A Proposal to Hybridize Multi-Objective Evolutionary Algorithms with Non-Gradient Mathematical Programming Techniques

Saúl Zapotecas Martínez and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, MÉXICO
saul.zapotecas@gmail.com, ccoello@cs.cinvestav.mx

Abstract. The hybridization of multi-objective evolutionary algorithms (MOEAs) with mathematical programming techniques has gained increasing popularity in the specialized literature in the last few years. However, such hybrids normally rely on the use of gradients and, therefore, normally consume a high number of extra objective function evaluations in order to estimate the gradient information required. The use of direct (nonlinear) optimization techniques has been, however, less common in the specialized literature, although several hybrids of this sort have been proposed for single-objective evolutionary algorithms. This paper proposes a hybridization between a well-known MOEA (the NSGA-II) and two direct search methods (Nelder and Mead's method and the golden section algorithm). The aim of the proposed approach is to combine the global search mechanisms of the evolutionary algorithm with the local search mechanisms provided by the aforementioned mathematical programming techniques, such that a more efficient (i.e., with a lower number of objective function evaluations) approach can be produced.

1 Introduction

The use of evolutionary algorithms for solving multi-objective optimization problems has become very popular in the last 10 years, finding applications in a wide variety of areas [1]. However, one of the current limitations of MOEAs is their computational cost, which turns out to be unaffordable in certain real world applications. The design of hybrid approaches combining MOEAs and mathematical programming techniques is not new (see for example [2]). However, these hybridation schemes normally rely on gradient-based information to guide the search. This may be inappropriate, since estimating such gradients normally requires additional objective function evaluations, which is precisely what we are trying to avoid in computationally expensive problems. Although the use of surrogate models is a possible alternative to deal with such problems (see for example [3]), these approximate models tend to produce accumulated errors that sometimes generate a significant deviation with respect to the original model. In this paper, we propose a new multi-objective hybrid algorithm based

on the NSGA-II [4], coupled with two mathematical programming methods: Nelder and Mead's method (which is used for multidimensional optimization) and the golden section (which is used for unidimensional optimization). The aim of this proposed approach is to speed up the convergence of the baseline MOEA, through the introduction of powerful local search engines (based on direct search methods taken from the mathematical programming literature). This sort of hybrid aims at introducing information obtained from mathematical programming techniques (which are deterministic algorithms) to refine the search performed by a global search engine (a genetic algorithm in this case) without having to perform additional objective function evaluations.

2 Basic Concepts

2.1 The Nonlinear Simplex Method

Spendley et al. [5] presented the basic simplex method, which is an efficient sequential optimization method for minimizing real and multidimensional functions. Later on, Nelder and Mead presented an improvement of this method which was called the Nonlinear Simplex Search (NSS) method [6] (also known as the Nelder and Mead method). The convergence towards a minimum value at each iteration of the NSS is conducted by four movements in a geometric shape called *simplex*. A *simplex* or *n-simplex* Δ is a convex hull of a set of $n+1$ affinely independent points Δ_i ($i = \{0, 1, \ldots, n\}$), in some Euclidean space of dimension n. To define the full algorithm, it is necessary to specify four scalar parameters to control the movements performed in the simplex: reflection (α), expansion (γ), contraction (β) and shrinkage (δ). At each iteration, the $n+1$ vertices of the simplex Δ_i represent the solutions evaluated and are sorted according to: $f(\Delta_0) \leq f(\Delta_1) \leq \cdots \leq f(\Delta_n)$. In this way, the movements performed in the simplex by the NSS method are defined as:

1. *Reflection*: $x_r = (1+\alpha)x_c - \alpha\Delta_n$.
2. *Expansion*: $x_e = (1+\alpha\gamma)x_c - \alpha\gamma\Delta_n$.
3. *Contraction*:
   (a) *Outside*: $x_{co} = (1+\alpha\beta)x_c - \alpha\beta\Delta_n$.
   (b) *Inside*: $x_{ci} = (1-\beta)x_c + \beta\Delta_n$.
4. *Shrinkage*: The new vertices of the simplex at the next iteration will be: $\{\Delta_0, v_1, v_2, \ldots, v_n\}$, where $v_j = \Delta_0 + \delta(\Delta_j - \Delta_0)$, for all $j = \{1, 2 \ldots, n\}$.

where x_c is called centroid of the simplex, and is computed as: $x_c = \frac{1}{n}\sum_{i=0}^{n-1}\Delta_i$; Δ_0 and Δ_n are the best and the worst solutions identified within the simplex, respectively. At each iteration, the initial simplex is modified by one of the above movements, according to the following rules:

1. If $f(\Delta_0) \leq f(x_r) \leq f(\Delta_{n-1})$, then $\Delta_n = x_r$.
2. If $f(x_e) < f(x_r) < f(\Delta_0)$, then $\Delta_n = x_e$,
   otherwise $\Delta_n = x_r$.
3. If $f(\Delta_{n-1}) \leq f(x_r) < f(\Delta_n)$ and $f(x_{co}) \leq f(x_r)$,
   then $\Delta_n = x_{co}$; otherwise, perform a shrinkage.
4. If $f(x_r) \geq f(\Delta_n)$ and $f(x_{ci}) < f(\Delta_n)$,
   then $\Delta_n = x_{ci}$; otherwise, perform a shrinkage.

We chose Nelder and Mead's method for two reasons: it is a widely used multidimensional optimization technique, and there exists previous evidence of success in being hybridized with evolutionary algorithms. However, the use of other (more powerful) mathematical programming techniques (e.g., Powell's conjugate direction method) is left for future work.

2.2 The Golden Section Method

The *golden section*, represented by φ, is a line segment sectioned in two parts according to the *golden ratio*, which refers to the ratio between length and height of a rectangle that is required in order to make it more aesthetically pleasant to our senses. The golden ratio has a value of $\varphi \approx 0.618033$. The golden section search method finds the minima of a function within a certain (given) search interval. This approach is very efficient to optimize unimodal, unidimensional and unconstrained functions, and it is based on the main principle of the *region elimination methods*, which establishes that if we assume a function to be minimized f to be strictly unimodal on the interval $a \leq x \leq b$ with a minimum at x^*, and having two points x_1 and x_2 in this interval, such that $a < x_1 < x_2 < b$, then we can conclude [7]:

1. If $f(x_1) > f(x_2)$, then the minimum of $f(x)$ does not lie in the interval (a, x_1). In other words, $x^* \in (x_1, b)$.
2. If $f(x_1) < f(x_2)$, then the minimum does not lie in the interval (x_2, b). In other words, $x^* \in (a, x_2)$.

By using the golden section, the region to be eliminated at each iteration can be maximized, so that the minimum can be found in a more efficient way (i.e., requiring less iterations). We decided to adopt this approach, because it is the most efficient region elimination method and it is a direct search approach (i.e., does not require derivatives). Although other (more powerful) unidimensional optimization techniques exist, they rely either on polynomial approximations (e.g., quadratic estimation) or gradient-based information (e.g., secant, Newton-Raphson and cubic search) and thus limit the type of functions to be optimized.

2.3 Low-discrepancy Sequences

A *low-discrepancy sequence* is also called a quasi-random or sub-random sequence, and it offers a high degree of uniformity in comparison with the more

common uniformly distributed random numbers. Low-discrepancy sequences are commonly used in Monte Carlo simulations of integrals that do not have a closed-form expression in order to achieve variance reduction. Here, we adopt low-discrepancy sequences to construct the simplex in the Nelder and Mead method. Next, we present the two low-discrepancy sequences that are adopted in this work.

Halton Sequence The *Halton sequence* [8] is a variation of the *van der Corput sequence* [9], differing only in the representation, since the van der Corput sequence uses binary representation and the Halton sequence adopts a different base for each vector coordinate. The Halton sequence is constructed according to a deterministic method based on number theory. For constructing the i-th vector of the Halton sequence in $\mathbb{R}^n$, the first step is to choose n relatively prime numbers $p_1, p_2, \ldots, p_n$. We consider the representation in base p of i, which takes the $i = a_0 + pa_1 + p^2a_2 + \ldots$ form. Each coordinate of the vector is in $[0, 1]$ and is obtained by:

$$r(i, p) = \frac{a_0}{p} + \frac{a_1}{p^2} + \frac{a_2}{p^3} + \frac{a_3}{p^4} + \cdots$$

In this way, the i-th vector in the Halton sequence (starting with $i = 0$) is defined as:

$$\langle r(i, p_1), r(i, p_2), \ldots, r(i, p_n) \rangle \tag{1}$$

Hammersley sequence The *Hammersley sequence* [10] is an adaptation of the Halton sequence, which uses $n-1$ relatively prime numbers. Starting with $i = 0$, the i-th vector in the Hammersley sequence for a set of k vectors is defined as:

$$\left\langle \frac{i}{k}, r(i, p_1), r(i, p_2), \ldots, r(i, p_{n-1}) \right\rangle \tag{2}$$

In an analogous way, each component of the vector in the Hammersley sequence is in $[0, 1]$.

3 Our proposed approach

Our proposed approach is called *Nonlinear Simplex Search Genetic Algorithm* (NSS-GA), and combines the explorative power of a MOEA with the exploitative power of the Nelder and Mead method, which acts as a local search engine. The general scheme of the NSS-GA is detailed in Figure 1.

3.1 Local search

The general idea of this phase is to intensify the search towards better solutions for each objective function, based on an individual of the population. Genetic traits of the best individuals found for each objective function are reproduced using the evolutionary operators of a genetic algorithm. The main goal of this phase

n = number of decision variables
λ = set of locally optimal solutions found by the
local search mechanism
1. $t = 0$.
2. Randomly initialize a population P_t of size N.
3. Evaluate the fitness of each individual in P_t.
4. Generate the offspring Q_t, applying the selection,
crossover and mutation operators to P_t.
5. $R_t = P_t \cup Q_t$ (thus, R_t is now of size $2N$).
6. Assign to P^* the N better individuals from R_t
according to the crowded comparison operator ($\prec_n$).
7. If ($t \mod \frac{n}{2} = 0$) then:
i. Get λ set according to the exploration phase.
ii. $R_t^* = P_t^* \cup \lambda$.
iii. Assign to P_{t+1} the N best
individuals from R_t^* according to the crowded comparison operator ($\prec_n$).
Else: $P_{t+1} = P^*$.
8. $t = t + 1$.
9. If $t > t_{max}$ stop, else go to step 4.

Fig. 1. Main algorithm of our proposed Nonlinear Simplex Search Genetic Algorithm (NSS-GA).

is to obtain the λ set using classical optimization methods. Because the Nelder and Mead algorithm was designed to optimize multidimensional functions, when dealing with unidimensional optimizations, the golden section method is used instead. Thus, the λ set is defined as:

$$\lambda = \lambda_1 \cup \lambda_2 \cup \cdots \lambda_k \cup \Upsilon$$

where λ_i is a set of the best solutions found for the i-th objective function of the MOP and Υ is a set of the best solutions found for the aggregating function described later in this section. If the i-th objective function to be optimized is unidimensional, the size of λ_i is 1. In this case, the golden section method is adopted to find the minimum of such objective function. Otherwise, if the i-th objective function is multidimensional, then the size of the λ_i set is $n + 1$ and corresponds to all the vertices of the final simplex found by the NSS algorithm. Next, we describe the different components of our local search engine.

Selection Mechanism In the population P, we choose the individual $x_\Delta \in P$ to optimize its i-th objective:

$$x_\Delta = x_l | x_l = \min_{\forall x_l \in \mathcal{P}^*} \{f_i(x_l)\} \tag{3}$$

where $\mathcal{P}^*$ is a nondominated solutions set within the population P. In other words, the selected individual is the best nondominated solution with respect to the i-th objective function.

Aggregating Function The vector $\boldsymbol{H} = [f_1^*, f_2^*, \ldots, f_k^*]$, consists of the minimum values f_i^* of the k objective functions in the current generation. We select the individual x_a from the population P, such that we minimize:

$$G(x_a) = \sum_{i=1}^{k} \frac{|\boldsymbol{H}[i] - f_i(x_a)|}{|\boldsymbol{H}[i]|} \tag{4}$$

In this way, the local search minimizes the aggregating function defined by:

$$\psi(x) = ED(\boldsymbol{H}, \boldsymbol{F}(x)) \tag{5}$$

where $\boldsymbol{F}$ is vector of objective functions values of each individual and ED is the Euclidean distance between the $\boldsymbol{F}$ and $\boldsymbol{H}$ vectors. Summarizing, the search first focuses on finding the extremes of the Pareto front (using equation 3). In this phase, we select as many individuals as objectives of the problem. Then, we select one additional individual using equation 4, aiming to reach the "knee" of the Pareto front.

Note that there are functions for which the NSS algorithm becomes inoperable (for example, McKinnon's function [11]). In order to deal with these and other more complex functions, the NSS method has undergone some modifications in the specialized literature (see for example [12, 13]). We propose here a new strategy to guide the improvement process towards promising areas during each generation of the hybrid algorithm. Such strategy is described next.

Building the Simplex The selected solution x_Δ (x_a for the case of the aggregating function) is called "*simplex-head*", which is the first vertex of the n-simplex. The remaining n vertices are created in two phases:

1. *Reducing the Search Domain*: We use a strategy based on genetic analysis of a sample taken from the population. From this sample, we identify the average and standard deviation of the genes (decision variables) in each individual. Based on that information, we define the new search space as:

$$\begin{aligned} low\_bound_{new}^j &= m(P_m(j)) - \sigma(P_m(j)) \\ up\_bound_{new}^j &= m(P_m(j)) + \sigma(P_m(j)) \end{aligned} \tag{6}$$

   where P_m represents the individuals in the sample taken from the population (such individuals are those with the best fitness with respect to the objective function to optimize), $m(P_m(j))$ is the average and $\sigma(P_m(j))$ is the standard deviation in the j-th parameter of the sample P_m. The size of the sample taken in this work is 20% of the total population size.
2. *Build Simplex Vertices*: Once the new search domain has been defined, the remaining vertices are determined using either the Halton or the Hammersley sequence (each has a 50% probability of being selected).
   For both sequences, the components are in $[0, 1]$ and are mapped to the new interval acording to:

$$c' = low\_bound_{new} + c \cdot (up\_bound_{new} - low\_bound_{new})$$

where c is the component to be mapped to the interval $[0, 1]$ and c' is the component already mapped to the desired interval. Although we do not have a mathematical proof regarding the suitability of this scheme to generate a non-degenerate simplex (i.e., a simplex whose volume is greater than zero), we empirically found that this procedure worked well in the numerous experiments that we performed.

Bounded Variables for NSS The NSS method was conceived for unbounded domain problems. When dealing with bounded variables, the created vertices can be located outside the allowable bounds after some movements of the NSS method. Luersen et al. [14] proposed a simple strategy to deal with bounded variables, which is the one we adopted in this work:

Let Δ_{new} be the new vertex created by some NSS movement. The j-th component of the vertex is established as:

$$\Delta_i^{(j)} = \begin{cases} low\_bound_j & \text{, if } \Delta_i^{(j)} < low\_bound_j \\ up\_bound_j & \text{, if } \Delta_i^{(j)} > up\_bound_j \\ \Delta_i^{(j)} & \text{, otherwise.} \end{cases} \tag{7}$$

where low_bound_j and up_bound_j are the lower and upper bounds in the j-th parameter, respectively.

However, this strategy can degenerate the simplex. We propose here to rebuild the simplex if it has been degenerated, i.e., if its volume is different from zero. Since we need a procedure to compute the volume of the simplex, we adopt the proposal from [15] for that sake.

Stopping Criteria Two stopping criteria are adopted in this work. The first criterion imposes convergence towards a vertex better than the worst vertex within the simplex (x_w). This criterion is taken from Lagarias et al. [15]. However, adopting this stopping criterion does not guarantee that the NSS method has an efficient performance. Convergence can be slow and may require a large number of evaluations of the objective function. For this reason, we use a second stopping criterion which consists of defining a convergence threshold ϵ (for the experiments reported in this paper, $\epsilon = 1 \times 10^{-3}$). Thus, the local search is stopped if:

1. It does not generate a vertex better than x_w after performing $(n + 1)$ iterations, or
2. if after performing $2(n+1)$ iterations, the convergence towards a better point is $\leq \epsilon$.

where n is the number of decision variables of the function to be optimized.

4 Comparison of Results

In order to evaluate the performance of the proposed hybrid algorithm, we compare its performance with respect to the NSGA-II [4]. The test problems adopted

are five from the ZDT test suite [16] (except from ZDT5, which is a binary test problem). We also adopted two problems from the DTLZ test suite (DTLZ1 and DTLZ2) [17]. The description of these test problems is omitted due to space constraints. We adopted three performance measures to assess our results: Inverted Generational Distance ($\mathcal{IGD}$) [18, 1], Spacing ($\mathcal{S}$) [19] and the Coverage Indicator ($\mathcal{CI}$) [16]. Their description is also omitted due to space constraints.

4.1 Experimental Setup

Our proposal (the NSS-GA) is compared with respect to the baseline algorithm adopted (the NSGA-II). Since our approach does not require any additional parameters for the main search engine (i.e., the NSGA-II), the comparison was done with the same parameter values for both approaches in order to allow a fair comparison. Thus, we adopted the following values: Population size (S_p) = 100, crossover probability (P_c) = 0.9, mutation probability (P_m) = $\frac{1}{N}$, where N is the number of decision variables. The nonlinear simplex search was implemented with: $\alpha = 1, \beta = 2, \gamma = \frac{1}{2}$ and $\delta = \frac{1}{2}$. For each MOP, we performed 30 independent runs with each of the two approaches. The results are presented in Tables 1 to 3. Each table displays both the average and the standard deviation (σ) of each performance measure, for each of the test problems adopted. The best average results obtained in each test function are displayed in **boldface**. Each run is restricted to 4,000 fitness function evaluations, which is a very low value when compared to those adopted by most MOEAs nowadays. From these results, it is evident that our proposed approach (NSS-GA) outperforms the NSGA-II in all the test problems, with respect to both the $\mathcal{IGD}$ (which measures closeness to the true Pareto front) and the $\mathcal{CI}$ performance measure (which determines if the solutions generated by one algorithm dominate the solutions generated by the other). In fact, the low values obtained by our NSS-GA indicate that our approach has practically converged to the true Pareto front, except for ZDT4 and DTLZ1 (we could corroborate this by plotting the results, but such graphs were omitted due to space restrictions). With respect to the $\mathcal{S}$ performance measure, our approach obtained better results in 4 test problems, and the NSGA-II obtained better results in the other three. However, since convergence is more important than even distribution of nondominated solutions, we do not consider this to be a major drawback of our proposed approach.

5 Conclusions and Future Work

In this paper, we have introduced a hybridization scheme in which a MOEA (the NSGA-II) is coupled to two direct search methods (Nelder and Mead's method and the golden section method). Our proposed approach (called NSS-GA), was found to be competitive with respect to the original NSGA-II over a set of test functions taken from the specialized literature, when performing only 4,000 fitness function evaluations. As part of our future work, we are interested in experimenting with other direct search methods, such as the Hooke-Jeeves

Table 1. Results of $\mathcal{IGD}$ for the NSS-GA and the NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**0.001149**	0.000598	0.005582	0.000905
ZDT2	**0.002101**	0.001785	0.015385	0.004631
ZDT3	**0.001221**	0.000832	0.004217	0.000798
ZDT4	**0.122063**	0.058813	0.156509	0.051699
ZDT6	**0.008980**	0.004758	0.046699	0.007258
DTLZ1	**0.658650**	0.107311	0.779135	0.168162
DTLZ2	**0.000403**	0.000022	0.000428	0.000024

Table 2. Results of $\mathcal{S}$ for the NSS-GA and NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**0.014620**	0.005329	0.023731	0.004730
ZDT2	**0.021928**	0.014674	0.029762	0.006576
ZDT3	**0.013990**	0.005108	0.023994	0.004774
ZDT4	**0.455495**	0.416654	3.098866	2.822281
ZDT6	0.171233	0.117406	**0.106812**	0.055624
DTLZ1	17.965977	7.564753	**16.132116**	6.874782
DTLZ2	0.055607	0.005740	**0.055528**	0.004754

pattern search method and Powell's conjugate direction method. We are also interested in devising mechanisms that help us to decide whether the local search needs to be triggered or not.

Acknowledgments: The first author acknowledges support from CINVESTAV-IPN and CONACyT. The second author acknowledges support from CONACyT through project number 45683-Y.

References

1. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Second edn. Springer, New York (2007) ISBN 978-0-387-33254-3.
2. Shukla, P.K.: On Gradient Based Local Search Methods in Unconstrained Evolutionary Multi-objective Optimization. In Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T., eds.: Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matshushima, Japan, Springer. Lecture Notes in Computer Science Vol. 4403 (2007) 96–110
3. Mack, Y., Goel, T., Shyy, W., Haftka, R.: Surrogate Model-Based Optimization Framework: A Case Study in Aerospace Design. In Yang, S., Ong, Y.S., Jin, Y., eds.: Evolutionary Computation in Dynamic and Uncertain Environments. Springer (2007) 323–342 ISBN 978-3-540-49772-1.
4. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions Evolutionary Computation **6**(2) (2002) 182–197

Table 3. Results of $\mathcal{CI}$ for the NSS-GA and NSGA-II

Problem	NSS-GA		NSGA-II	
	average	σ	*average*	σ
ZDT1	**1.000000**	0.000000	0.000000	0.000000
ZDT2	**0.971111**	0.155571	0.004444	0.023934
ZDT3	**0.969534**	0.064908	0.009305	0.022992
ZDT4	**0.686486**	0.322281	0.332336	0.388879
ZDT6	**0.769754**	0.273523	0.144094	0.230425
DTLZ1	**0.590605**	0.147409	0.222936	0.112005
DTLZ2	**0.150000**	0.072019	0.025667	0.022462

5. Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential Application of Simplex Designs in Optimization and Evolutionary Operation. Technometrics **4**(4) (1962) 441–461
6. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. The Computer Journal **7** (1965) 308–313
7. Ravindran, A., Ragsdell, K., Reklaitis, G.: Engineering Optimization. Methods and Applications. John Wiley & Sons, Inc., Hoboken, New Jersey, USA (2006)
8. Halton, J.H.: On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. Numerische Mathematik **2** (1960) 84–90
9. van der Corput, J.G.: Verteilungsfunktionen. Akademie van Wetenschappen **38** (1935) 813–821
10. Hammersley, J.M.: Monte-Carlo methods for solving multivariable problems. Annals of the New York Academy of Science **86** (1960) 844–874
11. McKinnon, K.I.M.: Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point. SIAM Journal on Optimization **9**(1) (1998) 148–158
12. Trabia, M.B., Lu, X.B.: A Fuzzy Adaptive Simplex Search Optimization Algorithm. Journal of Mechanical Design **123** (2001) 216–225
13. Rahman, M.K.: An intelligent moving object optimization algorithm for design problems with mixed variables, mixed constraints and multiple objectives. Structural and Multidisciplinary Optimization **32**(1) (2006) 40–58
14. Luersen, M.A., Le Riche, R.: Globalized Nelder-Mead method for engineering optimization. Computers & Structures **82** (2004) 2251–2260
15. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM Journal of Optimization **9** (1998) 112–147
16. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
17. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., Jain, L., Goldberg, R., eds.: Evolutionary Multiobjective Optimization. Theoretical Advances and Applications. Springer, USA (2005) 105–145
18. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
19. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts (1995)