EVOLUTIONARY ALGORITHMS AND INTELLIGENT TOOLS
IN ENGINEERING OPTIMIZATION
W. Annicchiarico, J. Périaux, M. Cerrolaza and G. Winter (Eds.)

USING EVOLUTION STRATEGIES TO SOLVE CONSTRAINED OPTIMIZATION PROBLEMS

Efrén Mezura-Montes[*]
Carlos A. Coello Coello[D]
Evolutionary Computation Group (EVOCINV)
CINVESTAV-IPN
Computer Science Section
Av. IPN No. 2508, Col. San Pedro Zacatenco
México D.F., México 07359
Email: emezura@computación.cs.cinvestav.mx
ccoello@cs.cinvestav.mx
Web page: http://www.cs.cinvestav.mx/~EVOCINV

Arturo Hernández-Aguirre[•]
Center of Research in Mathematics
CIMAT
Department of Computer Science
A.P. 402, Guanajuato, Gto. C.P. 36000
Guanajuato, Guanajuato., México
Email: artha@cimat.mx
Web page: http://www.cimat.mx/~artha/

Abstract. In this chapter, we present an overview of the main approaches that have been used to solve constrained optimization problems using evolution strategies. Furthermore, we propose a novel approach to incorporate constraints into an evolution strategy used to solve global optimization problems. This new approach uses a set of simple rules to guide the search towards the feasible region and a diversity mechanism to maintain good infeasible solutions in the population during all the evolutionary process. The approach combines two recombination operators (discrete and intermediate) and also adopts a reduced initial stepsize for the mutation operator. The new approach is not only simple, but also highly competitive with respect to the algorithms more representative of the state-of-the-art in the area.

Key words: Evolution strategies, global optimization, constraint-handling.

1 INTRODUCTION

Evolution strategies (ES) are a class of evolutionary algorithm (EA) that have been found to perform quite well in a wide variety of unconstrained optimization problems. When dealing with constrained problems, evolution strategies are normally combined with a penalty function whose definition is problem-dependant. In fact, the study of alternative mechanisms to incorporate constraints into an evolution strategy have been only scarcely studied.

[*] The author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN in México City.
[•] The author acknowledges support from CONCyTEG through project No. 04-02-K117-037 Part 1
[Δ] The author acknowledges support from CONACyT through project No. 42435-Y

In this chapter, we present an overview of the main approaches that have been used to solve constrained optimization problems using evolution strategies. Furthermore, we propose a novel approach to incorporate constraints into an evolution strategy used to solve global optimization problems. The new approach is not only simple, but also highly competitive with respect to the algorithms more representative of the state-of-the-art in the area.

The chapter is organized as follows. First, we briefly introduce some basic concepts related to evolution strategies. We cover aspects such as encoding of the solutions, the selection process, types of mutation and recombination operators available and self-adaptation. Also, we briefly discuss the differences between an evolution strategy and any of the two other main evolutionary algorithms in current use (i.e., evolutionary programming and genetic algorithms). After that, we provide a taxonomy of constraint-handling techniques that have been proposed for evolutionary algorithms and we discuss the main approaches currently available (e.g., different types of penalty functions, the separation of objective and constraints, repair algorithms, etc.).

As indicated before, the chapter also provides a review of the main approaches used to solve constrained optimization problems which are based on an evolution strategy. We describe, for example, the Adaptive Segregational Constraint Handling Evolution Strategy (ASCHEA), Stochastic Ranking and the Pareto Archive and Dominance Selection with Shrinkable Search Space (PAS[4]).

After the review of the state-of-the-art techniques, we proceed to describe our proposal for a novel approach which exploits the main features of an evolution strategy to solve global optimization problems. This new approach uses a set of simple rules to guide the search towards the feasible region and a diversity mechanism to maintain good infeasible solutions in the population during all the evolutionary process. To enhance the search power of the evolution strategy adopted, we propose to combine two recombination operators (discrete and intermediate) and also the use of reduced initial stepsizes for the mutation operator in order to take advantage of finer movements in the search space and inside the feasible region as well. We test our approach in some well-known benchmark problems. Our proposed approach is also compared with respect to other algorithms which are representative of the state-of-the-art in the area. Finally, we provide our conclusions and some potential paths for future research.

2 THE EVOLUTION STRATEGY

The ES were proposed by Bienert, Rechenberg and Schwefel who used them to solve hydrodynamical problems[1,2] . The main idea was to allow the evolutionary process to evolve, besides the solutions of a problem, the parameters of the algorithm. The first ES version was the (1+1)-ES which uses just one individual that is mutated using a normally distributed random number with mean zero and an identical stepsize value for each decision variable. The best solution between the parent and the offspring is chosen and the other one is eliminated. Rechenberg derived a convergence rate theory and proposed a rule for changing the stepsize value of mutations in a (1+1)-ES. This is the so-called *"1/5-success rule"* [3].

The first multimembered ES was the (μ+1)-ES, which was designed by Rechenberg and is described in detail by Bäck et al.[4]. In this approach, "μ" parent solutions recombine to generate one offspring. This solution is also mutated and, if it is better, it will replace the worst parent solution. Note however that the (μ+1)-ES has not been too popular in the literature. However, it provided the transition to the state-of-the-art multimembered ES.

The (μ+λ)-ES and the (μ,λ)-ES were proposed by Schwefel. In the first one, the best "μ" individuals out of the union of the "μ" original parents and their "λ" offspring will survive for the next generation. On the other hand, in the (μ,λ)-ES, the best "μ" will be selected only from the "λ" offspring.

The (μ+λ)-ES uses an implicit elitist mechanism and solutions can survive more than one generation. Meanwhile, in the (μ,λ)-ES, solutions only survive one generation (this is the type of selection traditionally adopted in genetic algorithms[5]). Instead of the *"1/5-success rule"*, each individual includes a stepsize value for each decision variable. Moreover, for each combination of two stepsize values, a rotation angle is included. These angles are used to perform a correlated mutation. This mutation allows each individual to look for a search direction. The stepsize values and the angles of each individual are called strategy parameters. They are also recombined and mutated. A (μ+λ)-ES or (μ,λ)-ES individual can be seen as follows: $a(i)(\vec{x},\vec{s},\vec{q})$, where *"i"* is the number of individual in the population, $x \in \Re^n$ is a vector of *"n"* decision variables, $\vec{s}$ is a vector of *"n"* stepsize values and $\vec{q}$ is a vector of *"n(n-1)/2"* rotation angles where $q_i \in [-p, p]$. One of the main differences between a genetic algorithm and en evolution strategy relies on the way in which a solution is represented (see Figure 1).

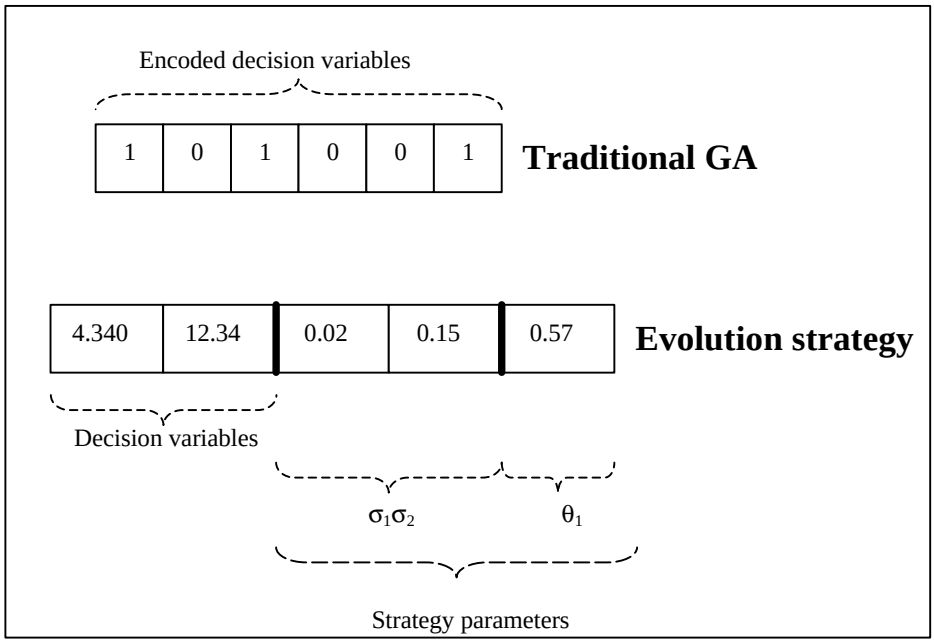

Figure 1: Representation of individuals of a genetic algorithm and an evolution strategy.

Recombination can be sexual (two parents) or panmictic (more than two parents). There are two main types of recombination: (1) Discrete and (2) Intermediate. Both can be either sexual or panmictic. Also, Schwefel[6] proposed to generalize intermediate recombination by allowing arbitrary weight factors from the interval *[0,1]* to be used anew for each component of the chromosome. For a complete description of the recombination operators normally available we provide a list in Table 1 (Refer to Bäck[7] for details).

	Operation	Type of recombination
$offspring_i$ =	$Parent\_1_i$ or $Parent\_2_i$	Discrete
	$Parent\_1_i$ or $Parent\_J_i$	Panmictic discrete
	$(Parent\_1_i + Parent\_2_i)/2$	Intermediate
	$(Parent\_1_i + Parent\_J_i)/2$	Panmictic intermediate
	$(1-\chi)(Parent\_1_i)+(\chi)(Parent\_2_i)$	Generalized intermediate
	$(1-\chi)(Parent\_1_i)+(\chi)(Parent\_J_i)$	Panmictic generalized intermediate

Table 1: Different recombination operators used in ES.

In Table 1, "Parent_1" and "Parent_2" are the parents used for the sexual recombination. "Parent_J" means a different parent for each gene (variable of the problem) in the chromosome. "χ_i" is the weight factor created anew for each decision variable and used in the generalized recombination. The mutation is calculated in the following way:

$$s_i' = s_i \cdot \exp(t' \cdot N(0,1) + t \cdot N_i(0,1)) \quad (1)$$

$$q_j' = q_j + b \cdot N_j(0,1) \quad (2)$$

$$\vec{x}' = \vec{x} + \vec{N}(0, C(\vec{s}', \vec{q}')) \quad (3)$$

where t and t' are interpreted as "learning rates" and are defined by Schwefel[6] as: $t = \left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $t' = \left(\sqrt{2n}\right)^{-1}$ and $b \approx 0.0873$.

Some authors use correlated mutation, but this implies an extra computational effort in order to process the value of each angle and also to rotate the individual. Moreover, some extra memory space is needed to store all the different angles per individual (the angles are formed by the combination of all the axis based on the number of decision variables of the problem). If non-correlated mutation is preferred, the computational cost and the storage space for each individual gets lower. If a non-correlated mutation is used, the mutation expressions are:

$$s_i' = s_i \cdot \exp(t' \cdot N(0,1) + t \cdot N_i(0,1)) \quad (4)$$

$$\vec{x}_i' = \vec{x}_i + s_i' \cdot N_i(0,1) \quad (5)$$

As can be noted, the genetic operators (recombination and mutation) are applied to the values of the decision variables as well as the strategy parameters. In this way, the ES is able to evolve both, the solutions of the problem and its own parameters. This is another feature that distinguishes ES from other evolutionary computation paradigms. For a more detailed summary of differences see Table 2. The detailed ES algorithm is shown in Figure 2.

	Evolution Strategies[7]	**Genetic Algorithms**[5]	**Evolutionary Programming**[8]
Encoding	Real numbers	Binary (typically)	Real numbers
Self-adaptive	Yes	No (typically)	No (typically)
Mutation	Gaussian. Main operator	Bit inversions. Secondary operator	Gaussian (unique operator)
Recombination	Discrete and intermediate. Secondary operator	1 point, 2 points, n points. Main operator	None
Selection	Deterministic	Probabilistic	Probabilistic

Table 2: Main differences among evolutionary computation paradigms.

```
Begin
    t=0
    Create μ random solutions for the initial population.
    Evaluate all μ individuals
    Assign a fitness value to all μ individuals
    For t=1 to MAX_GENERATIONS Do
        Produce λ offspring by recombination of the μ parents
        Mutate each child
        Evaluate all λ offspring
        Assign a fitness value to all λ individuals
        If Selection = "+" Then
            Select the best μ individuals from the μ+λ individuals
        Else
            Select the best μ individuals from the λ individuals
        End If
    End For
End
```

Figure 2: Detailed ES algorithm.

3 CONSTRAINT HANDLING IN EVOLUTIONARY ALGORITHMS

EAs are unconstrained search techniques. Thus, incorporating constraints into the fitness function of an EA is an open research area. There is a considerable amount of research regarding mechanisms that allow EAs to deal with equality and inequality constraints[9,10]. Constraint-handling approaches tend to incorporate information about infeasibility (or distance to the feasible region) into the fitness function in order to guide the search. In this chapter, we present a classification and descriptions of several constraint-handling approaches used in EAs.

3.1 Statement of the problem

We are interested in the general nonlinear programming (NLP) problem in which we want to:

$$\text{Find } \bar{x} \text{ which optimizes } f(\bar{x}) \tag{6}$$

Subject to:

$$\begin{aligned} g_i(\vec{x}) &\leq 0, \qquad i = 1,\ldots,m \\ h_j(\vec{x}) &= 0, \qquad j = 1,\ldots,p \end{aligned} \tag{7}$$

where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots x_n]^T$, m is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). If we denote with F to the feasible region and with S to the whole search space, then it should be clear that $F \subseteq S$. For an inequality constraint that satisfies $g_i(\vec{x}) = 0$, we will say that is active at $\vec{x}$; it is said to be inactive if $g_i(\vec{x}) \leq 0$. All equality constraints h_j (regardless of the value of $\vec{x}$ used) are considered active at all points of F.

3.2 Penalty function

The most common approach adopted to deal with constrained search spaces is the use of penalty functions[11]. When using a penalty function, the amount of constraint violation is used to punish or "penalize" an infeasible solution so that feasible solutions are favored by the selection process.

There are two types of penalty functions:

- **Exterior**: More commonly used in Evolutionary Algorithms. In this case, the algorithm starts with infeasible solutions and the search will be guided towards the feasible region of the search space. The penalty value will be low at the beginning of the search and it will be increased over time (i.e. iterations). The idea is to allow the search to move towards the feasible region and, once reached, stay there.
- **Interior**: Also known as barrier penalties. In this case, the algorithm starts with a feasible solution (which, for some problems, is not easy or computationally efficient to get[12]) and moves inside the feasible region. In this case, the penalty factor is low in zones far from the boundaries of the feasible region and it will be high in zones close to the boundaries. This allows the search to move inside the feasible region trying to locate the global optimum.

The general formula of a penalty function is the following:

$$f(\vec{x}) = f(\vec{x}) \pm \left[\sum_{i=1}^{m} r_i \cdot G_i + \sum_{j=1}^{p} c_j \cdot L_j \right] \tag{8}$$

where $f(\vec{x})$ is the expanded objective function to be optimized, G_i and L_j are functions of the constraints of the problem $g(\vec{x})$ and $h(\vec{x})$, respectively, and r_i y c_j are positive constants called "penalty factors" which determine the severity of the penalty. The most common form of G_i and L_j is:

$$G_i = \max[0, g_i(\vec{x})]^b \tag{9}$$

$$L_j = \left| h_j(\vec{x}) \right|^g \tag{10}$$

where b and g are normally 1 or 2.

The main drawback of penalty functions is that they require a careful fine tuning of the penalty factors that accurately estimates the degree of penalization to be applied so that we can approach efficiently the feasible region[9,12].

Several approaches have been proposed to avoid this dependency of the values of the penalty factors. The most known are the following:

- **Death Penalty**: In this case, infeasible solutions either get a zero fitness regardless of their amount of constraint violation or are just discarded. This idea was proposed by Schwefel[4].
- **Static Penalties**: In this case, the penalty factors remain without change during all the evolutionary process[13].
- **Dynamic Penalties**: The idea of a dynamic penalty approach is to use time (i.e. the current generation number) to influence the computation of the penalty factor of an individual[14].
- **Annealing Penalties**: This is a particular case of dynamic penalty functions based on the idea of simulated annealing[15],
- **Adaptive Penalties**: The aim of adaptive penalties is to use information of the evolutionary process itself (instead of a pre-defined variation function as in the case of dynamic penalties) to update the value of the penalty factors[16].
- **Co-evolutionary penalties**: The main idea is to evolve penalty factors in one subpopulation and in the other one the solutions of the original problem[17].
- **Segregated genetic algorithm**: Proposes a balance between heavy and moderated penalty factors[18].
- **Fuzzy Penalties**: In this approach, a set of fuzzy rules is used to update the value of the penalty factors[19].

3.3 Special representations and operators

When the traditional representation of solutions (i.e. binary) is not suitable, some researchers have opted to propose alternative representations and associated operators suitable for the proposed representation. In most cases, special encodings are adopted to generate feasible solutions and ad-hoc operators are used to preserve their feasibility during all the evolutionary process. The main application of this approach is in problems in which it is extremely difficult to locate at least a single feasible solution, or in problems in which traditional encodings do not perform well[20,21,22]

3.4 Repair algorithms

Repair in the context of constraint handling means to make feasible an infeasible solution. This idea has been widely used in combinatorial optimization, more than in numerical optimization. Some of the open questions related to repair algorithms are, for example, if the repaired solution must be inserted in the population or if it should be used for evaluating fitness[23]. Another question is how to design efficient and effective (and even generalizable) repair algorithms. One application of repair algorithms for numerical optimization was proposed by Michalewicz and his GENOCOP III[24]. The aim was to incorporate the original GENOCOP system[20] (which handles only linear

constraints) and also use two different populations where results in one population influence evaluations of individuals in the other population. Individuals in the first population are search points which satisfy linear constraints of the problem. These solutions are kept as feasible by using special operators. Solutions in the second population are feasible reference points. Then, solutions from the first population are repaired in order to be similar to those of the second population. The main drawback of the approach is that the effort to repair an infeasible solution can become more costly than the entire algorithm. Also, repair methods are not usually easy to generalize. For combinatorial optimization, Liepins et al.[23] have shown, through an empirical study on a diverse set of constrained combinatorial optimization problems, that a repair algorithm can provide better results than other approaches in both speed and performance. Other area of application of repair algorithms is robotics. Xiao et al.[25] used a repair algorithm to transform an infeasible path of a robot trying to move between two points in the presence of obstacles, so that the path would become feasible. The difficult part of this work was the design of the repair operators.

3.5 Separation of Constraints and Objectives

Unlike penalty functions which combine the value of the objective function and the constraints of a problem to assign fitness, these approaches handle constraints and objectives separately. The most representative are: the use of coevolution by Paredis[26], the approach based on the superiority of feasible points by Deb[27] and the use of multiobjective optimization concepts by Coello & Mezura[28].

3.6 Hybrid Methods

Within this category, we consider methods that are coupled with another technique (another heuristic or a mathematical programming approach) to deal with constrained spaces: Bilchev & Parmee[29] proposed to use Ant System concepts to solve constrained problems. The use of Lagrangian multipliers to solve constrained problems has been proposed by some authors like Kim & Myung[30]. Other ideas adopted to deal with constrained search spaces are the use of Cultural Algorithms (Chung and Reynolds[31]) and Artificial Immune System (Hajela & Lee[32]).

4 EVOLUTION STRATEGIES TO SOLVE CONSTRAINED PROBLEMS

After a brief description of the different approaches proposed to incorporate the constraints of a problem into the fitness function of an evolutionary algorithm, we now focus on those approaches whose search engine is an evolution strategy.

Oyman et al.[33] used Deb's approach[27] (listed in Section 3.5), but with an evolution strategy as a search engine. They compared their approach against a death penalty. Their approach outperformed the death penalty scheme. However, they tested it using just three problems (two of them are from the well-known benchmark from Michalewicz & Schoenauer[34]). Moreover, they only tested evolution strategies without a self-adaptation mechanism (hence, they do not use correlated mutation as well). The authors used a (1+1)-ES, with no recombination operator (the ES adopted is single-membered), and using different parameters for each test problem with a number of evaluations of the objective function which oscillated between 331 and 330,491, depending of the test problem solved. One interesting conclusion found by Oyman et al. is the importance of defining a correct value for the stepsize used in the mutation operator. In our proposed approach detailed in the next Section, we also emphasize the

importance of the definition of the initial stepsize value in a self-adaptive ES.

Hamida & Schoenauer[35] proposed the Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA). ASCHEA is based on three components:

- **An adaptive penalty function**: The expression used is:

$$\text{fitness}(\vec{x}) = \begin{cases} f(\vec{x}) & \text{if feasible} \\ f(\vec{x}) - \text{penal}(\vec{x}) & \text{otherwise} \end{cases} \tag{11}$$

where

$$\text{penal}(\vec{x}) = a \sum_{j=1}^{q} g_j^+(\vec{x}) + a \sum_{j=q+1}^{m} \left| h_j(\vec{x}) \right| \tag{12}$$

where $g_j^+(\vec{x})$ is the positive part of $g_j(\vec{x})$ and α is the penalty coefficient for all the constraints of the problem. The penalty factor is adapted according to a desired ratio of feasible solutions t_{target} and the current ratio in the generation t, t_t in the following way:

$$\begin{array}{ll} \text{if } (t_t > t_{target}) & a(t+1) = a(t)/\ fact \\ \text{otherwise} & a(t+1) = a(t) *\ fact \end{array} \tag{13}$$

where *fact* >1 and t_{target} are used-defined parameters and

$$a(0) = \left| \frac{\sum_{i=1}^{n} f_i(\vec{x})}{\sum_{i=1}^{n} V_i(\vec{x})} \right| * 1000 \tag{14}$$

where $V_i(\vec{x})$ is the sum of constraint violation of individual *i*.

- **Constraint-driven recombination (crossover)**: Combine an infeasible solution with a feasible one and apply it when there is a low number of feasible solutions with respect to t_{target}. If $t_t > t_{target}$, the recombination is performed in the traditional way.
- **Segregational Selection based on feasibility**: The aim is to choose a defined ratio t_{select} of feasible solutions based on their fitness to be part of the population for the next generation. The remaining individuals are selected in the traditional way (proportional selection) based on their penalized fitness. t_{select} is another user-defined parameter.

In ASCHEA's new version[35], the authors propose to use a penalty factor for each constraint of the problem. Each factor is adapted independently:

$$\text{penal}(\vec{x}) = \sum_{j=1}^{q} a_j g_j^+(\vec{x}) + \sum_{j=q+1}^{m} a_j \left| h_j(\vec{x}) \right| \tag{15}$$

and the adaptation process is now as follows:

$$\begin{array}{ll} \text{if } \left(t_{\mathrm{t}}(j) > t_{target}\right) & a_j(\mathrm{t}+1) = a_j(\mathrm{t}) / \ fact \\ \text{otherwise} & a_j(\mathrm{t}+1) = a_j(\mathrm{t}) * \ fact \end{array} \quad (16)$$

also, the authors used a niching mechanism to improve the performance of the algorithm in multimodal functions. Finally, they added both, a dynamic and an adaptive scheme to decrease the tolerance value used to handle equality constraints. All these three new mechanisms also add more user-defined parameters, which makes more difficult to tune them to solve an specific problem. The approach uses a (100+300)-ES and requires 1,500,000 fitness function evaluations to provide good results in 11 functions of the aforementioned benchmark for constrained evolutionary optimization[34]. The authors used standard arithmetical recombination (similar to generalized intermediate recombination) using a crossover rate (which is not usually adopted when using an ES). The main drawback of the approach is the definition by the user of several extra parameters required by the technique.

One of the most competitive approaches found in the literature is the Stochastic Ranking (SR) by Runarsson & Yao[36]. The aim of this approach is to balance the influence of the objective function and the penalty function when assigning fitness to a solution. SR does not require the definition of a penalty factor. Instead, the selection process is based on a ranking process and a user-defined parameter called P_f that sets the probability of using only the objective function to compare two solutions when sorting them. Then, when the solutions are sorted using a bubble-sort like algorithm, sometimes, depending of the P_f value, the comparison between two adjacent solutions will be performed using only the objective function. The remaining comparisons will be performed using only the penalty function that consists in this case, of the sum of constraint violation. The suggested range for the P_f value is $0.4 < P_f < 0.5$. The results obtained using all the functions of the well-known benchmark from Michalewicz & Schoenauer[34] (plus one test function) are the best reported to date in the literature. Runarsson & Yao used a (30,200)-ES with 350,000 evaluations of the fitness function. The authors used panmictic intermediate recombination for the strategy parameters and they did not use any recombination operator for the decision variables nor correlated mutation. One drawback of the approach is that the user needs to define the parameter P_f. The sorting algorithm adopted by this approach (assuming minimization) is shown in Figure 3.

One of the most recent approaches based on an evolution strategy used to solve constrained problems is the Pareto Archived and dominance Selection with Shrinkable Search Space (PASSSS or PAS^4)[37]. PAS^4 is based on a multiobjective optimization technique called Pareto Archived Evolution Strategy (PAES) originally proposed by Knowles & Corne[38], whose main feature is the use of an external population, stored in an adaptive grid, that keeps nondominated solutions found along the evolutionary process. PAS^4 also uses an adaptive grid, but it reduces its size over time in order to focus the search on the most promising regions of the search space. Besides, PAS^4 uses a shrinking mechanism which performs four tasks:

1. To select the best 15% individuals found in the adaptive grid (these solutions are feasible or infeasible solutions whose values of violation of each constraint are the lowest).
2. To find the extreme values of each decision variable from the selected solutions.

3. To shrink the feasible space around the potential solutions enclosed in the hypervolume defined by the bounds found for each decision variable. This trimming process will be performed using a percentage of shrinking β defined by the user.
4. These new bounds are used to re-initialize the values for the stepsizes σ for the mutation operator.

```
Begin
   For i=1 to N Do
      For j=1 to N-1 Do
         U=random(0,1)
         If ( (f (Ij)= f (Ij+1) =0)  or (u<Pf) ) Then
            If ( f(Ij) > f(Ij+1) ) Then
               swap(Ij, Ij+1)
            End If
          Else
            If ( f (Ij) > f(Ij+1) ) Then
               swap(Ij, Ij+1)
            End If
         End If
      End For
      If (not swap performed) Then
         Break
      End If
   End For
End
```

Figure 3: Stochastic Ranking sort algorithm. I is an individual of the population. f (I_j) is the sum of constraint violation of individual I_j. $f(I_j)$ is the objective function value of individual I_j.

The authors used a (150+200)-ES with 350,000 evaluations of the objective function to solve the extended benchmark of Michalewicz & Schoenauer[34]. They applied discrete crossover on the decision variables and intermediate crossover on the strategy parameters, and did not use correlated mutation. The main drawback of the approach is that it requires the definition by the user of several parameters. Furthermore, the implementation of PAS[4] is far from being simple.

5 A MULTIMEMBERED EVOLUTION STRATEGY TO SOLVE CONSTRAINED PROBLEMS

In this section we present a novel approach which exploits the features of an evolution strategy in order to solve global optimization problems with constraints. Motivated by the fact that the most recent and competitive approaches to solve constrained optimization problems are based on an Evolution Strategy (e.g. Stochastic Ranking[36] and ASCHEA[35]) we hypothesized the following:

1. The self-adaptation mechanism of an ES helps to sample the search space well enough as to reach the feasible region reasonably fast.
2. The simple addition of feasibility rules to an ES should be enough to guide the search in such a way that the global optimum can be approached efficiently.

Thus, based on these ideas, we implemented a generic ES-based approach to solve constrained optimization problems. Then, we performed an empirical study in which we varied the type of selection ("+" or ",") and the type of mutation (noncorrelated or correlated)[39]. We also implemented a variation of a (μ+1)-ES with the *"1/5 successful rule"* to adapt on-line the sigma value. Constraints were handled using rules based on feasibility similar to those used by Deb[27] and shown below:

1. A feasible solution is always preferred over an infeasible one.
2. Between 2 feasible solutions, the one with the best value of the objective function is preferred.
3. Between 2 infeasible solutions, the one with the lowest amount of constraint violation is preferred.

From our ES's comparative study, the best results were provided by the variation of a (μ+1)-ES[39] in which one child created from μ mutations of the current solution competes against it and the best one is selected as the new current solution. The use of correlated mutation showed no positive impact on the performance of the ES. However, the approach presented premature convergence in some test functions. Therefore, a (1+λ)-ES was proposed[40], which improved the robustness and quality of the previous ES proposed. In this version, a diversity mechanism was added. Its function was to maintain infeasible solutions with a good value of the objective function.

Both aforementioned approaches provided good results. However, their exploratory power to sample large search spaces was limited because they are single-membered ES. Therefore, our more recent approach is based on a (μ+λ)-ES which is called a *Simple Multimembered Evolution Strategy (SMES)*. The detailed features of our approach are described next.

5.1 Diversity mechanism

With an idea similar to that used in the (1+λ)-ES version, we allow infeasible solutions to remain in the population. However, unlike this previous approach, where the best parent based only on the objective function (regardless of its feasibility) can survive, in this new approach we allow the infeasible individual with the best value of the objective function and with the lowest amount of constraint violation to survive for the next generation. This solution (called by us the best infeasible solution) can be chosen either from the parents or the offspring population, with 50% probability. This process of allowing this solution to survive for the next generation happens 3 times every 100 during the same generation. However, it is a desired behavior because a few copies of this solution will allow its recombination with several solutions in the population, especially with feasible ones. Recombining feasible solutions with infeasible solutions in promising areas (based on the good value of the objective function) and close to the boundary of the feasible region will allow the ES to reach global optimum solutions located precisely on the boundary of the feasible region of the search space (which are normally the most difficult solutions to reach). Following the idea of allowing just a few infeasible solutions (one in the case of the (1+λ)-ES approach), we allow the best infeasible solution to be copied into the population for the next generation just 3 times for every 100 attempts. This works in the following way: When the deterministic replacement is used to form the population for the next generation in an ES, the best individuals from the union of parents and offspring are selected based on the comparison mechanism previously indicated (in a deterministic way). The process will pick feasible solutions with a better value of the objective function first, followed by infeasible solutions with a lower value of constraint

violation. However, 3 times from every 100 picks, the best infeasible solution is copied in the population for the next generation. The pseudocode is listed in Figure 4.

Based on the empirical evidence observed in the previous version of the approach[40] where we used a population of 3 offspring, we decided to use a small number of copies of the best infeasible solutions for the next generation of our approach. For values larger than 3, the quality and robustness of our approach tend to decrease. It is worth remarking that in the case where no infeasible solutions are found in the population, a random solution is copied to the population for the next generation. Therefore, it is possible, at any given generation, to have an entirely feasible parents population. However, the mechanism will allow, when the offspring are generated, to have infeasible individuals again.

```
Function population for next generation()
Begin
   For i=1 to μ Do
      If  flip(0.97)  Then
          Select the best individual based on the comparison mechanism
          from the union of the parents and offspring population,
          add it to the population for the next generation and delete
          it from this union.
      Else
          If  flip(0.5)  Then
              Select the best infeasible individual from the parents
              population and add it to the population for the next
              generation.
          Else
              Select the best infeasible individual from the offspring
              population and add it to the population for the next
              generation.
          End If
      End If
   End For
End
```

Figure 4: Pseudocode of the generation of the population for the next generation with the diversity mechanism incorporated. flip(*P*) is a function that returns TRUE with probability *P*.

5.2 Combined recombination

We use panmictic recombination, but with a combination of the discrete and intermediate recombination operators. Each gene in the chromosome can be processed with any of these two recombination operators with 50% probability. This operator is applied to both, strategy parameters (sigma values) and decision variables of the problem. The pseudocode is shown in Figure 5. Note that we use intermediate recombination by just computing the average between the values of the variable of each parent (as originally proposed by Schwefel[6]).

```
Function combined recombination()
Begin
   Select mate 1 from the parents population
   For i=1 to NUMBER_OF_VARIABLES Do
      Select mate 2 from the parents population
      If  flip(0.5)  Then
         If  flip(0.5)  Then
            child_i = mate _1_i
         Else
            child_i = mate _2_i
         End If
      Else
            child_i = (mate_1_i + mate_2_i) / 2
      End If
   End For
End
```

Figure 5: Pseudocode of the panmictic combined (discrete-intermediate) recombination operator used by our approach. flip(P) is a function that returns TRUE with probability P.

5.3 Reduction of the initial stepsize of the ES

The previous versions of our algorithm were based on a variation of a (μ + 1)-ES[39] and a (1 + λ)-ES[40]. These approaches do not use a population of solutions and employ the most simple scheme of an ES where only one sigma value is used for all the decision variables. We observed that when this sigma value was close to zero, the previous approaches were capable of reaching the global optimum, or at least improve the value of the final solution. Therefore, in our new approach based on a multimembered ES, we decided to favor finer movements in the search space. We experimented with just a percentage of the quantity obtained by the formula proposed by Schwefel[7]. We initialize the sigma values (we use one for each decision variable) for each individual in the initial population with only a 40% of the value obtained by the following formula (where n is the number of decision variables):

$$s_i(0) = 0.4 \times \left(\frac{\Delta x_i}{\sqrt{n}} \right) \tag{17}$$

where Δx_i is approximated with the expression (suggested by Runarsson & Yao[36]), $\Delta x_i \approx x_i^u - x_i^l$, where $x_i^u - x_i^l$ are the upper and lower bounds of the decision variable i.

Summarizing, our approach works over a simple multimembered evolution strategy: (μ+λ)-ES. The only modifications introduced are the reduction of the initial stepsize of the sigma values, the panmictic combined (discrete-intermediate) recombination and the changes to the original deterministic replacement of the ES (made by sorting the solutions with respect to the comparison mechanism based on feasibility discussed at the beginning of this section), allowing the best infeasible solution, from either the parents or the offspring population, to remain in the next generation. The details of our approach are presented in Figure 6.

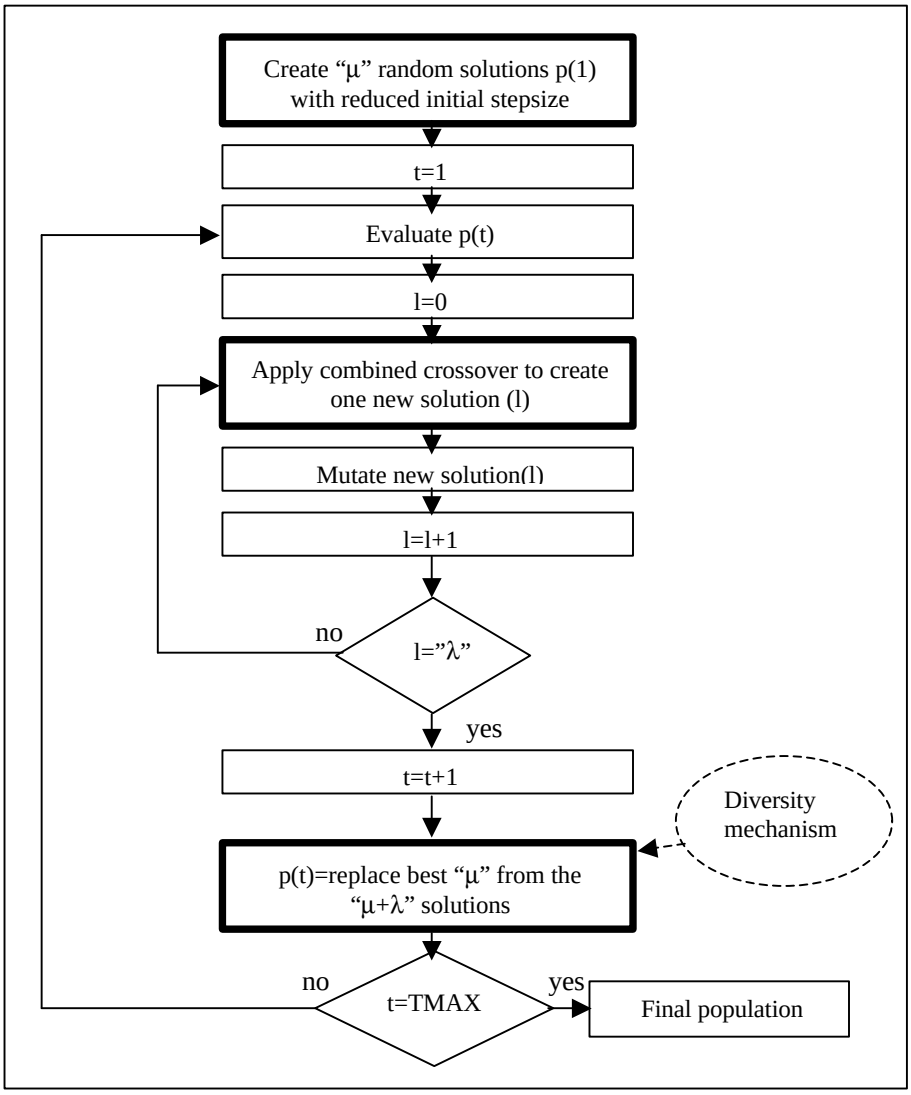

Figure 6: Algorithm of our (μ+λ)-ES (SMES). The thick boxes indicate the three modifications made to the original ES

5.4 A graphical example

A graphical example of the expected behavior of the approach can be found in Figure 7. We used a 2-dimensional test problem g08, which is a problem easy to solve by the approach; it requires about 5400 evaluations of the objective function (18 generations) to reach the global optimum, but it helps to visualize how our approach works. The definition of this problem is the following:

$$\text{Maximize: } f(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \tag{18}$$

$$\text{Subject to: } g_1(\vec{x}) = x_1^2 - x_2 + 1 \le 0$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \le 0$$

where $0 \le x_1 \le 10$ and $0 \le x_2 \le 10$. The global optimum is located at x* = (1.2279713,

4.2453733) where f(x*) = 0.095825.

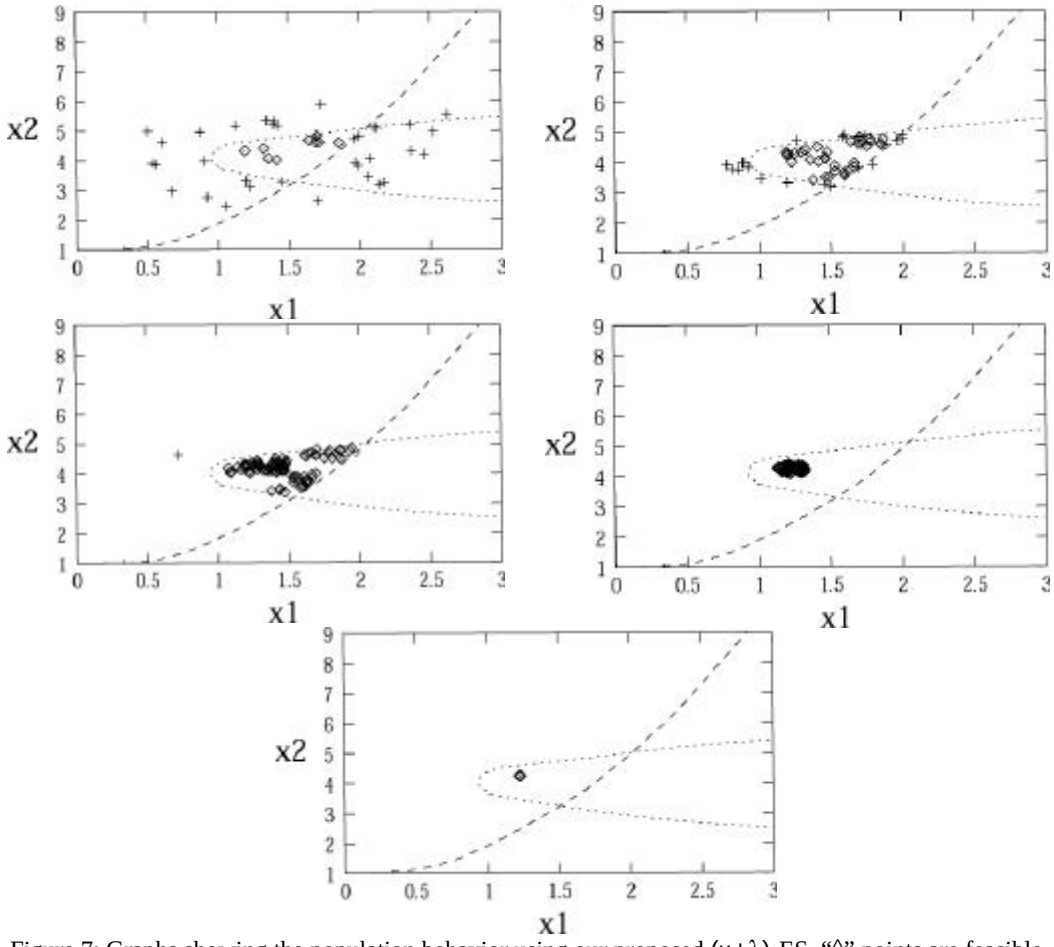

Figure 7: Graphs showing the population behavior using our proposed (μ+λ)-ES. "◊" points are feasible solutions, "+" points are infeasible ones. The dashed line represents constraint $g_1(x)$ of the problem and the dotted line represents constraint $g_2(x)$.

As it can be observed, in generation 1 there are a few feasible as well as several infeasible solutions. The behavior of the approach can be observed in generation 3, where there are more feasible solutions than those in generation 1 and there are also infeasible solutions surrounding the feasible region. In this way, helped by the combined crossover and the finer mutation movements the feasible region is sampled well-enough as to find promising areas (three areas in the example). This is shown in generation 6, where there is still an infeasible solution in the population. It is worth noticing that this infeasible solution is close to the area where the global optimum is located; this can be seen in generation 10 where the infeasible solution has disappeared but the approach has found the vicinity of the constrained global optimum. Our algorithm has converged to the constrained global optimum in generation 18.

5.5 Experimental results

To evaluate the performance of the proposed approach we used the 13 test functions commonly adopted to test constraint handling techniques[36]. Their expressions can be found elsewhere[36]. We performed 30 independent runs for each test function. The learning rates values were calculated using the formulas proposed by Schwefel[6] and discussed in Section 2.

The initial values for the stepsizes were calculated using equation 17. In the experiments, the following parameters were used:

- (100+300)-ES
- Number of generations = 800.
- Number of objective function evaluations = 240,000.

The combined recombination operator was used both for the decision variables of the problem and for the strategy parameters (sigma values). Note that we do not use correlated mutation[39]. To deal with equality constraints, a dynamic mechanism originally proposed in ASCHEA[35] and used in some of our previous work[40] is adopted. The tolerance value ε is decreased with respect to the current generation using the following expression:

$$e_j(t+1) = \frac{e_j(t)}{1.00195} \tag{19}$$

Statistical Results of the Simple Multimembered Evolution Strategy (SMES)						
Problem	**Optimal**	**Best**	**Mean**	**Median**	**Worst**	**St. Dev.**
g01	**-15.000**	**-15.000**	**-15.000**	**-15.000**	**-15.000**	0
g02	0.803619	0.803601	0.785238	0.792549	0.751322	1.67E-2
g03	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	2:09E-4
g04	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**	0
g05	5126.498	5126.599	5174.492	5160.198	5304.167	50.06E+0
g06	**-6961.814**	**-6961.814**	-6961.284	**-6961.814**	-6952.482	1.85E+0
g07	24.306	24.327	24.475	24.426	24.843	1.32E-1
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0
g09	680.63	680.632	680.643	680.642	680.719	1.55E-2
g10	7049.25	7051.903	7253.047	7253.603	7638.366	136.02E+0
g11	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**	1.52E-4
g12	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0
g13	0.053950	0.053986	0.166385	0.061873	0.468294	1.77E-1

Table 3 : Statistical results obtained by our SMES for the 13 test functions over 30 independent runs. A result in **boldface** indicates that the global optimum (or best known solution) was reached.

The initial ε_0 was set to 0.001. Note that the use of the value 1.00195 in equation 19 causes the allowable tolerance for the equality constraints to go from 0.001 (initial value) to 0.0004 (final value) given the number of iterations adopted by our approach (if more iterations are performed, this value will tend to zero). For problem g13, ε_0 was set to a much larger value (3.0), because in this case it is very difficult to generate feasible solutions during the initial generations of our approach. Thus, by using a large tolerance value, more individuals will be able to satisfy the equality constraints and will serve as reference solutions that the algorithm will improve over time. Given that this larger value is adopted, we also changed the constant decreasing value. So, instead of using 1.00195, we adopt, in this case, a value of 1.0145. Such a value causes the allowable equality constraint violation to go from 3.0 (initial value) to 0.00003 (final value) given the number of iterations adopted by our approach. Note that the final allowable tolerance is smaller in this case, despite the initial larger value. As a matter of fact, we recommend to use this second setup for the tolerance of the equality constraints in problems in which no feasible solutions can be found by our algorithm when using a small initial ε_0. Additionally, for problems g03 and g13 the initial stepsize required a more dramatic decrease. They were defined as 0.01 (just a 5% instead of the 40% used for the other test functions) for g03 and 0.05 (2.5%) for g13. Those two test functions

seem to provide better results with very smooth movements. It is important to note that those two problems share the following features: moderately high dimensionality (five or more decision variables), nonlinear objective function, one or more equality constraints, and moderate size of the search space (based on the range of the decision variables). Those common features suggest that for these types of problems, finer movements provide a better sampling of the search space using an evolution strategy. The statistical results of our SMES are summarized in Table 3.

As described in Table 3, our approach was able to find the global optimum in seven test functions (g01, g03, g04, g06, g08, g11 and g12) and it found solutions very close to the global optimum in the remaining six (g02, g05, g07, g09, g10, g13). Furthermore, we compared these results with respect to three state-of-the-art techniques previously discussed (Stochastic Ranking (SR)[36], ASCHEA[35] and PAS[4 37]). The best result found by each approach is compared in Table 4. Analogously, in Tables 5 and 6 the mean and worst values are compared.

5.6 Discussion of results

With respect to SR, our approach was able to find a "better" best result in functions g02 and g10. In addition, it found a "similar" best solution in seven problems (g01, g03, g04, g06, g08, g11 and g12). Slightly "better" best results were found by SR in the remaining functions (g05, g07, g09 and g13). Our approach found "better" mean and worst results in four test functions (g02, g06, g09 and g10). It also provided "similar" mean and worst results in six functions (g01, g03, g04, g08, g11 and g12). Finally, SR found again "better" mean and worst results in function g05, g07 and g13.

Compared against ASCHEA, the SMES found "better" best solutions in three problems (g02, g07 and g10) and it found "similar" best results in six functions (g01, g03, g04, g06, g08, g11). ASCHEA found slightly "better" best results in function g05 and g09. Additionally, our approach found "better" mean results in four problems (g01, g02, g03 and g07) and it found "similar" mean results in three functions (g04, g08 and g11). ASCHEA surpassed our mean results in four functions (g05, g06, g09 and g10). We did not compare the worst results because they were not available for ASCHEA. Also, we did not perform comparisons with respect to ASCHEA using functions g12 and g13 for the same reason.

Comparison of the best solution found					
Problem	**Optimal**	**SR[36]**	**ASCHEA[35]**	**PAS[4 37]**	**SMES**
g01	**-15.000**	**-15.000**	**-15.0**	-14.9998	**-15.000**
g02	0.803619	0.803515	0.785	0.80346	**0.803601**
g03	**1.000**	**1.000**	**1.0**	**1.000**	**1.000**
g04	**-30665.539**	**30665.539**	-30665.5	-30665.530	**-30665.539**
g05	5126.498	**5126.497**	5126.5	5126.52	5126.599
g06	**-6961.814**	**-6961.814**	-6961.81	-6961.810	**-6961.814**
g07	24.306	24.307	24.3323	24.33060	24.327
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.63	**680.630**	**680.630**	**680.630**	680.632
g10	7049.25	7054.316	7061.13	7059.84	**7051.903**
g11	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**
g12	**1.000**	**1.000**	NA	**1.000**	**1.000**
g13	0.053950	0.053957	NA	**0.053950**	0.053986

Table 4 : Comparison of the best solutions found by the Stochastic Ranking (SR), ASCHEA, PAS[4] and our SMES. NA = Not available. A result in **boldface** indicates either that the global optimum (or best known solution) was reached or a better solution was found by the corresponding approach.

Comparison of the mean solution found					
Problem	**Optimal**	**SR[36]**	**ASCHEA[35]**	**PAS[4] [37]**	**SMES**
g01	**-15.000**	**-15.000**	-14.84	-14.88731	**-15.000**
g02	0.803619	0.781975	0.59	**0.79901**	0.785238
g03	**1.000**	**1.000**	0.99989	**1.000**	**1.000**
g04	**-30665.539**	**30665.539**	-30665.5	-30665.530	**-30665.539**
g05	5126.498	**5128.881**	5141.65	5180.15545	5174.492
g06	**-6961.814**	-6875.940	**-6961.81**	-6961.810	-6961.284
g07	24.306	**24.374**	24.66	24.57961	24.475
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	**0.095825**
g09	680.63	680.656	680.641	**680.63243**	680.643
g10	7049.25	7559.192	**7193.11**	7366.9965	7253.047
g11	**0.75**	**0.75**	**0.75**	**0.75**	**0.75**
g12	**1.000**	**1.000**	NA	**1.000**	**1.000**
g13	0.053950	**0.057006**	NA	0.22022	0.166385

Table 5 : Comparison of the mean solutions found by the Stochastic Ranking (SR), ASCHEA, PAS[4] and our SMES. NA = Not available. A result in **boldface** indicates either that the global optimum (or best known solution) was reached or a better solution was found by the corresponding approach.

Comparison of the worst solution found					
Problem	**Optimal**	**SR[36]**	**ASCHEA[35]**	**PAS[4] [37]**	**SMES**
g01	**-15.000**	**-15.000**	NA	-12.4477	**-15.000**
g02	0.803619	0.726288	NA	**-0.78548**	0.751322
g03	**1.000**	**1.000**	NA	**-1.000**	**1.000**
g04	**-30665.539**	**30665.539**	NA	-30665.530	**-30665.539**
g05	5126.498	**5142.472**	NA	5558.7	5304.167
g06	**-6961.814**	-6350.262	NA	**-6961.81**	-6952.482
g07	24.306	**24.642**	NA	25.3666	24.843
g08	**0.095825**	**0.095825**	NA	**-0.095825**	**0.095825**
g09	680.63	680.763	NA	**680.6360**	680.719
g10	7049.25	8835.655	NA	7803.11	**7638.366**
g11	**0.75**	**0.75**	NA	**0.75**	**0.75**
g12	**1.000**	**1.000**	NA	**1.000**	**1.000**
g13	0.053950	**0.216915**	NA	0.44512	0.468294

Table 6 : Comparison of the worst solutions found by the Stochastic Ranking (SR), ASCHEA, PAS[4] and our SMES. NA = Not available. A result in **boldface** indicates either that the global optimum (or best known solution) was reached or a better solution was found by the corresponding approach.

Our SMES provided "better" best results than PAS[4] in six functions (g01, g02, g04, g06, g07 and g10) and it found "similar" best results in four problems (g03, g08, g11 and g12). In the remaining three, PAS[4] surpassed our best result (g05, g09 and g13). In addition, the SMES found "better" mean results in six functions (g01, g04, g05, g07, g10 and g13) and it found "similar" mean results for four problems (g03, g08, g11 and g12). PAS[4] provided better mean results in the remaining three (g02, g06 and g09). Finally, the SMES found a "better" worst result in five functions (g01, g04, g05, g07 and g10), it found "similar" worst solutions in other four (g03, g08, g11 and g12). PAS[4] found better worst solutions in the remaining four problems (g02, g06, g09 and g13).

As we can see, our approach showed a very competitive performance with respect to these three state-of-the-art approaches. SMES can deal with moderately constrained problems (g04), highly constrained problems, problems with low (g06, g08), moderated

(g09) and high (g01, g02, g03, g07) dimensionality, with different types of combined constraints (linear, nonlinear, equality and inequality) and with very large (g02), very small (g05 and g13) or even disjoint (g12) feasible regions. Also, the algorithm is able to deal with large search spaces (based on the intervals of the decision variables) and with a very small feasible region (g10). Furthermore, the approach can find the global optimum in problems where such optimum lies on the boundaries of the feasible region (g01, g02, g04, g06, g07, g09). This behavior suggests that the mechanism of maintaining the best infeasible solution helps the search to sample the boundaries between the feasible and infeasible regions.

The computational cost (measured in terms of the number of fitness function evaluations (FFE) performed by any approach) is lower for the SMES than for others with respect to which it was compared. This is an additional (and important) advantage, mainly if we wish to use this approach for solving real-world problems. The SMES performed 240,000 FFE, the Stochastic Ranking and PAS[4] performed 350,000 FFE, and ASCHEA required 1,500,000 FFE.

It is also worth mentioning that the SMES had some problems to find consistently good results in presence of more than one nonlinear equality constraints (g05 and g13). This issue deserves a more in-depth analysis in the future.

5.7 Finding the strength of the SMES

Once we corroborated the effectiveness of our approach, it became particularly relevant to identify the key component (or combination of them) that was mainly responsible for the good performance of our algorithm. For that sake, we designed two experiments.

1. **Cross-validation of our ES' mechanisms**: We tested our SMES using each of its mechanisms (diversity mechanism, combined recombination and stepsize reduction) separately and combining them in pairs, in order to recognize which of them was mandatory. It is important to note that removing the diversity mechanism implies disallowing the best infeasible solution to remain in the population for the next generation of the algorithm. The comparison mechanism based on feasibility remains in all cases in order to guide the search to the feasible region of the search space.
2. **ES against GA**: Our second experiment consisted on implementing a real-coded GA with the same combined recombination and the same diversity mechanism used in our SMES. Here, we wanted to see if the use of a GA instead of an ES would make any significant difference in terms of performance.

The parameters used in these experiments are exactly the same used in the experiments described in Section 5.5. Thus, the number of evaluations of the objective function is also the same (240,000). We performed 30 independent runs for each different version of the algorithm (with different combination of mechanisms) and also for the version using the GA.

In Table 7 we present the version which provided the best results (from the cross validation experiments). Also, we present the results provided by the approach but using a GA instead of a ES. For the sake of the GA experiment, we tested different mutation operators for real-coded GAs and non-uniform mutation provided the best results. Furthermore, we intended that the GA used the same features of the ES (except for the self-adaptive mutation which we hypothesized was the main strength of our ES-based approach). Finally, the same dynamic mechanism to handle the tolerance for equality constraints was employed. The parameters used by our real-coded GA were the following:

- Population size: 200
- Maximum number of generations: 1200
- Crossover rate: 0:8
- Mutation rate: 0:6
- Number of objective function evaluations: 240,000 (the same performed by our SMES).

Statistical Results for the cross validation experiments and the GA test

		Best		Mean		Worst	
P	**Optimal**	**Rec. & Stepsize**	**GA**	**Rec. & Stepsize**	**GA**	**Rec. & Stepsize**	**GA**
g01	**-15.000**	**-15.000**	-14.440	**-15.000**	-14.236	**-15.000**	-14.015
g02	0.803619	0.803592	0.796231	0.798786	0.788588	0.785255	0.779140
g03	**1.000**	**1.000**	0.990	**1.000**	0.976	0.999	0.956
g04	**-30665.539**	-30665.42	-30626.05	-30661.10	-30590.45	-30647.48	-30567.10
g05	5126.498	5216.998	-	5158.739	-	5201.935	-
g06	**-6961.814**	**-6961.814**	-6952.472	**-6961.814**	-6872.204	**-6961.814**	-6784.255
g07	24.306	24.343	31.097	24.474	34.980	24.789	38.686
g08	**0.095825**	**0.095825**	**0.095825**	**0.095825**	0.095799	**0.095825**	0.095723
g09	680.63	680.631	685.994	680.637	692.064	680.664	698.297
g10	7049.25	7062.754	9079.770	7193.887	10003.225	7368.333	11003.533
g11	**0.75**	**0.75**	**0.75**	0.752	**0.75**	0.767	0.752
g12	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999
g13	0.053950	0.058037	0.134057	0.247404	-	0.466266	-

Table 7 : Statistical results obtained by the best combination of mechanisms of the SMES and the GA experiment for the 13 test functions over 30 independent runs. "-" means no feasible solutions were found. A result in **boldface** indicates that the global optimum (or best known solution) was reached

It is worth mentioning that the version with only the combined recombination (without stepsize reduction and also without diversity mechanism) provided the best results (based on quality and robustness) among the versions with only one active mechanism. However, this version with only the combined recombination was clearly surpassed by the version with the combined recombination and also the stepsize reduction (which is shown in Table 7). These results suggest that the combined recombination is the dominant mechanism, which is assisted by the fine mutation movements provided by the reduction of the initial stepsize.

It is clear that the results provided by this version with only the combined recombination and the stepsize reduction are indeed very competitive compared with respect to the complete version of the SMES (with the three mechanisms active). The main question that arose at this point was: what is the role of the diversity mechanism in the success of our approach? In order to answer this question, we compared the results of the version with combined recombination and stepsize reduction against the version with the three mechanisms (from Tables 7 and 3 respectively) . From this comparison we observed that our approach provides results of a better quality when using the diversity mechanism. However, the price paid for this higher quality of results is a slight decrease in robustness. Also, the overall results (providing competitive results in all 13 test functions) are better when the diversity mechanism is incorporated into our SMES. It is also worth reminding that the goal of the diversity mechanism is to allow the search to generate solutions in the boundaries of the feasible region (which is something critical when dealing with constraints that are active in the global optimum). Hence, the use of such diversity mechanism seems a logical choice for dealing with active constraints.

Finally, for the case of the version of the approach using a GA both the quality and robustness of the results provided by the GA are significantly poorer than those obtained with the evolution strategy in all the test functions adopted. The exceptions are g08, g11 and g12, in which the GA was able to find competitive results. These results highlight the strong influence (positive in this case) of using a more adequate search engine, in our case an ES over a GA. Therefore, the results seem to confirm our initial hypothesis about the usefulness of an ES to sample constrained search spaces in a more appropriate way.

6 CONCLUSIONS

In this chapter, we presented a review of constraint handling techniques which are based on an evolution strategy as a search engine to solve global optimization problems in the presence of constraints. After a brief introduction to evolution strategies, we provided a review of constraint handling techniques used in evolutionary algorithms. Afterwards, we focused on the techniques proposed to deal with constrained search spaces which are based on an evolution strategy. Besides, we proposed a novel approach which exploits the features of an evolution strategy to solve constrained optimization problems, we called it a Simple Multimembered Evolution Strategy (SMES). The approach was based on three modifications to the original evolution strategy algorithm: (1) a diversity mechanism which allows infeasible solutions close to the feasible region of the search space and with a good value of the objective function to remain in the population for the next generation. This infeasible solution is called the "best infeasible solution". The aim is to have a few copies of the best infeasible solution (either from the parent or the offspring population) in the population at each stage of the evolutionary process. (2) A combined panmictic (discrete-intermediate) recombination operator applied to the decision variables of the problem an the strategy parameters as well. The goal is to improve the exploitation feature of the operator and to allow infeasible solutions to combine with feasible ones in order to sample the boundaries of the feasible region. And (3) a reduction of the initial stepsize of the mutation operator. The objective is to favor finer movements in the search space. The combination of these three mechanisms provided competitive results with respect to three state-of-the-art approaches also based on an evolution strategy. Furthermore, we performed experiments in order to know which mechanism (or combination of them) was the main responsible of the good performance of the SMES. This analysis suggested that the dominant mechanism was the combined recombination, which is assisted by the finer movements of the mutation operator (due to the reduction of the initial stepsize) to provide competitive results. It was also found in this study that the diversity mechanism helps the SMES to provide results with a better quality but decreasing slightly its robustness. Finally, we empirically showed that the use of a genetic algorithm with the same mechanisms used in the evolution strategy decreases considerably the quality and robustness of the approach. This result suggests that the use of an evolution strategy is more suitable to solve this set of constrained problems. As a final conclusion we can state that the choice of the search engine and the genetic operators used to solve constrained optimization problems seems to be more important than the constraint handling mechanism.

7 FUTURE WORK

As future paths of research we want to use other genetic operators commonly adopted in evolution strategies like the derandomized self-adaptation proposed by Hansen and Ostermeier[41]. Furthermore, we want to experiment with other recombination operators like panmictic generalized intermediate recombination. The aim will be to reduce the number of evaluations required to approximate the global optimum.

8 REFERENCES

[1] Ingo Rechenberg., "Cybernetic solution path of an experimental problem", Royal Aircraft Establishment, Library Translation No. 1122, Farnborough, Hants, UK, August (1965).

[2] H.P. Schwefel, "Projekt MHD-Staustrahlrohr: Experimentelle Optimierung einer zweiphasendüse", Teil I. Technical Report Technischer Bericht 11.034/68, 35, AEG Forschungsinstitut, Berlin, (1968).

[3] Ingo Rechenberg, "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution", *Frommann-Holzboog*, Stuttgart, (1973).

[4] Thomas Bäck, Frank Hoffmeister, and Hans-Paul Schwefel., "A Survey of Evolution Strategies"., In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algori thms*, pages 2-9, San Mateo, California, Morgan Kaufmann Publishers (1991)

[5] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Co., Reading, Massachusetts, (1989).

[6] Hans-Paul Schwefel, *Evolution and Optimization Seeking*, John Wiley & Sons, New York, (1995).

[7] Thomas Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, (1996).

[8] Lawrence J. Fogel, *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*, John Wiley & Sons, New York, (1999).

[9] Carlos A. Coello Coello, "Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art, *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245-1287, January (2002).

[10] Zbigniew Michalewicz and Marc Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems", *Evolutionary Computation*, 4(1):1-32, (1996).

[11] Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard, "Some Guidelines for Genetic Algorithms with Penalty Functions". In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pages 191-197, San Mateo, California, June, George Mason University, Morgan Kaufmann Publishers, (1989)

[12] Alice E. Smith and David W. Coit. "Constraint Handling Techniques: Penalty Functions", In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, (1997).

[13] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained Optimization via Genetic Algorithms", *Simulation*, 62(4):242–254, (1994).

[14] J. Joines and C. Houck. "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Gas". In David Fogel, editor,

Proceedings of the first IEEE Conference on Evolutionary Computation, pages 579–584, Orlando, Florida, IEEE Press. (1994).

[15] Zbigniew Michalewicz and Naguib F. Attia, "Evolutionary Optimization of Constrained Problems", In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, (1994).

[16] Raziyeh Farmani and Jonathan A. Wright, "Self-Adaptive Fitness Formulation for Constrained Optimization". *IEEE Transactions on Evolutionary Computation*, 7(5):445—455, October (2003).

[17] Carlos A. Coello Coello, "Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems", *Computers in Industry*, 41(2):113–127, January (2000).

[18] Rodolphe G. Le Riche, Catherine Knopf-Lenoir, and Raphael T. Haftka, "A Segregated Genetic Algorithm for Constrained Structural Optimization", In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pages 558–565, San Mateo, California, July, University of Pittsburgh, Morgan Kaufmann Publishers, (1995)

[19] Baolin Wu and Xinghuo Yu, "Fuzzy Penalty Function Approach for Constrained Function Optimization with Evolutionary Algorithms", In *Proceedings of the 8th International Conference on Neural Information Processing*, pages 299–304, Shanghai, China, November, Fudan University Press (2001).

[20] Zbigniew Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, third edition, (1996).

[21] Marc Schoenauer and Zbigniew Michalewicz, "Evolutionary Computation at the Edge of Feasibility". In H.-M. Voigt,W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Fourth Conference on Parallel Problem Solving from Nature (PPSN IV)*, pages 245–254, Heidelberg, Germany, September, Berlin, Germany, Springer-Verlag, (1996)

[22] Slawomir Koziel and Zbigniew Michalewicz, "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization", *Evolutionary Computation*, 7(1):19–44, 1999.

[23] Gunar E. Liepins and W. D. Potter, "A Genetic Algorithm Approach to Multiple-Fault Diagnosis", In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 17, pages 237–250. Van Nostrand Reinhold, New York, New York, 1991.

[24] Zbigniew Michalewicz and G. Nazhiyath, "Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints", In David B. Fogel, editor, *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pages 647–651, Piscataway, NJ, IEEE Press. (1995)

[25] Jing Xiao, Zbigniew Michalewicz, and Krzysztof Trojanowski. "Adaptive Evolutionary Planner/Navigator for Mobile Robots", *IEEE Transactions on Evolutionary Computation*, 1(1):18–28, (1997).

[26] J. Paredis. "Co-evolutionary Constraint Satisfaction", In Proceedings of the 3rd Conference on Parallel Problem Solving from Nature, pages 46–55, New York, Springer Verlag, (1994)

[27] Kalyanmoy Deb. "An Efficient Constraint Handling Method for Genetic Algorithms", *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, (2000).

[28] Carlos A. Coello Coello and Efrén Mezura-Montes. Handling Constraints in Genetic Algorithms Using Dominance-Based Tournaments. In I.C. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing in Design and Manufacture (ACDM'2002)*, volume 5, pages 273–284, University of

Exeter, Devon, UK, April, Springer-Verlag. (2002)
[29] George Bilchev and Ian C. Parmee, "Constrained and Multi-Modal Optimisation with an Ant Colony Search Model", In Ian C. Parmee and M. J. Denham, editors, *Proceedings of 2nd International Conference on Adaptive Computing in Engineering Design and Control*. University of Plymouth, UK, March (1996).
[30] J.-H. Kim and H. Myung. "Evolutionary programming techniques for constrained optimization problems", *IEEE Transactions on Evolutionary Computation*, 1:129–140, July (1997).
[31] Chan-Jin Chung and Robert G. Reynolds, "A Testbed for Solving Optimization Problems Using Cultural Algorithms", In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 225–236, Cambridge, Massachusetts, MIT Press, March, (1996)
[32] P. Hajela and J. Lee, "Constrained Genetic Search via Schema Adaptation. An Immune Network Solution", *Structural Optimization*, 12:11–15, (1996).
[33] Ahmet Irfan Oyman, Kalyanmoy Deb, and Hans-Georg Beyer, "An Alternative Constraint Handling Method for Evolution Strategies", In *Proceedings of the Congress on Evolutionary Computation 1999 (CEC'99)*, volume 1, pages 612–619, Piscataway, New Jersey, IEEE Service Center, July (1999)
[34] Zbigniew Michalewicz and Marc Schoenauer, "Evolutionary Algorithms for Constrained Parameter Optimization Problems", *Evolutionary Computation*, 4(1):1–32, (1996).
[35] Sana Ben Hamida and Marc Schoenauer, "ASCHEA: New Results Using Adaptive Segregational Constraint Handling", In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 884–889, Piscataway, New Jersey, IEEE Service Center, May (2002).
[36] Thomas P. Runarsson and Xin Yao. "Stochastic Ranking for Constrained Evolutionary Optimization", *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September (2000).
[37] Arturo Hernández-Aguirre, Salvador Botello-Rionda, and Carlos A. Coello Coello, "PASSSS: An Implementation of a Novel Diversity Strategy for Handling Constraints". In *Proceedings of the Congress on Evolutionary Computation 2004 (CEC'2004)*, volume 1, pages 403-410, Piscataway, New Jersey, Portland, Oregon, USA, IEEE Service Center. June (2004)
[38] Joshua D. Knowles and David W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy", *Evolutionary Computation*, 8(2):149–172, (2000).
[39] Efrén Mezura-Montes and Carlos A. Coello Coello. "A Simple Evolution Strategy to Solve Constrained Optimization Problems". In Erick Cantú-Paz et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, pages 640–641, Heidelberg, Germany, Chicago, Illinois, Springer Verlag. Lecture Notes in Computer Science Vol. 2723, July (2003).
[40] Efrén Mezura-Montes and Carlos A. Coello Coello. "Adding a Diversity Mechanism to a Simple Evolution Strategy to Solve Constrained Optimization Problems". In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 1, pages 6–13, Piscataway, New Jersey, Canberra, Australia, IEEE Service Center, December (2003).
[41] Nikolaus Hansen and Andreas Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies", *Evolutionary Computation*, 9(2):159–195, (2001).