An Extension of Generalized Differential Evolution for Multi-objective Optimization with Constraints

Saku Kukkonen and Jouni Lampinen

Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20
FIN-53851 Lappeenranta, Finland
Saku.Kukkonen@lut.fi

Abstract. In this paper an extension of Generalized Differential Evolution for constrained multi-objective (Pareto-)optimization is proposed. The proposed extension adds a mechanism for maintaining extent and distribution of the obtained non-dominated solutions approximating a Pareto front. The proposed extension is tested with a set of five benchmark multi-objective test problems and results are numerically compared to known global Pareto fronts and to results obtained with the elitist Non-Dominated Sorting Genetic Algorithm and Generalized Differential Evolution. Results show that the extension improves extent and distribution of solutions of Generalized Differential Evolution.

Keywords: multi-objective optimization, Pareto-optimization, constraint handling, evolutionary algorithms, differential evolution

1 Introduction

Many situations in engineering and economics deal with optimization. One may want to optimize, *e.g.*, manufacturing processes, shape of products, and number of different products to be manufactured. Typical goals are minimizing costs, maximizing profits, and improving performance. Several natural aspects limit feasible solutions, *e.g.*, resources may cause limitations and/or the number of products cannot be a negative number.

Optimization is an intensively studied problem field in mathematics. However, functions to be optimized in traditional mathematics are relatively simple (continuous, convex, unimodal, differentiable, *etc.*), yet functions to be optimized in practice are often far more complicated (discontinuous, non-convex, multi-modal, non-differentiable, *etc.*). In such cases various stochastic optimization methods have shown their effectiveness.

Most optimization research deals with single-objective optimization problems. The basic nature of many optimization problems is, however, multi-objective and these problems are usually first converted to single-objective problems.

Single-objective problems are commonly considered easier to solve but conversion from a multi-objective problem to a single-objective problem requires some *a priori* knowledge which is not necessarily available or which is hard to determine, *e.g.*, the relative importance of each individual sub-objective. For this reason interest exists in solving multi-objective problems in multi-objective form.

Several extensions of Differential Evolution (DE) for multi-objective optimization have already been proposed [1–5]. Most of these methods use a non-dominated sorting for reproduction in each generation and some distance metric to prevent crowding.

This paper continues with the following parts: In Section 2 the concept of multi-objective optimization with constraints is handled briefly. Section 3 describes Differential Evolution algorithm and Section 4 describes proposed extension for constrained multi-objective optimization. Section 5 describes experiments and finally conclusions are given in Section 6.

2 Multi-objective Optimization with Constraints

Many practical problems have multiple objectives. For example, designing a wing of an aircraft may have objectives such as maximizing strength, minimizing weight, minimizing manufacturing costs, maximizing lifting force, minimizing drag, *etc.* Multiple objectives are almost always more or less conflicting.

Several aspects cause constraints to problems. In the previous example of the wing, the thickness of the metal parts used must be a positive number, shape limitations exist, some parts are available only in some predefined standard sizes, *etc.* Constraints can be divided into box or boundary constraints and constraint functions. Boundary constraints are used when the value of some optimized variable is limited to some range and constraint functions are representing more complicated constraints which are expressed as functions.

Multi-objective problems are often converted to single-objective problems by predefining weighting factors for different objectives, expressing the relative importance of each objective. However, this is impossible in many cases because a decision-maker does not necessarily know beforehand how different objectives should be weighted. Thus, a more convenient way is to keep multiple objectives of multi-objective problems and try to solve them in this form even though this may be harder to do in practice. Optimizing several objectives simultaneously without articulating the relative importance of each objective *a priori*, is often called Pareto-optimization [6]. An obtained solution is Pareto-optimal if none of the objectives can be improved without impairing at least one other objective [7, p. 11–12]. If the obtained solution can be improved in such way that at least one objective improves and other objectives do not decline, then the new solution dominates the original solution. A set of Pareto-optimal solutions form a Pareto front. An approximation of the Pareto front is called a set of non-dominated solutions because the solutions in this set are not dominating each other in the space of objective functions. From the set of non-dominated solutions the decision-maker may select one which has suitable values for different objectives.

This can be viewed as *a posteriori* articulation of the decision-makers preferences concerning the relative importance of each objective.

A mathematically constrained multi-objective optimization problem can be presented in the form [7, p. 37]

$$\begin{array}{l} \text{minimize } \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})\} \\ \text{subject to } \boldsymbol{x} \in S = \{\boldsymbol{x} \in \mathbf{R}^D | \boldsymbol{g(x)} = (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), \ldots, g_M(\boldsymbol{x}))^T \leq \mathbf{0}\} \end{array} \tag{1}$$

Thus, there are K functions to be optimized and M constraint functions.

The major part of earlier mathematical research has concentrated on optimization problems where the functions are linear, differentiable, convex, or otherwise mathematically well behaving. However, in practical problems objective functions are often nonlinear, non-differentiable, discontinuous, multi-modal, *etc.* and no presumptions can be made about their behavior. Variables may also be integers or discrete instead of being continuous. Most traditional optimization methods cannot handle such complexity or do not perform well in these cases in which the assumptions they are based on do not hold. For such problems stochastic optimization methods such as Simulated Annealing (SA) and Evolution Algorithms (EAs) have been demonstrated to be effective because they do not rely on any assumptions concerning the objective and constraint functions.

3 Differential Evolution

The Differential Evolution (DE) algorithm [8, 9] [10, pp. 79–108] belongs to the family of Evolution Algorithms and was introduced by Storn and Price in 1995 [11]. Design principles in DE were simplicity, efficiency, and use of floating-point encoding instead of binary numbers.

Like in a typical EA, the idea in DE is to have some random initial population which is then improved using selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides an appropriate stopping condition.

A trial vector $\boldsymbol{u}_{i,G}$ created by mutation and crossover operations is compared to an old objective vector $\boldsymbol{x}_{i,G}$. Here i is an index of the vector in the population and G is a generation index. If the trial vector has equal or lower objective value, then it replaces the old vector. This selection operation can be presented as follows [10, p. 82]:

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} \text{ if } & f(\boldsymbol{u}_{i,G}) \leq f(\boldsymbol{x}_{i,G}) \\ \boldsymbol{x}_{i,G} \text{ otherwise} \end{cases} \tag{2}$$

The average objective value of the population will never increase, because the trial vector replaces the old vector only if it has equal or lower objective value.

4 An Extension of Generalized Differential Evolution

Generalized Differential Evolution (GDE) [12–17] extends the selection operation of the basic DE algorithm for constrained multi-objective optimization. GDE has been demonstrated to have good convergence properties but distribution of solutions and extent of the obtained non-dominated front need to be improved. GDE does not contain any mechanism for maintaining these. As an attempt to improve GDE from this point of view, a modified selection operation for GDE is proposed in this paper. The proposed selection operation for M constraint and K objective functions is presented formally in (3).

$$\boldsymbol{x}_{i,G+1} = \begin{cases} \boldsymbol{u}_{i,G} \text{ if } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \exists j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) > 0 \\ \wedge \\ \forall j \in \{1,\ldots,M\} : g'_j(\boldsymbol{u}_{i,G}) \leq g'_j(\boldsymbol{x}_{i,G}) \end{array} \right. \\ \vee \\ \left\{ \begin{array}{l} \forall j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) \leq 0 \\ \wedge \\ \exists j \in \{1,\ldots,M\} : g_j(\boldsymbol{x}_{i,G}) > 0 \end{array} \right. \\ \vee \\ \left\{ \begin{array}{l} \forall j \in \{1,\ldots,M\} : g_j(\boldsymbol{u}_{i,G}) \leq 0 \wedge g_j(\boldsymbol{x}_{i,G}) \leq 0 \\ \wedge \\ \left\{ \begin{array}{l} \forall k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) \leq f_k(\boldsymbol{x}_{i,G}) \\ \vee \\ \left\{ \begin{array}{l} \neg \left[\forall k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) \geq f_k(\boldsymbol{x}_{i,G}) \wedge \right. \\ \quad \left. \exists k \in \{1,\ldots,K\} : f_k(\boldsymbol{u}_{i,G}) > f_k(\boldsymbol{x}_{i,G}) \right] \\ \wedge \\ d_{\boldsymbol{u}_{i,G}} \geq d_{\boldsymbol{x}_{i,G}} \end{array} \right. \end{array} \right. \end{array} \right. \end{array} \right. \\ \boldsymbol{x}_{i,G} \text{ otherwise} \end{cases} , \quad (3)$$

where $g'_j(\boldsymbol{x}_{i,G}) = \max(g_j(\boldsymbol{x}_{i,G}), 0)$ and $g'_j(\boldsymbol{u}_{i,G}) = \max(g_j(\boldsymbol{u}_{i,G}), 0)$ are representing the constraint violations, and d_i is a distance measure for measuring the distance from a particular solution i to its neighbor solutions.

The selection rule given in (3) selects the trial vector $\boldsymbol{u}_{i,G}$ to replace the old vector $\boldsymbol{x}_{i,G}$ in the following cases:

1. Both the trial vector and the old vector violate at least one constraint but the trial vector does not violate any of the constraints more than the old vector does.
2. The old vector violates at least one constraint whereas the trial vector is feasible.
3. Both vectors are feasible and
   – the trial vector dominates the old vector or has equal value for all objectives, or
   – the old vector does not dominate the trial vector and the old vector resides in a more crowded region of the objective space.

Otherwise the old vector $\boldsymbol{x}_{i,G}$ is preserved.

The basic idea in the selection rule is that the trial vector is required to dominate the compared old population member in constraint violation space or in objective function space, or at least provide an equally good solution as the old population member. If both vectors are feasible and they do not dominate each other, then the one residing in a less crowded region of the objective space is chosen to the population of the next generation. The principle of constraint handling is effectively rather similar to the method described in [18] even though the formulation is different. The main difference is in the case of two infeasible solutions. In this case the selection rule given in (3) compares solutions based on dominance of the constraint violations whereas the selection method described in [18] compares solutions based on a sum of the constraint violations which needs evaluation of all constraint functions and normalization of their values.

The whole selection rule given in (3) is effectively almost same as a constrained tournament method in [19, pp. 301–308]. In the selection rule given in (3) the trial vector is preferred over the old vector also in the cases when the trial vector is equally good as the old vector, *i.e.*, constraint violations are equal or objective function values are equal.

The selection rule given in (3) can be implemented in such a way that the number of function evaluations is reduced because not always all the constraints and objectives need to be evaluated, *e.g.*, inspecting constraint violations (even one constraint) is often enough to determine which vector to select for the next generation [13, 14]. However, in the case of feasible solutions all the objectives need to be evaluated which was not always necessary in earlier GDE. This will increase the total number of function evaluations as well as execution time. Also calculation of the distance measure, d_i, will increase execution time. In principle, any measure of distance from a solution to its neighbor solutions can be applied. A crowding distance [19, pp. 248–249] was applied here as the distance measure, d_i, because it does not need any extra parameters.

After the selected number of generations the final population presents a solution for the optimization problem. The non-dominated solutions can be separated from the final population if desired. There is no sorting of non-dominated solutions during the optimization process.

Later on in this paper the proposed method with the selection rule given in (3) is called *Generalized Differential Evolution 2 (GDE2)*. The selection rule given in (3) handles any number M of constraints and any number K of objectives. When $M = 0$ and $K = 1$, the selection rule is identical to the selection rule of the basic DE algorithm.

Usually large values (such as 0.9) are suggested as initial settings for the crossover rate CR and mutation factor F in the case of single-objective problems. In the case of multiple objectives, it was observed that using a large crossover rate often leads to faster convergence along one objective compared to another [15–17]. This causes the solution to converge to a single point of the Pareto front, which is not desired. Based on this observation, our initial recommendations for the control parameter values used for multi-objective optimization problems are $CR \in [0.05, 0.5]$ and $F \in [0.05, 1+)$ for initial settings. In line with these observations, use of small values for CR was also reported by other researchers in [2, 3]. However, the current recommendations are based on limited experimentation, and the problem of selecting the control parameter values is remaining mostly open.

5 Experiments

GDE and GDE2 were implemented in C and tested with a set of five bi-objective benchmark problems described in [20] and [21, pp. 57–59]. These problems are known as ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 [19, pp. 356–360]. They are designed to test the ability of a multi-objective optimization method to handle convexity (ZDT1), non-convexity (ZDT2), discontinuity (ZDT3), multi-modality (ZDT4), and non-uniformity (ZDT6) of the Pareto front.

5.1 Experimental Results and Discussions

In all the test problems the size of the population was 100, the number of generations was 250, and the control parameter values for GDE and GDE2 were $CR = 0.05$ and $F = 0.1$. In preliminary tests control parameter values 0.05, 0.1, 0.2, 0.3, and 0.4 were tested and suitable crossover rate and mutation factor were thereby approximately determined. It was noticed that suitable values for the crossover rate and mutation factor should be drawn from a rather narrow range for some problems, *e.g.*, problem ZDT4, while the underlying reason for this remains open.

The results of a single run of GDE and GDE2 for solving the multi-objective benchmark problems are shown in Fig. 1, where known global Pareto fronts and the results obtained with the Strength Pareto Evolutionary Algorithm (SPEA) [22] and the elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [23] are also shown for comparison and visual assessment.

Tests for NSGA-II, GDE, and GDE2 were repeated 100 times with different random number generator seeds and the results were compared with different metrics. NSGA-II was selected for comparison because of its good performance in previous comparison tests [23] and since it is well known within the multi-objective optimization community.

Closeness to the Pareto front was measured with an error ratio (ER) and a generational distance (GD) [19, pp. 324–327]. Diversity of the obtained solution was measured using spacing (S), spread (Δ), and maximum spread (D) metrics [19, pp. 328–331]. Smaller values for the error ratio, generational distance, spacing, and spread are preferable. The optimal value for the maximum spread is 1.

Average numbers of needed function evaluations for GDE and average execution times for the methods are reported in Table 1. For NSGA-II and GDE2 2×25100 function evaluations were needed on each run. All the tests were run on a Sun Sparc Ultra2. Table 2 contains the performance measurements solving the benchmark problems. Solutions contained the non-dominated members of the final population.

The results show that GDE2 improved extent and diversity of solutions over GDE without impairing the convergence property of GDE and increasing execution time only by little. NSGA-II was slightly better than GDE2 in most of the problems according to the metrics but NSGA-II needed more execution

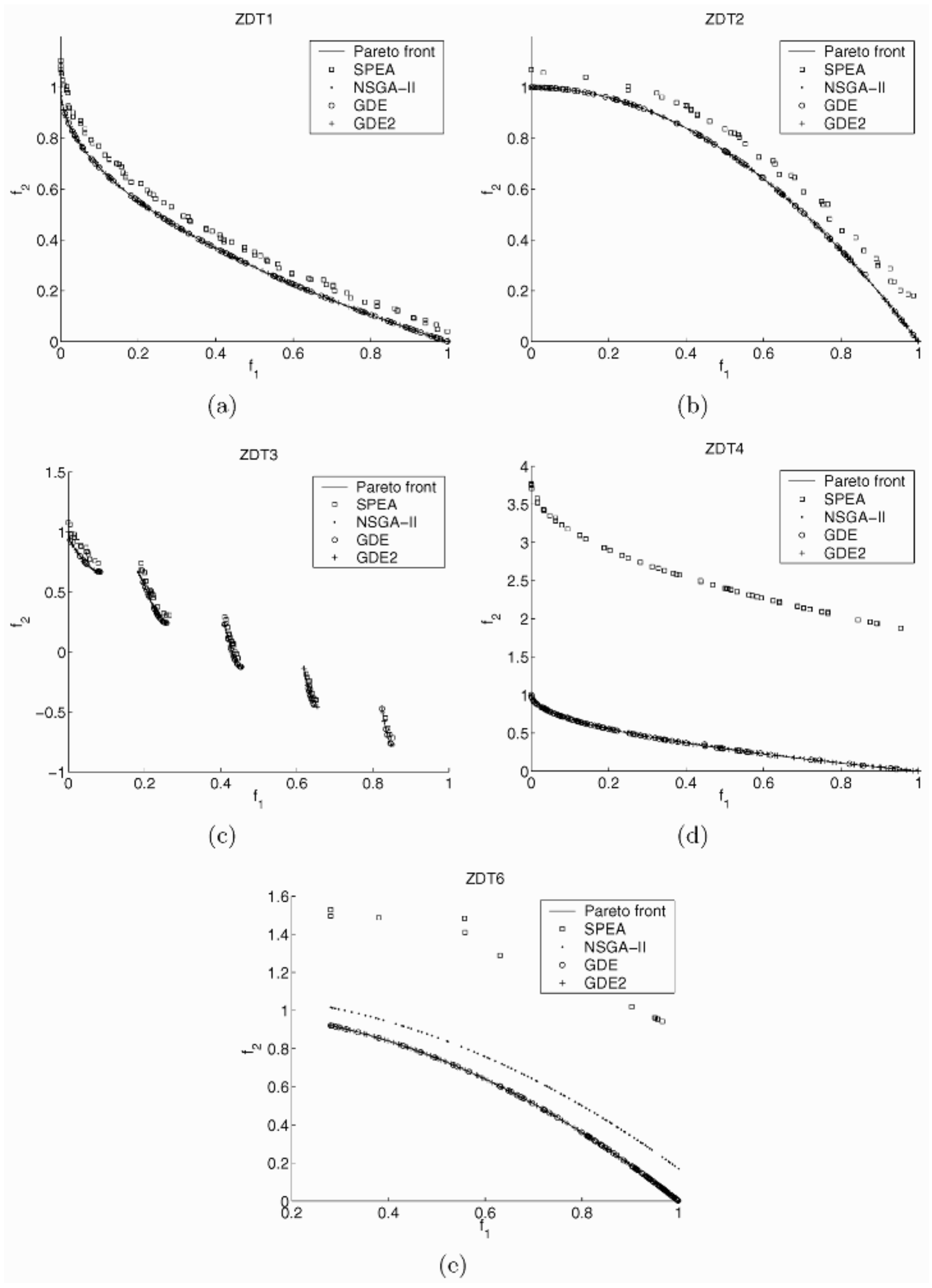

Fig. 1. Global Pareto front and solutions obtained with SPEA, NSGA-II, GDE, and GDE2 for a) ZDT1 b) ZDT2 c) ZDT3 d) ZDT4 e) ZDT6.

Table 1. Average execution times of NSGA-II, GDE, and GDE2 solving multi-objective benchmark test problems. Average number of needed function evaluations (f_1 and f_2) of GDE are also shown. Standard deviations are in parenthesis.

	NSGA-II	GDE			GDE2
	Execution time	Execution time	f_1	f_2	Execution time
ZDT1	4.0040(0.3661) s	1.0166(0.0071) s	25100(0.0)	24079.2(29.3)	1.0875(0.0091) s
ZDT2	4.0315(0.2389) s	1.0230(0.0163) s	25100(0.0)	24080.2(27.9)	1.0820(0.0067) s
ZDT3	3.9735(0.3963) s	1.0477(0.0085) s	25100(0.0)	24078.3(32.8)	1.1169(0.0081) s
ZDT4	3.9341(0.4769) s	0.5537(0.0085) s	25100(0.0)	23280.5(37.1)	0.6329(0.0151) s
ZDT6	4.0762(2.6311) s	0.5782(0.0097) s	25100(0.0)	22885.3(68.0)	0.6554(0.0106) s

Table 2. Means of the solution cardinality ($\aleph$), error ratio (ER), generational distance (GD), spacing (S), spread (Δ), and maximum spread (D) of NSGA-II, GDE, and GDE2 for the multi-objective benchmark test problems. Standard deviations are in parenthesis.

ZDT1	$\aleph$	ER	GD	S	Δ	D
NSGA-II	91.7(2.7)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	**0.418**(0.036)	**1.000**(0.000)
GDE	**98.9**(1.3)	**0.000**(0.000)	**0.000**(0.000)	0.012(0.003)	0.764(0.045)	0.949(0.028)
GDE2	83.6(4.7)	**0.000**(0.000)	**0.000**(0.000)	0.011(0.001)	0.518(0.048)	**1.000**(0.000)

ZDT2	$\aleph$	ER	GD	S	Δ	D
NSGA-II	74.7(37.1)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	0.535(0.236)	0.800(0.402)
GDE	78.1(9.8)	**0.000**(0.001)	**0.000**(0.000)	0.018(0.005)	0.864(0.067)	0.978(0.024)
GDE2	**87.9**(4.4)	0.020(0.141)	**0.000**(0.000)	0.010(0.001)	**0.470**(0.052)	**1.000**(0.001)

ZDT3	$\aleph$	ER	GD	S	Δ	D
NSGA-II	**92.9**(2.3)	**0.000**(0.000)	**0.000**(0.000)	**0.006**(0.001)	**0.573**(0.036)	0.971(0.083)
GDE	69.1(4.8)	0.003(0.007)	**0.000**(0.000)	0.018(0.008)	1.044(0.068)	0.968(0.020)
GDE2	40.3(3.8)	0.007(0.014)	**0.000**(0.000)	0.020(0.005)	0.712(0.063)	**1.000**(0.001)

ZDT4	$\aleph$	ER	GD	S	Δ	D
NSGA-II	**95.5**(16.8)	**0.031**(0.113)	**0.001**(0.001)	**0.007**(0.001)	**0.389**(0.113)	0.971(0.172)
GDE	66.7(25.4)	0.235(0.357)	0.003(0.007)	0.026(0.015)	0.775(0.123)	0.968(0.052)
GDE2	55.2(21.1)	0.318(0.384)	0.004(0.006)	0.019(0.010)	0.532(0.067)	**1.006**(0.025)

ZDT6	$\aleph$	ER	GD	S	Δ	D
NSGA-II	89.5(2.9)	1.000(0.000)	0.008(0.001)	**0.008**(0.001)	0.513(0.031)	0.965(0.006)
GDE	**99.8**(2.1)	0.010(0.100)	0.002(0.021)	0.017(0.005)	1.018(0.079)	0.988(0.050)
GDE2	97.2(2.3)	**0.000**(0.000)	**0.000**(0.000)	**0.008**(0.001)	**0.388**(0.046)	**1.000**(0.001)

time than GDE and GDE2, probably because of additional operations, *e.g.*, the non-dominated sorting. GDE2 outperforms NSGA-II in the problem ZDT6.

6 Conclusions and Future Research

In this paper an extension of Generalized Differential Evolution algorithm is proposed. The extension, GDE2, adds to GDE a mechanism for improving extent and diversity of the obtained Pareto front approximation without impairing convergence speed of GDE and increasing execution time only little. GDE2 is demonstrated to be effective, and does not introduce any extra control parameters to be preset by the user.

GDE and GDE2 were tested with a set of five benchmark multi-objective test problems. The numerical results show that GDE2 is able to provide a solution for all the test problems and performs comparably to NSGA-II and GDE providing a relatively good approximation of the Pareto front. However, the proposed method was found rather sensitive to control parameter values.

The effect of parameters on the optimization process, extensive comparison of GDE2 with latest multi-objective evolutionary algorithms and test problems, and applying GDE2 for practical multi-objective problems remains among the topics to be studied.

Acknowledgements

Authors wish to thank K. Deb and E. Zitzler for providing source codes of NSGA-II and results of SPEA respectively, and reviewers for useful commends.

References

1. Chang, C.S., Xu, D.Y., Quek, H.B.: Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mass transit system. IEE Proceedings on Electric Power Applications **146** (1999) 577–583
2. Abbass, H.A., Sarker, R.: The Pareto Differential Evolution algorithm. International Journal on Artificial Intelligence Tools **11** (2002) 531–552
3. Madavan, N.K.: Multiobjective optimization using a Pareto Differential Evolution approach. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii (2002) 1145–1150
4. Babu, B.V., Jehan, M.M.L.: Differential Evolution for multi-objective optimization. In: Proceedings of the 2003 Congress on Evolutionary Computation, CEC'03, Canberra, Australia (2003) 2696–2703
5. Feng Xue, Arthur C. Sanderson, R.J.G.: Multi-objective differential evolution and its application to enterprise planning. In: Proceedings of IEEE International Conference on Robotics and Automation, Taiwan (2003) 3535–3541
6. Pareto, V.: Cours D'Economie Politique. Libraire Droz, Geneve (1964 (the first edition in 1896))
7. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1998)
8. Storn, R., Price, K.V.: Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11** (1997) 341–359
9. Lampinen, J., Storn, R.: Differential Evolution. In: New Optimization Techniques in Engineering. Springer (2004) 123–166
10. Corne, D., Dorigo, M., Glover, F.: New Ideas in Optimization. McGraw-Hill, London (1999)
11. Storn, R., Price, K.V.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI (1995) [Online] Available: ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.gz, 3.5.2004.

12. Lampinen, J.: A constraint handling approach for the Differential Evolution algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii (2002) 1468–1473
13. Lampinen, J.: DE's selection rule for multiobjective optimization. Technical report, Lappeenranta University of Technology, Department of Information Technology (2001) [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 3.5.2004.
14. Lampinen, J.: Multi-constrained nonlinear optimization by the Differential Evolution algorithm. Technical report, Lappeenranta University of Technology, Department of Information Technology (2001) [Online] Available: http://www.it.lut.fi/kurssit/03-04/010778000/DECONSTR.PDF, 3.5.2004.
15. Kukkonen, S., Lampinen, J.: A Differential Evolution algorithm for constrained multi-objective optimization: Initial assessment. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Innsbruck, Austria (2004) 96–102
16. Kukkonen, S., Lampinen, J.: Mechanical component design for multiple objectives using Generalized Differential Evolution. In: Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture (ACDM2004), Bristol, United Kingdom (2004) 261–272
17. Kukkonen, S., Lampinen, J.: Comparison of Generalized Differential Evolution algorithm to other multi-objective evolutionary algorithms. In: Proceedings of the 4th European Congress on Computational Methods in Applied Sciences and Engineering, Jyväskylä, Finland (2004) Accepted for publication.
18. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186** (2000) 311–338
19. Deb, K.: Multi-Objective Optimization using Evolutionary algorithms. John Wiley & Sons, Chichester, England (2001)
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation **8** (2000) 173–195 Also available: ftp.tik.ee.ethz.ch/pub/people/zitzler/ZDT2000.ps, 15.1.2004.
21. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany (1999)
22. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. IEEE Transactions on Evolutionary Computation **4** (1999) 257–271
23. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197