evolutionary algorithms has been proposed by Takahama and Sakai in 2005 (Takahama and Sakai, 2005), where a relaxation of constraints is permitted to explore constrained regions. This tolerance level over the relaxation, called the ε level, indicates the limit under which solutions are considered as feasible. Once the feasibility of solutions has been identified by means of the ε level, the lexicographical order (i.e. Deb's feasibility rules) is used for selecting the surviving individuals for the next generation. This technique has proven to be especially efficient in highly constrained problems, such as those involving equality constraints, because this relaxation, allowed at the early generations within a certain level, promotes exploration of regions that would be impossible to reach by simple feasibility rules.

The main drawback of this method is the difficulty for setting the ε parameter. It has been remarked that ε level enables a good exploration of the search space in early generations but also it is clear that ε must be 0 at some point of the evolutionary process in order to obtain feasible solutions. In (Takahama and Sakai, 2006), the authors proposed a dynamic control of ε level, according to:

$$\varepsilon(0) = \phi(\mathbf{x}_\theta) \tag{7}$$
$$\varepsilon(t) = \begin{cases} \varepsilon(0)(1-\frac{t}{T_c})^{cp}, & 0 < t < T_c, \\ 0, & t \geq T_c \end{cases}$$

where $\mathbf{x}_\theta$ is the best θ-th individual (in terms of constraint violation) in the first generation, *cp* is a parameter to control the speed ε level decrease and T_c represents the generation after which the ε level is set to 0 (after that, Deb's feasibility rules are considered). According to the authors, the following parameter setting works well in many problems: $\theta = 0.2NP$, $cp = 5$, $T_c \in 0.2T_{max}$. However, their tuning still constitutes a disadvantage, as it might become a harsh task. Additionally, it is important to recall that the use of the ε level according to Eq.(7) is only recommended for highly constrained problems in which the feasibility rules do not work properly, otherwise ε constrained method may get worse results than the feasibility rules in terms of efficiency and efficacy.

It should be underlined that the ε constrained method obtained the first place in the competition on constrained optimization of the Congress of Evolutionary Computation (CEC 2006) and very competitive results in CEC 2010 (Takahama and Sakai, 2006, 2010). Due to this success, ε constrained method has also been embedded in a number of algorithms for multiobjective optimization (Yang et al., 2014; Fan et al., 2016, 2017). However, it is worth noting that the excellent results obtained by this method in the two above-mentioned competitions did not only depend on the use of the ε level relaxation strategy by itself, but also on a additional technique, namely gradient-based repair method (presented in the next section) (Chootinan and Chen, 2006). It is difficult to determine which of both methods contribute the most in obtaining such excellent results. In this study, the ε constrained method has been implemented with and without the gradient-based repair, this way, the performance of both algorithms can be compared.

3.5. Gradient-based repair

The gradient-based repair method, proposed by Chootinan and Chen in 2006 (Chootinan and Chen, 2006), is a constraint-handling technique that uses the gradient information derived from the constraint set to systematically repair infeasible solutions. Basically, the gradient of constraint violation is used to direct infeasible solutions toward the feasible region. The vector of constraint violations $\Delta C(\mathbf{x})$ is defined as:

$$\begin{aligned} \Delta C(\mathbf{x}) = [&\Delta g_1(\mathbf{x}), \ldots, \Delta g_m(\mathbf{x}), \\ &\Delta h_1(\mathbf{x}), \ldots, \Delta h_p(\mathbf{x})]^T \end{aligned} \tag{8}$$

where $\Delta g_i(\mathbf{x}) = \max\{0, g_i(\mathbf{x})\}$ and $\Delta h_j(\mathbf{x}) = h_j(\mathbf{x})$. This information, additionally to the gradient of constraints $\nabla C(\mathbf{x})$, is used to determine the step $\Delta\mathbf{x}$ to be added to the solution $\mathbf{x}$, according to:

$$\nabla C(\mathbf{x})\Delta\mathbf{x} = -\Delta C(\mathbf{x}) \tag{9}$$
$$\Delta\mathbf{x} = -\nabla C(\mathbf{x})^{-1}\Delta C(\mathbf{x}) \tag{10}$$

Although the gradient matrix ∇C is not invertible in general, the Moore-Penrose inverse or pseudoinverse $\nabla C(\mathbf{x})^+$ (Campbell and Meyer, 2009), which gives an approximate or best (least square) solution to a system of linear equations, can be used instead in Eq.(10). Thus, once the step $\Delta\mathbf{x}$ has been computed, the infeasible point $\mathbf{x}$ is moved to a *less* infeasible point $\mathbf{x} + \Delta\mathbf{x}$. This repair operation is performed with a probability P_g and repeated R_g times while the point is infeasible.

In this work, the computation of the gradient $\nabla C(\mathbf{x})$ is done numerically using forward finite differences, for all problems. Also, it is worth noting that in (Chootinan and Chen, 2006) only non-zero elements of $\Delta C(\mathbf{x})$ are repaired, i.e., the gradient is only computed for constraints that are actually being violated. On the contrary, in (Takahama and Sakai, 2006) all constraints are considered in the repair process, even those that are already satisfied. The former approach has the disadvantage that a given constraint may be fulfilled at one iteration but violated in the next one, nevertheless, this is usually more efficient compared to the latter approach, in terms of number of iterations needed to get to the feasible region. In this study, the former approach is considered, i.e., only non-zero elements of $\Delta C(\mathbf{x})$ are taken into account within the repair process. Note that this procedure can produce situations where some variables lie outside their allowed variation range, so that two inequality constraints may be added for each variable, accounting for their bounds. Due to the associated computational burden in real-world optimization problems, where the number of variables may be high, these additional constraints are not considered here. Instead, an additional repair process, performed at each iteration, sets the variable value to the violated bound if necessary. The pseudo-code of the gradient-based repair procedure used in this study is presented in Fig. 2.

```
for i = 1 to NP
    t = 0
    sample u ~ U(0,1)
    while t < R_g and φ(x) > 0 and u < P_g
        compute ∇C(x) of violating constraints
        compute ∇C(x)^+
        compute Δx
        x ← x + Δx
        repair x to its bounds
        compute ΔC(x)
        t = t + 1
    end
end
```

Figure 2: Gradient-based repair method.

Even if the gradient-based repair can be considered as a constraint-handling technique itself, using it alone would be computationally expensive, since, in highly constrained spaces, this procedure might require many iterations to reach the feasible region, and in extreme cases, a feasible solution could be impossible to obtain. Therefore, usually this technique is coupled with another constraint-handling technique, e.g., stochastic ranking or Deb's feasibility rules. In this work, gradient-based repair is coupled with the ε constrained method.

4. Computational experiments

To illustrate the benefits of the above-mentioned constraint-handling techniques in process engineering applications, 12 problems have been selected as representative in the specialized literature. Some trivial or simple examples have been excluded from this study. These problems present some mathematical characteristics typically found in engineering, e.g., non-linearities, equality and inequality constraints, binary and continuous variables. Some characteristics of these examples are provided in Table 1. In Appendix AAppendix A, the complete formulation of these problems is presented in details and additional information concerning local and global optimal solutions is also given.

The algorithms previously presented were implemented with MATLAB R2017b and all the following computational experiments were carried out with a processor Intel Xeon E3-1505M v6 and 32 Go RAM.

4.1. Parameters settings

In order to perform a fair comparison of the different constraint-handling methods, the parameters tuning has been set constant for all the test problems, so that, for a given technique, the best overall performance is obtained, excepting, obviously, the static penalty function where the tuning of the parameter r for each problem is intrinsic to this method. The actual parameters used are:

- Static penalty function. Parameter r is tuned following the *minimum penalty rule*. The precision of the parameter is set according to $r = x \times 10^y$, where x and y are integer numbers.
- Stochastic ranking. $P_f = 0.45$.
- ε constrained method. $\theta = 0.2NP$, $cp = 5$, $T_c = 0.2T_{max}$.
- Gradient-based repair. Identical parameters as for the ε constrained method above. Additionally, $P_g = 1$, $R_g = 3$.

Regarding the jDE algorithm, the only parameter to be tuned is the population size (NP), as the scaling factor (F) and crossover rate (CR) are adjusted by the algorithm. The population size is calculated as $NP = \min(100, 10n)$ where n is the number of decision variables. The algorithm stops if the current best solution is as close as 0.01% to the reported global optimal solution or if the number of function evaluations (NFEs) exceeds 200 000. Due to the stochastic nature of evolutionary algorithms, 50 independent executions are carried out for each problem and each method.

5. Results and discussion

The results obtained for the 12 optimization test problems are summarized in Table 2. The results are analyzed through the best, median and worst objective function value, "–" means no feasible solution was found. Feasibility and success rates represent respectively the rates of feasible and optimal solutions found out of 50 independent runs (considering the best solution found in each run). Complementary results regarding NFEs needed to achieve convergence are presented in Table 3. The computational times in Table 3 represent the overall elapsed time for the 50 runs.

For problem 1, the static penalty function and feasibility rules present a good performance, achieving an acceptable success rate. Even if the success rate of SR is acceptable, this method needs higher times for solving the problem. The constraint relaxation done by ε constrained method seems to have a negative effect on feasibility rules for this problem. It is worth nothing that in (Babu and Angira, 2006; Srinivas and Rangaiah, 2007), in order to efficiently obtain the global optimal solution, this problem was reformulated removing all equality constraints and eliminating dependent variables, and then a static penalty function was used. In contrast, the results obtained by the gradient-based repair method suggest that this reformulation is not necessary since this method finds the global optimum in all runs, with short CPU times (lower than 1 second per run).

Problem 2 constitutes a difficult case, containing 6 non-linear equality constraints that involve all the decision

Table 1: Brief description of example problems

Example	Decision variables Binary	Decision variables Continuous	Constraints (active)	Description
1	0	6	5(5)	Reactor network design
2	0	10	6(6)	Flowsheeting
3	1	1	2(1)	Process synthesis
4	1	2	2(2)	Process synthesis
5	1	2	3(2)	Process synthesis
6	3	2	5(3)	Process synthesis
7	2	6	8(6)	Reactor network design
8	4	3	9(5)	Process synthesis
9	3	8	9(7)	Planning problem
10	5	7	13(7)	Batch plant
11	5	7	13(8)	Batch plant
12	12	16	61(15)	Batch plant

variables. Indeed, not any one of the tested constraint-handling technique except gradient-based repair was able to found the optimum in any run. It is worth highlighting that, in (Rangaiah, 2009), this problem was also addressed by DE, considering the constraints as a system of non-linear equations which is then solved by an exact algorithm, so that the original problem is transformed into an unconstrained one. However, such repair process is computationally expensive, as it is performed for every individual at each generation. In this study, the same approach has been carried out for comparison purposes. The system of non-linear equations has been solved using the Levenberg-Marquardt algorithm embedded in the MATLAB software. This approach took approximately 40 seconds per run, i.e., about 20 times more than the gradient-based repair procedure.

Problems 3 and 5 consist in small and rather simple MINLP examples. All constraint-handling methods obtained an overall good performance in terms of success rate.

Regarding problem 4, this problem is modelled as a MINLP involving one non-linear equality constraint and one binary variable, which together, yield a rather high difficulty for the solution by feasibility rules, since this technique gets trapped in an "easy-to-access" local optimum. In addition, all the other techniques obtain a very good performance. It is noteworthy that in (Costa and Oliveira, 2001; Yiqing et al., 2007), the problem is reformulated by reducing one continuous variable and thus eliminating the equality constraint. This approach, although efficient, is problem-devoted and may not be practical in highly constrained real-world problems.

Problem 6 takes into account a MINLP problem with 3 binary variables and 2 equality constraints. Although this problem can be considered as a small one, its characteristics are not easy to overcome by feasibility rules, meaning that the first feasible solution is likely to be found far from the global minimum region. Further, the relaxation done by ε constrained method does not manage to obtain acceptable success rates, at least not with the parameters used here. Regarding stochastic ranking, static penalty function and gradient-based repair, they solve the problem efficiently, with much lower CPU times reported for the gradient-based repair technique. In (Cardoso et al., 1997; Srinivas and Rangaiah, 2007), the same problem was tackled, but the model was simplified by eliminating the continuous variables by means of the equality constraints.

For problem 7, feasibility rules and ε constrained method present a poor performance due to the existence of 2 binary variables and 4 equality constraints. Stochastic ranking and static penalty function present a fair to good performance. On the contrary, gradient-based repair method enables the algorithm to search in the whole search space before converging to an optimum. Again, this example was addressed in previous works (Cardoso et al., 1997; Costa and Oliveira, 2001; Yiqing et al., 2007; Srinivas and Rangaiah, 2007) by reformulating the problem in order to eliminate equality constraints and simultaneously, reducing the number of decision variables. Then, the remaining constraints are then handled by a static penalty function.

For problems 8 and 10, all constraint-handling techniques performed excellently, finding the global optimum in almost all runs. Indeed, these problems are easier instances in which the first feasible solutions found coincide with the region where the global optimal solution lies, even if the problem may present some difficulties regarding its mathematical properties (4 binary variables with 9 constraints and 5 binary variables with 7 constraints, respectively).

Problem 9 constitutes a difficult problem with 3 binary variables and 5 equality constraints involving all the continuous variables. For SR, the balance between feasible and infeasible solutions that have been stochastically generated is not sufficient to reach the global optimum region in most cases. According to Deb's feasibility rules, convergence to the global optimum is highly unlikely, since

Table 2: Experimental results in terms of objective function values

Problem (`optimum`)	Constr-handling	Best	Median	Worst	Mean	Std
1	St. penalty fcn.	−0.38881	−0.38871	−0.38802	−0.38870	1.4E-04
`-0.38881`	SR	−0.38881	−0.38871	−0.37867	−0.38813	1.9E-03
	Feas. rules	−0.38881	−0.38871	−0.38377	−0.38854	7.8E-04
	ε-constrained	−0.38877	−0.38871	−0.38720	−0.38848	4.2E-04
	Grad-based repair	−0.38881	−0.38876	−0.38872	−0.38878	3.9E-05
2	St. penalty fcn.	10 041 455	12 562 687	16 932 972	12 626 104	1.3E+06
`9490593`	SR	–	–	–	–	–
	Feas. rules	10 148 136	12 421 765	18 043 019	12 667 483	1.4E+06
	ε-constrained	10 603 444	–	–	13 457 709	1.9E+06
	Grad-based repair	9 490 594	9 490 600	9 490 603	9 490 600	2.5E+00
3	St. penalty fcn.	2.000	2.000	2.236	2.005	3.3E-02
`2.000`	SR	2.000	2.000	2.000	2.000	3.0E-05
	Feas. rules	2.000	2.000	2.236	2.019	6.5E-02
	ε-constrained	2.000	2.000	2.236	2.014	5.7E-02
	Grad-based repair	2.000	2.000	2.000	2.000	3.0E-05
4	St. penalty fcn.	2.124	2.124	2.558	2.168	1.3E-01
`2.124`	SR	2.124	2.124	2.558	2.142	8.6E-02
	Feas. rules	2.124	2.558	2.558	2.549	6.1E-02
	ε-constrained	2.124	2.124	2.558	2.150	1.0E-01
	Grad-based repair	2.124	2.124	2.124	2.124	1.7E-05
5	St. penalty fcn.	1.0766	1.0766	1.2500	1.0801	2.5E-02
`1.0765`	SR	1.0766	1.0766	1.2500	1.0835	3.4E-02
	Feas. rules	1.0766	1.0766	1.2500	1.0880	4.2E-02
	ε-constrained	1.0766	1.0766	1.0766	1.0766	1.8E-05
	Grad-based repair	1.0765	1.0766	1.0766	1.0766	3.1E-05
6	St. penalty fcn.	7.667	7.667	7.931	7.688	7.2E-02
`7.667`	SR	7.667	7.667	7.931	7.693	8.0E-02
	Feas. rules	7.667	7.931	8.240	7.928	1.1E-01
	ε-constrained	7.667	7.931	7.931	7.846	1.2E-01
	Grad-based repair	7.667	7.667	7.667	7.667	1.5E-05
7	St. penalty fcn.	99.239	99.240	107.374	101.355	3.6E+00
`99.238`	SR	99.238	99.239	107.374	100.703	3.7E+00
	Feas. rules	99.240	107.374	–	111.974	2.3E+01
	ε-constrained	99.239	107.374	107.374	103.795	4.1E+00
	Grad-based repair	99.239	99.240	99.240	99.239	2.7E-04
8	St. penalty fcn.	4.57958	4.57962	4.57968	4.57962	3.2E-05
`4.57958`	SR	4.57960	4.57967	4.57968	4.57966	1.7E-05
	Feas. rules	4.57959	4.57966	4.57968	4.57966	1.9E-05
	ε-constrained	4.57959	4.57966	4.57968	4.57966	2.1E-05
	Grad-based repair	4.57958	4.57958	4.57964	4.57958	9.0E-06
9	St. penalty fcn.	−1.9231	−1.7236	−1.4125	−1.6925	2.0E-01
`-1.9231`	SR	−1.9231	−1.7235	−0.2202	−1.5924	4.5E-01
	Feas. rules	−1.4125	−1.2138	0.7607	−0.8621	6.7E-01
	ε-constrained	−1.4099	−0.0011	0.7431	−0.0370	3.7E-01
	Grad-based repair	−1.9231	−1.9231	−1.9230	−1.9230	1.3E-04
10	St. penalty fcn.	38 500.0	38 500.1	38 500.2	38 500.1	5.3E-02
`38499.5`	SR	38 499.7	38 499.8	38 499.8	38 499.8	2.4E-02
	Feas. rules	38 499.9	38 500.1	38 500.2	38 500.1	5.8E-02
	ε-constrained	38 499.8	38 500.2	40 977.5	38 747.9	7.5E+02
	Grad-based repair	38 499.5	38 499.7	38 499.8	38 499.7	8.8E-02
11	St. penalty fcn.	106 756.6	106 756.8	106 756.9	106 756.8	8.7E-02
`106755.8`	SR	106 755.9	106 755.9	112 947.6	107 009.4	1.2E+03
	Feas. rules	106 756.6	112 947.2	122 607.8	110 739.1	4.3E+03
	ε-constrained	106 756.8	122 607.8	136 009.7	126 123.0	1.0E+04
	Grad-based repair	106 755.8	106 755.8	106 755.9	106 755.8	1.7E-02
12	St. penalty fcn.	304 660.5	310 155.0	311 349.9	308 282.6	2.7E+03
`285509.6`	SR	286 826.0	308 092.0	–	313 469.9	1.5E+04
	Feas. rules	310 350.1	322 711.5	332 793.1	322 466.2	7.0E+03
	ε-constrained	305 311.9	330 042.1	370 131.6	330 407.5	1.5E+04
	Grad-based repair	285 550.6	285 868.6	286 497.8	285 911.8	2.4E+02

Table 3: Experimental results in terms of NFEs needed for the algorithm to achieve convergence

Example	Constr-handling	Best	Median	Worst	Mean	Std	Feas. rate	Succ. rate	CPU Time(s)
1	St. penalty fcn.	14 340	64 230	200 000	88 734	73 334	100	90	10.12
	SR	45 480	156 840	200 000	153 434	44 795	100	70	91.58
	Feas. rules	27 660	149 490	200 000	136 008	53 187	100	86	15.61
	ε-constrained	32 700	193 320	200 000	156 421	61 826	100	54	19.06
	Grad-based repair	211	2142	5490	2176	1226	100	100	2.90
2	St. penalty fcn.	200 000	200 000	200 000	200 000	0	100	0	19.77
	SR	200 000	200 000	200 000	200 000	0	0	0	142.17
	Feas. rules	200 000	200 000	200 000	200 000	0	100	0	19.31
	ε-constrained	200 000	200 000	200 000	200 000	0	20	0	20.96
	Grad-based repair	15 300	19 100	22 800	19 094	1366	100	100	90.29
3	St. penalty fcn.	500	870	200 000	4851	28 162	100	98	1.24
	SR	640	1350	2640	1423	444	100	100	1.06
	Feas. rules	520	1110	200 000	17 006	54 511	100	92	4.25
	ε-constrained	420	1550	200 000	13 583	47 582	100	94	3.79
	Grad-based repair	33	203	387	226	89	100	100	0.31
4	St. penalty fcn.	1680	2775	200 000	22 441	59 792	100	90	3.57
	SR	3540	6825	200 000	14 887	38 229	100	96	6.63
	Feas. rules	10 200	200 000	200 000	196 214	26 843	100	2	31.19
	ε-constrained	24 090	28 245	200 000	38 545	41 240	100	94	7.41
	Grad-based repair	100	108	676	197	151	100	100	0.24
5	St. penalty fcn.	1830	2895	200 000	6804	27 885	100	98	1.16
	SR	4620	6465	200 000	14 212	38 332	100	96	6.02
	Feas. rules	2610	4065	200 000	19 623	53 738	100	92	3.20
	ε-constrained	22 800	28 575	31 200	27 988	2182	100	100	5.96
	Grad-based repair	93	175	1046	244	173	100	100	0.25
6	St. penalty fcn.	2400	4125	200 000	19 780	53 686	100	92	2.48
	SR	10 150	14 500	200 000	33 204	56 229	100	90	15.80
	Feas. rules	3900	200 000	200 000	184 422	53 365	100	8	22.18
	ε-constrained	14 550	200 000	200 000	143 018	83 935	100	32	18.23
	Grad-based repair	195	198	360	207	37	100	100	0.31
7	St. penalty fcn.	16 880	22 280	200 000	68 891	78 739	100	74	6.05
	SR	39 120	69 600	200 000	92 301	52 791	100	82	60.87
	Feas. rules	92 720	200 000	200 000	190 869	23 914	86	22	16.41
	ε-constrained	35 280	200 000	200 000	128 550	81 429	100	44	11.82
	Grad-based repair	3761	9591	20 639	10 082	3574	100	100	10.30
8	St. penalty fcn.	5810	6720	8540	6891	714	100	100	0.91
	SR	8330	11 480	14 770	11 739	1485	100	100	8.20
	Feas. rules	7840	9415	11 270	9362	785	100	100	1.25
	ε-constrained	8120	25 760	30 730	23 408	6563	100	100	3.28
	Grad-based repair	407	2133	9879	3691	3276	100	100	3.33
9	St. penalty fcn.	47 800	200 000	200 000	159 822	65 129	100	28	15.96
	SR	149 200	200 000	200 000	197 052	9930	100	18	149.53
	Feas. rules	200 000	200 000	200 000	200 000	0	100	0	20.78
	ε-constrained	200 000	200 000	200 000	200 000	0	100	0	20.90
	Grad-based repair	16 197	41 450	77 610	41 982	10 260	100	100	66.25
10	St. penalty fcn.	41 600	47 300	59 100	48 120	3810	100	100	5.83
	SR	66 000	76 900	105 000	79 904	9492	100	100	66.11
	Feas. rules	54 900	62 650	84 600	63 884	5679	100	100	7.93
	ε-constrained	68 400	85 850	200 000	95 892	35 719	100	90	12.28
	Grad-based repair	31 684	48 641	63 401	47 392	7172	100	100	61.88
11	St. penalty fcn.	43 100	48 150	58 500	48 542	3222	100	100	6.23
	SR	136 000	183 700	200 000	178 034	21 151	100	68	153.10
	Feas. rules	69 000	200 000	200 000	144 760	60 730	100	46	17.57
	ε-constrained	87 900	200 000	200 000	191 724	28 443	100	8	24.21
	Grad-based repair	2744	18 337	32 626	18 142	9130	100	100	27.32
12	St. penalty fcn.	200 000	200 000	200 000	200 000	0	100	0	84.35
	SR	200 000	200 000	200 000	200 000	0	62	0	177.20
	Feas. rules	200 000	200 000	200 000	200 000	0	100	0	72.45
	ε-constrained	200 000	200 000	200 000	200 000	0	100	0	88.10
	Grad-based repair	200 000	200 000	200 000	200 000	0	100	0	1136.68

this technique always prefers feasible solutions over the infeasible ones, whatever the quality of the objective function value. For the ε constrained method, the relaxation conducted on the equality constraints seems not to be sufficient to reach the global optimum. However, when this relaxation is combined with a reparation based on the constraint gradient, the algorithm is able to search over the entire space before converging to a suboptimal solution.

Problem 11 is the same as problem 10, just changing one parameter (see Appendix AAppendix A). Nevertheless, it seems that this slight modification makes the problem much more difficult, since SR, Deb's feasibility rules and ε constrained methods exhibit very different performance levels from those observed for problem 10. The feasible space has been modified in such a way that the global optimum lies now in a region that is difficult to reach. Yet, Table 3 highlights that the robustness of the static penalty function and gradient-based repair methods remained unchanged for the solution of this problem.

Problem 12 represents the optimization of a multiproduct batch plant (this is also the case for problems 10 and 11). This bigger instance of the problem is the most difficult example treated in this study. It is modelled as a MINLP problem involving 6 integer variables with 4 possible values each (equivalent to 12 binary variables). Additionally, it has 16 continuous variables and 61 inequality constraints. The global optimum corresponds to an ill-conditioned point, since variations as small as 0.01% in any of the 16 continuous variables produce infeasibility. For this problem, no constraint-handling technique could obtain the reported optimal solution. However, stochastic ranking and gradient-based repair techniques are the only ones to be able to find the global optimum region: SR in 20% of the runs and 100% of the runs for gradient-based repair. It seems that once the global optimal region has been identified, new solutions generated by DE operator are very likely to be infeasible, and even if the repair process acts upon them, the direction in which constraint violation is minimized is not necessarily the same as the direction in which the objective function decreases, so that the optimization process gets very slow.

Thus, in order to speed up the convergence to the global optimum, a local search is performed: the local optimizer Successive Quadratic Programming (SQP) is applied with a probability $0.1/NP$ for each individual once the ε level is equal to 0, i.e., once the algorithm has likely identified the optimal region. In this way, one individual is improved on average every 10 generations. The results obtained are reported in Table 4. It can be noted that the use of the local search greatly improves the performance of the algorithm. The global optimum is now found in all the runs with an acceptable computational time (approx. 7 seconds per run). The NFEs reported in Table 4 takes into account the evaluations of the objective function performed by both jDE and SQP. It is worth highlighting that the local search is carried out on average only on 5 individuals before solving the problem.

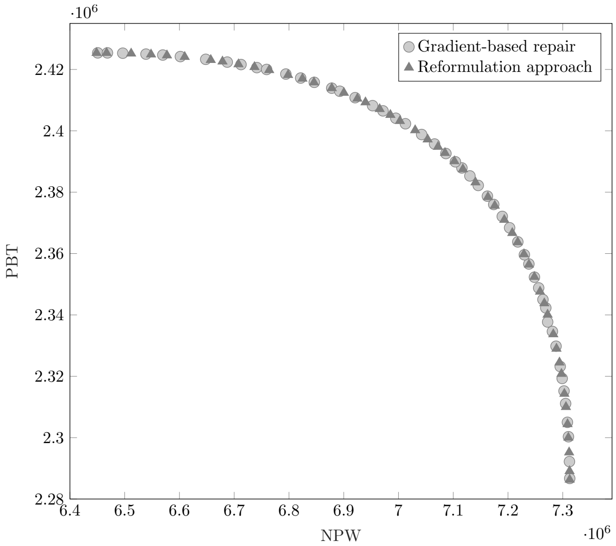

Figure 3: Approximation of the Pareto front of the biobjective Williams & Otto process example using NSGA-II with gradient-based repair method and the reformulation approach.

Summarizing, the above empirical study highlighted the importance of using an efficient constraint-handling technique when solving PE optimization problems, which are in general highly constrained problems. The numerical results obtained point out that gradient-based repair method is the most robust and thus promising method studied here. Also, as mentioned previously, one of the main motivations for using metaheuristics in PE area is their ability to simultaneously optimize multiple criteria. To illustrate this, in the next subsection, the gradient-based repair method is used to solve a biobjective version of the flow sheet example (Problem 2).

5.1. Additional example: Biobjective case study

This problem, presented in (Rangaiah, 2009), considers the maximization of two objectives: the profit before taxes (PBT) and the net present value (NPW) for the Williams & Otto process problem. The nondominated sorting genetic algorithm II (NSGA-II)(Deb et al., 2002) coupled with gradient-based repair as constraint-handling technique is used as a solution technique. Also, for comparison purposes, the constraint-handling strategy presented in (Rangaiah, 2009), in which all equality constraints are eliminated by means of solving a system of nonlinear equations, is explored. The obtained approximations to the Pareto front of this problems are presented in Figure 3.

The non-dominated solutions are obtained in one single run, unlike mathematical programming techniques in which multiple runs are needed to produce an approximation of the Pareto front. The importance of an efficient constraint-handling technique is also to be highlighted: no other constraint-handling technique studied in this work was able to find a non-dominated solution in the real Pareto front, actually no feasible solutions can be found except with the gradient-based repair procedure. Besides,

Table 4: NFEs for Problem 12 using gradient-based repair with local search.

Best	Median	Worst	Mean	Std	Feas. rate	Succ. rate	CPU Time(s)
40424	52658	129562	59368	18061	100	100	335.37

the reformulation approach proposed in (Rangaiah, 2009) is time consuming, taking approximately 45 sec per run (since the solution of the system of nonlinear equations has to be performed for every evaluation of the objective function) while, in contrast, the gradient-based repair approach takes approximately 1.5 sec per run. Also, it is worth mentioning that the Pareto front approximations obtained by both approaches are comparable, i.e., no approach outperforms the other.

6. Conclusions and perspectives

In this study, the performance of several constraint-handling techniques for EAs has been compared for the solution of a set of 12 test bench problems from the PE area. The empirical analysis conducted showed that the results' quality greatly depends on the constraint-handling technique used for the solution of problems with high number of constraints or binary variables.

The analysis of the dedicated literature has shown that the most widely used approach within EAs considers the reformulation of the model and the use of static penalty functions or feasibility rules as constraint-handling techniques. However, the results obtained in this work highlighted that the performance of this strategy, though acceptable in some cases, proved to be poor in others. Besides, among the constraint-handling techniques considered in this study, the gradient-based repair method deserves a special attention, as this constraint-handling technique is the only one capable of finding the global optimum region in all test problems. Coupled with ε constrained method, the search algorithm promotes the exploration of promising regions over the entire search space instead of getting trapped into a local optimum. It is worth emphasizing that, even if this method needs supplementary information (computation of constraints gradient), its excellent results both in terms of computational time and solution quality encourage its use. In addition, the use of gradient-based repair method in highly constrained mixed-integer problems seems to be not only adequate, but necessary in order to obtain satisfactory results. Finally, this work highlighted the unquestionable benefits obtained using this constraint-handling method, usually under-estimated in the devoted literature. Therefore, these conclusions allow reconsidering evolutionary algorithms as a serious approach for solving highly-constrained real-world optimization problems.

Besides, the good performance exhibited in the solution of the biobjective case study, permits to contemplate the solution of bigger instances of PE multi-objective problems. Also, as the gradient-based repair method can be coupled with any multi-objective evolutionary algorithm (MOEA), the solution of multi-objective MINLP problems related to PE, using more sophisticated MOEAs is under the scope of future work.

Appendix A. Appendix A. Test problems

This appendix describes the 12 global optimization test problems considered in this study. For all problems, the global optimum solution is reported as found in the literature. Additional information related to local optima and active constraints is also given.

Example 1. Reactor network design. Proposed in (Ryoo and Sahinidis, 1995), this problem involves the design of a sequence of two reactors of type CSTR, where the consecutive reactions A → B → C takes place. The objective is to maximize the concentration of product B (x_4) in the exit stream. The mathematical model is as follows:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}) = -x_4 \\ \text{s.t.} \quad & g_1(\mathbf{x}) = x_5^{0.5} + x_6^{0.5} - 4 \le 0 \\ & h_1(\mathbf{x}) = x_1 + k_1 x_1 x_5 - 1 = 0 \\ & h_2(\mathbf{x}) = x_2 - x_1 + k_2 x_2 x_6 = 0 \\ & h_3(\mathbf{x}) = x_3 + x_1 + k_3 x_3 x_5 - 1 = 0 \\ & h_4(\mathbf{x}) = x_4 - x_3 + x_2 - x_1 + k_4 x_4 x_6 = 0 \\ & \quad 0 \le x_i \le 1, \quad i = \{1,2,3,4\} \\ & \quad 1\mathrm{e}-5 \le x_i \le 16, \quad i = \{5,6\} \end{aligned}$$

where $k_1 = 0.09755988$, $k_2 = 0.99k_1$, $k_3 = 0.0391908$, $k_4 = 0.9k_3$. The global optimum is $\mathbf{x}^* = [0.771462,$ $0.516997,$ $0.204234,$ $0.388812,$ $3.036504,$ $5.096052]$ and $f(\mathbf{x}^*) = -0.388812$. Constraint g_1 is active. This example possesses a local minimum with an objective function value that is very close to that of the global solution. This local solution is at $\mathbf{x} = [1,\ 0.393,\ 0,\ 0.388,\ 0,\ 16]$ with $f = -0.3881$. Interestingly, this solution utilizes only one of the two reactors whereas the global solution makes use of both reactors.

Example 2. Flowsheeting. This problems considers the optimization of a flow sheet example of the Williams & Otto process (Biegler et al., 1997; Pintaric and Kravanja, 2006). Reactants A and B and the recycle stream enter the continuous-flow stirred-tank reactor, where the main product P is produced together with one by-product E and the waste product G, while C is an intermediate.

$$\mathrm{A} + \mathrm{B} \rightarrow \mathrm{C}$$

$$\mathrm{C} + \mathrm{B} \rightarrow \mathrm{P} + \mathrm{E}$$
$$\mathrm{P} + \mathrm{C} \rightarrow \mathrm{G}$$

In the decanter, component G is entirely removed from the other components. Product P is removed from the overhead of the distillation column, but some of the product is retained in the bottom due to the formation of an azeotrope. Part of the bottom stream is purged in order to avoid accumulation of the by-product, while most of it is recycled to the reactor. The purge stream has a substantial fuel value and can be sold on the market. The optimization variables account for the reactor volume, the reaction temperature, the purge fraction and the mass flow for each component, except for component P which is equal to 2160 kg/h. The objective is to minimize the total annual cost. The model is formulated as:

$$\begin{aligned}
\min \quad f(\mathbf{x}) = 2.2\Big[&168x_5 + 252x_1 + 2.22\big[x_1 + \\
&x_5 + \sum_{i=6}^{8}(1-x_4)x_i + 1.1(1-x_4)x_9\big] \\
&+ 84x_{10} + \frac{1041.6}{2.2} + 60x_2\rho\Big]
\end{aligned}$$

s.t.

$$\begin{aligned}
h_1(\mathbf{x}) &= x_5 + x_6(1-x_4) - \frac{k_1x_6x_7x_2\rho}{q_3^2} - x_6 = 0 \\
h_2(\mathbf{x}) &= x_1 + x_7(1-x_4) - \frac{(k_1x_6 + k_2x_8)x_7x_2\rho}{q_3^2} \\
&\quad - x_7 = 0 \\
h_3(\mathbf{x}) &= x_8(1-x_4) \\
&\quad + \frac{(2k_1x_6x_7 - 2k_2x_7x_8)x_2\rho}{q_3^2} \\
&\quad + \frac{(-k_3x_8(2160 + 0.1x_9))x_2\rho}{q_3^2} \\
&\quad - x_8 = 0 \\
h_4(\mathbf{x}) &= x_9(1-x_4) + \frac{2k_2x_7x_8x_2\rho}{q_3^2} - x_9 = 0 \\
h_5(\mathbf{x}) &= \frac{x_9(1-x_4)}{10} \\
&\quad + \frac{(k_2x_7 - 0.5k_3(2160 + 0.1x_9))x_8x_2\rho}{q_3^2} \\
&\quad - 2160 + 0.1x_9 = 0 \\
h_6(\mathbf{x}) &= \frac{1.5k_3(2160 + 0.1x_9)x_8x_2\rho}{q_3^2} - x_{10} = 0
\end{aligned}$$

$$\begin{gathered}
1\mathrm{e}4 \leq x_1 \leq 1.5\mathrm{e}4 \\
0.85 \leq x_2 \leq 10 \\
322 \leq x_3 \leq 378 \\
0 \leq x_4 \leq 0.99 \\
0 \leq x_i \leq 1\mathrm{e}5 \quad \forall i \in \{5, \ldots, 10\}
\end{gathered}$$

where

$$\begin{aligned}
q_3 &= x_6 + x_7 + x_8 + 1.1x_9 + x_{10} + 2160 \\
k_1 &= 5.9755\mathrm{e}9 \cdot \exp\left(\frac{-1.2\mathrm{e}4}{x_3}\right) \\
k_2 &= 2.5962\mathrm{e}12 \cdot \exp\left(\frac{-1.5\mathrm{e}4}{x_3}\right) \\
k_3 &= 9.6283\mathrm{e}15 \cdot \exp\left(\frac{-2\mathrm{e}4}{x_3}\right) \\
\rho &= 801
\end{aligned}$$

The optimum lies at $\mathbf{x}^* = [10878.60, 7.90, 342.11, 0.102, 4807.37, 11122.40, 39668.61, 2874.52, 61925.59, 1101.336]$ with $f(\mathbf{x}^*) = 9490592.6$.

Example 3. Process synthesis MINLP. This is a little process synthesis problem with only two decision variables. It was proposed by (Kocis and Grossmann, 1988), and also found in (Ryoo and Sahinidis, 1995):

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 2x_1 + x_2 \\
\text{s.t.} \quad & g_1(\mathbf{x}) = 1.25 - x_1^2 - x_2 \leq 0 \\
& g_2(\mathbf{x}) = x_1 + x_2 - 1.6 \leq 0 \\
& 0 \leq x_1 \leq 1.6 \\
& x_2 = \{0, 1\}
\end{aligned}$$

The global minimum is [0.5, 1] with $f = 2$. There is a local minimum at [1.118, 0] with $f = 2.236$. Constraint g_1 is active.

Example 4. MINLP. This example is taken from (Kocis and Grossmann, 1987):

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 2x_1 + x_2 - x_3 \\
\text{s.t.} \quad & g_1(\mathbf{x}) = -x_1 + x_2 + x_3 \leq 0 \\
& h_1(\mathbf{x}) = x_1 - 2\exp(-x_2) = 0 \\
& 0.5 \leq x_1 \leq 1.4 \\
& 0 \leq x_2 \leq 2 \\
& x_3 = \{0, 1\}
\end{aligned}$$

There is one local optimum at [0.853, 0.853, 0] with $f = 2.558$. The global minimum is $\{\mathbf{x}^*; f(\mathbf{x}^*)\} = \{1.375, 0.375, 1; 2.124\}$. Constraint g_1 is active.

Example 5. MINLP. Problem taken from (Floudas, 1995):

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = -0.7x_3 + 5(x_1 - 0.5)^2 + 0.8 \\
\text{s.t.} \quad & g_1(\mathbf{x}) = -\exp(x_1 - 0.2) - x_2 \leq 0 \\
& g_2(\mathbf{x}) = x_2 + 1.1x_3 + 1 \leq 0 \\
& g_3(\mathbf{x}) = x_1 - 1.2x_3 - 0.2 \leq 0 \\
& 0.2 \leq x_1 \leq 1 \\
& -2.22554 \leq x_2 \leq -1 \\
& x_3 = \{0, 1\}
\end{aligned}$$

The global minimum is at $[0.94194, -2.1, 1]$ where $f(\mathbf{x}^*) = 1.07654$. Constraints g_1 and g_2 are active.

Example 6. MINLP. Proposed in (Kocis and Grossmann, 1988), and also reported in (Floudas et al., 1989; Ryoo and Sahinidis, 1995; Cardoso et al., 1997):

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 2x_1 + 3x_2 + 1.5x_3 + 2x_4 - 0.5x_5 \\
\text{s.t.} \quad & g_1(\mathbf{x}) = x_1 + x_3 - 1.6 \leq 0 \\
& g_2(\mathbf{x}) = 1.333x_2 + x_4 - 3 \leq 0 \\
& g_3(\mathbf{x}) = -x_3 - x_4 + x_5 \leq 0 \\
& h_1(\mathbf{x}) = x_1^2 + x_3 - 1.25 = 0 \\
& h_2(\mathbf{x}) = x_2^{1.5} + 1.5x_4 - 3 = 0 \\
& 0 \leq x_1 \leq 1.5 \\
& 0 \leq x_2 \leq 2.2 \\
& x_i = \{0, 1\}, \quad i = \{3, 4, 5\}
\end{aligned}$$

There are 2^3 different combinations of the binary variables, of these only one combination is infeasible because it violates the pure integer constraint. The global solution is $\mathbf{x}^* = [1.118, 1.3310, 0, 1, 1]$ with $f(\mathbf{x}^*) = 7.667$. Constraint g_3 is active.

Example 7. Reactor network design. This problem, taken from (Kocis and Grossmann, 1989) and also studied in (Cardoso et al., 1997), is a two-reactor problem, where selection is to be made among two candidate reactors the one that minimizes the cost of producing a desired product. The MINLP formulation is given as:

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 7.5x_5 + 5.5x_6 + 7x_1 + 6x_2 \\
& \qquad + 5(x_3 + x_4) \\
\text{s.t.} \quad & g_1(\mathbf{x}) = x_1 - 10x_5 \leq 0 \\
& g_2(\mathbf{x}) = x_2 - 10x_6 \leq 0 \\
& g_3(\mathbf{x}) = x_3 - 20x_5 \leq 0 \\
& g_4(\mathbf{x}) = x_4 - 20x_6 \leq 0 \\
& h_1(\mathbf{x}) = x_5 + x_6 - 1 = 0 \\
& h_2(\mathbf{x}) = x_7 - 0.9x_3(1 - \exp(-0.5x_1)) = 0 \\
& h_3(\mathbf{x}) = x_8 - 0.8x_4(1 - \exp(-0.4x_2)) = 0 \\
& h_4(\mathbf{x}) = x_7 + x_8 - 10 = 0 \\
& x_i \geq 0, \quad i = \{1, 2, 3, 4, 7, 8\} \\
& x_i = \{0, 1\}, \quad i = \{5, 6\}
\end{aligned}$$

The global minimum is $\mathbf{x}^* = [3.514,\ 0,\ 13.428,\ 0,\ 1,\ 0,\ 10,\ 0.0001]$ with $f = 99.238$. Constraints g_2 and g_4 are active.

Example 8. Process synthesis MINLP. This example is taken from (Yuan et al., 1989), and is also found in (Floudas et al., 1989; Ryoo and Sahinidis, 1995; Cardoso et al., 1997; Yiqing et al., 2007):

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = (x_4 - 1)^2 + (x_5 - 2)^2 + (x_6 - 1)^2 \\
& - \ln(x_7 + 1) + (x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2 \\
\text{s.t.} \quad & g_1(\mathbf{x}) = \sum_{i=1}^{6} x_i - 5 \leq 0 \\
& g_2(\mathbf{x}) = \sum_{i=1}^{4} x_i^2 - 5.5 \leq 0 \\
& g_3(\mathbf{x}) = x_4 + x_1 - 1.2 \leq 0 \\
& g_4(\mathbf{x}) = x_5 + x_2 - 1.8 \leq 0 \\
& g_5(\mathbf{x}) = x_6 + x_3 - 2.5 \leq 0 \\
& g_6(\mathbf{x}) = x_7 + x_1 - 1.2 \leq 0 \\
& g_7(\mathbf{x}) = x_5^2 + x_2^2 - 1.64 \leq 0 \\
& g_8(\mathbf{x}) = x_6^2 + x_3^2 - 4.25 \leq 0 \\
& g_9(\mathbf{x}) = x_5^2 + x_3^2 - 4.64 \leq 0 \\
& x_i \geq 0, \quad i = \{1, 2, 3\} \\
& x_i = \{0, 1\}, \quad i = \{4, 5, 6, 7\}
\end{aligned}$$

The global minimum is $\{\mathbf{x}^*; f(\mathbf{x}^*)\} = \{0.2,\ 0.8,\ 1.9079,\ 1,\ 1,\ 0,\ 1;\ 4.579582\}$. Constraints g_3, g_4, g_6, g_7 and g_9 are active.

Example 9. Planning problem. First introduced in (Kocis and Grossmann, 1988), this example represents a small planning problem, in which several alternatives are proposed for obtaining product C. The goal is to produce the profitable product C from B that is purchased from a market or produced from raw material A. There are also two paths to produce B from A. The problem is modelled as a MINLP:

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = 3.5x_1 + x_2 + 1.5x_3 + 7x_5 + x_6 \\
& \qquad + 1.2x_7 + 1.8x_8 - 11x_{11} \\
\text{s.t.} \quad & g_1(\mathbf{x}) = x_4 - 5x_1 \leq 0 \\
& g_2(\mathbf{x}) = x_9 - 5x_2 \leq 0 \\
& g_3(\mathbf{x}) = x_{10} - 5x_3 \leq 0 \\
& g_4(\mathbf{x}) = x_{11} - 1 \leq 0 \\
& h_1(\mathbf{x}) = x_6 - \ln(1 + x_9) = 0 \\
& h_2(\mathbf{x}) = x_7 - 1.2\ln(1 + x_{10}) = 0 \\
& h_3(\mathbf{x}) = x_{11} - 0.9x_4 = 0 \\
& h_4(\mathbf{x}) = -x_4 + \sum_{i=5}^{7} x_i = 0 \\
& h_5(\mathbf{x}) = -x_8 + x_9 + x_{10} = 0 \\
& x_i = \{0, 1\}, \quad i = \{1, 2, 3\} \\
& x_i \geq 0, \quad \forall i \\
& x_6 \leq 5 \\
& x_{11} \leq 1
\end{aligned}$$

The model contains three binary variables and five continuous variables. The global minimum is $\mathbf{x}^* = [1,\ 0,\ 1,\ 1.11111081,\ 0,\ 0,\ 1.11111081,\ 1.5242038,\ 0,\ 1.5242038,\ 0.99999978]$ and $f(\mathbf{x}^*) = -1.9231$. Constraints g_2 and g_4 are active. There is a local optimum at $\mathbf{x} = [1,\ 1,\ 1,\ 1.111,$

0, 0.446744, 0.664156, 1.30208, 0.563058, 0.739121, 1] with $f(\mathbf{x}) = -1.41252645$.

Examples 10/11/12. Multi-product batch plant design. The multi-product batch plant consists of M processing stages in series where fixed amounts Q_i of N products have to be manufactured. The objective is to determine for each stage j the number of parallel units N_j and their sizes V_j and for each product i the corresponding batch sizes B_i and cycle times $T_{\mathrm{L}i}$. The problem data are the horizon time H, the size factors S_{ij} and processing times t_{ij} of product i in stage j, the required productions Q_{ij}, and appropriate cost functions α_j and β_j. The mathematical formulation of this problem is as follows (Grossmann and Sargent, 1979; Kocis and Grossmann, 1988):

$$\begin{aligned}
\min \quad & \sum_{j=1}^{M} \alpha_j N_j V_j^{\beta_j} \\
\text{s.t.} \quad & \sum_{i=1}^{N} \frac{Q_i T_{\mathrm{L}i}}{B_i} - H \leq 0 \\
& S_{ij} B_i - V_j \leq 0 \\
& t_{ij} - N_j T_{\mathrm{L}i} \leq 0 \\
& 1 \leq N_j \leq N_j^{\mathrm{u}} \\
& V_j^{\mathrm{l}} \leq V_j \leq V_j^{\mathrm{u}} \\
& T_{\mathrm{L}i}^{\mathrm{l}} \leq T_{Li} \leq T_{\mathrm{L}i}^{\mathrm{u}} \\
& B_j^{\mathrm{l}} \leq B_j \leq B_j^{\mathrm{u}} \\
& N_j \text{ integer}
\end{aligned}$$

The bounds N_j^{u}, V_j^{l}, V_j^{u} are specified by the problem and appropriate bounds for $T_{\mathrm{L}i}$ and B_i can be determined as follows:

$$\begin{aligned}
T_{\mathrm{L}i}^{\mathrm{l}} &= \max_j \frac{t_{ij}}{N_j^{\mathrm{u}}} \\
T_{\mathrm{L}i}^{\mathrm{u}} &= \max_j t_{ij} \\
B_i^{\mathrm{l}} &= \frac{Q_i}{H} T_{\mathrm{L}i}^{\mathrm{l}} \\
B_i^{\mathrm{u}} &= \min\left(Q_i, \min_j \frac{V_j^{\mathrm{u}}}{S_{ij}}\right)
\end{aligned}$$

The data corresponding to these problems are presented in Table A.5. For all examples the parameters α_j, β_j and H are 250, 0.6 and 6000, respectively.

References

Ali L, Sabat SL, Udgata SK. Particle swarm optimisation with stochastic ranking for constrained numerical and engineering benchmark problems. International Journal of Bio-Inspired Computation 2012;4(3):155–66.

Almaraz SDL, Azzaro-Pantel C, Montastruc L, Boix M. Deployment of a hydrogen supply chain by multi-objective/multi-period optimisation at regional and national scales. Chemical Engineering Research and Design 2015;104:11–31.

Ayala HVH, Keller P, de Fátima Morais M, Mariani VC, dos Santos Coelho L, Rao RV. Design of heat exchangers using a novel multiobjective free search differential evolution paradigm. Applied Thermal Engineering 2016;94:170–7.

Babu B, Angira R. Modified differential evolution (MDE) for optimization of non-linear chemical processes. Computers & Chemical Engineering 2006;30(6-7):989–1002.

Biegler LT, Grossmann IE, Westerberg AW. Systematic methods for chemical process design. Prentice Hall, Old Tappan, NJ (United States), 1997.

Bozorg-Haddad O, Solgi M, Loaiciga HA. Meta-heuristic and evolutionary algorithms for engineering optimization. volume 294. John Wiley & Sons, 2017.

Brest J, Greiner S, Boskovic B, Mernik M, Zumer V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 2006;10(6):646–57.

Campbell SL, Meyer CD. Generalized inverses of linear transformations. SIAM, 2009.

Cardoso M, Salcedo R, de Azevedo SF, Barbosa D. A simulated annealing approach to the solution of MINLP problems. Computers & Chemical Engineering 1997;21(12):1349–64.

Chootinan P, Chen A. Constraint handling in genetic algorithms using a gradient-based repair method. Computers & operations research 2006;33(8):2263–81.

Coello CAC. Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 2000;41(2):113–27.

Coello Coello CA. Constraint-handling techniques used with evolutionary algorithms. In: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion. ACM; 2016. p. 563–87.

Costa L, Oliveira P. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems. Computers & Chemical Engineering 2001;25(2-3):257–66.

Deb K. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering 2000;186(2-4):311–38.

Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6(2):182–97. doi:10.1109/4235.996017.

Dowling AW, Biegler LT. A framework for efficient large scale equation-oriented flowsheet optimization. Computers & Chemical Engineering 2015;72:3–20.

Fan Z, Li H, Wei C, Li W, Huang H, Cai X, Cai Z. An improved epsilon constraint handling method embedded in MOEA/D for constrained multi-objective optimization problems. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE; 2016. p. 1–8.

Fan Z, Li W, Cai X, Huang H, Fang Y, You Y, Mo J, Wei C, Goodman E. An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions. Soft Computing 2017;:1–20.

Fan Z, Liu J, Sorensen T, Wang P. Improved differential evolution based on stochastic ranking for robust layout synthesis of MEMS components. IEEE Transactions on Industrial Electronics 2009;56(4):937–48.

Floudas C, Aggarwal A, Ciric A. Global optimum search for nonconvex NLP and MINLP problems. Computers & Chemical Engineering 1989;13(10):1117–32.

Floudas CA. Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford University Press, 1995.

Floudas CA, Gounaris CE. A review of recent advances in global optimization. Journal of Global Optimization 2009;45(1):3–38.

Grossmann IE, Sargent RWH. Optimum design of multipurpose chemical plants. Industrial & Engineering Chemistry Process Design and Development 1979;18(2):343–8. doi:10.1021/i260070a031.

Kaiser NM, Flassig RJ, Sundmacher K. Probabilistic reactor design in the framework of elementary process functions. Computers & Chemical Engineering 2016;94:45–59.

Table A.5: Input data for Examples 10/11/12.

Example	M	N	N_j^{u}	V_j^{l}	V_j^{u}	S_{ij}	t_{ij}	Q_i
10	3	2	3	250	2500	$\begin{bmatrix} 2 & 3 & 4 \\ 4 & 6 & 3 \end{bmatrix}$	$\begin{bmatrix} 8 & 20 & 8 \\ 16 & 4 & 4 \end{bmatrix}$	$\begin{bmatrix} 40000 \\ 20000 \end{bmatrix}$
11	3	2	3	250	2500	$\begin{bmatrix} 2 & 3 & 4 \\ 4 & 6 & 3 \end{bmatrix}$	$\begin{bmatrix} 8 & 20 & 8 \\ 16 & 4 & 4 \end{bmatrix}$	$\begin{bmatrix} 200000 \\ 100000 \end{bmatrix}$
12	6	5	4	300	3000	$\begin{bmatrix} 7.9 & 2.0 & 5.2 & 4.9 & 6.1 & 4.2 \\ 0.7 & 0.8 & 0.9 & 3.4 & 2.1 & 2.5 \\ 0.7 & 2.6 & 1.6 & 3.6 & 3.2 & 2.9 \\ 4.7 & 2.3 & 1.6 & 2.7 & 1.2 & 2.5 \\ 1.2 & 3.6 & 2.4 & 4.5 & 1.6 & 2.1 \end{bmatrix}$	$\begin{bmatrix} 6.4 & 4.7 & 8.3 & 3.9 & 2.1 & 1.2 \\ 6.8 & 6.4 & 6.5 & 4.4 & 2.3 & 3.2 \\ 1.0 & 6.3 & 5.4 & 11.9 & 5.7 & 6.2 \\ 3.2 & 3.0 & 3.5 & 3.3 & 2.8 & 3.4 \\ 2.1 & 2.5 & 4.2 & 3.6 & 3.7 & 2.2 \end{bmatrix}$	$\begin{bmatrix} 250000 \\ 150000 \\ 180000 \\ 160000 \\ 120000 \end{bmatrix}$

Kocis GR, Grossmann IE. Relaxation strategy for the structural optimization of process flow sheets. Industrial & Engineering Chemistry Research 1987;26(9):1869–80.

Kocis GR, Grossmann IE. Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis. Industrial & Engineering Chemistry Research 1988;27(8):1407–21. doi:10.1021/ie00080a013.

Kocis GR, Grossmann IE. A modelling and decomposition strategy for the MINLP optimization of process flowsheets. Computers & Chemical Engineering 1989;13(7):797–819.

Mallipeddi R, Suganthan PN. Ensemble of constraint handling techniques. IEEE Transactions on Evolutionary Computation 2010;14(4):561–79.

Mezura-Montes E, Coello CAC. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary computation 2005;9(1):1–17.

Mezura-Montes E, Coello CAC. Constraint-handling in nature-inspired numerical optimization: past, present and future. Swarm and Evolutionary Computation 2011;1(4):173–94.

Nanakorn P, Meesomklin K. An adaptive penalty function in genetic algorithms for structural design optimization. Computers & Structures 2001;79(29-30):2527–39.

Padhye N, Mittal P, Deb K. Feasibility preserving constraint-handling strategies for real parameter evolutionary optimization. Computational Optimization and Applications 2015;62(3):851–90.

Pintaric ZN, Kravanja Z. Selection of the economic objective function for the optimization of process flow sheets. Industrial & engineering chemistry research 2006;45(12):4222–32.

Ponsich A, Coello CC. Differential evolution performances for the solution of mixed-integer constrained process engineering problems. Applied Soft Computing 2011;11(1):399–409.

Qin AK, Suganthan PN. Self-adaptive differential evolution algorithm for numerical optimization. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. IEEE; volume 2; 2005. p. 1785–91.

Rangaiah GP. Multi-objective optimization: techniques and applications in chemical engineering. volume 1. World Scientific, 2009.

Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Transactions on Evolutionary Computation 2000;4(3):284–94.

Ryoo HS, Sahinidis NV. Global optimization of nonconvex NLPs and MINLPs with applications in process design. Computers & Chemical Engineering 1995;19(5):551–66.

Srinivas M, Rangaiah G. Differential evolution with tabu list for solving nonlinear and mixed-integer nonlinear programming problems. Industrial & Engineering Chemistry Research 2007;46(22):7126–35.

Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 1997;11(4):341–59.

Takahama T, Sakai S. Constrained optimization by ε constrained particle swarm optimizer with ε-level control. In: Abraham A, Dote Y, Furuhashi T, Köppen M, Ohuchi A, Ohsawa Y, editors. Soft Computing as Transdisciplinary Science and Technology. Berlin, Heidelberg: Springer Berlin Heidelberg; 2005. p. 1019–29.

Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: 2006 IEEE International Conference on Evolutionary Computation. IEEE; 2006. p. 1–8.

Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation. In: Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE; 2010. p. 1–9.

Tanabe R, Fukunaga A. Success-history based parameter adaptation for differential evolution. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE; 2013. p. 71–8.

Tawarmalani M, Sahinidis NV, Sahinidis N. Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications. volume 65. Springer Science & Business Media, 2002.

Tessema B, Yen GG. A self adaptive penalty function based algorithm for constrained optimization. In: Evolutionary Computation, 2006. CEC 2006. IEEE Congress on. IEEE; 2006. p. 246–53.

Vesterstrom J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753). IEEE; volume 2; 2004. p. 1980–7.

Woo Yb, Cho S, Kim J, Kim BS. Optimization-based approach for strategic design and operation of a biomass-to-hydrogen supply chain. International Journal of Hydrogen Energy 2016;41(12):5405–18.

Yang Z, Cai X, Fan Z. Epsilon constrained method for constrained multiobjective optimization problems: some preliminary results. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. ACM; 2014. p. 1181–6.

Yiqing L, Xigang Y, Yongjian L. An improved PSO algorithm for solving non-convex NLP/MINLP problems with equality constraints. Computers & chemical engineering 2007;31(3):153–62.

Yuan X, Zhang S, Pibouleau L, Domenech S. Une methode d'optimization nonlineaire en variables mixtes pour la conception de procedes. RAIRO - Operations Research 1989;.

Zhang J, Sanderson AC. JADE: Adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 2009;13(5):945–58. doi:10.1109/TEVC.2009.2014613.

Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences 2008;178(15):3043–74.

Zhu Z, Xu D, Liu X, Zhang Z, Wang Y. Separation of acetonitrile/methanol/benzene ternary azeotrope via triple column pressure-swing distillation. Separation and purification technology 2016;169:66–77.