On the use of Projected Gradients for Constrained Multiobjective Optimization Problems

Alfredo G. Hernandez-Diaz[a], Carlos A. Coello[b], Luis V. Santana-Quintero[b], Fatima Perez[c], Julian Molina[c], and Rafael Caballero[c]

[a] Department of Economics, Quantitative Methods and Economic History, Pablo de Olavide University, Seville (Spain), agarher@upo.es
[b] Centro de Investigacion y de Estudios Avanzados, Mexico D.F., (Mexico), ccoello@cs.cinvestav.mx, lvspenny@hotmail.com
[c] Department of Applied Economics(Mathematics), University of Malaga, Malaga (Spain), fperez@uma.es, julian.molina@uma.es, rafael.caballero@uma.es

Abstract. Recent works have shown how hybrid variants of gradient-based methods and evolutionary algorithms perform better than the pure evolutionary method both for single-objective and multiobjective optimization. This same idea has been used with Evolutionary Multiobjective Optimization (**EMO**), obtaining also very promising results. In most of the cases, gradient information is used as part of the mutation operator (and only for unconstrained MOPs), in order to move every generated point to the exact Pareto front. In our approach, we use the Karush-Kuhn-Tucker optimality condition for constrained optimization problems to combine the information provided by the gradient vector of each objective function and the gradient vectors of constraint functions to obtain a feasible movement direction in those points near the border. In our approach, gradients of the objective functions will be approximated using quadratic regressions, trying to avoid local optima. The proposed algorithm is able to converge on several nonlinear constrained multiobjective optimization problems obtained from a benchmark, consuming few objective function evaluations (between 150 and 1000). Our results indicate that our proposed scheme may produce a significant reduction in the computational cost, while producing results of good quality, if it is incorporated in a hybrid MOEA or if it is used to seed an EMO algorithm.

Key words: Gradient-based method, constrained optimization, nonlinear multiobjective programming, quadratic approximation

1 Introduction

MOEAs have been very successful in the solution of a wide variety of problems, mainly during the last few years [2]. However, for certain types of applications, MOEAs result particularly expensive (computationally speaking), since they require a large number of objective function evaluations in order to produce an

acceptable approximation of the true Pareto front, specially for constrained problems, where a suitable constraint-handling mechanism is needed.

On the other hand, the classical (exact) methods for (multi-objective) optimization (gradient based methods) consume just a few number of evaluations, but can be trapped in local optima and require a lot of assumptions about the problem: continuity, differentiability, explicit mathematical formulation, etc.

Besides, under proper assumptions, Newton's method is quadratically convergent, but its efficiency is reduced by its expensive computational cost, especially, for the middle-large scale problems. The key point is to evaluate the gradient and the Hessian efficiently, and two different approaches can be found:

- **Use analytical derivatives** The first option is manually obtaining analytic derivatives of each objective and constraint functions. But this is only possible if an explicit mathematical formulation is available, and this is the main weakness of this approach as many interesting problems could not be solve: simulation based problems, design problems, etc. On the other hand, it is an error-prone activity, because if the formulation is complicated, obtaining analytical derivatives can be a hard task.
- **Use estimated derivatives** In this category we can find the Newton-like methods, where derivatives are estimated in some efficient way. These methods do not require explicit formulae of derivatives but, on the other hand, consume some more evaluations in order to compute the estimation.

On the other hand, the use of gradient information for constrained optimization problem has been widely use for many years. Techniques such that Barrier Methods and Penalty functions (see [3]), Interior-Point Methods ([7]) or Projected Gradient ([5], [8]) have been successfully used in continuous optimization.

Barrier and penalty methods are designed to face the problem by solving a sequence of specially constructed unconstrained optimization problems. In a penalty method, the feasible region is expanded from the feasible region to all of $\mathbb{R}^n$, but a large cost or "penalty" is added to the objective function for points that lie outside of the original feasible region. In a barrier method, we assume we are given a starting point in the interior of the feasible region, and we impose a very large cost on feasible points that lie ever closer to the boundary of feasible region, thereby creating a "barrier" to exiting the feasible region. Interior-point methods moves through the interior of the feasible region following the gradient vector of the objective function generating approximated solutions that asymptotically converge to the exact solution, while projected gradient orthogonally projects new generated unfeasible solutions over the feasible region.

In this work we propose an easy-to-implement method to iteratively generate nondominated solutions for constrained multiobjective optimization problems. In this method we **use "global" estimated derivatives** for the objective functions (and **analytical derivatives** for constraint functions) but consuming the less evaluations as possible while maintaining a high quality on the results. We propose its use to seed and EMO method instead of using it along the whole process (consuming too many evaluations).

2 Definitions and basic concepts

We assume the following definition of a constrained MOP problem[1]:

$$\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) := (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_s(\boldsymbol{x})) \tag{1}$$

subject to:

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

$$h_i(\boldsymbol{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ is the vector of decision variables (normally bounded $a_i \leq x_i \leq b_i$), $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., s$ are the objective functions, and $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$, $j = 1, ..., p$ are the continuously differentiable constraint functions of the problem.

Given a function $f : \mathbb{R}^n \to \mathbb{R}$, for $x \in \mathbb{R}^n$, a direction $v \in \mathbb{R}^n$ is a *descent direction* if ($\cdot$ denotes the inner product and ∇f the gradient vector of f):

$$\nabla f(x) \cdot v < 0 \tag{4}$$

A generalized gradient method can be summarized in the following equation:

$$x^{k+1} = x^k + \alpha^k v^k$$

where v^k is a descent direction and α^k is the step size. One of the most commonly used choice for the descent direction is the following (*steepest descent direction*):

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

Obviously, one of the main difficulties for constrained problems is the feasibility of x^{k+1}. Specially when the constraints are nonlinear, a balance has to be achieved between satisfying the constraints and reducing the objective function.

Moreover, choosing the optimum step size α^k is desirable, but it may be computationally expensive. Some other approaches, which has good properties (e.g., convergence), are quite efficient. One of the most efficient is the Armijo's rule:

Let $\beta \in (0, 1)$ be a prespecified value, let v be a descent direction and let x be the current point. The condition to accept t (the step size) is:

$$f(x + tv) \leq f(x) + \beta t \nabla f(x) \cdot v$$

where we start with $t = 1$ and while this condition is not satisfied we set $t := t/2$.

The choice of β can be critical because the bigger the value of β, the bigger the steps we can implement at the beginning; but the more evaluations that can be consumed if too many reductions of t must be done to achieve the condition. Armijo's rule is mathematically correct and the "t" value always exists. However, this value could be very small, which would be translated into an insignificant

[1] Without loss of generality, we will assume only minimization problems.

progress (this is, in fact, the main disadvantage of Armijo's rule). This problem is more significant for box-constrained problems or, in general, for constrained problems when the current solution is over or close to the boundary between the feasible and infeasible regions, or to the boundary of one of the decision variables, and the descent direction moves it outside of the feasible space. Depending on the violated constraints, we distinguish two cases: (a) if the new solution x^{k+1} violates constraints in (2) or (3) or, (b) if some variables of x^{k+1} is out of its range.

(a) Lets denote by $c_1(x), c_2(x), ..., c_q(x)$ the violated constraints by x^{k+1}. The Karush-Kuhn-Tucker optimality condition (for equality constraint problems) states that x^* is a local minimizer if there exist real numbers $\lambda_1, ..., \lambda_q$ (Lagrange multipliers) such that

$$\nabla f(x^*) = \sum_{i=1}^{q} \lambda_i \nabla c_i(x^*).$$

In a regular situation (for example, if $\nabla f(x^*)$ and $\nabla c_i(x^*), i = 1, ..., q$, are independent), the above condition is imposible to achieve. This means there exists a feasible direction v obtained by decomposing $\nabla f(x^*)$ in its projection over the space generated by $\{\nabla c_1(x^*), ..., \nabla c_q(x^*)\}$ and its normal component, v. So, this normal vector v is computed taken into account that $v = \nabla f(x^*) - \sum_{i=1}^{q} \lambda_i \nabla c_i(x^*)$ has to be orthogonal to $\nabla c_i(x^*)$, for all i. So, the coefficient vector $(\lambda_1, ..., \lambda_q)$ may be obtained by solving the system (all gradient vectors are evaluated in x^*)

$$\begin{pmatrix} \nabla c_1 \cdot \nabla c_1 & \nabla c_1 \cdot \nabla c_2 & \cdots & \nabla c_1 \cdot \nabla c_q \\ \nabla c_2 \cdot \nabla c_1 & \nabla c_2 \cdot \nabla c_2 & \cdots & \nabla c_2 \cdot \nabla c_q \\ \vdots & \vdots & \vdots & \vdots \\ \nabla c_q \cdot \nabla c_1 & \nabla c_q \cdot \nabla c_2 & \cdots & \nabla c_q \cdot \nabla c_q \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_q \end{pmatrix} = \begin{pmatrix} \nabla f \cdot \nabla c_1 \\ \nabla f \cdot \nabla c_2 \\ \vdots \\ \nabla f \cdot \nabla c_q \end{pmatrix}.$$

Then, if x^{k+1} violates constraints $c_1, c_2, ..., c_q$ and the current solution x^k is close enough to them (in order to consider these constraints as active constraints), the feasible direction considered is the above normal vector (reducing the step size until x^{k+1} is feasible). The key issue is the following: The closer x^k to these violated constrains, the more precise the feasible direction v but, due to some of these constraints are nonlinear, the closer x^k to constrains, the smaller the step size to obtain a feasible move. In our experiments, x is considered close, or ε-active, to constraint c_i if $\frac{|c_i(x)|}{\nabla c_i(x) \cdot \nabla c_i(x)} < \varepsilon$. ($\varepsilon = 0.001$ for linear constraints and $\varepsilon = 0.1$ for nonlinear constraints).

(b) In this case, we apply the following rules for each violated variable i:
- If $x_i^{k+1} < a_i$, then $x_i^{k+1} = \frac{a_i + x_i^k}{2}$.
- If $x_i^{k+1} > b_i$, then $x_i^{k+1} = \frac{b_i + x_i^k}{2}$.

This way, the current solution x^k is moved in a intermediate direction between the direction induced by $\nabla f(x^k)$ and its projection over the violated

constraints (like in (a)). In an informal sense, this kind of transformation is also an adapted Armijo's rule but considering different step sizes in each coordinate.

3 Gradient Based method for Multi-Objective Optimization

The goal now is trying to adapt some of the principles of single-objective optimization to obtain a number of efficient points of the MOP problem. The main idea is based on the Fritz-John optimality condition for MOP problems (see for example [4])

– Given a point $x \in X$, a necessary condition to be Pareto optimal solution is the existence of $\boldsymbol{\lambda} \geq 0$ such that:

$$\sum_{i=1}^{p} \lambda_i \nabla f_i(x) = 0$$

For a bi-objective optimization problem, this condition means that for any Pareto optimal solution, we can find some $\lambda \geq 0$ such that $\nabla f_1(x) = -\lambda \nabla f_2(x)$. This is, for any Pareto optimal point, gradients of both objective functions are parallel but in the opposite direction. It means that if we are placed in the minimum of one of the objectives (for example the minimum of f_1, a Pareto optimal solution) and follow the direction of $\nabla f_2(x)$, we will keep in the Pareto front. This is shown graphically in Figure 1.

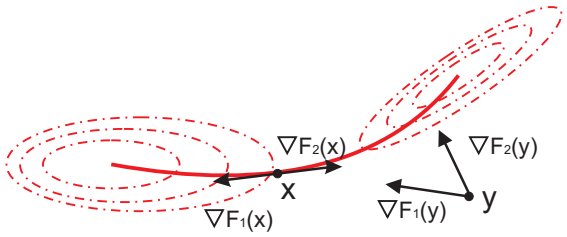

Fig. 1. Pareto front on a bi-objective problem

This idea was used in [10], where they link $s+1$ local searches (more precisely, tabu searches). The first local search starts from an arbitrary point and attempts to find the optimal solution to the problem with the single objective f_1. Let x_1 be the last point visited at the end of this search. Then, a local search is applied again to find the best solution to the problem with the single objective f_2 using x_1 as the initial solution. This process is repeated until all the single-objective problems associated with the s objectives have been solved. At this point, they solve again the problem with the first objective f_1 starting from x_s, to finish

a cycle around the efficient set. This phase yields the s efficient points that approximate the best solutions to the single-objective problems that result from ignoring all but one objective function, and additional efficient solutions may be found during this phase because all visited points are checked for inclusion in the approximation of the Pareto front, as probably most of the intermediate points will lie on the Pareto front. This way, they obtain an initial set of efficient points to be used as an initial population for the EMO method developed in [10].

In this work, we are going to use the same idea, link $s+1$ single objective local searches, but using a single-objective gradient based method instead of a tabu search. The next subsection is devoted to show the main features on this gradient-based local search mechanism.

3.1 Single-Objective Gradient based method

For our local search engine, we are going to use an steepest descent method, this is, given the current point x^k, the next point will be computed as follows:

$$x^{k+1} = x^k - t \cdot \widetilde{\nabla} f(x^k)$$

where $\widetilde{\nabla} f(x^k)$ is an estimation of $\nabla f(x^k)$ (or its projection/modification seen in the above section), and t will be computed following our adapted Armijo's rule with $\beta = 0.1$ and starting with the value of $t = 1$. The reason to choose a low value for β is the fact that small steps are also interesting for us while we are on the Pareto front, as we are checking every intermediate solution for being included in the final approximation. This is, we are not only interested in the final point of each search, but also in the intermediate points.

To estimate the gradient of a function f, we will use a quadratic approximation over all its domain:

$$f(x) \approx \beta_0 + \sum_{i=1}^{n} \beta_i^1 \cdot x_i + \sum_{i=1}^{n} \sum_{j=i}^{n} \beta_{i,j}^2 \cdot x_i \cdot x_j$$

This means that we are interested in global gradients instead of the local information provided by a precise estimation of the gradients at each solution.

The number of parameters (N) to adjust such an approximation for a function with n variables is: $N = 1 + n + \frac{n(n+1)}{2} = \frac{n^2+3n+2}{2}$. N represents the minimum number of points needed to adjust such an approximation. For a problem with 30 variables, for example, at least 496 will be needed. In order to generate these N points efficiently, we used Latin-Hypercubes [9], which is a method that guarantees a good distribution of the initial population in a multidimensional space, as it is required in order to better fit the function with this quadratic approximation. Once these points are generated and evaluated, we compute the values of each parameter solving the corresponding system of equations using a pseudo-inverse (due to its complexity when N is increased). This system of equations can be formulated using matrices: $X \cdot B = Y$, so $X \cdot B = (X^t X)^{-1} X^t Y$.

Finally, we assumed the following stopping conditions:

1. The step is too small, this is, the estimated gradient or the projected gradient is too small: $t \cdot \|\widetilde{\nabla} f(x_k)\| < 0.01$, or
2. The improvement is too small: $|f(x_{k+1}) - f(x_k)| < 0.001$.

The complete method is summarized in Algorithm 1.

Algorithm 1 Constrained Multi-Objective Gradient Based method: CMGBM

Generate a set *InitPop* with N initial points using Latin-Hypercubes.
Send each point in *InitPop* to the list of nondominated solutions: PF.
Use the set *InitPop* to adjust a quadratic approximation of each objective function over all its domain.
for each objective function f_i (repeating the first one) **do**
  x^0 =*last point visited* or *random solution in* PF *when* $i = 0$
  while stopping conditions = FALSE or x^{k+1} is unfeasible **do**
    Obtain x^{k+1} through the gradient-based method using $\widetilde{\nabla} f_i(x^k)$.
    If x^{k+1} is unfeasible, check ε-active constraints for x^k.
    If x^k is an interior point (it has no ε-active constraints), reduce the step size. Otherwise, obtain a new x^{k+1} through the projected gradient over the ε-active constraints.
    Send x^{k+1} to PF.
  end while
end for

4 Preliminary Results

To test the performance of CMGBM, we solved several constrained optimization problems from the benchmark: Srinivas ([13]), Osyczka and Osyczka2 ([11]), Tanaka ([14]), Binh ([1]) and Jimenez ([6]). All of them are nonlinear bi-objective optimization problems with several nonlinear constraints. Moreover, they all have 2 decision variables, but Osyczka which has 6 variables. For a quick overview of these test functions, it can be seen *www.cs.cinvestav.mx/* $\sim$ *emoobook/*.

Results obtained by CMGBM are not compare against other algorithms because the main aim is to show the viability of this scheme to obtain nondominated solutions over the true Pareto front. This is why we perform 11 independent runs, measured using the Inverted Generational Distance, IGD ([15]). IGD measures the euclidean distance from the true Pareto front to the approximated front, previously normalized to allow a fair comparison. So, the closer the IGD value to zero, the better the approximation. $IGD = 0$ is obtained only when the approximated front is over the true Pareto front and the extremes have been also achieved.

It can be observed in Table 1 that CMGBM produced IGD values really close to zero. The first column shows the best IGD values (min), next one shows the mean IGD values and its corresponding standard deviation. The third column show the worst (max) of the eleven IGD values. Finally, last two columns show

the average of the number of nondominated solutions obtained by CMGBM and the mean value of the number of evaluations consumed. For these problems, the CMGBM is able to find a high number of exact efficient points using very few evaluations.

Test Function / IGD	**Min**	**Mean (SD)**	**Max**	**N. Points**	**N. Eval.**
Srinivas	0.057	0.089 (0.013)	0.107	74.455	406.6
Osyczka	0.171	0.296 (0.136)	0.570	64.000	544.2
Osyczka2	0.102	0.140 (0.018)	0.159	37.546	917.4
Tanaka	0.091	0.489 (0.410)	1.227	17.091	239.8
Binh	0.046	0.072 (0.012)	0.084	224.727	594.1
Jimenez	0.010	0.043 (0.063)	0.216	129.091	262.3

Table 1. IGD values for the selected six test problems.

Figure 2 shows the nondominated solutions got by CMGBM. These plots correspond to the best run (top) and the run in the median value (bottom) with respect to the IGD metric. We can clearly see that in all problems, but Osycka2, CMGBM converged to the true Pareto front after only about 500 fitness function evaluations. And even for the hardest problem (Osyczka2), CMGBM obtained some good solutions after only 1000 evaluations. On the other hand, CMGBM was able to obtain the extreme points (of each objective function) in most of the cases.

5 Conclusions

We have introduced a Constrained Multi-Objective Gradient-Based Method (CMGBM) in order to generate efficient solutions of nonlinear constrained multi-objective optimization problems with a low number of objective function evaluations. The main contribution is the way we use estimated gradient vector of the objective functions and the gradient vector of the constraints to obtain an improvement direction. Results show the efficiency of this method over several nonlinear constrained MOPs, since CMGBM obtain good approximations of the Pareto front consuming few objective function evaluations. With this preliminary results we show how the use of gradient information could reduce the computational cost while quality is not decreased. We think, this gradient information could be so useful to seed EMO algorithms and enhance their convergence. This strategy could be more efficient than using gradients through all the EMO execution because once the EMO method is provided with solutions close (or in) to the Pareto front, the use of gradient information is consuming a lot of evaluations while not providing sensible advantages.

In the future, besides completing a comprehensive set of experiments, we would like to improve this scheme using also approximated gradient vectors of

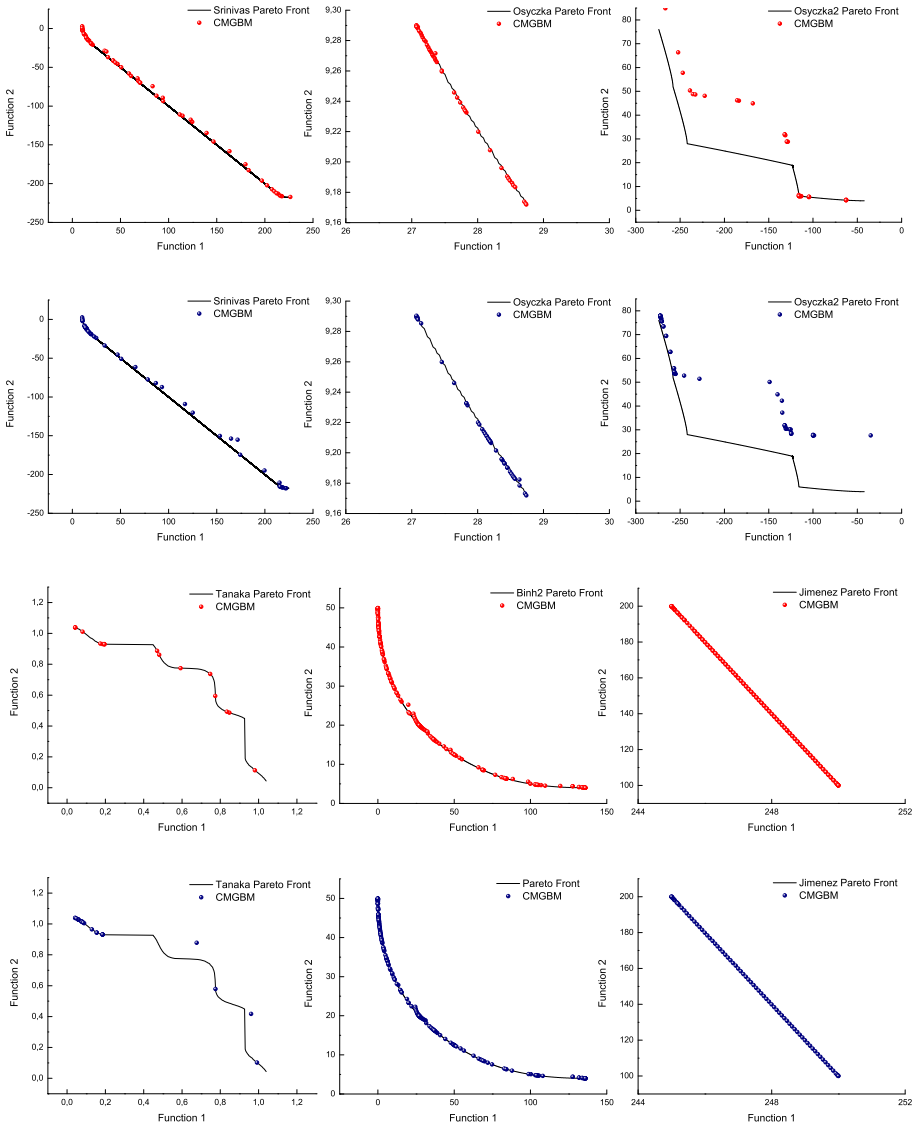

Fig. 2. Best (top) and median (bottom) IGD values obtained by CMGBM for selected problems

the constraints, and adapt this mechanism for problems with more than two objective functions.

Acknowledgments

The second author gratefully acknowledges support from CONACyT project 45683-Y. The third author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN. Rest of the authors acknowledge support from Andalusian Regional Government and Spanish Ministry of Educacion y Ciencia

References

1. T. T. Binh, U. Korn, MOBES: A multiobjective evolution strategy for constrained optimization problems, In *The Third International Conference on Genetic Algorithms (Mendel 97)*, 176-182, Brno, Czech Republic, (1997).
2. C. A. Coello Coello, G. B. Lamont, Applications of Multi-Objective Evolutionary Algorithms, World Scientific, Singapore, (2004).
3. A. V. Fiacco and G. P. McCormick, Nonlinear Programming, Classics Appl. Math. 4, SIAM, Philadelphia, PA, 1990. Reprint of the 1968 original.
4. J. Fliege, B. Svaiter: Steepest Descent Methods for Multicriteria Optimization. Mathematical Methods of Operations Research, 51(3), pp. 479-494, (2000).
5. A. A. Goldstein, Convex programming in Hilbert Space, Bulletin of the American Mathematical Society 70, pp. 709-710, (1964).
6. F. Jiménez, J. L. Verdegay, A. F. Gómez-Skarmeta. Evolutionary Techniques for Constrained Multiobjective Optimization Problems. In A. S. Wu, editor, Proc. of GECCO, pp. 115-116, Orlando, Florida, (1999).
7. N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4, pp. 373-395, (1984).
8. E. S. Levitin, B. T. Polyak, Constrained Minimization Problems, USSR Computational Mathematics and Mathematical Physics 6, pp. 1-50, (1966).
9. M.D. McKay, R.J. Beckman, W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 21(2), pp. 239-245, (1979).
10. J. Molina, M. Laguna, R. Marti, R. Caballero: SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization. INFORMS Journal on Computing 19(1), pp. 91-100, (2007).
11. A. Osyczka, S. Kundu: A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm, *Structural Optimization 10*, pp. 94-99, (1995).
12. S. Schaffler, R. Schultz, K. Weinzierl: Stochastic method for the solution of unconstrained vector optimization problems. Journal of Opt. Theory and Applications, 114(1), pp. 209-222, (2002).
13. N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3) pp. 221-248, (1994).
14. M. Tanaka, H. Watanabe, Y. Furukawa, T. Tanino:GA-Based Decision Support System for Multicriteria Optimization, *Proc. of the International Conference on Systems, Man, and Cybernetics 2*, pp. 1556-1561, IEEE Piscataway, NJ, (1995).

15. D. A. Van Veldhuizen: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Phd. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, (1999).