A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model

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 development of multi-objective evolutionary algorithms (MOEAs) assisted by meta-models has increased in the last few years. However, the use of local search engines assisted by meta-models for multi-objective optimization has been less common in the specialized literature. In this paper, we propose the use of a local search mechanism which is assisted by a meta-model based on support vector machines. The local search mechanism adopts a free-derivative mathematical programming technique and consists of two main phases: the first generates approximations of the Pareto optimal set. Such solutions are obtained by solving a set of aggregating functions which are defined by different weighted vectors. The second phase generates new solutions departing from those obtained during the first phase. The solutions found by the local search mechanism are incorporated into the evolutionary process of our MOEA. Our experiments show that our proposed approach can produce good quality results with a budget of only 1,000 fitness function evaluations in test problems having between 10 and 30 decision variables.

1 Introduction

Evolutionary algorithms (EAs) have been successfully adopted for solving multi-objective optimization problems (MOPs) in a wide variety of engineering and scientific problems [1]. However, in real-world applications is common to find objective functions which are very expensive to evaluate (computationally speaking). This considerably limits the application of EAs. This has motivated the development of numerous strategies for reducing the number of fitness function evaluations when using EAs [2]. From such strategies, the use of meta-models has been one of the most commonly adopted. Several authors have reported the use of surrogate models which aim to model a function by means of a simple linear regression, polynomial regression or by more elaborated models such as Artificial Neural Networks (ANNs), Radial Basis Functions (RBFs), Support Vector Machines (SVMs), Gaussian processes (also known as Kriging), among others.

* The first author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN.
** The second author acknowledges support from CONACyT project no. 103570.

Most of this work, however, focuses on single-objective optimization problems, and relatively few refer to multi-objective optimization tasks. In this paper, we present a strategy which combines an approximation function model (based on support vector machines) combined with a local search engine (which adopts a non-gradient mathematical programming technique) and a multi-objective evolutionary algorithm (MOEA). Our goal was to reduce the number of fitness function evaluations, while still producing reasonably good approximations of the Pareto optimal set.

The remainder of this paper is organized as follows. In Section 2, we present a brief survey of previous related work reported in the specialized literature. In Section 3, we describe in detail our proposed approach. In Section 4, we show the results of our proposal. Finally, in Section 5 we present our conclusions and provide some possible paths for future research.

2 Previous Related Work

The incorporation of meta-models in EAs to approximate the real fitness function of a problem, aiming to reduce the total number of fitness evaluations performed has been studied by several researchers. However, most of these approaches have been developed to deal with single-objective optimization problems (see for example [2]). Here, however, our review of previous work will focus only on MOEAs.

Ong et al. [3] proposed an approach that incorporates surrogate models for solving computationally expensive problems with constraints. The authors used a combination of a parallel EA coupled with sequential quadratic programming in order to find optimal solutions of an aircraft wing design problem. A local surrogate model based on RBFs is the strategy adopted to approximate the objective and the constraint functions.

Emmerich and Naujoks [4] proposed several metamodel-assisted MOEAs. Gaussian field (Kriging) models fitted by results from previous evaluations are used in order to pre-screen candidate solutions and decide whether they should be evaluated or rejected. Three different rejection mechanisms were proposed and integrated into MOEA variants (NSGA-II and ϵ-MOEA).

In [5], Knowles proposed "ParEGO", which consists of a hybrid algorithm based on a single optimization model (EGO) and a Gaussian process, which is updated after each function evaluation, coupled to an evolutionary algorithm. EGO is a single-objective optimization algorithm that uses Kriging to model the search landscape from the previously visited solutions.

Isaacs et al. [6] proposed a MOEA with spatially distributed surrogate models based on RBFs. In this approach, the objective functions are analyzed with their actual values for the initial population and then periodically, at every few generations. The approach maintains an external archive of these actual objective function values, since these values are used to train the surrogate models. The data points are divided into multiple partitions using clustering techniques (the k-means algorithm). The surrogate model is built for each partition using a fraction of the points lying in that partition. The rest of the points in the

partition are used as validation data to decide the prediction accuracy of the surrogate model.

Finally, Georgopoulou and Giannakoglou [7] proposed a metamodel-assisted memetic algorithm for multi-objective optimization. This approach uses several RBFs, each of them corresponding to a small portion of the search space. The local search mechanism uses a function which corresponds to an ascent method that incorporates gradient values provided by the metamodels. Each RBF is retrained by considering the current offspring, parent and elite populations. The performance of this approach was evaluated with three benchmark problems and a combined cycle power plant problem. This approach outperformed a conventional MOEA in all the test problems adopted.

3 Our Proposed Approach

In this section, we present a new *Multi-Objective Meta-Model Assisted Memetic Algorithm* (MO-MAMA) which incorporates a local search mechanism based on non-gradient mathematical programming techniques. Our algorithm is characterized by using an approximation model based on support vector regression [8]. Additionally, our approach adopts an external archive $\mathcal{A}$ and a solutions set $\mathcal{R}$ (obtained by the local search mechanism) to create the offspring population in the EA. The meta-model is trained with the set $\mathcal{D}$, which consist of all the solutions evaluated with the real fitness function values obtained up to the current generation. The details of this approach are described next.

3.1 The Multi-Objective Meta-Model Assisted Memetic Algorithm

Initially, we create a sample $\mathcal{S}$ of size $2N$ (where N is the population size) which is randomly distributed in the search space using the Latin hypercube sampling method [9]. The initial population $\mathcal{P}_0$ is defined by N solutions randomly chosen from $\mathcal{S}$. Then, the normal evolutionary process of the MOEA is carried out. The proposed approach uses the current population $\mathcal{P}_t$, a set of solutions $\mathcal{R}_t$ (obtained by the local search mechanism) and a (bounded) external archive $\mathcal{A}_t$ (defined by all the nondominated solutions found throughout the evolutionary process) to create the offspring population $\mathcal{Q}_t$ at generation t. The next population $\mathcal{P}_{t+1}$ is obtained by selecting N individuals from $\mathcal{P}_t \cup \mathcal{Q}_t$ according to Pareto ranking. This procedure is called $SelectNextPopulation$ in the algorithm of Figure 1, which shows the complete scheme of our proposed MO-MAMA. Its details are explained next.

Archiving solutions: Our algorithm uses an external archive $\mathcal{A}$ which stores all the nondominated solutions found at each generation of the MOEA. The archive $\mathcal{A}$ is bounded according to the population size. Thus, the maximum number of solutions in $\mathcal{A}$ is N. Since we are interested in obtaining a well-distributed set of solutions along the Pareto front, we adopted a strategy based on the k-means algorithm [10]. At each generation, the archive $\mathcal{A}$ is updated

```
// t_max = maximum number of generations
1. t = 0, A = ∅;
2. Generate S of size 2N // using the Latin Hyper-cubes method;
3. Evaluate(S);              // using the real fitness function
4. P_t = {x_i ∈ S} such that: x_i is randomly chosen from S and |P_t| = N;
5. R_t = S \ P_t;
6. A = UpdateArchive(R_t, A);
7. D = S;
8. while (t < t_max)do
9.       A = UpdateArchive(P_t, A);
10.      Q_t = CreateOffspring(P_t, R_t, A_t);
11.      Evaluate(Q_t);       // using the real fitness function
12.      D = D ∪ Q_t;
13.      P_{t+1} = SelectNextPopulation(P_t, Q_t);
14.      R_{t+1} = SurrogateLocalSearch(P_t, A);
15.      t = t + 1;
16. end while
```

Fig. 1. Main algorithm of our proposed MO-MAMA.

with the new nondominated solutions found in the population $\mathcal{P}$. If the number of solutions is greater than N, then we define k-means ($k = N$) from $\mathcal{A}$. In this way, the archive is updated with the nearest solutions to each mean. This procedure is called *UpdateArchive* in the algorithm of Figure 1.

Generating offspring population: We consider the set $\mathcal{D}$ as the set of all solutions obtained by the MOEA, and $\mathcal{R}$ as the set of solutions obtained by the local search mechanism. Furhermore, we assume that our approach will eventually converge to the Pareto optimal set (or, at least, to a reasonably good approximation of it). Therefore, in the last generations of the algorithm, a well-distributed sample of the Pareto set is achieved and maintained in $\mathcal{D}$. For this, the improvement mechanism (which approximates solutions to the Pareto optimal set) generates solutions, which, when evaluated into the meta-model, correspond to good approximations of the real fitness values. Furthermore, since the set $\mathcal{R}$ is the result of an improvement procedure, we consider that both the $\mathcal{R}$ set and the $\mathcal{A}$ set (the nondominated set) have solutions of similar quality. Based on the previous discussion, crossover takes place between each individual of the population $\mathcal{P}$ (the current population) and an individual which can be chosen from either $\mathcal{R}$ or $\mathcal{A}$. Therefore, we define the parents for the crossover operator according to the following procedure:

$$\begin{aligned} parent^1 &= x_i \in \mathcal{P} \quad \forall i = 1, \ldots, N \\ parent^2 &= \begin{cases} y \in \mathcal{R}, & \textbf{if } \left(g < 1 - \frac{|\mathcal{A}|}{2N}\right) \\ y \in \mathcal{A}, & \textbf{otherwise} \end{cases} \end{aligned} \tag{1}$$

where g is a uniformly distributed random number within $(0, 1)$ and y is a solution randomly chosen from $\mathcal{A}$ or $\mathcal{R}$. Clearly, when the archive pool $\mathcal{A}$ is full, $|\mathcal{A}| = N$ and equation (1) guarantees to choose a solution from either $\mathcal{R}$

or $\mathcal{A}$ (both have the same probability). The mutation operator is applied (with a certain probability) to each child generated by the crossover operator. In this work, we adopted the genetic operators from the NSGA-II [11] (Simulated Binary Crossover (SBX) and Parameter-Based Mutation (PBM)). Figure 2 shows the complete procedure for creating the offspring population.

```
// P = current population
// R = set of solutions obtained by the local search mechanism
// A = external archive
1. Q = ∅;
2. forall (x ∈ P)do
3.     parent^1 = x;
4.     Define parent^2 according to equation (1);
5.     Generate child^1 and child^2 performing SBX(parent^1, parent^2);
6.     y^1 = PBM(child^1) and y^2 = PBM(child^2);
7.     Q = Q ∪ {y^1, y^2};
8. end forall
9. return Q;
```

Fig. 2. Creating the offspring population ($CreateOffspring(\mathcal{P}, \mathcal{R}, \mathcal{A})$).

Local search mechanism: The main goal of the local search mechanism incorporated into our meta-model is to find new solutions nearby the solutions provided by the MOEA (such solutions should be at least nondominated with respect to the current and previous populations). While the local search engine explores promising areas into the meta-model, the MOEA performs a broader exploration of the search space. All this procedure is called $SurrogateLocalSearch$ within the algorithm of Figure 1.

Approximating solutions: There exist several mathematical programming methods designed for solving multi-objective optimization problems (see e.g., [12, 13]). Here, we are interested in solving the *weighted Tchebycheff problem* which is of the form:

$$\min_{x \in \mathbb{R}^n} \max_{i=1,\ldots,k} \{w_i | f_i(x) - z_i^*|\} \tag{2}$$

where z^* denotes the ideal vector, w is a vector in $\mathbb{R}^k$ such that $\mathbf{0} < w$ and $\sum_{i=1}^{k} w_i = 1$ (a convex combination of weights). It is well known that, for each Pareto optimal point there exists a weighting vector $\mathbf{0} < w \in \mathbb{R}^k$ such that it is the optimum solution of (2). Unfortunately, if the solution of the Tchebycheff problem is not unique, the solutions generated will be weakly Pareto optimal. In order to identify the Pareto optimal solutions, the following *augmented weighted Tchebycheff* problem is suggested:

$$\min_{x \in \mathbb{R}^n} \max_{i=1,\ldots,k} \{w_i | f_i(x) - z_i^*|\} + \rho \sum_{i=1}^{k} |f_i(\boldsymbol{x}) - z_i^\star| \tag{3}$$

where ρ is a sufficiently small positive scalar and z^* represents the utopian vector.

Initially, a set of n_w well-distributed weighted vectors $W \subset \mathbb{R}^k$ is defined (for this task, we use the method proposed by Zhang and Li [14]). The approximate solutions to the Pareto optimal set are obtained by solving the Tchebycheff problem for each weighted vector. For each weighted vector $w_j \in W$, a set of solutions λ_j is found, which consists of all the solutions evaluated so far into the meta-model by solving the above Tchebycheff problem. The utopian vector z^* is constructed with the minimum of each objective function at the current generation. Moreover, here, we use the well-known *pattern search* (or Hooke and Jeeves) algorithm [15], in order to solve each Tchebycheff problem. Clearly, all the candidate solutions are evaluated into the surrogate model. The initial search point x_s for solving the first problem corresponding to the weighted vector w_1, is defined according to the next equation:

$$x_s = x^* \in \{\mathcal{P}_t \cup \mathcal{A}\}, \text{ such that } x^* \text{ minimizes equation (3)} \tag{4}$$

where $\mathcal{P}_t$ and $\mathcal{A}$ are the population and the external archive at the current generation, respectively. The remaining sets λ_j $(j = 2, \ldots, n_w)$ are obtained by solving the Tchebycheff problem for the weighted vector w_j. The initial search point for obtaining λ_j is given by the decision vector which minimizes the Tchebycheff problem for the weighted vector w_{j-1}. Therefore, we define the set $\mathit{\Lambda}$ as the union of all the sets λ found by solving the n_w Tchebycheff problems, that is:

$$\mathit{\Lambda} = \bigcup_{j=1}^{n_w} \lambda_j \tag{5}$$

Generating new solutions: We consider $\mathit{\Lambda}$ to be the set of solutions found by the above process. Furthermore, we consider:

$$P(\exists p \in \mathbb{R}^n : ||q^* - p|| < \delta \text{ and } q^* \not\prec p) = 1 \tag{6}$$

for any small $\delta \in \mathbb{R}_+$. Here, q^* is at least a locally nondominated solution. That is, the probability that p is nondominated with respect to q^* is equal to one, which implies that p is also nondominated. We generate more approximate solutions using an evolutionary algorithm available within the meta-model. The differential evolution (DE) [16] algorithm with a *DE1/rand/bin* strategy is adopted for this task. Furthermore, the following dominance rule is used to select the new individuals for the next generation:

$$x_{i,g+1} = \begin{cases} x^*_{i,g} & \textbf{if } (x^*_{i,g} \prec x_{i,g}) \\ & \textbf{or } (x^*_{i,g} \text{ and } x_{i,g} \text{ are nondominated}) \\ x_{i,g} & \textbf{otherwise} \end{cases} \tag{7}$$

where x_i is a solution in the current population, x^* is the test vector and g is the current iteration of the DE algorithm. For more details about DE see [17]. The initial population is given by $\mathcal{G}_0 = \mathit{\Lambda}$. Each new individual $x_{i,g+1}$ is stored

(or not) in an external archive $\mathcal{L}$ according to the dominance rule. The archive strategy can make that the set of solutions $\mathcal{L}$ increases or decreases its size. Given the probability defined by equation (6), we generate more nondominated solutions from $\mathcal{L}$. Thus, the next population for the DE algorithm is defined by $\mathcal{G}_{g+1} = \mathcal{L}$.

Since all the solutions in the archive $\mathcal{L}$ are nondominated, we can say that the algorithm has converged (at least to a local Pareto front) when it has obtained N different nondominated solutions from the evolutionary process. That is:

$$\textbf{if } |\mathcal{L}| = N \textbf{ then} \qquad \text{stop the DE algorithm} \tag{8}$$

Therefore, the solutions set $\mathcal{R}$ obtained by our local search mechanism is given by $\mathcal{R} = \mathcal{L}$. However, this stopping criteria is not always satisfied. Thus, we can define the $\mathcal{R}$ set by selecting N individuals from $\Lambda \cup \mathcal{L}$ using Pareto ranking after a certain number of iterations.

4 Comparison of Results

In order to assess the performance of our proposed approach, we compare it with respect to NSGA-II [11]. The test problems adopted are the ZDT (Zitzler-Deb-Thiele) test suite [18] (except from ZDT5, which is a binary test problem). We adopted three performance measures to assess our results: Inverted Generational Distance ($\mathcal{IGD}$) [19], Spacing ($\mathcal{S}$) [20] and the Set Coverage ($\mathcal{SC}$) [18].

4.1 Experimental Setup

As indicated before, we compared our proposed approach with respect to the NSGA-II. For each MOP, we performed 25 independent runs with each approach. The parameters used in the algorithms are shown below.

Since our approach adopts the same genetic operators included in the NSGA-II (SBX and PBM), we adopted the same parameter values for these operators in both algorithms, that is: crossover index $\eta_c = 15$ and mutation index $\eta_m = 20$. Furthermore, for both algorithms we used: crossover probability $P_c = 1.0$, mutation probability $P_m = \frac{1}{n}$ (where n represents the number of decision variables of the MOP) and a population size $N = 100$.

The Hooke-Jeeves algorithm was implemented with: $\delta_i = \frac{up_i - low_i}{2}$ (up_i and low_i are the upper and lower bounds of the i^{th} decision variable component, respectively), the reduction factor was set to $\alpha = 2$ and $\varepsilon = 1 \times 10^{-3}$. The differential evolution algorithm was implemented using a weighting factor $F = 0.5$ and a crossover constant $CR = 1.0$. Finally, for the approximation phase, we set $n_w = 5$ (which is equal to 5% of the population size) as the number of weighted vectors which define the number of Tchebycheff problems. We should consider that more weight vectors implies more local search and with this, greater computational effort.

Table 1. Results for $\mathcal{IGD}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**0.000068** (0.000036)	0.055665 (0.005467)
ZDT2	**0.000186** (0.000400)	0.065110 (0.006909)
ZDT3	**0.000965** (0.000028)	0.055609 (0.006253)
ZDT4	0.151272 (0.033721)	**0.143483** (0.022090)
ZDT6	**0.000483** (0.000210)	0.023264 (0.001469)

Table 2. Results for $\mathcal{S}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**0.020216** (0.013534)	1.285481 (0.848476)
ZDT2	**0.034953** (0.062620)	1.690465 (1.171313)
ZDT3	**0.022872** (0.007906)	1.455995 (1.258330)
ZDT4	**6.522628** (8.981982)	19.108133 (24.636678)
ZDT6	0.415136 (0.320719)	**0.226461** (0.162008)

Table 3. Results for $\mathcal{SC}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**1.000000** (0.000000)	0.000000 (0.000000)
ZDT2	**0.979200** (0.048656)	0.000000 (0.000000)
ZDT3	**1.000000** (0.000000)	0.000000 (0.000000)
ZDT4	0.673600 (0.093590)	**0.684000** (0.073103)
ZDT6	**1.000000** (0.000000)	0.000000 (0.000000)

4.2 Discussion of results

Our results are summarized in Tables 1 to 3. Each table displays both the average (showing the best results in **boldface**) and the standard deviation (σ) of each performance measure, for each of the test problems adopted. Each run was restricted to $1,000$ fitness function evaluations. These results clearly show that our proposed approach (MO-MAMA) outperformed the NSGA-II in most of the test problems adopted (except for ZDT4), not only with respect to $\mathcal{IGD}$ but also with respect to $\mathcal{SC}$. It is worth noticing that the NSGA-II performed better with respect to $\mathcal{S}$, which indicates that it produced solutions with a better distribution. However, a better distribution of solutions is relevant only when a good approximation of the true Pareto front has been achieved. Since in our case, we were emphasizing efficiency (i.e., only a fairly limited number of fitness function evaluations was allowed). Furthermore, according to the Wilcoxon rank-sum test [21], our MO-MAMA is significantly better than NSGA-II over the $\mathcal{IGD}$ metric (which is the most important metric that we considered in this work) in most of the adopted test problems (except for ZDT4) with a significance level of 0.05. In the other hand, for the ZDT4 problem the Wilcoxon test did not show a significant variation. Therefore, we considered these results to be satisfactory. Finally, a picture of the convergence for the $\mathcal{IGD}$ metric in the ZDT1 problem is shown in Figure 3.

5 Conclusions and Future Work

We have proposed a multi-objective memetic algorithm assisted by support vector machines, with the aim of performing an efficient exploration of the search space. Our local search engine was based on a weighted Tchebycheff function and the Hooke-Jeeves method was adopted as our minimizer for each problem defined by each weighted vectors under consideration. Our proposed approach was found to be competitive with respect to the NSGA-II over a set of test functions taken from the specialized literature, when performing only $1,000$ fitness function evaluations.

As part of our future work, we plan to use our approach in problems having more objectives (three or more) and we aim to experiment with other search engines (e.g., with multi-objective scatter search [22]). The introduction of alternative approaches to improve the uniform distribution of our solutions as well as the use of more difficult test problems (e.g., the Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems [23] and the Walking-Fish-Group (WFG) test problems [24]) is also part of our future work. Finally, we are also interested in testing our approach with real-world problems having computationally expensive objective functions, and that is indeed part of our ongoing research.

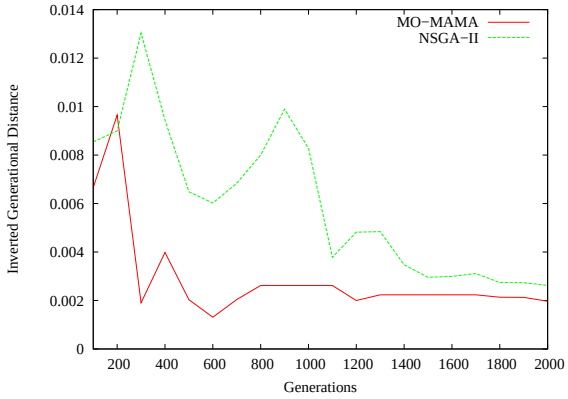

Fig. 3. Convergence for $\mathcal{IGD}$ metric over ZDT1 problem using 2,000 real fitness function evaluations

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. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing **9**(1) (2005) 3–12
3. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal **41**(4) (2003) 687–696
4. Emmerich, M.T.M., Naujoks, B.: Metamodel-assisted multiobjective optimisation strategies and their application in airfoil design. In: Adaptive Computing in Design and Manufacture VI, Berlyn, Germany: Springer-Verlag (2004) 249–260
5. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1) (2006) 50–66
6. Isaacs, A., Ray, T., Smith, W.: An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization. In: ACAL. (2007) 257–268

7. Georgopoulou, C.A., Giannakoglou, K.C.: A multi-objective metamodel-assisted memetic algorithm with strengthbased local refinement. Engineering Optimization **41**(10) (2009) 909–923
8. Vapnik, V., Golowich, S.E., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. In: Advances in Neural Information Processing Systems 9, MIT Press (1997) 281–287
9. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2) (1979) 239–245
10. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume 1., University of California Press (1967) 281–297
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation **6**(2) (2002) 182–197
12. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston, Massachuisetts (1999)
13. Hillermeier, C.: Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach. Birkhuser Basel (2000)
14. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation **11**(6) (2007) 712–731
15. Hooke, R., Jeeves, T.A.: "direct search" solution of numerical and statistical problems. J. ACM **8**(2) (1961) 212–229
16. Storn, R.M., Price, K.V.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, Berkeley, CA (1995)
17. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution. A Practical Approach to Global Optimization. Springer, Berlin (2005) ISBN 3-540-20950-6.
18. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
19. 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)
20. 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)
21. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6) (1945) 80–83
22. Nebro, A.J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J.J., Beham, A.: AbYSS: Adapting Scatter Search to Multiobjective Optimization. IEEE Transactions on Evolutionary Computation **12**(4) (2008) 439–457
23. 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
24. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE Transactions on Evolutionary Computation **10**(5) (2006) 477–506