COMPDYN 2011
3rd ECCOMAS Thematic Conference on
Computational Methods in Structural Dynamics and Earthquake Engineering
M. Papadrakakis, M. Fragiadakis, V. Plevris (eds.)
Corfu, Greece, 25–28 May 2011

Normalized domination selection criteria for differential evolution algorithms in constrained optimization for seismic engineering

J. Avakian[1], A. Fiore[2], D. Serio[1], R. Greco[2] G.C. Marano[1]

1Department of Environmental Engineering and Sustainable Development, Technical University of Bari, viale del Turismo 10, 74100 Taranto, Italy
[2]Department of Civil and Architectural Engineering, Technical University of Bari, viale del Turismo 10, 74100 Taranto, Italy

Keywords: Structural optimization, constrained optimization, evolution algorithms, selection criteria

Abstract. *Optimization is a central aspect of structural engineering, but its practical application hasn't been supported by mathematical and numerical tools because of inner strong nonlinear aspects involved. Moreover during last few decades Evolutionary Algorithms (EAs) gives new interest and horizons in this specific topic, thanks to their strong capacity in treatment of these problems more efficiently than standard methods. But a common criticism to EAs is lack of efficiency and robustness in handling constraints, mainly because they were originally developed for unconstraint problems only. For this reason during past decade hybrid algorithms combining evolutionary computation and constraint-handling techniques have shown to be effective in this specific area. Moreover still now this is a crucial point for practical applications in structural optimization. In this paper a Normalized Domination Selection-based (NDS) rule is proposed to solve constrained-handling optimization problems using a modified version of proposed Differential Evolution algorithm (NDS - DEa). The strategy developed doesn't requires any additional parameter, increasing the appeal for a simple implementation in many real problems by structural designer without a specific knowledge in the field. Mainly it is based on a domination criteria in selection phase. Actually a common way for constrained handling is introducing a specific role for selection step, so that all other phase of EA aren't modified; in this way DE flow chart scheme doesn't present any modification from a standard unconstrained one. Anyway the specific constrained selection scheme plays an important role in solution search efficiency, certainly more than in unconstrained cases. Unconstrained selection is based only on comparing individuals OF values, but in constrained one it seemed somewhat different and complicated. The more simple, common and intuitive way for approaching this phase is the penalty function, where OF values are reduced for those individuals don't satisfying constraints disqualifies (unfeasible individuals). It is immediate (and well known in literature) that depending on penalty low adopted, a more drastic or permissive surviving of unfeasible solutions happened. But this is a central point in this problems, because of in many cases indeed real optimal solutions lies*

just on one constraints, so that its correct evaluation needs of specific research around the boundary, not only in the feasible space. to develop this strategy the domination concept is related to the specific selection that has to be implemented. If in a unconstrained contest it means simply that the domination coincide with the OF ranking, in the constrained contest the question has to be properly treated. In fact there are three possible scenarios:

- *both two individuals are feasible -> selection based on rank*
- *both two individuals are unfeasible -> selection of the feasible one*
- *one is feasible and the other is unfeasible -> selection of less unfeasible*

Moreover the last case presents same some ambiguity because in general there are many constraints with different scales, so that it is impossible to rank correctly to different unfeasible individuals. For this reason a normalized criteria is here proposed and analyzed with different cross over methodology. A comparative analysis using different test cases is performed.

1 INTRODUCTION

Optimization techniques play an important role in many scientific, economical, social and many other problems, the very purpose of which is to find the best way to do something or alternatively to help decision makers to derive the maximum benefit from limited available resources. A common way in many cases has been using past experiences in similar conditions or extending indications from comparable situations. These approaches will not lead in general to the best solution, but the shortcomings of indirect design can be overcame by adopting a direct or optimal design procedure.

An optimization problem or an optimal design procedure consists in finding a maximum or a minimum function problem under some constraint conditions. To deal with this class of problems many different approaches are possible in theory and many heuristic intelligent algorithms have been developed for different classes of optimization problems.

Within the framework of the soft computing methodologies, an incredible number of non-conventional paradigms has been explored in order to create efficient and user-friendly optimizers. Nowadays, a wide variety of biological, social and physical metaphors has been analyzed and tested. Evolutionary Algorithms EAs are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social behavior of species. EAs are ubiquitous nowadays, having been effectively applied to several problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems. Usually grouped under the term Evolutionary Computation or Evolutionary Algorithms, one can find the domains of Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Genetic Programming.

These are stochastic search and optimization heuristics derived from the classic evolution theory, which are implemented on computers in the majority of cases. The basic idea is that if only those individuals of a population reproduce, which meet a certain selection criteria, and the other individuals of the population die, the population will converge to those individuals that best meet the selection criteria. If imperfect reproduction is added, the population can begin to explore the search space and will move to individuals that have an increased selection probability and that inherit this property to their descendants. These population dynamics follow the basic rule of the Darwinist evolution theory, which can be described in short as the "survival of the fittest" . To solve optimization problems with an evolutionary heuristic the individuals of a population have to represent a possible solution of a given problem and the selection probability is set proportional to the quality of the represented solution.

The interest toward this class of optimizers is continuously attracting consensus, substantially because the objective function and constraints are not required to be differentiable, continuous or even explicit. In effect, no preliminary assumptions or *a priori* information are needed for solving constrained optimization problems by means of EAs.

In these author knowledge, at the moment, there isn't a unique accepted classification of different EAs proposed in literature, so that a possible non exhaustive one is proposed in the following, where similar techniques differ in the implementation details and in the nature of the particular applied problem.

- ***Genetic algorithm****: This is the most popular (and older) type of EA. One seeks the solution of a problem in the form of strings of numbers (traditionally binary, although the best representations are usually those that reflect something about the problem being solved), by applying operators such as recombination and mutation (sometimes one, sometimes both). This type of EA is often used in optimization problems;*
- ***Genetic programming****: Here the solutions are in the form of computer programs, and their fitness is determined by their ability to solve a computational problem.*

- ***Evolutionary programming****: Like genetic programming, only the structure of the program is fixed and its numerical parameters are allowed to evolve;*
- ***Evolution strategy*** *- This works with vectors of real numbers as representations of solutions, and typically uses self-adaptive mutation rates;*
- ***Differential evolution****: It is based on vector differences and it is therefore primarily suited for numerical optimization problems.*
- ***Particle swarm optimization****: This is based on the ideas of animal flocking behavior and it is also primarily suited for numerical optimization problems.*
- ***Ant colony optimization****: This is based on the ideas of ant foraging by pheromone communication to form paths. It is primarily suited for combinatorial optimization problems.*
- ***Invasive weed optimization algorithm****: It is based on the ideas of weed colony behavior in searching and finding a suitable place for growth and reproduction.*
- ***Harmony search****: Based on the ideas of musicians' behavior in searching for better harmonies. This algorithm is suitable for combinatorial optimization as well as parameter optimization.*
- ***Gaussian adaptation****: This is based on information theory. Used for maximization of manufacturing yield, mean fitness or average information. See for instance Entropy in thermodynamics and information theory.*

Among these approaches one of the more promising one is the Differential Evolution algorithm (DEa): this can be defined a recent stochastic, population-based global optimization method and was proposed about a decade ago [10][11]. The algorithm is based on the use of a special crossover-mutation operator, based on the linear combination of three different individuals and one subject-to-replacement parent. The selection process is performed via deterministic tournament selection between the parent and the child created by it. It is immediate noting that the general structure of the DEa shares similar features with other evolutionary algorithms like GAs. For instance, both optimizers adopt the same terminology to define the key elements of the algorithm (i.e. a collection of solutions is called “population”, each solution is called “individual” and each iteration is called “generation”) and incorporate operators (like mutation, crossover and selection scheme) that work in similar manners. Nonetheless, it is different in handling distance and direction information to move from the population at the current generation toward the next one because it has constructive cooperation between individuals: in this sense, it behaves like Particle Swarm Optimization algorithms (PSOs). Another interesting feature of DEa deals with the selection operator that performs very well and sometimes it is more efficient and faster than other population based algorithms, because of the one-to-one competition scheme. In more general terms, its fashion can be imputable to two positive features; firstly, the DEs provides more simple operators in comparison to the most advanced GAs. Moreover, it requires only few embedded control parameters (typically the total number of control parameters is less than the adopted ones for GAs and PSOAs), so that the parameter tuning stage is less time consuming and maybe more practical for non-experts in the field of soft computing techniques.

All attractive features of EAs are in opposition to some criticisms. For instance, EAs suffer the lack of well posed theories about their convergence and a larger computational time is typically required. Moreover, in their original formulation they was limited to unconstrained problems and do not include a method to incorporate feasibility information into the fitness function. In effect, many real-world optimization problems in science and engineering involve a number of constraints which the optimal solution must satisfy. Actually, due to constraints, the feasible space might be reduced to some portion, sometimes very narrow if compared with

the overall search space. Sometimes simply finding feasible solutions itself could be a daunting challenge for some specific practical problems.
As it is know, the ultimate goal of constrained optimization problem is to find the feasible optimal solution. To achieve this goal in EAs, it is required that more feasible individuals are involved in the evolution process. However, on the other hand, some infeasible individuals may carry some information, sometime important, for the final solution than their feasible counterparts in some generations. Hence, these two aspects lead to a contradiction in constrained evolutionary optimization. To address this contradiction, the main challenge is to handle the constraints and to optimize the objective function simultaneously. One possible way is to determinate the tradeoff between the constraint violations and the objective function.
This has trigger a considerable amount of researches and a wide variety of approaches have been suggested in the last few years to incorporate constraints into the fitness function of an evolutionary algorithm. The most popular approach is the use of (mainly exterior) penalty function [12] where the aim is to decrease the fitness of infeasible solutions in order to favor the selection of feasible solutions. Several alternative constraint-handling techniques have been proposed [9].
Actually it is generally accepted that performance of an algorithm largely depends on the underlying mechanism of constraint handling. Motivated by this fact, a number of constraint-handling techniques have been proposed for evolutionary algorithms, and over last few years several methods have been proposed.
These methods have been grouped by different authors into the following categories [1][2][4]:

- *Methods based on preserving the feasibility of solutions. The idea behind the method is based on specialized operators which transform feasible parents into feasible offspring. The method assumes linear constraints only and a feasible starting point or feasible initial population.*
- *Methods based on penalty functions. Many evolutionary algorithms incorporate a constraint-handling method based on the concept of exterior penalty functions which penalize infeasible solutions. These methods differ in important details, such as how the penalty function is designed and applied to infeasible solutions.*
- *Methods which make a clear distinction between feasible and infeasible solutions. There are a few methods which emphasize the distinction between feasible and infeasible solutions in the search space. One of those methods distinguishes between feasible and infeasible individuals: for any feasible individual* $\boldsymbol{x}$ *and any infeasible individual* $\boldsymbol{y}$*:* $f(\boldsymbol{x}) \lhd f(\boldsymbol{y})$*, i.e. any feasible solution is better than any infeasible one.*
- *Other hybrid methods. These methods combine evolutionary computation techniques with deterministic procedures for numerical optimization problems. Most constrained problems can be handled by the penalty function method. A measure of the constraint violation is often useful when handling constraints.*

According to the no free lunch theorem, it is impossible for a single constraint handling technique to outperform all other techniques on every problem. In other words, depending on several factors such as the ratio between feasible search space and the whole search space, multimodality of the problem, the chosen EA and global exploration/local exploitation stages of the search process, different constraint handling techniques can be effective on different problems and during different stages of the search process.
The most used techniques incorporate constraints into the fitness function, such as in penalty functions approach, but they usually eliminate all unfeasible individuals. On the other hand it is particularly important to maintain diversity in the population and to be able to keep solu-

tions both inside and outside the feasible region. In fact, several studies have shown that, despite their popularity, traditional (external) penalty functions, even when used with dynamic penalty factors, tend to have difficulties to deal with highly constrained search spaces and with problems in which the constraints are active in the optimum. In these situations infeasible individuals may carry more important information for the final solution than their feasible counterparts in some generations. Hence, these two aspects lead to a contradiction in constrained evolutionary optimization. The use of (exterior) penalty functions is one of the most popular methods to deal with constrained search spaces when using Constrained optimization Evolution Algorithms (COEAs).

For these reasons an efficient and adequate constraint-handling technique is a key element in the design of competitive COEAs to solve complex optimization problems. In this way, this subject deserves special research efforts, with the main aim of proposing approaches able to prevent a too fast convergence without an adequate global (feasible and unfeasible space) research process. Some recent proposes in this way investigated modification of what is generally known as the *Constraint Domination Selection (CDS) rule*.

This methodology was initially proposed by Deb [3] as a modification of Powell and Skolnick method which gives a dynamic penalty to each element so that at each generation the best unfeasible element has a rank that is better than the worst unfeasible one. The Deb's method uses a tournament selection operator, where two solutions are compared. In this method, any feasible solution is preferred to any infeasible solution; among two feasible solutions, the one having a better objective function value is preferred and among two infeasible solutions, the one having smaller constraint violation is preferred. Moreover all these method are static application of a *Constraint Domination Selection* (CDS) rule, that practically prefers feasible solutions to unfeasible ones. Finally this was implemented on different types of Evolution Strategies in which the results were very promising [7][6]

The main motivation of this work was to increase this selection criteria to better perform constrained optimization by correctly using selection criteria with specific crossover operators. Standard differential evolution algorithms in unconstrained Optimum Design

The feature of the optimal design is that it consists of only logical decisions in a mathematical way, setting out constraints, and minimizes or maximizes an objective function, that is generally either cost, benefit or a generic merit function. Many of the methods give rise to local minimum/maximum. This, however, depends on the mathematical nature of the optimization problem, that generally can be described as follows:

Find the best vector $\mathbf{x} \in \boldsymbol{\Omega}$ *that minimizes* $f(\mathbf{x})$ (1)

Satisfying the following constraints

$$g_i(\mathbf{x}) \le 0 \quad \mathrm{i} = 1, 2, .., n_g \tag{2}$$

$$h_j(\mathbf{x}) = 0 \quad j = 1, 2, .., n_h \tag{3}$$

in which $\mathbf{x} = \{x_1,\ldots,x_j,\ldots,x_n\}$ is a vector whose components are real numbers, $f(\mathbf{x})$ is the objective function (OF) to be minimized and $\boldsymbol{\Omega}$ is a box-type search space. For instance, if $[x^l_j, x^u_j]$ is the admissible interval for the j^{th} variable (x^l_j and x^u_j are its lower and the upper bounds, respectively), then:

$$\boldsymbol{\Omega} = \left[x_1^l, x_1^u\right] \otimes \left[x_2^l, x_2^u\right] \ldots \otimes \left[x_j^l, x_j^u\right] \otimes \ldots \left[x_n^l, x_n^u\right] \subseteq \mathbb{R} \tag{4}$$

where the symbol $\otimes$ denotes the Cartesian product between intervals.

The constraints of the optimization problems can be both inequalities $g_i(\mathbf{x})$ or equalities $h_j(\mathbf{x})$. Without loss of generality, all equalities can be converted into inequalities using the transformation $\left|h_j(\mathbf{x})\right| - \varepsilon \le 0, \quad j = 1,2,..,n_h$, where ε is a tolerance parameter.

Therefore, in the following we will refer only to inequalities-based constraints, *e.g.* $g_p(\mathbf{x}) \le 0$ with $p = 1,\ldots,n_g, n_g+1,\ldots,n_g+n_h$.

A solution $\mathbf{x}$ is regarded as *feasible if*:

$$\sum_{k=1}^{n_p} \max\left(0, g_k(\mathbf{x})\right) \le 0 \tag{5}$$

otherwise it is called as *unfeasible*.

A general optimization problem can be formulated as a typical minimization problem in the form

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) \right\} \tag{6}$$

s.t. $\mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u$

in which $\mathbf{x} = \{x_1,\ldots, x_j, ...,x_n\}$ is the design vector (for example the collection of n system parameters to be identified), $\mathbf{x}^l = \{x_1{}^l,\ldots, x_j^l, ...,x_n{}^l\}$ and $\mathbf{x}^u = \{x_1{}^u,\ldots, x_j^u, ...,x_n{}^u\}$are its lower and upper bounds, respectively. The shape of the objective function may have many local optima and high complex topology, therefore, when preliminary information are not available it may not always be convex. In these circumstances special optimizers have to be used.

In the following it is illustrated the state of the art of DEa for problems in form (6).

Differential evolution is a simple but powerful population-based stochastic search technique for solving global optimization problem which is characterized by simplicity, effectiveness and robustness. Its main idea is to construct at each generation, for each element of the population a mutant vector. This mutant vector is constructed trough a specific mutation operation based on adding differences between randomly selected elements of the population to another element. The original DE algorithm is described in the following briefly. The main variation introduced for constrained problems is in selection process, where use of CDS gives to algorithms for unconstrained problem a natural extension for constrained ones. Moreover the CDS is also applied to evaluation of "best" individual over the entire population. This because some proposed DE algorithm present a mutation that take into account information about the "best" individual over the entire population. In this case a "direction" to escape from constrained space is available at each iteration.

The general structure of a DE is typically for evolutionary algorithms, the particularity of the algorithm being related with the mutation and crossover operators. By combining different mutation and crossover operators various schemes have been developed. In literature different DE schemes are denoted using the convention DE/a/b/c where a denotes the way of constructing the mutant vector, b denoted the number of differences in the construction of the mutant vector and, finally, c denotes the crossover type.

The working of DEa depends on the manipulation and the efficiency of three main operators: mutation, crossover and selection.

2 MUTATION

The main idea of DEa is to construct at each generation for each element of the population a mutation vector. The mutant vector is constructed through a specific mutation operation based on adding differences between randomly selected element of the population to another element.

DEa uses the differences betiween two randomly selected individuals as the source of random variations for a third indivdual referred to as the *target vector*. *Trial solutions* are generated by adding weighted difference vectors to the target vector. This process is referred to as the mutation operator: its main goal is to enable diversity in the current population as well as to direct the individuals in such a way a better result is expected. By computing the differences between two individuals randomly chosen from the population, the algorithm estimates the gradient in that zone rather than in a point. Let consider ${}^{k}\mathbf{x}_i = \{{}^{k}x_{i1},\ldots,{}^{k}x_{ij},\ldots,{}^{k}x_{in}\}$ the i^{th} individual (for $i = 1,\ldots,N$) at iteration k. The initial population ${}^{0}\mathbf{x}_i$ for $i = 1,\ldots, N$ is defined by generating pseudo-randomly the collection of N solutions within the specified search space. In this study the Latin Hypercube Sampling (LHS) technique has been iteratively used to generate the best initial population with minimum correlation between samples [8]. At iteration k+1, for each individual ${}^{k}\mathbf{x}_i$ a mutation vector ${}^{(k+1)}\mathbf{v}_i$ is computed by means of one of the following alternatives:

rand /1/bin

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_{r1} + F^1\left({}^{k}\mathbf{x}_{r2} - {}^{k}\mathbf{x}_{r3}\right) \tag{7} $$

best /1/bin

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_{best} + F^1\left({}^{k}\mathbf{x}_{r1} - {}^{k}\mathbf{x}_{r2}\right) \tag{8} $$

current-to-best /1/bin

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_i + F^2\left({}^{k}\mathbf{x}_{best} - {}^{k}\mathbf{x}_i\right) + F^1\left({}^{k}\mathbf{x}_{r1} - {}^{k}\mathbf{x}_{r2}\right) \tag{9} $$

best /2/bin

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_{best} + F^2\left({}^{k}\mathbf{x}_{r1} - {}^{k}\mathbf{x}_{r2}\right) + F^1\left({}^{k}\mathbf{x}_{r3} - {}^{k}\mathbf{x}_{r4}\right) \tag{10} $$

rand /2/bin

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_{r1} + F^2\left({}^{k}\mathbf{x}_{r2} - {}^{k}\mathbf{x}_{r3}\right) + F^1\left({}^{k}\mathbf{x}_{r4} - {}^{k}\mathbf{x}_{r5}\right) \tag{11} $$

Two new possible mutation strategy candidates are thus here reported to evaluate their effectiveness in

rand /1/bin (based on tournament selection

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_i + {}^{k}F_{r3,i}\left({}^{k}\mathbf{x}_{r3} - {}^{k}\mathbf{x}_i\right) + {}^{k}F_{r1,r2}\left({}^{k}\mathbf{x}_{r1} - {}^{k}\mathbf{x}_{r2}\right) \quad if\ \ k \le 0.5L \tag{12} $$

rand/1/bin (based on attraction-repulsion paradigm)

$$ {}^{(k+1)}\mathbf{v}_i = {}^{k}\mathbf{x}_{r1} + {}^{k}F_{best,i}\left({}^{k}\mathbf{x}_{best} - {}^{k}\mathbf{x}_i\right) + {}^{k}F_{r1,r2}\left({}^{k}\mathbf{x}_{r1} - {}^{k}\mathbf{x}_{r2}\right) \quad if\ \ k > 0.5L \tag{13} $$

The mutation coefficients in (12) and in (13) are calculated as follows:

$$ {}^{k}F_{r3,i} = \max\left\{\left|\frac{f\left({}^{k}\mathbf{x}_{r3}\right) - f\left({}^{k}\mathbf{x}_i\right)}{{}^{k}f_{\max} - {}^{k}f_{\min}}\right|, 0.5\right\} \tag{14} $$

$$ {}^{k}F_{r1,r2} = \begin{cases} \max\left\{\left|\dfrac{f\left({}^{k}\mathbf{x}_{r1}\right) - f\left({}^{k}\mathbf{x}_{r2}\right)}{{}^{k}f_{\max} - {}^{k}f_{\min}}\right|, 0.5\right\} & if\ \ k \le 0.5L \\ \left|\dfrac{f\left({}^{k}\mathbf{x}_{r1}\right) - f\left({}^{k}\mathbf{x}_{r2}\right)}{{}^{k}f_{\max} - {}^{k}f_{\min}}\right| & if\ \ k > 0.5L \end{cases} \tag{15} $$

$$ {}^{k}F_{best,i} = \left| \frac{{}^{k}f_{\min} - f\left({}^{k}\mathbf{x}_i\right)}{{}^{k}f_{\max} - {}^{k}f_{\min}} \right| \qquad (16) $$

in which

$$ {}^{k}f_{\min} = \min_{i=1,\ldots,N} \left\{ f\left({}^{k}\mathbf{x}_i\right) \right\} \quad {}^{k}f_{\max} = \max_{i=1,\ldots,N} \left\{ f\left({}^{k}\mathbf{x}_i\right) \right\} \qquad (17) $$

From Equation (7) to Equation (11) $r1$, $r2$, $r3$ and $r4$ denote integers randomly selected within the set $\{1,\ldots,i\text{-}1,i\text{+}1,\ldots,N\}$ and $r1 \neq r2 \neq r3 \neq r4$. The individual ${}^{k}\mathbf{x}_{best}$ is the best performer in the population at the iteration k. The coefficients F^1 and F^2 are the so-called mutation coefficients and they are real positive constants. These parameters control the amplification level due to the mutation operator (for this reason they are also dubbed scale factors). Any alternative mutation operator leads to different versions of DEAs [10]: rand/1/bin, best/1/bin, current-to-best/1/bin, best/2/bin, rand/2/bin, respectively. Storn and Price shown that the usage of two difference vectors may improve the diversity of the population, especially when the population size is high enough.

As before stated, only mutation in (8), (9) and (10) are used in the following for constrained handling problems. This because they are specifically implemented with a "best" selection based on a CDS rule.

3 CROSSOVER

The perturbed individual ${}^{(k+1)}\mathbf{v}_i$ and the current population ${}^{k}\mathbf{x}_i$ are then subject to crossover operation. The crossover follows the mutation phase and for each mutated vector ${}^{(k+1)}\mathbf{v}_i$ a trial vector ${}^{(k+1)}\mathbf{u}_i$ (offspring) is generated by using the binomial crossover formalized in Equation (18).

$$ {}^{(k+1)}u_{ij} = \begin{cases} {}^{(k+1)}v_{ij} & \text{if } u \le p^c \text{ for } j = randint(0,n) \\ {}^{k}x_{ij} & \text{otherwise} \end{cases} \qquad (18) $$

In Equation (18) u is a pseudo-random number generated by using the uniform probability density functions in the range [0,1]. The parameter p^c is the probability of crossover (or crossover ratio or probability of reproduction) and it takes values between 0 and 1 and it is set by the user. Moreover, *randint*(0,n) is a pseudo-randomly integer selected within the set $\{1,\ldots,j,\ldots,n\}$: based on its realization, an additional condition is introduced to ensure that at least one parameter is taken into account for constructing the vector ${}^{(k+1)}\mathbf{u}_i$.

4 SELECTION

The selection operator in case of unconstrained problems employs a very simple one-to-one competition scheme between ${}^{(k+1)}\mathbf{u}_i$ and ${}^{(k+1)}\mathbf{x}_i$ as follows

$$ {}^{(k+1)}\mathbf{x}_i = \begin{cases} {}^{(k+1)}\mathbf{u}_i & \text{if } f\left({}^{(k+1)}\mathbf{u}_i\right) < f\left({}^{(k+1)}\mathbf{x}_i\right) \\ {}^{k}\mathbf{x}_i & \text{otherwise} \end{cases} \qquad (19) $$

Therefore, the winner in the selection stage is the best performer between the parent individual and its trial one. The output of this operator is a new population for the next generation, if a stopping criteria has not been satisfied. Likely to the evolutionary algorithms, the required number of iterations L is not known a priori and therefore a stopping criterion is needed. In more general terms, the stopping criteria can be the same typically adopted for GAs, see for instance[8] and its references. In this study, we stop the search once a maximum number of iterations L is achieved.

A general way the most used way to deals with constraint handling problems consists in replacing standard selection criteria with one that considers not only performance (OF) of solution but feasibility too. In this sense, this selection criteria is obtained by operating a solutions ranking by a generalized domination concept between different individuals: given two solutions, namely ${}^{(k+1)}\mathbf{u}_i$ and ${}^{(k+1)}\mathbf{x}_i$ the concept of dominance can be introduced:

$$ {}^{(k+1)}\mathbf{x}_i \succ {}^{(k+1)}\mathbf{u}_i \tag{20} $$

that denotes that ${}^{(k+1)}\mathbf{u}_i$ is dominated by ${}^{(k+1)}\mathbf{x}_i$

In a constrained based selection the concept of domination must be expressed in a wider sense than those obtained considering simply the OF. Taking into account individuals feasibility, violation function for the i^{th} individual is evaluated by the violation function:

$$ \Phi\left({}^{k}\mathbf{x}_i\right) = \sum_{p=1}^{n_q+n_h} \max\left\{0, g_p\left({}^{k}\mathbf{x}_i\right)\right\} \geq 0 \tag{21} $$

so that its value is zero if and only if all constrains are satisfied (the i^{th} individual lies in the feasible design variable space); differently (the i^{th} individual lies in the unfeasible design variable space) the violation function is a positive scalar number.

The selection method able to consider properly feasibility in dominance should be formulated in the following way:

$$ {}^{(k+1)}\mathbf{x}_i \succ {}^{(k+1)}\mathbf{u}_i \Leftrightarrow \begin{cases} \left(f\left({}^{(k+1)}\mathbf{x}_i\right) < f\left({}^{(k+1)}\mathbf{u}_i\right)\right) \textit{ and } \left(\Phi\left({}^{(k+1)}\mathbf{x}_i\right)=0\right) \wedge \left(\Phi\left({}^{(k+1)}\mathbf{u}_i\right)=0\right) \\ \textit{or} \\ \left(\Phi\left({}^{(k+1)}\mathbf{x}_i\right)=0\right) \textit{ and } \left(\Phi\left({}^{(k+1)}\mathbf{u}_i\right)>0\right) \\ \textit{or} \\ \Phi\left({}^{(k+1)}\mathbf{x}_i\right) < \Phi\left({}^{(k+1)}\mathbf{u}_i\right) \end{cases} \tag{22} $$

These should be summarized in simply words by the following rule:

${}^{(k+1)}\mathbf{x}_i$ dominates ${}^{(k+1)}\mathbf{u}_i$ if :

- ${}^{(k+1)}\mathbf{x}_i$ *and* ${}^{(k+1)}\mathbf{u}_i$ *are both feasible solutions, and* ${}^{(k+1)}\mathbf{x}_i$ *has the minimum value of OF*
- ${}^{(k+1)}\mathbf{x}_i$ *is feasible and* ${}^{(k+1)}\mathbf{u}_i$ *is unfeasible*
- ${}^{(k+1)}\mathbf{x}_i$ *and* ${}^{(k+1)}\mathbf{u}_i$ *are both unfeasible solutions, and* ${}^{(k+1)}\mathbf{x}_i$ *has the minimum value of violation.*

This domination criterion should be simply expressed in words as the rule that *"feasible solutions survive to the infeasible in any cases"*.

Moreover, selection scheme (20) based on (22) is well defined in cases 1 and 2. The first is nothing else than standard selection (both two individuals are feasible, so that the selection can't deal with constraints. The second simply defines that when are compared two individuals that are one feasible and the other unfeasible, the feasible survives.

The third case presents some drawbacks because it is not so well defined as the first two. Actually, because of number of constraints involved in a optimization problem (usually greater than one) it is impossible to correctly ranking violation levels of unfeasible individuals by using directly the violation function presented in (21). To properly clarify this point, in a generic search space selecting between two unfeasible individuals, we have to prefer those more closer to the feasible space; unfortunately the violation function, as it has been obtained in (21) is not able to evaluate this distance, so that it isn't able to ranking solutions according their distance to the feasible space because it is obtained simply adding violation levels of different constraints that are not homogeneous according their effective distance from the feasible

space. In simply words it means that, once two solutions are both unfeasible, it isn't possible evaluating correctly which of them is the *"less far"* from the feasible space boundary. This piece of information is needed in evolutionary search algorithms to properly select the best solution among a group of unfeasible ones. So that, the dominance based selection criterion for two unfeasible individuals is still an open question and there isn't a metric measure to evaluate what is the "less worst".

In order to overcome this limitation, modified versions of the standard rule are here proposed, to evaluate if and how they increase algorithm performance in convergence to a feasible space. This aspect is extremely important especially in engineering optimizations, where the necessity of a feasible solution is priority respect to those of a performance ones.

A first variation of standard Deb's Dominance Selection rule consists in using a normalization of violation function in (21). In this way, one scraps from the effective numerical value of the violated constraint, whose value should be very different regardless of the effective distance of individual from the admissible domain. The final goal in this manner is to obtain a ranking in which individuals nearest
to the admissible domain work better.

For this aim, each constraint violation is normalized according to:

$$ {}^{k}\chi_r^i = \frac{\max\left(0, {}^{k}g_r^i\right)}{g_r^{\max}} \tag{23} $$

where:

$$ g_r^{\max} = \max_{j=1,n_p}\left({}^{k}g_r^j\right) \tag{24} $$

being n_p the number of the population.

It is clear that ${}^{k}\chi_r^i \in [0,1]$ for all *k* considered, so that having a normalized weight for all constraints considered.

The violation function is thus defined as

$$ {}^{k}\Phi^i = \sum_{r=1}^{n_v} {}^{k}\chi_r^i \tag{25} $$

5 NUMERICAL ANALYSIS

The proposed domination-based selection schemes have been applied to eleven mathematical and three engineering benchmark optimization problems.

A complete presentation of these benchmark problems is given in Appendix. The optimization problems are solved fifty times by using the NDS_DE and the final results are recorded. The initial population is different for each run. The best, the worst, the mean value of the OF at hand as well as its standard deviation are calculated over the fifty simulated runs. For better understanding the inner work of the proposed selection schemes, the behaviors of an additional indicator is also analyzed.

With this aim, a measure of the difficulty of solving each test problem, a metric measure ρ (as suggested by Koziel and Michalewicz [4]) was introduced as the ratio of the feasible and total population number at each generation, so that considering the following set:

$$ {}^{k}S = \left\{ {}^{k}\mathbf{x}_i \in \mathbf{\Omega} \middle| g_p\left({}^{k}\mathbf{x}_i\right) \le 0 \ \forall p = 1,\ldots,n_q + n_r \right\} \tag{26} $$

whose cardinality is denoted $\#({}^{k}S) \le N$, the ratio ${}^{k}\rho$ is:

$$^{k}\rho = \frac{\#\left(^{k}S\right)}{N} \in [0,1] \tag{27}$$

whose value is 0 when no individual lies in feasible space, and is 1 when all of them are feasible. To properly measure the ratio between feasible and infeasible space dimension, in table 1 it is reported the value of ρ for the mathematical test functions here analyzed, using 10^6 individuals randomly generated [7].

Problem	N	Type of function	ρ	LI	NI	LE	NE
g01	13	*quadratic*	**0,0003%**	9	0	0	0
g02	20	*nonlinear*	**99,9973%**	2	0	0	0
g03	10	*nonlinear*	**0,0026%**	0	0	0	1
g04	5	*quadratic*	**27,0079%**	4	2	0	0
g05	4	*nonlinear*	**0,0000%**	2	0	0	3
g06	2	*nonlinear*	**0,0057%**	0	2	0	0
g07	10	*quadratic*	**0,0000%**	3	5	0	0
g08	2	*nonlinear*	**0,8581%**	0	2	0	0
g09	7	*nonlinear*	**0,5199%**	0	4	0	0
g10	8	*linear*	**0,0020%**	6	0	0	0
g11	2	*quadratic*	**0,0973%**	0	0	0	1

Table 1: Values of ρ (evaluated in a pure random way) for the 11 mathematical test problems chosen

The different values of ρ for each of the functions chosen are shown in Table 1, where *n* is the number of decision variables, LI is the number of linear inequalities, NI the number of nonlinear inequalities, LE is the number of linear equalities and NE is the number of nonlinear equalities.
A standard and a normalized domination-based selection scheme have been adopted and they are indicated in Table 2.

6 RESULTS ANALYSIS

This study takes into consideration designers basic rule that is to prefer a feasible but not economuc solution to a more economic but unfeasible ones. This is an important point in structural and seismic design, where constraints violations usually are associated to an unaceptable low safety level. So that, differently from standard analysis to constrained optimization problems, in this research is analysed firstly the ability of alghoritms to produce feasible solutions and only after is analised their performances. In this research are analyzed 12 test functions from literatures [3].
Tree main efficience indicators are used to evaluate performances, with references to differ selection criteria and cross over alghoritms; the performane indives are :

- *Objective Function*
- *Stagnation*
- *ratio between Unfeasible individuals and total population size* ρ

Due to random nature of DE, performanes are evaluatred as statistically over 100 indepenedent runs, and in particolar as:

- Best
- Worst

- Mean
- Standard Deviation

The first analysis is performer over a population of 50 individuals and a number of 300 generations. As first analyzed index the ration between unfeasible and total population size ρ.

This ratio ρ is evaluated with reference to how many iterations are necessary to reach two different goals: the first is to have at least one feasible individual over all independent runs (figure 1) and the second is to have at least one feasible individuals over all independent runs. From a pragmatic point of view the first indication concerns performance while the second concerns robustness of algorithms with reference to ability to reach feasible solutions.

Smaller is number of generations needs to reach both to indicators, and greater is effectiveness of algorithms in reaching feasible solutions. If finally generations number is 300, than any feasible solution is reached at all.

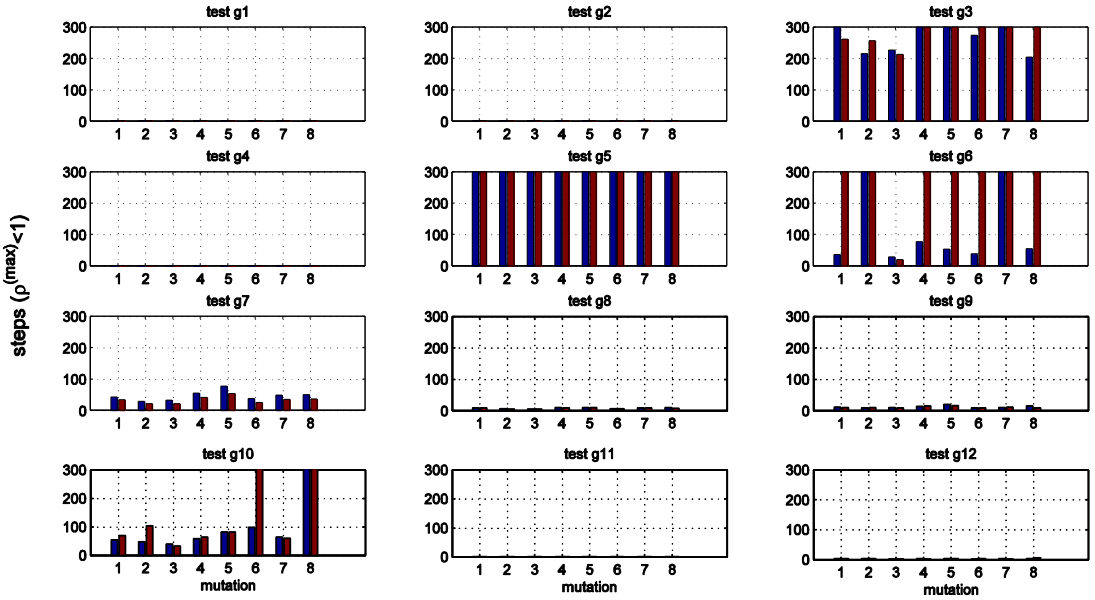

Figura 1: number of steps necessary to have that all runs gives at least one feasible solution. Mutation are those reported in table XX, while blu bars are for normalised approach, while red ones are for standard approach

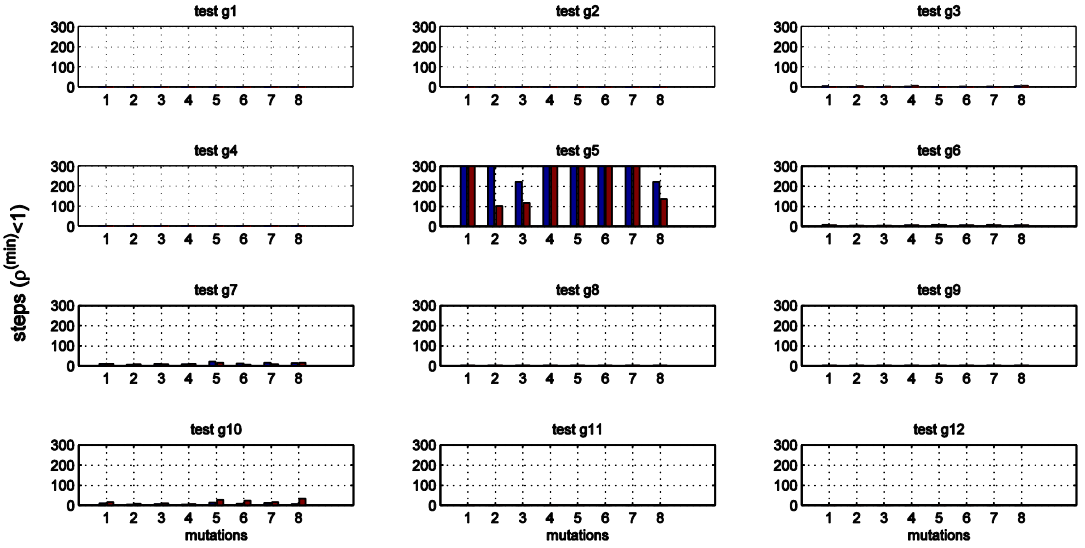

Figura 2: number of steps necessary to have that at least one runs presents a single feasible solution. Mutation are those reported in table XX, while blu bars are for normalised approach, while red ones are for standard approach

Mutation type	denomination	Equation reference
1	rand/1/bin	(7)
2	best/1/bin	(8)
3	current-to-best/1/bin	(9)
4	best/2/bin	(10)
5	rand/2/bin	(11)
6	Proposed 1	(12)
7	Proposed 2	(13)
8	Random (1-7)	(7) - (13)

Table 2: mutation type used in the analysis.

It is immediate that the function g5 is the more difficult to deal with. From figure 2 it appears that without any combination of cross over and selection criteria we have all independent runs with at all a single feasible solution. That means that the worst case is that after 300 generations we are not able to reach any feasible individual.

To better understand performances for ratios ρ are reported for each test function, for all considered combinations of cross over and selection criteria, that are Normalized (on the left) and Standard (on the right). Results are reported after 100, 200 and 300 generations (to evaluate efficiency under different number of generations) in best case (minimum value) in table 3 and worst case (maximum value) in table 4. When rho is equal to one means that no feasible solution are present, and when is equal to 0 means that all solutions are feasible.

generations	test functior	mutation															
		rand/1/bin		best/1/bin		rent-to-best/1/		best/2/bin		rand/2/bin		rand/1/bin		rand/1/bin		random	
		N	S	**N**	S	**N**	S	**N**	S	**N**	S	**N**	S	**N**	S	**N**	S
100	1	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,02**	0
	2	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	3	**0,92**	0,9	**0,88**	0,86	**0,22**	0,84	**0,2**	0,18	**0,9**	0,9	**0,9**	0,9	**0,92**	0,94	**0,86**	0,92
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	6	**0**	0	**0**	0	**0,06**	0	**0,34**	0	**0,42**	0	**0**	0,84	**0**	0,98	**0,46**	0,98
	7	**0**	0	**0**	0	**0**	0	**0,12**	0	**0,36**	0	**0**	0	**0**	0	**0**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	10	**0**	0	**0**	0	**0,02**	0	**0**	0	**0,1**	0,2	**0**	0,06	**0**	0	**0**	0
	11	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	12	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
200	1	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,02**	0
	2	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	3	**0,88**	0,88	**0,82**	0,74	**0,08**	0,74	**0,06**	0,04	**0,86**	0,86	**0,86**	0,84	**0,88**	0,9	**0,78**	0,84
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	0	**1**	0,04	**1**	1	**1**	1	**1**	1	**1**	1	**1**	0,48
	6	**0**	0	**0**	0	**0,04**	0	**0**	0	**0**	0	**0**	0	**0**	0,98	**0**	0,98
	7	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	10	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	11	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	12	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
300	1	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,02**	0
	2	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	3	**0,84**	0,84	**0,7**	0,64	**0,06**	0,6	**0,04**	0	**0,8**	0,78	**0,78**	0,76	**0,86**	0,86	**0,74**	0,8
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	0	**0,22**	0	**1**	1	**1**	1	**1**	1	**1**	1	**0,08**	0
	6	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0,98	**0**	0,98
	7	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	10	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	11	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	12	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0

table 3: ratio ρ, best solutions over 100 independent runs

generations	test function	mutation															
		rand/1/bin		best/1/bin		rent-to-best/1/		best/2/bin		rand/2/bin		rand/1/bin		rand/1/bin		random	
		N	S	N	S	N	S	N	S	N	S	N	S	N	S	N	S
100	1	**0**	0	**0**	0	**0,08**	0	**0,02**	0	**0**	0	**0,04**	0	**0**	0	**0,08**	0
	2	**0,02**	0,02	**0,04**	0,04	**0,04**	0,04	**0,04**	0,04	**0,04**	0,04	**0**	0	**0,02**	0,02	**0,04**	0,02
	3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	6	**0,56**	1	**1**	1	**0,54**	0,3	**0,86**	1	**0,88**	1	**0,86**	1	**1**	1	**0,92**	1
	7	**0,18**	0	**0,08**	0	**0,1**	0,02	**0,64**	0,16	**0,82**	0,26	**0,06**	0	**0,22**	0	**0,24**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0,02**	0	**0,02**	0	**0,02**	0	**0,02**	0	**0,02**	0	**0**	0	**0**	0	**0,02**	0
	10	**0,42**	0,44	**0,12**	1	**0,4**	0,08	**0,66**	0,36	**0,86**	0,9	**0,98**	1	**0,44**	0,48	**1**	1
	11	**0**	0	**0,54**	0	**0,1**	0,12	**0**	0	**0**	0	**0**	0	**0,02**	0,02	**0,12**	0
	12	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,04**	0
200	1	**0**	0	**0**	0	**0,06**	0	**0,02**	0	**0**	0	**0,02**	0	**0**	0	**0,08**	0
	2	**0**	0	**0,02**	0,02	**0,02**	0,02	**0,04**	0,04	**0,04**	0,04	**0**	0	**0**	0,02	**0,02**	0
	3	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	6	**0**	1	**1**	1	**0,44**	0,12	**0,54**	1	**0,66**	1	**0,32**	1	**1**	1	**0,62**	1
	7	**0,02**	0	**0,02**	0	**0,04**	0	**0,1**	0	**0,16**	0	**0,02**	0	**0,02**	0	**0,04**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0**	0	**0**	0	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0
	10	**0,02**	0	**0,04**	0,08	**0,16**	0,04	**0,08**	0,02	**0,04**	0,04	**0,04**	1	**0,02**	0,02	**1**	1
	11	**0**	0	**0,06**	0	**0,06**	0,1	**0**	0	**0**	0	**0**	0	**0,02**	0	**0,08**	0
	12	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,04**	0
300	1	**0**	0	**0**	0	**0,06**	0	**0,02**	0	**0**	0	**0,02**	0	**0**	0	**0,06**	0
	2	**0**	0	**0**	0,02	**0,02**	0	**0,04**	0,04	**0,04**	0,04	**0**	0	**0**	0	**0**	0
	3	**1**	0,98	**0,96**	0,96	**0,98**	0,96	**1**	1	**1**	1	**0,98**	1	**1**	1	**0,98**	1
	4	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	5	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1	**1**	1
	6	**0**	1	**1**	1	**0,42**	0,08	**0,42**	1	**0,4**	1	**0**	1	**1**	1	**0,48**	1
	7	**0,02**	0	**0,02**	0	**0,02**	0	**0,02**	0	**0,04**	0	**0,02**	0	**0,02**	0	**0,02**	0
	8	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	9	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0**	0
	10	**0**	0	**0,02**	0	**0,16**	0,04	**0,04**	0,02	**0**	0	**0,02**	1	**0**	0	**1**	1
	11	**0**	0	**0**	0	**0,04**	0,06	**0**	0	**0**	0	**0**	0	**0**	0	**0,06**	0
	12	**0**	0	**0**	0	**0,02**	0	**0**	0	**0**	0	**0**	0	**0**	0	**0,02**	0

table 4: ratio ρ, worst solutions over 100 independent runs

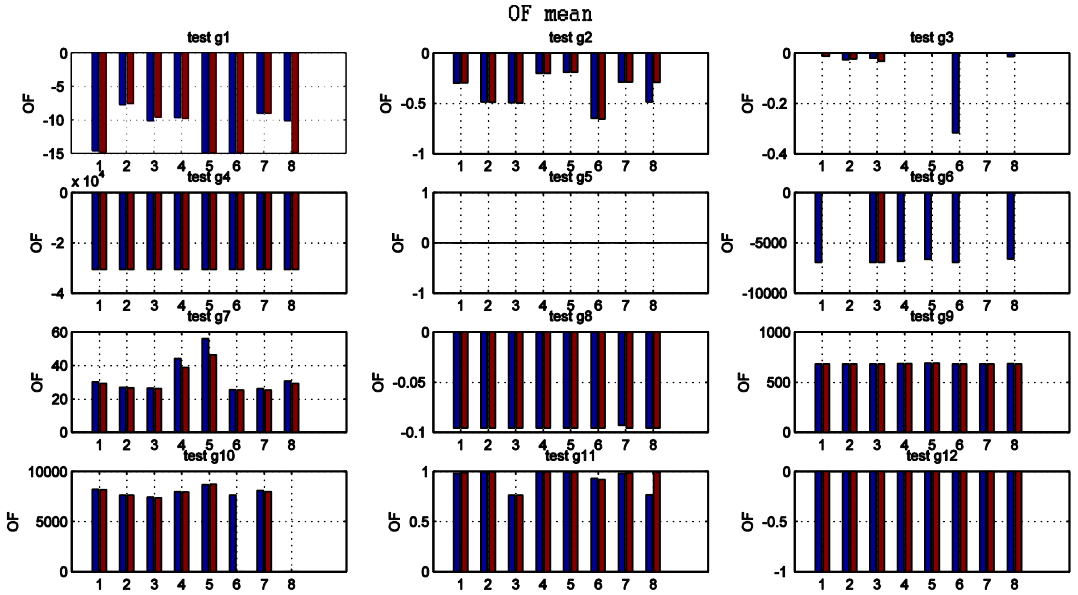

Figure 3: mean values of OF

Finally an aveluation of OF is reported in figure 3, where are reported

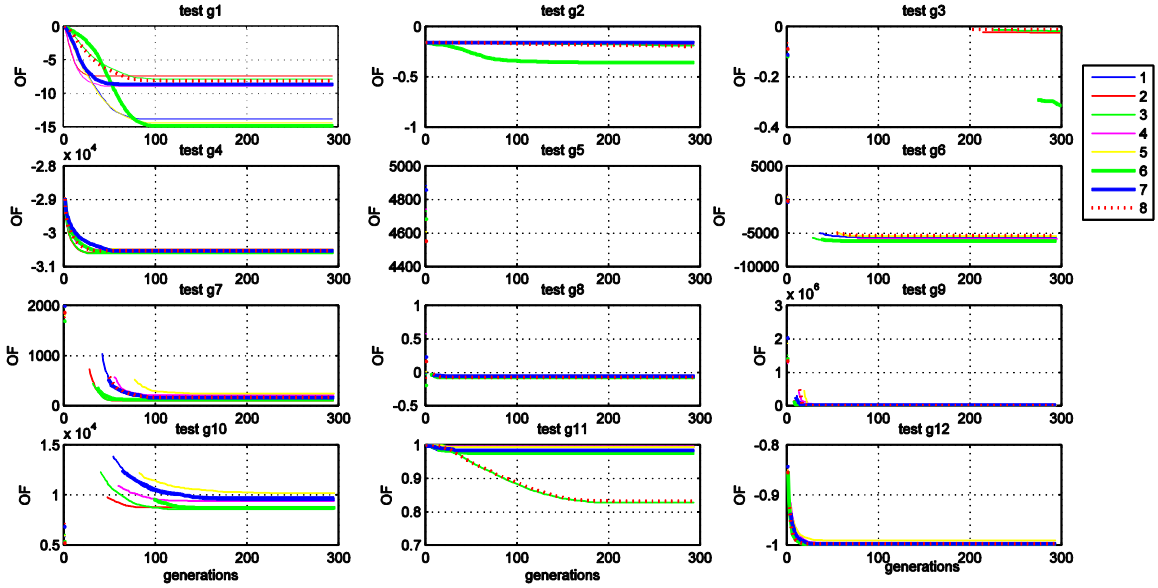

Figura 4: mean values of OF using a normalised selection criteria

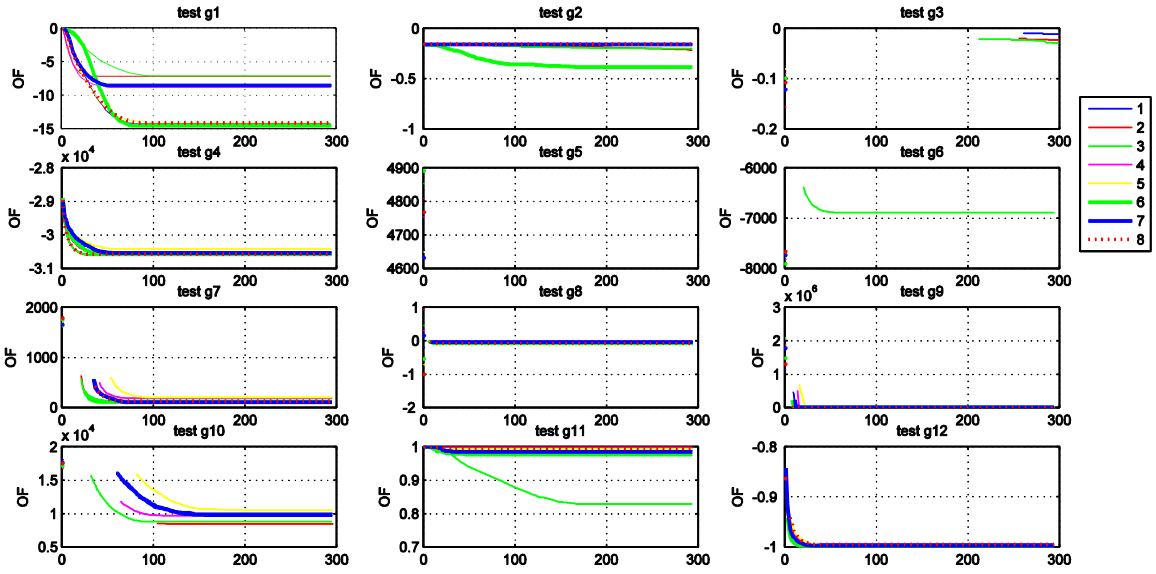

Figure 5: mean values of OF using a standard selection criteria

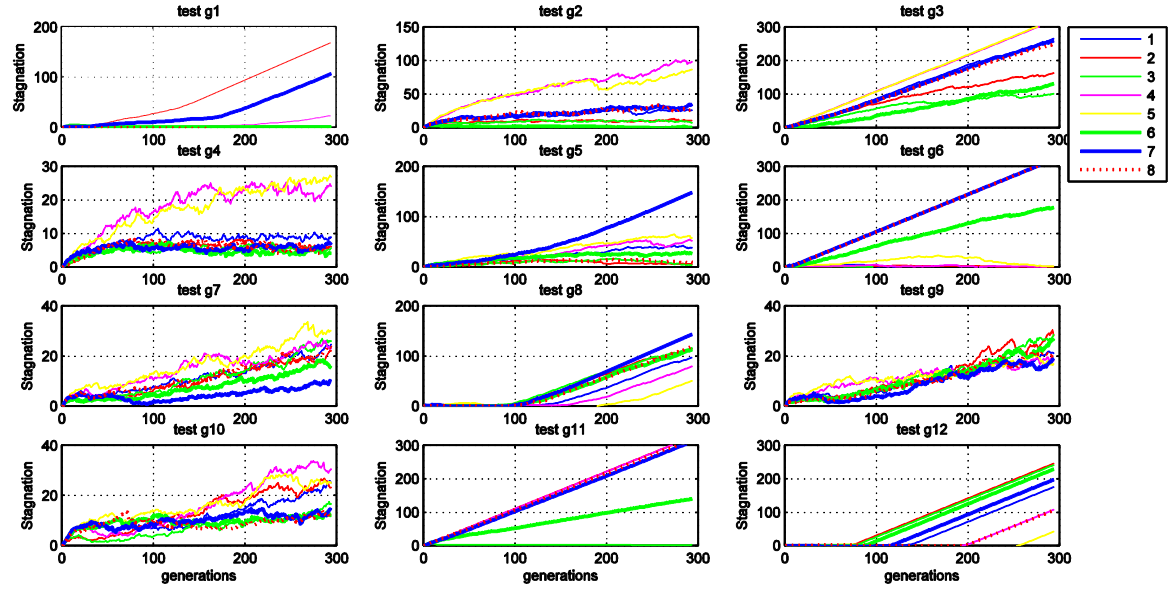

Figure 6: mean values of stagnations using a standard selection criteria

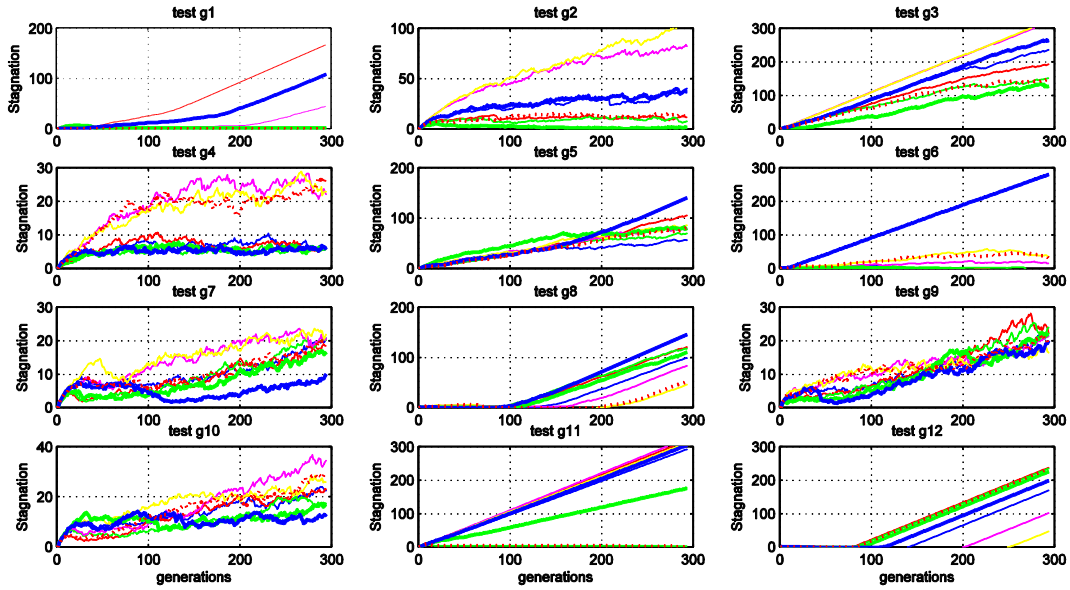

Figura 7 mean values of stagnations using a Normalized selection criteria

A deeper analysis of the two more difficult functions to be optimized - from the feasibility point of view - is thus developed. In details two more selection criteria have been used, in addition to the Standard - eq. (21) - and Normalized - eq. (25) - before considered.
In details the first one is a Normalized modified to take into consideration number of active violations in each selection, that means violation table search

$$ {}^{k}\Phi_{m}{}^{i} = n_a\left(1+\sum_{r=1}^{n_v}{}^{k}\chi_r^i\right) \tag{28}$$

where n_a is the number of active constraints. Finally a complete different criteria is considered, based on the Kreisselmeier–Steinhsauser (KS) function that was first presented by G. Kreisselmeierand R. Steinhauser in [5], that is in the following form:

$$ {}^{k}\Phi_{KS}{}^{i} = \ln\left(\sum_{r=1}^{n_v} e^{{}^{k}\chi_r^i}\right) \tag{29}$$

The selection used for a deep analysis og functions g3 and g5 are thus reported in table

Violation type	Denomination	Equation reference
Standard	A	(25)
Normalized	B	(21)
Normalized modified	C	(28)
Kreisselmeier–Steinhsauser	D	(29)

Table 5: mutation type used in the analysis.

In table 6 and 7 are reported results obtained for the four selection criteria. Are reported the results in terms of best, worst, mean and standard deviation of the OF, evaluated over 2000 generations using 100 independent runs. Results in terms of mean and standard deviation are reported only if all 100 independent runs produce at least a single feasible solution. As reprted results of g5 presents only few positive results. In details considering this function's results (table 5) only mutation *current-to-best/1/bin* type - eq (9) - gives positive results (all indepen-

dent runs produce at least one feasible solution) associated with selection C and D. Moreover in tables 4 and 5 are reported the better four combinations in green, and the worsted four in red. The best one is thus underlined to be well recognizable from the others.
An interesting observation is done in terms of the worst results, that deals directly with robustness of the algorithms. In this since we should notice that the here proposed mutation criteria (eq 12) presents in all analyzed cases the best results, independently from the specific selection criteria used.

mutation type		violation type			
		A	B	C	D
best	1	-0.590085	-0.5166404	-0.5104957	-0.4439761
	2	-0.8749872	-0.8037016	-0.7836352	-0.8001065
	3	-0.8250566	-0.7726869	-0.9282868	-0.8733928
	4	-0.141305	-0.1859609	-0.1059074	-0.1886541
	5	-0.3008128	-0.1389607	-0.0194375	-0.0587828
	6	-0.8889512	-0.9072688	-0.90685	-0.9351505
	7	***-0.945052***	-0.2654813	-0.0521805	-0.9321886
mean	1	-0.0878921	-0.0981476	-0.1108218	-0.1014194
	2	-0.2492862	-0.2529975	-0.2851858	-0.2946302
	3	-0.3248937	-0.3045726	-0.3084063	-0.3298275
	4	-0.0047889	-0.0080551	-0.0023269	-0.0028046
	5	-0.0049531	-0.0019552	-0.0005526	-0.0017174
	6	-0.7156717	-0.7066208	-0.7563377	***-0.773117***
	7	-0.7230272	-0.0065023	-0.0016844	-0.7615129
worst	1	-6.731E-05	-0.0001251	-0.0004741	-1.921E-05
	2	-0.0003809	0	0	-0.0034442
	3	-0.0056007	-0.007803	-0.0073629	-0.0138165
	4	0	0	0	0
	5	0	0	0	0
	6	-0.5242883	-0.4700976	-0.5002015	***-0.536492***
	7	-0.4452317	0	0	-0.425281
std	1	0.0872463	0.1027619	0.1094379	0.0962279
	2	0.2139432	0.1894743	0.2145767	0.2263379
	3	0.215557	0.2118199	0.2321309	0.2112057
	4	0.0182519	0.0306952	0.0113859	0.0191684
	5	0.0313684	0.0141752	***0.0027731***	0.0070985
	6	0.0863034	0.0845136	0.0843159	0.0901844
	7	0.0983988	0.02907	0.0081292	0.0931918

Table 4: Objective Functions results evaluated using 2000 generations for test function g3

	mutation type	violation type			
		A	B	C	D
best	1	5167.1009	5134.2465	5126.4989	5126.4971
	2	5126.4967	5126.4967	***5126.4967***	***5126.4967***
	3	5126.4967	***5126.4967***	***5126.4967***	***5126.4967***
	4				
	5				
	6	5126.4967	5126.4967	5126.4967	5126.4967
	7				
mean	1				
	2				
	3			5484.539	***5422.7768***
	4				
	5				
	6				
	7				
worst	1	5716.0382	5853.421	5696.6796	5560.2758
	2	6112.2237	6112.2238	6112.2237	6112.2237
	3	6111.1807	6076.805	6041.9826	6055.8937
	4				
	5				
	6	5127.7672	5126.5732	5155.8446	***5126.4988***
	7				
std	1				
	2				
	3			357.31161	***345.25123***
	4				
	5				
	6				
	7				

Table 5: Objective Functions results evaluated using 2000 generations for test function g3

7 CONCLUSIONS

With the aim of better explore possibilities of Differential Evolutionary (DE) algorithms in solving structural and seismic optimization problems, a numerical analysis has been conducted over standard test functions with the aim of compare different DE strategies using combinations of selection criteria and cross over. The constrained nature of the problem has been approached by using a number of selection criteria able to take into consideration elements feasibility starting from the original Deb's proposal in this way. Main advantages of this technique is inner simplicity due to absence of additional parameters to be opportunely regulated by tray and error initial procedure. Tree variations of original Deb's one are pro-

posed, using each of them with different cross over criteria. Five are just proposed in literature, while 2 are here proposed to evaluate their performances in accordance with selection criteria. A first numerical analysis has been carried on a reduced number of generations (300) and using a small population size over 12 standard test functions. Algorithms performances in reaching feasible solutions has been evaluated over a set of 100 independent runs,; best mean and worst conditions have been analyzed. Finally a deeper analysis has been evaluate over the two more problematic functions , using a more wide number of possible cross over - selection criteria . Using a greater number of generations (2000), results have been carried on using four selections and seven cross over different criteria (for 28 different DE). Results have indicated that results are strongly influenced by proper selection of those two DE elements, and that a modified Deb's selection criteria shows a general higher performance instead of the original one.

REFERENCES

[1] C.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, 2002, N. 191 (11–12), pp.1245–1287.

[2] N. Cruz-Cortés, Handling constraints in global optimization using artificial immune systems, E. Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Studies in Computational Intelligence Series, 2009, N. 198, Springer-Verlag, ISBN 978-3-642-00618-0, pp. 237–262.

[3] K. Deb, An efficient constraint handling method for genetic algorithms, Computer Methods in Applied Mechanics and Engineering, 2000, N. 186 (2–4), pp. 311–338.

[4] Koziel S., Michalewicz Z., Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, Evolutionary Computation, 1999, N. 7(1), pp. 19–44.

[5] G.KreisselmeierandR.Steinhauser.Systematiccontroldesignbyoptimizingavectorperform ance index.In International Federation of Active Controls Syposium on Computer Aided Design of Control Systems, Zurich, Switzerland, 1979.

[6] E. Mezura-Montes (Ed.), Constraint-Handling in Evolutionary Optimization, Studies in Computational Intelligence, 2009, N. 198, Springer-Verlag.

[7] E. Mezura-Montes, C.A. Coello Coello, Constrained optimization via multiobjective evolutionary algorithms, J. Knowles, D. Corne, K. Deb (Eds.), Multiobjective Problems Solving from Nature: From Concepts to Applications, Natural Computing Series, Springer-Verlag, 2008, ISBN 978-3-540- 72963-1, pp. 53–76.

[8] Monti G, Quaranta G, Marano GC, Genetic-algorithm-based strategies for dynamic identification of nonlinear systems with noise-corrupted response, Journal of Computing in Civil Engineering ASCE, 2009

[9] A. Oyama, K. Shimoyama, K. Fujii, New constraint-handling method for multi-objective and multi-constraint evolutionary optimization, Transactions of the Japan Society for Aeronautical and Space Sciences, 2007, N. 50 (167), pp. 56–62.

[10] Storn R, Price K, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 1997, N. 11(4), pp. 359–431.

[11] K. Price, R. Storn, J. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Natural Computing Series, Springer-Verlag, 2005.

[12] A.E. Smith, D.W. Coit, Constraint handling techniques—penalty functions, T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Oxford University Press, Institute of Physics Publishing, 1997. pp. C 5.2:1–C 5.2:6.