Constraint-Handling in Genetic Algorithms Through the Use of Dominance-based Tournament Selection*

Carlos A. Coello Coello† and Efrén Mezura Montes*
†CINVESTAV-IPN
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300, MEXICO
ccoello@cs.cinvestav.mx

*Maestría en Inteligencia Artificial
Universidad Veracruzana
Sebastián Camacho # 5
Xalapa, Veracruz 91000, MEXICO
emezura@mail.mia.uv.mx

Abstract

In this paper, we propose a dominance-based selection scheme to incorporate constraints into the fitness function of a genetic algorithm used for global optimization. The approach does not require the use of a penalty function and, unlike traditional evolutionary multiobjective optimization techniques, it does not require niching (or any other similar approach) to maintain diversity in the population. The algorithm is validated using several test functions taken from the specialized literature on evolutionary optimization. The results obtained indicate that the approach is a viable alternative to the traditional penalty function, mainly in engineering optimization problems.

Keywords: genetic algorithms, constraint handling, multiobjective optimization, self-adaptation, evolutionary optimization, numerical optimization.

*Technical Report EVOCINV-02-2001, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México, September 2001.

1 Introduction

Genetic algorithms (GAs) have been very successful in a wide variety of optimization problems, presenting several advantages with respect to traditional optimization techniques such as the following [14, 21, 23]: (a) GAs do not require the objective function to be continuous or even to be available in algebraic form, (b) GAs tend to escape more easily from local optima because of its population-based nature, (c) GAs do not require specific domain information although they can exploit it if such information is available, (d) GAs are conceptually simple and relatively easy to implement.

GAs have been successfully applied both to unconstrained and constrained problems [21]. However, despite the considerable number of constraint-handling methods that have been developed GAs in the last few years (see for example [22]), most of them either require a large number of fitness function evaluations, complex encodings or mappings, or are limited to problems with certain (specific) characteristics.

The aim of this work is to show that using concepts from multiobjective optimization [6] is possible to derive new constraint-handling techniques that are not only easy to implement, but also computationally efficient, and competitive with traditional approaches in terms of the quality of results that they produce.

The organization of this paper is the following: Section 2 contains the basic concepts used throughout this paper. Section 3 presents the most relevant previously published related work. Then, our approach is described in Section 4 and validated with the examples of Section 5. Our results are briefly discussed in Section 7 and our conclusions and some paths of future research are provided in Section 8.

2 Basic concepts

The problem that is of particular interest to us in this paper is the general nonlinear optimization problem in which we want to:

$$\text{Find } \vec{x} \text{ which optimizes } f(\vec{x}) \tag{1}$$

subject to:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{3}$$

where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear). Only inequality constraints will be considered in this work, because equality constraints can be transformed into inequalities using:

$$|h_j(\vec{x})| - \epsilon \leq 0 \tag{4}$$

where ϵ is the tolerance allowed (a very small value).

If we denote with $\mathcal{F}$ to the feasible region and with $\mathcal{S}$ to the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$.

For an inequality constaint that satisfies $g_i(\vec{x}) = 0$, then we will say that is active at $\vec{x}$. All equality constraints h_j (regardless of the value of $\vec{x}$ used) are considered active at all points of $\mathcal{F}$.

A point $\vec{x}^* \in \mathcal{F}$ is **Pareto optimal** if for every $\vec{x} \in \mathcal{F}$ and $I = \{1, 2, \ldots, k\}$ either,

$$\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{5}$$

or, there is at least one $i \in I$ such that

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{6}$$

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. The phrase "Pareto optimal" is considered to mean with respect to the entire decision variable space unless otherwise specified.

A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to **dominate** (in the Pareto sense) $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.

3 Related Work

The idea of using evolutionary multiobjective optimization techniques [6] to handle constraints is not entirely new. A few researchers have reported approaches that rely on the use of multiobjective optimization techniques as we will see in this section.

The most common approach is to redefine the single-objective optimization of $f(\vec{x})$ as a multiobjective optimization problem in which we will have $m+1$ objectives, where m is the number of constraints. Then, we can apply any multiobjective optimization technique [12] to the new vector $\vec{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x}))$, where $f_1(\vec{x}), \ldots, f_m(\vec{x})$ are the original constraints of the problem. An ideal solution $\vec{x}$ would thus have $f_i(\vec{x})$=0 for $1 \leq i \leq m$ and $f(\vec{x}) \geq f(\vec{y})$ for all feasible $\vec{y}$ (assuming maximization).

Surry et al. [35, 34] proposed the use of Pareto ranking [11] and the *Vector Evaluated Genetic Algorithm* (VEGA) [31] to handle constraints. In their approach, called COMOGA, the population was ranked based on constraint violations (counting the number of individuals dominated by each solution). Then, one portion of the population was selected based on constraint ranking, and the rest based on real cost (fitness) of the individuals. COMOGA compared fairly with a penalty-based approach in a pipe-sizing problem, since the resulting GA was less sensitive to changes in the parameters, but the results achieved were not better than those found with a penalty function [35]. It should be added that COMOGA [35] required several extra parameters, although its authors argue [35] that the technique is not particularly sensitive to the values of such parameters.

Parmee and Purchase [24] implemented a version of VEGA [31] that handled the constraints of a gas turbine problem as objectives to allow a GA to locate a feasible region within the highly constrained search space of this application. However, VEGA was not used to further explore the feasible region, and instead they opted to use specialized operators that would create a variable-size hypercube around each feasible point to help the GA to remain within the feasible region at all times [24]. This approach was specially developed for a heavily constrained search space and it proved to be appropriate to reach the feasible region. However, this application of a multiobjective optimization technique does not aim at finding the global optimum of the problem, and the use of special operators suggested by the authors certainly limits the applicability of the approach.

Camponogara & Talukdar [4] proposed the use of a procedure based on an evolutionary multiobjective optimization technique. Their proposal was to restate a single objective optimization problem in such a way that two objectives would be considered: the first would be to optimize the original objective function and the second would be to minimize:

$$\Phi(\vec{x}) = \sum_{i=1}^{n} \max[0, g_i(\vec{x})]^{\beta} \tag{7}$$

where β is normally 1 or 2.

Once the problem is redefined, non-dominated solutions with respect to the two new objectives are generated. The solutions found define a search direction $d = (x_i - x_j)/|x_i - xj|$, where $x_i \in S_i$, $x_j \in S_j$, and S_i and S_j are Pareto sets. The direction search d is intended to simultaneously minimize all the objectives [4]. Line search is performed in this direction so that a solution x can be found such that x dominates x_i and x_j (i.e., x is a better compromise than the two previous solutions found). Line search takes the place of crossover in this approach, and mutation is essentially the same, where the direction d is projected onto the axis of one variable j in the solution space [4]. Additionally, a process of eliminating half of the population is applied at regular intervals (only the less fitted solutions are replaced by randomly generated points).

Camponogara & Talukdar's approach [4] has obvious problems to keep diversity (a common problem when using evolutionary multiobjective optimization techniques [6]), as it is indicated by the fact that the technique discards the worst individuals at each generation. Also, the use of line search increases the cost (computationally speaking) of the approach and it is not clear what is the impact of the segment chosen to search in the overall performance of the algorithm.

Jiménez and Verdegay [18] proposed the use of a min-max approach [5] to handle constraints. The main idea of this approach is to apply a set of simple rules to decide the selection process:

1. If the two individuals being compared are both feasible, then select based on the minimum value of the objective function.
2. If one of the two individuals being compared is feasible and the other one is infeasible, then select the feasible individual.
3. If both individuals are infeasible, then select based on the maximum constraint violation ($\max g_j(\vec{x}), \;$ for $j = 1, \ldots, m$, and m is the total number of constraints). The individual with the lowest maximum violation wins.

A subtle problem with this approach is that the evolutionary process first concentrates only on the constraint satisfaction problem and therefore it samples points in the feasible region essentially at random [35]. This means that in some cases (e.g., when the feasible region is disjoint) we might land in an inappropriate part of the feasible region from which we will not be able to escape. However, this approach (as in the case of Parmee and Purchase's [24] technique) may be a good alternative to find a feasible point in a heavily constrained search space.

Coello [7] proposed the use of a population-based multiobjective optimization technique similar to VEGA [31] to handle each of the constraints of a single-objective optimization problem as an objective. At each generation, the population is split into $m+1$ sub-populations (m is the number of constraints), so that a fraction of the population is selected using the (unconstrained) objective function as its fitness and another fraction uses the first constraint as its fitness and so on.

For the sub-population guided by the objective function, the evaluation of such objective function for a given vector $\vec{x}$ is used directly as the fitness function (multiplied by (-1) if it is a minimization problem), with no penalties of any sort. For all the other sub-populations, the algorithm used was the following [7]:

$$
\begin{array}{lll}
\textbf{if } g_j(\vec{x}) < 0.0 & \textbf{then} & \text{fitness} = g_j(\vec{x}) \\
\textbf{else if } v \neq 0 & \textbf{then} & \text{fitness} = -v \\
\textbf{else} & & \text{fitness} = f(\vec{x})
\end{array}
$$

where $g_j(\vec{x})$ refers to the constraint corresponding to sub-population $j+1$ (this is assuming that the first sub-population is assigned to the original objective function $f(\vec{x})$), and v refers to the number of constraints that are violated ($\leq m$).

This approach provided good results in several optimization problems, but required a relatively large number of fitness function evaluations to converge [7].

Ray et al. [28] proposed an approach in which solutions were ranked separately based on the value of their objective functions and their constraints. Then, a set of mating restrictions were applied based on the information that each individual had of its own feasibility (this idea was inspired on an earlier approach by Hinterding and Michalewicz [16]), so that the global optimum could be reached through cooperative learning. This approach seems to be very efficient, but it has as a drawback the fact that it requires a lot of extra knowledge about the problem at hand in order to be effective. Although this knowledge is extracted directly from the problem itself, the approach is certainly far more complicated than using straightforward dominance relationships, and its implementation seems to be cumbersome.

The limitations of the previously reported multiobjective optimization techniques used to handle constraints were the main motivation of this work.

4 The Proposed Approach

The concept of nondominated vector is used in multiobjective optimization to denote solutions that represent the best possible compromise, given a set of objective functions. None of the objective function values of these nondominated vectors can be improved without worsening another one (see [6] for details). Our hypothesis is that this concept can be used to extend evolutionary multiobjective optimization techniques to be used as single-objective optimization approaches in which the constraints are handled as additional objectives. Although this sort of approach can be quite useful to reach the feasible region in highly constrained search spaces, is not straightforward to extend it to solve single-objective optimization problems. The main difficulty is that we could bias the search towards a certain specific portion of the feasible region and, as a consequence, we could be unable to reach the global optimum. This paper presents a proposal based on a technique known as the Niched-Pareto Genetic Algorithm (NPGA) [17] that uses tournament selection decided through nondominance. In the original proposal of the NPGA, the idea was to use a sample of the population to determine who is the winner between two candidate solutions to be selected, and to choose one of them based on nondominance with respect to the sample taken. Since only a portion of the population is used with this technique, it has a lower computational complexity with respect to traditional Pareto ranking approaches (the most common evolutionary multiobjective optimization technique) [6].

To adapt the NPGA to solve single-objective constrained optimization problems, we performed the following changes:

- The tournament performed is not completely deterministic. We use a parameter called selection ratio (S_r), which indicates the minimum number of individuals that will not be selected through tournament selection. These individuals will be selected using a probabilistic procedure. This means that $(1 - Sr)$ individuals in the population are probabilistically selected.

- When comparing two individuals, we can have four possible situations:

  1. **Both are feasible**. In this case, the individual with a better fitness value wins.
  2. **One is infeasible, and the other is feasible**. The feasible individual wins, regardless of its fitness function value.

3. **Both are infeasible**. The nondominated individual is selected, only if the other candidate is dominated.
4. **Both are infeasible and both are either nondominated or dominated**. The individual with the lowest amount of constraint violation wins, regardless of its fitness function value.

- Our approach does not require niching or any other similar approach to keep diversity, since the value of S_r will control the diversity of the population. For the experiments reported in this paper, a value close to one (≥ 0.8) was adopted for S_r.

The pseudocode of our approach is presented below. The following notation is adopted: *oldpop* is the current population, t_{dom} is the size of the comparison set and $flip(P)$ is a function that returns TRUE with probability P:

```
function select
  begin
    shuffle(tournlist);
    candidate_1 = tournlist[0];
    candidate_2 = tournlist[1];
    if (flip(Sr)) /* fitness-feasibility-nondominance based tournament */
      begin
        candidate_1_dominated = FALSE;
        candidate_2_dominated = FALSE;
        if (oldpop[candidate_1]=feasible AND oldpop[candidate_2]=feasible)
        /* fitness checking */
          if (oldpop[candidate_1].fitness ≥ oldpop[candidate_2].fitness)
            winner=candidate_1;
          else
            winner=candidate_2;
        else/* feasibility checking */
          if (oldpop[candidate_1]=feasible AND oldpop[candidate_2]=nonfeasible)
            winner=candidate_1;
          else
            if (oldpop[candidate_1]=nonfeasible AND oldpop[candidate_2]=feasible)
              winner=candidate_2;
            else
              begin /* nondominance checking */
                for (i = 2 to tdom + 2)
                  begin
                    comparison_individual=tournlist[i];
                    if (oldpop[comparison_individual] dominates oldpop[candidate_1])
                      candidate_1_dominated=TRUE;
                    if (oldpop[comparison_individual] dominates oldpop[candidate_2])
                      candidate_2_dominated=TRUE;
                  end
```

```
                    if (candidate_1_dominated=TRUE AND candidate_2_dominated=FALSE)
                       winner=candidate_2;
                    else
                       if (candidate_1_dominated=FALSE AND candidate_2_dominated=TRUE)
                          winner=candidate_1;
                       else /* tie break with accumulated constraint violation */
                          if (oldpop[candidate_1].sumviol < oldpop[candidate_2].sumviol)
                             winner=candidate_1;
                          else
                             winner=candidate_2;
                 end
         end
      else
         if (flip(0.5))
            winner=candidate_1;
         else
            winner=candidate_2;
      return(winner);
   end
```

The way in which our algorithm works is described next. First, the algorithm tries to reach the feasible region of the search space in two ways: finding non-dominated solutions and choosing those with a lower accumulation of constraint violation. When the number of feasible solutions in the population increases, those with a better fitness value will be preferred, because the feasible region is sufficently sampled and the aim then becomes to reach the global optimum.

During all this process, some individuals are probabilistically selected. These individuals can be either infeasible or dominated. The reason for this is to avoid that the GA stagnates and prematurely converges to a local optimum. This mechanism is thus responsible for keeping the diversity required in the population to ensure that the search progresses.

A significant difference of our approach with respect to the NPGA is the way in which dominance between vectors is checked. In order to move the search towards the feasible region of the problem, we eliminate the value of the fitness function of the dominance checking. In other words, we only check dominance between the values of the constraints. This reduces the computational cost of the approach and helps the GA to move efficiently towards the feasible region.

In the following experiments, we use a GA with binary representation, two-point crossover, and uniform mutation. The parameters used for our GA are the following: population size = 200 individuals, maximum number of generations = 400, crossover rate = 0.6, mutation rate = 0.03, S_r = 0.99 (i.e., one out of every one hunded selections will be done probabilistically, rather than in a deterministic way), tournament size = 10.

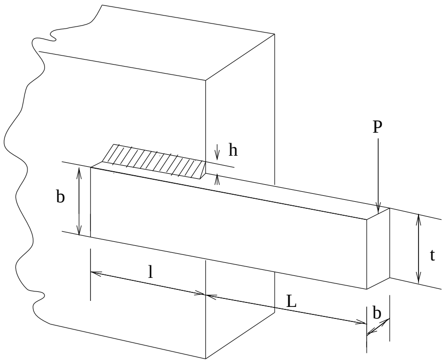

Figure 1: The welded beam used for the first example.

5 Examples

Several examples taken from the optimization literature will be used to show the way in which the proposed approach works. These examples have linear and nonlinear constraints, and have been previously solved using a variety of other techniques (both GA-based and traditional mathematical programming methods), which is useful to determine the quality of the solutions produced by the proposed approach.

5.1 Example 1 : Welded Beam Design

A welded beam is designed for minimum cost subject to constraints on shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints [27]. There are four design variables as shown in Figure 1 [27]: h (x_1), l (x_2), t (x_3) and b (x_4).

The problem can be stated as follows:

Minimize:

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \tag{8}$$

Subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0 \tag{9}$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0 \tag{10}$$

$$g_3(\vec{x}) = x_1 - x_4 \leq 0 \tag{11}$$

$$g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \tag{12}$$

$$g_5(\vec{x}) = 0.125 - x_1 \leq 0 \tag{13}$$

$$g_6(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0 \tag{14}$$

$$g_7(\vec{x}) = P - P_c(\vec{x}) \leq 0 \tag{15}$$

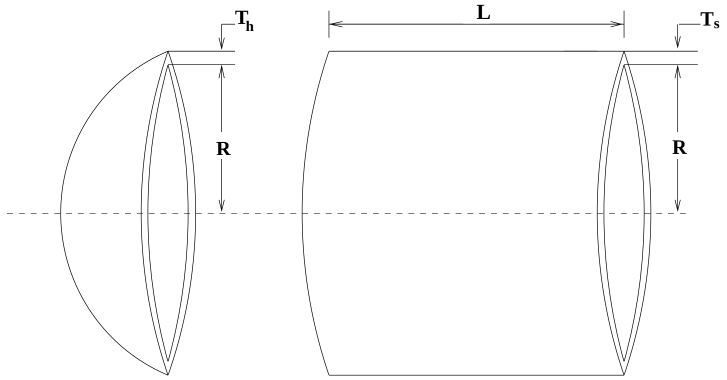

Figure 2: Center and end section of the pressure vessel used for the second example.

where

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \tag{16}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right) \tag{17}$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{18}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \tag{19}$$

$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4} \tag{20}$$

$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{21}$$

$$P = 6000\ lb,\quad L = 14\ in,\quad E = 30 \times 10^6\ psi,\quad G = 12 \times 10^6\ psi \tag{22}$$

$$\tau_{max} = 13,600\ psi,\quad \sigma_{max} = 30,000\ psi,\quad \delta_{max} = 0.25\ in \tag{23}$$

5.2 Example 2 : Design of a Pressure Vessel

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 2. The objective is to minimize the total cost, including the cost of the material, forming and welding. There are four design variables: T_s (thickness of the shell), T_h (thickness of the head), R (inner radius) and L (length of the

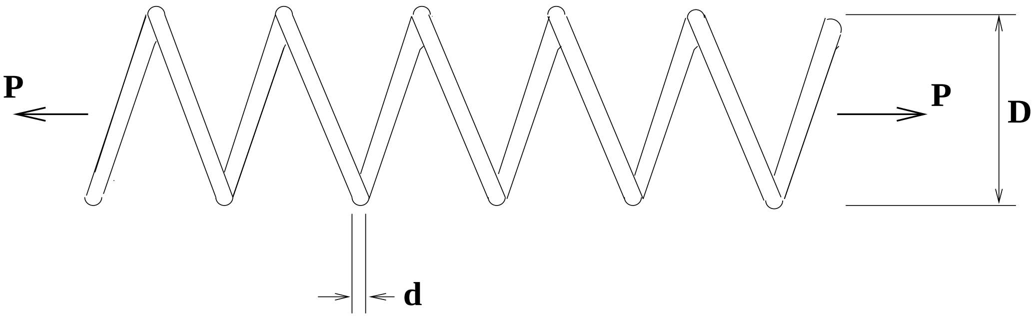

Figure 3: Tension/compression string used for the third example.

cylindrical section of the vessel, not including the head). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and R and L are continuous. Using the same notation given by Kannan and Kramer [19], the problem can be stated as follows:

Minimize :

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \qquad (24)$$

Subject to :

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0 \qquad (25)$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0 \qquad (26)$$

$$g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \leq 0 \qquad (27)$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0 \qquad (28)$$

5.3 Example 3: Minimization of the Weight of a Tension/Compression String

This problem was described by Arora [1] and Belegundu [2], and it consists of minimizing the weight of a tension/compression spring (see Figure 3) subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and on design variables. The design variables are the mean coil diameter D (x_2), the wire diameter d (x_1) and the number of active coils N (x_3).

Formally, the problem can be expressed as:

$$\text{Minimize} \quad (x_3 + 2)x_2x_1^2 \qquad (29)$$

Subject to

$$g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0 \qquad (30)$$

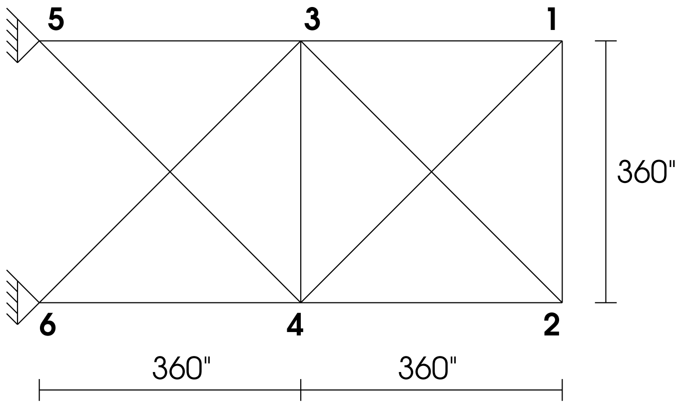

Figure 4: 10-bar plane truss used for the fifth example.

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{31}$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{32}$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0 \tag{33}$$

5.4 Example 4: Disjoint Feasible Region

This problem was proposed by Michalewicz & Schoenauer [22], and its search space consists of 9^3 disjoint spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that below inequality holds. The optimum is located at $x^* = (5, 5, 5)$ where $f(x^*) = 1$. The solution lies within the feasible region.

$$\text{Maximize} \quad f(\vec{x}) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \tag{34}$$

Subject to

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0 \tag{35}$$

where $0 \leq 10$ $i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$.

5.5 Example 5 : Design of a 10-bar plane truss

Consider the 10-bar plane truss shown in Figure 4 [2]. The problem is to find the moment of inertia of each member of this truss, such that we minimize its weight, subject to stress and displacement constraints. The weight of the truss is given by:

$$f(x) = \sum_{j=1}^{10} \rho \, A_j \, L_j \tag{36}$$

where x is the candidate solution, A_j is the cross-sectional area of the jth member ($A_j = \sqrt{I_j}$, where I is the moment of inertia of member j), L_j is the length of the jth member, and ρ is the weight density of the material.

The assumed data are: modulus of elasticity, $E = 1.0 \times 10^4$ ksi 68965.5 MPa), $\rho = 0.10$ lb/in^3 (2768.096 kg/m^3), and a load of 100 kips (45351.47 Kg) in the negative y-direction is applied at nodes 2 and 4. The maximum allowable stress of each member is called σ_a, and it is assumed to be ± 25 ksi (172.41 MPa). The maximum allowable displacement of each node (horizontal and vertical) is represented by u_a, and is assumed to be 2 inches (5.08 cm).

There are 10 stress constraints, and 12 displacement constraints (we can really assume only 8 displacement constraints because there are two nodes with zero displacement, but they will nevertheless be considered as additional constraints by the new approach). The moment of inertia of each element can be different, thus the problem has 10 design variables.

6 Comparison of Results

6.1 Example 1

This problem has been solved by Deb [8] using a simple genetic algorithm with binary representation, and a traditional penalty function as suggested by Goldberg [14]. It has also been solved by Ragsdell and Phillips [26] using geometric programming. Ragsdell and Phillips also compared their results with those produced by the methods contained in a software package called "Opti-Sep" [33], which includes the following numerical optimization techniques: ADRANS (Gall's adaptive random search with a penalty function), APPROX (Griffith and Stewart's successive linear approximation), DAVID (Davidon-Fletcher-Powell with a penalty function), MEMGRD (Miele's memory gradient with a penalty function), SEEK1 & SEEK2 (Hooke and Jeeves with 2 different penalty functions), SIMPLX (Simplex method with a penalty function) and RANDOM (Richardson's random method).

Their results are compared against those produced by the approach proposed in this paper, which are shown in Table 1. In the case of Siddall's techniques [33], only the best solution produced by the techniques contained in "Opti-Sep" is displayed. The solution shown for the technique proposed here is the best produced after 30 runs, and using the following ranges for the design variables: $0.1 \le x_1 \le 2.0$, $0.1 \le x_2 \le 10.0$, $0.1 \le x_3 \le 10.0$, $0.1 \le x_4 \le 2.0$.

The mean from the 30 runs performed was $f(\vec{x}) = 1.792654$, with a standard deviation of 0.074713. The worst solution found was $f(\vec{x}) = 1.993408$, which

Design Variables	**Best solution found**			
	This paper	**Deb [8]**	**Siddall [33]**	**Ragsdell [26]**
$x_1(h)$	0.205986	0.2489	0.2444	0.2455
$x_2(l)$	3.471328	6.1730	6.2189	6.1960
$x_3(t)$	9.020224	8.1789	8.2915	8.2730
$x_4(b)$	0.206480	-0.2533	0.2444	0.2455
$g_1(\vec{x})$	-0.074092	-5758.603777	-5743.502027	-5743.826517
$g_2(\vec{x})$	-0.266227	-255.576901	-4.015209	-4.715097
$g_3(\vec{x})$	-0.000495	-0.004400	0.000000	0.000000
$g_4(\vec{x})$	-3.430043	-2.982866	-3.022561	-3.020289
$g_5(\vec{x})$	-0.080986	-0.123900	-0.119400	-0.120500
$g_6(\vec{x})$	-0.235514	-0.234160	-0.234243	-0.234208
$g_7(\vec{x})$	-58.666440	-4465.270928	-3490.469418	-3604.275002
$f(\vec{x})$	**1.728226**	**2.43311600**	**2.38154338**	**2.38593732**

Table 1: Comparison of the results for the first example (optimal design of a welded beam).

is better than any of the solutions produced by any of the other techniques depicted in Table 1. The number of fitness function evaluations of our approach was 80000.

6.2 Example 2

This problem has been solved by Deb [9] using GeneAS (Genetic Adaptive Search), by Kannan and Kramer using an augmented Lagrangian Multiplier approach [19], and by Sandgren [30] using a branch and bound technique. Their results were compared against those produced by the approach proposed in this paper, and are shown in Table 2. The solution shown for the technique proposed here is the best produced after 30 runs, and using the following ranges for the design variables: $1 \leq x_1 \leq 99$, $1 \leq x_2 \leq 99$, $10.0 \leq x_3 \leq 200.0$, $10.0 \leq x_4 \leq 200.0$. The values for x_1 and x_2 were considered as integer (i.e., real values were rounded up to their closest integer value) multiples of 0.0625, and the values of x_3 and x_4 were considered as real numbers.

The mean from the 30 runs performed was $f(\vec{x}) = 6177.253268$, with a standard deviation of 130.929702. The worst solution found was $f(\vec{x}) = 6469.322010$. We can see that in this case, our average solution was better than any of the solutions produced by any of the other techniques depicted in Table 2. The total number of fitness function evaluations performed was 80000.

6.3 Example 3

This problem has been solved by Belegundu [2] using eight numerical optimization techniques: CONMIN (a feasible directions method developed by Van-

Design Variables	Best solution found			
	This paper	**GeneAS [9]**	**Kannan [19]**	**Sandgren [30]**
$x_1(T_s)$	0.812500	0.9375	1.125	1.125
$x_2(T_h)$	0.437500	0.5000	0.625	0.625
$x_3(R)$	42.097398	48.3290	58.291	47.700
$x_4(L)$	176.654047	112.6790	43.690	117.701
$g_1(\vec{x})$	-0.000020	-0.004750	0.000016	-0.204390
$g_2(\vec{x})$	-0.035891	-0.038941	-0.068904	-0.169942
$g_3(\vec{x})$	-27.886075	-3652.876838	-21.220104	54.226012
$g_4(\vec{x})$	-63.345953	-127.321000	-196.310000	-122.299000
$f(\vec{x})$	**6059.946341**	**6410.3811**	**7198.0428**	**8129.1036**

Table 2: Comparison of the results for the second example (optimization of a pressure vessel).

derplaats [36]), OPTDYN (a feasible directions method developed by Bhatti and Pollack [3],LINMR, GRP-UI (a gradient projection technique developed by Haug and Arora [15]), SUMT (an exterior penalty approach implemented by Belegundu [2]), M-3 (a Lagrange multipliers code based on Powell's algorithm [25] which was implemented by Belegundu [2]), M-4 (a variation of M-3 implemented by Belegundu [2]), and M-5 (a Lagrange multipliers code based on Fletcher's method [10] which was implemented by Belegundu [2]). Additionally, Arora [1] also solved this problem using a numerical optimization technique called Constraint Correction at constant Cost (CCC).

In Belegundu's experiments, GRP-UI and CONMIN failed to solve this problem. Therefore, these two techniques were not included in the results presented in Tables 3 where the best result of our approach is compared.

The mean value from this problem after 30 runs was $f(\vec{x}) = 0.012742$. The worst solution found was $f(\vec{x}) = 0.012973$ with a standard deviation of 0.000059. The number of fitness function evaluations for each run was 80000. The mean value found is even better than the best solutions reported by Arora [1] which is infeasible [2].

6.4 Example 4

This problem has been solved by Runarsson and Yao using an evolution strategy [32] with Stochastic Ranking [29] and by Koziel and Michalewicz using a GA with a technique called Homomorphous Maps [20]. The values of the decision variables lie within the following ranges: $0 \leq x_i \leq 10$ $i = 1, 2, 3$.

After 30 runs performed, each one with 80000 fitness function evaluations, the mean was $f(\vec{x}) = 1.000000$, the worst solution also has a value of $f(\vec{x}) = 1.000000$ with a standard deviation of 0.000000. We can observe a very robust behavior of the proposed technique in this example. The best solution of our approach is compared in Table 4. Note that the approach proposed by

Design Variables	Best solution found			
	This paper	**Arora* [1]**	**M-5 [2]**	**OPTDYN* [2]**
$x_1(d)$	0.051989	0.053396	0.050000	0.0644
$x_2(D)$	0.363965	0.399180	0.315900	0.7488
$x_3(N)$	10.890522	9.185400	14.25000	2.9597
$g_1(\vec{x})$	-0.000013	0.000019	-0.000014	-0.005134
$g_2(\vec{x})$	-0.000021	-0.000018	-0.003782	0.002609
$g_3(\vec{x})$	-4.061338	-4.123832	-3.938302	-4.450398
$g_4(\vec{x})$	-0.722698	-0.698283	-0.756067	-0.457867
$f(\vec{x})$	**0.012681**	**0.01273027**	0.01283343	**0.01540256**

Table 3: Comparison of the results for the third problem (minimization of the weight of a tension/compression spring). * Infeasible solution.

Design Variables	Best solution found		
	This paper	**RY [29]**	**KM [20]**
$x_1(d)$	5.000000	5.000000	N.A.
$x_2(D)$	5.000000	5.000000	N.A.
$x_3(N)$	5.000000	5.000000	N.A.
$g(\vec{x})$	0.000000	0.000000	N.A.
$f(\vec{x})$	**1.000000**	**1.000000**	**0.999999857**

Table 4: Comparison of the results for the fourth problem (disjoint feasible region). RY = Runarsson & Yao [29], KM = Koziel & Michalewicz [20]. N.A. = Not Available.

Koziel & Michalewicz [20] required 1,400,000 evaluations of the fitness function to produce the result shown in Table 4. The approach of Runarsson & Yao [29] required 350,000 evaluations of the fitness function. In contrast, our approach only required 80,000 evaluations of the fitness function and it produced equiparable results.

6.5 Example 5

This problem was used by Belegundu [2] to evaluate the following numerical optimization techniques: Feasible directions (CONMIN and OPTDYN), Pshenichny's Recursive Quadratic Programming (LINRM), Gradient Projection (GRP-UI), Exterior Penalty Function (SUMT), Multiplier Methods (M-3, M-4 and M-5).

The results reported by Belegundu [2] are compared to the current approach in Tables 5 and 6 (all the solutions presented are feasible). To solve this problem, it was necessary to add a module responsible for the analysis of the plane truss. This module uses the matrix factorization method included in Gere and Weaver

Design Variables	Best solution found			
	This paper	**CONMIN**	**OPTDYN**	**LINRM**
x_1	985.808351	639.20	664.30	21.57
x_2	0.105877	3.60	0.01	10.98
x_3	519.966658	618.40	630.70	22.08
x_4	188.576078	250.50	375.90	14.95
x_5	0.102124	0.01	0.01	0.10
x_6	0.137725	3.05	0.01	10.98
x_7	690.171450	280.80	235.90	18.91
x_8	495.366009	389.20	413.00	18.42
x_9	467.438050	440.10	430.30	18.40
x_{10}	0.135133	6.30	1.30	13.51
$f(\vec{x})$	**5157.685516**	**5563.32**	**5471.48**	**6428.89**

Table 5: Comparison of results for the fifth example (10-bar plane truss). The value of all variables is given in in^4. Part I.

[13] together with the stiffness method [13] to analyze the structure, and returns the values of the stress and displacement constraints, as well as the total weight of the structure.

The solution shown for the technique proposed here is the best produced after 30 runs. The range $0.1 \leq x \leq 999.0$ was used for the 10 design variables (moments of inertia were used as the design variables, and their square roots were found in order to obtain the cross-sectional areas of each truss member).

The mean from the 30 runs performed was $f(\vec{x}) = 5198.085918$, with a standard deviation of 29.496938. The worst solution found was $f(\vec{x}) = 5274.693439$, which is better than any of the solutions produced by any of the other techniques depicted in Tables 5 and 6.

7 Discussion of Results

There are a few things about our approach that deserve to be mentioned. First, we have empirically shown the feasibility of using a multiobjective optimization technique to handle constraints. This sort of technique has as its main advantage its computational efficiency. Furthermore, we have shown that it is highly competitive and was even able to match (or even improve) the results produced by other algorithms, some of which are more complex constraint-handling techniques used with GAs.

The parameter S_r plays a crucial role in our approach, since it is responsible for providing the diversity needed. We monitored the population of our GA as the search progressed, and we found out that about 50% of the total population was feasible by the last generations in all the test functions adopted. This is a clear indicative of the efficacy of the approach to maintain diversity.

Design Variables	Best solution found				
	GRP-UI	**SUMT**	**M-3**	**M-4**	**M-5**
x_1	614.30	942.00	667.90	1000.0	667.70
x_2	17.40	5.60	9.40	139.40	8.30
x_3	614.40	1000.0	697.80	1000.0	699.40
x_4	208.80	135.90	163.10	306.40	162.60
x_5	0.01	0.01	0.01	1000.0	0.01
x_6	17.40	13.80	11.80	105.00	14.20
x_7	304.80	471.20	373.90	1000.00	375.50
x_8	370.90	467.00	367.60	1000.00	368.00
x_9	371.30	195.30	351.90	1000.00	352.20
x_{10}	27.70	10.60	19.50	1000.00	19.20
$f(\vec{x})$	**5727.05**	**5932.21**	**5719.19**	**11279.22**	**5726.08**

Table 6: Comparison of results for the fifth example (10-bar plane truss). The value of all variables is given in in^4. Part II.

It is worth reminding that traditional multiobjective optimization techniques normally cannot be used directly to handle constraints because their emphasis is to drive the GA towards the feasible region, but not necessarily to the global optimum.

The addition of a new parameter (S_r) may be debatable. However, at least in the test functions that we have used so far, the algorithm does not require a fine tuning of this parameter. If a value closer to 1 is used for S_r ($0.7 < S_r < 1.0$), the selection process allows the GA to reach and sample sufficiently well the feasible region of a problem as to reach the vicinity of the global optimum. However, if the problem turns out to be very difficult (i.e., a highly constrained search space), we suggest to use a lower value for S_r as those indicated before. This is to avoid premature convergence of the GA.

8 Conclusions and Future Work

This paper has introduced a new constraint-handling approach that is based on a multiobjective optimization technique called NPGA [17]. The approach is intended to be used with evolutionary algorithms as a way to reduce the burden normally associated with the fine-tuning of a penalty function.

The proposed approach performed well in several test problems both in terms of the number of fitness function evaluations required and in terms of the quality of the solutions found. The results produced were compared against those generated with other (evolutionary and mathematical programming) techniques reported in the literature.

As part of our future work, we are analyzing some possible improvements to this approach such as the elimination of the parameter S_r. Additionally, we are considering the extension of other multiobjective optimization techniques to handle constraints in EAs. Finally, we also have interest in using techniques such as the one proposed in this paper coupled to an evolutionary multiobjective optimization algorithm.

9 Acknowledgments

The first author acknowledges support from CONACyT through the NSF-CONACyT project number 32999 A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Maestría en Inteligencia Artificial of the Universidad Veracruzana.

References

[1] Jasbir S. Arora. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[2] Ashok Dhondu Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. PhD thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa, 1982.

[3] M. A. Bhatti, E. Polak, and K. S. Pister. OPTDYN—A General Purpose Optimization Program for Problems With or Without Dynamic Constraints. Technical Report UCB/EERC-79/16, University of California, Berkeley, 1979.

[4] Eduardo Camponogara and Sarosh N. Talukdar. A Genetic Algorithm for Constrained and Multiobjective Optimization. In Jarmo T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.

[5] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making: Theory and Methodology*. Systems Science and Engineering. North-Holland, 1983.

[6] Carlos A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.

[7] Carlos A. Coello Coello. Treating Constraints as Objectives for Single-Objective Evolutionary Optimization. *Engineering Optimization*, 32(3):275–308, 2000.

[8] Kalyanmoy Deb. Optimal Design of a Welded Beam via Genetic Algorithms. *AIAA Journal*, 29(11):2013–2015, November 1991.

[9] Kalyanmoy Deb. GeneAS: A Robust Optimal Design Technique for Mechanical Component Design. In Dipankar Dasgupta and Zbigniew

Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 497–514. Springer-Verlag, Berlin, 1997.

[10] R. Fletcher. An Ideal Penalty Function for Constrained Optimization. *J. Inst. Maths Applics.*, 15:319–342, 1975.

[11] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[12] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[13] James M. Gere and William Weaver. *Analysis of Framed Structures.* D. Van Nostrand Company, Inc., 1965.

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

[15] E. J. Haug and J. S. Arora. *Applied Optimal Design: Mechanical and Structural Systems.* Wiley-Interscience, New York, 1979.

[16] Robert Hinterding and Zbigniew Michalewicz. Your Brains and My Beauty: Parent Matching for Constrained Optimisation. In *Proceedings of the 5th International Conference on Evolutionary Computation*, pages 810–815, Anchorage, Alaska, May 1998.

[17] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[18] Fernando Jiménez and José L. Verdegay. Evolutionary techniques for constrained optimization problems. In *7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany, 1999. Springer-Verlag.

[19] B. K. Kannan and S. N. Kramer. An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and Its Applications to Mechanical Design. *Journal of Mechanical Design. Transactions of the ASME*, 116:318–320, 1994.

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

[21] Zbigniew Michalewicz, Dipankar Dasgupta, R. Le Riche, and Marc Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(4):851–870, September 1996.

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

[23] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, Massachusetts, 1996.

[24] I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth.

[25] M. J. D. Powell. Algorithms for Nonlinear Constraints that use Lagrangian Functions. *Mathematical Programming*, 14:224–248, 1978.

[26] K. M. Ragsdell and D. T. Phillips. Optimal Design of a Class of Welded Structures Using Geometric Programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976. Series B.

[27] Singiresu S. Rao. *Engineering Optimization.* John Wiley and Sons, third edition, 1996.

[28] Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.

[29] Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[30] E. Sandgren. Nonlinear integer and discrete programming in mechanical design. In *Proceedings of the ASME Design Technology Conference*, pages 95–105, Kissimine, Florida, 1988.

[31] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[32] Hans-Paul Schwefel. *Numerical Optimization of Computer Models.* John Wiley & Sons, Great Britain, 1981.

[33] James N. Siddall. *Analytical Design-Making in Engineering Design.* Prentice-Hall, 1972.

[34] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3), 1997.

[35] Patrick D. Surry, Nicholas J. Radcliffe, and Ian D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, Lecture Notes in Computer Science, pages 166–180. Springer-Verlag, Sheffield, U.K., 1995.

[36] G. N. Vanderplaats and F. Moses. Structural Optimization by Methods of Feasible Directions. *Computers and Structures*, 3:739–755, 1973.