An Evolutionary Programming Algorithm for Multi-Objective Optimisation

Andrew Lewis
Griffith University
Nathan Campus
Brisbane, QLD 4111, Australia
a.lewis@griffith.edu.au

David Abramson
Monash University
Caulfield East
Melbourne, VIC 3145, Australia
David.Abramson@infotech.monash.edu.au

Abstract: This paper describes a new Evolutionary Programming optimisation algorithm and a method of its application to multi-objective optimisation problems. Computational results are presented demonstrating the algorithm's ability to find Pareto-optimal solutions for a real-world problem in radio-frequency component design.

1 Introduction.

In the engineering design process there is now widespread use of sophisticated and realistic numerical simulations of physical phenomena influencing design decisions. However, much of this use is on an *ad hoc* basis, manually investigating a limited set of design alternatives. Considerable improvements could be achieved with a capability to perform automatic optimisation, minimising or maximising some derived quantity, a measure of "fitness" of the design. This is an extremely computationally intensive process when models must be run tens, or maybe hundreds, of times to effectively search design parameter space. Current High Performance Computing systems mostly derive their capacity from parallel architectures so for optimisation methods to be practical and effective the algorithms used must preferably have a large degree of concurrency.

Over the past decade the computational capacity available to scientists and engineers has increased to the point where population-based methods of optimisation, in which many instances of a problem are treated simultaneously, have become practical for the solution of real-world problems (for surveys of methods and applications to multi-objective problems see Coello 1999 or Van Veldhuizen 1999). Moreover, in engineering design, the evaluation of the objective function is so much slower than the rest of the algorithm, that such codes are capable of giving excellent speedup in spite of the need for global communication on each iteration. **Genetic Algorithms** (GA) now may be frequently encountered in application to engineering problems (for examples, see Alander 1995.) Often the design needs of a particular engineering problem require the optimisation of more than one objective. In addition to describing a new Evolutionary Programming optimization algorithm, this paper outlines a method of applying the algorithm to such multi-objective problems.

2 Evolutionary computation

In the general case, evolutionary computation seeks to use insight into natural processes to inform population-based computational methods. These resulting Evolutionary Algorithms are popularly differentiated into three main classes (Bäck 1996) namely, Genetic Algorithms, Evolutionary Strategies and Evolutionary Programming.

In the general, population-based method, multiple instances of a problem, each represented by a vector of parameter values, are subject to various operators so that a population of problem instances in a "parent" generation evolve into a "child" population. This process is repeated through a number of generations.

Widely known, **Genetic Algorithms** are generally accepted as having been developed by Holland (Holland 1975). Genetic Algorithms, in contrast to Evolutionary Strategies and Evolutionary Programming, work on bitstrings of fixed length. For problems of continuous variable parameters the bitstring has a mapping to the vector of parameters, which generally implies they are capable of returning approximate, rather than exact, global minima. The bitstring representations are subject to processes of recombination through a crossover operator, an analogue of genetic inheritance in sexual reproduction, mutation and selection.

Evolutionary Strategies, a joint development of Bienert, Rechenberg and Schwefel in the 1960s (see, for example, Schwefel 1965), operate on continuous parameters, using normally distributed mutation and recombination. Recombination is either discrete, choosing which parent will contribute each of the parameter values in turn to a child, or intermediate with child parameter values being formed from the (weighted) mean of parent parameters.

Evolutionary Programming was developed by Fogel (Fogel 1962) and refers to that class of methods in evolutionary computation that apply a (uniform) random mutation to each member of a population, generating a single offspring. However, unlike other methods, no

recombination operators are applied. Population members may be considered as representative of species, rather than individuals, so phenotypic effects are emphasised instead of genetic change. After mutation, selection takes place, and half the combined population of parents and offspring enter the next generation. Such methods are generally simple, robust and highly parallel. Underused for many years, they were further developed in the 1980s and their use became more widespread in single-objective optimisation. However, they remain underused in multi-objective optimisation. A 1999 review of 272 publications in multi-objective evolutionary algorithms found only one citation related to Evolutionary Programming (Van Veldhuizen 1999).

3 Self-organised Criticality

The theory of self-organised criticality gives an insight into emergent complexity in nature (Bak 1996, Bak 1993). Systems in stable equilibrium exhibit linear behaviour. The system's response to a disturbance is proportional to the size of the disturbance. Large fluctuations can only occur if several factors simultaneously combine to act in the same direction, which is unlikely to occur. Such a system, then, is also unlikely to adapt rapidly to the demands of an objective function.

At the other end of the spectrum, chaotic systems can react violently to change, as small perturbations of initial values are amplified in the system's response. Chaotic systems have no memory of their past states and cannot evolve.

However, at the transition from a stable system to chaos, complex behaviour can emerge. It is this critical state that may deliver efficient adaptation. Self-organised critical systems evolve to the critical state without any external organising force. This is advantageous because it implies no *a priori* information about the internal functioning of the systems is required to develop an effective means of optimising an objective function expressed in terms of externally exposed parameters and observed system response.

Bak contended that the critical state was "the most efficient state that can actually be reached dynamically". Inspection of many natural phenomena suggests the critical state is capable of efficient adaptation to environmental pressures using simple, robust systems. If optimisation is considered as the adaptation of a system described by its parameters to the selective pressure of an objective function, then it appears developing a critical state may be a highly effective method of optimisation.

Bak sought to model evolution of species with a simple model of inter-species interactions and selection. A number of species were arranged, arbitrarily and randomly, in a ring topology. At each time step, the least fit species and its two neighbours in the ring were replaced by randomly instantiated new species. This model was demonstrated to lead to complex behaviour, with gradual evolution of the fitness of the whole population.

Self-organised criticality is also exhibited by the "sandpile model" (Bak 1996). In this model, "grains" of sand are modelled numerically, stacking upon each other and toppling onto neighbouring stacks under simple rules. The structure of this simple system can be observed to evolve to an organised state with dynamics characteristic of criticality.

4 A New Evolutionary Programming Algorithm (EPSOC)

In earlier applications of self-organised criticality to optimisation, it has been proposed that a separately computed power-law extinction rate be imposed on a spatial diffusion model, or cellular GA (Krink 2001). Krink and Thomsen's model used pre-computed, stored dynamics of a sandpile model to control the size of extinction events in a diffusion model. The algorithm apparently does not attempt to evolve a population in a critical state, but indirectly imposes the observed behaviour of such a population.

In both the Bak-Sneppen nearest-neighbour, punctuated equilibrium model, and Krink and Thomsen's spatial diffusion model the population members are artificially arranged spatially: in a ring in the former, and in a toroidal, 2D grid in the latter. In contrast, by considering the trial solution parameter vectors as defining a location in an *n*-dimensional parameter space, the spatial behaviour of the Bak-Sneppen model is realized naturally in EPSOC.

In the following section we outline a new algorithm, called EPSOC, which is largely a straightforward implementation of Bak's model as an optimisation algorithm. It diverges in applying a high degree of greediness to the algorithm. Maintaining a large "elite" (in EPSOC, half the total population) can be viewed as a "constructive" operator. An analogous operator is "Maxwell's demon". Elitism has been clearly demonstrated to improved the performance of a GA (Zitzler 2000).

4.1 Single Objective Implementation

Restated, the general optimisation problem is:

$$\text{Minimize } f(\mathbf{x}) \text{ where: } f : \Re^n \rightarrow \Re^1 \text{ is an arbitrary non-linear function and } \mathbf{x} = \{x_0, ..., x_i, ..., x_n\}, x_i \in \Re \quad (1)$$

For a population-based method, the population, p, consists of a set of parameter vectors, $\mathbf{x}$: $p = \{\mathbf{x}_0, ..., \mathbf{x}_i, ..., \mathbf{x}_m\}$. For the real-world engineering design problems being considered, values for $f(\mathbf{x})$ are generally derived from execution of complex numerical simulations requiring considerable computation time.

The steps of the EPSOC algorithm are:

1. Initialise a random, uniformly-distributed population, p, and evaluate each trial solution, $\mathbf{x}_i$ $\forall$ i.
2. Sort the population by objective function value, $f(\mathbf{x})$.
3. Select a set, B, of the *nbad* worst members of the population. For each member of B, add to the set its two nearest neighbours in parameter space that are not already members of the set, *or from the best half of the sorted population.*
4. Apply a random, uniformly-distributed mutation to the selected set, B, i.e. re-initialise them. For all other members of the population, generate a "child" by applying a small (~10% of parameter range), random, uniformly-distributed mutation to the "parent" member.
5. Evaluate each new trial solution, $f(\mathbf{x})$.
6. If a child has a better objective function value than its parent, replace the parent with the child.
7. Repeat from step 2 until a preset number of iterations have been completed.

As each set of parameters defining a trial solution is independent of all others, it is immediately apparent that the evaluation of trial solutions at steps 1 and 5 can be performed concurrently. Since the evaluation of the objective function completely dominates the execution time, from Amdahl's Law we can expect extremely high parallel efficiency.

In a previous study we applied the single objective implementation of EPSOC and compared it to a range of optimization algorithms (Lewis 2003), including a GA (Genesis 5.0, Grefenstette 1984), a parallel gradient descent method (P-BFGS), Dennis and Torczon's MDS, the Simplex method of Nelder and Mead, a Reducing Set Concurrent Simplex (RSCS), and line-searching variants of Simplex and RSCS.

All the algorithms were tested on six case studies drawn from real-world problems, from antenna and aerofoil design to quantum electrodynamical simulations of laser-atom interactions. Each algorithm was run multiple times on each test case. The GA and EPSOC were limited to a maximum number of iterations determined empirically to be equivalent to the time the other algorithms took to converge. On each test case, tests were performed to determine optimal operational parameter settings for the GA and EPSOC. It was noted that statistically significant differences in results returned for different parameter settings were more common for the GA than for EPSOC, i.e. EPSOC was less sensitive to operational parameter tuning.

Shapiro-Wilk testing of the distributions of returned values indicated the results were not normally distributed, so non-parametric, descriptive statistical methods were used to analyse the results.

The Kruskal-Wallis H test statistic was used to rank the algorithms on each test case, and determine if any algorithm performed significantly better than others. A pair-wise comparison was performed on the two highest-ranked algorithms on each case, using the Mann-Whitney U test statistic. In 4 of 6 test cases EPSOC was demonstrated to be the best algorithm *on these test cases drawn from real-world problems*, to a significance level of better than 0.05. In the remaining cases there was no statistically significant difference between the performance of EPSOC and the other leading algorithm.

4.2 Multi-Objective Optimisation

When tackling real-world problems, particularly in the field of engineering design, the desired optimal design may not be expressed in terms of a single objective. Product designers may wish to maximise some element of the performance of a product, while minimising the cost of its manufacture, for example. Different objectives may be conflicting, with little *a priori* knowledge as to how they interact. In the past, common practice was to optimise for a single objective while applying other objectives as penalty functions or constraints, a less than ideal approach. Evolutionary Algorithms (EA) have recently become more widely used for their ability to work well with large-scale, multi-objective problems (Van Veldhuizen 1999).

In general, there remains a role for a human Decision Maker (DM) in choosing between competing objectives. Multi-objective optimisation algorithms can broadly be categorised by when in the optimisation process the DM intervenes (Zitzler 1999, Van Veldhuizen 2000):

- **Decision making before search**: the DM aggregates the objectives into a single objective, including preference information (or weights). The problem is essentially reduced to a single objective optimisation.
- **Decision making during search**: the DM interactively supplies preference information to guide the search
- **Decision making after search**: the DM selects from a set of candidate solutions resulting from the search.

It may be noted that the first two of these approaches would seem to require some *a priori* knowledge of the problem domain in order to effectively provide preference information, particularly for methods involving aggregation. It has been asserted (Zitzler 1999, Coello 1998) that the first approach does not return Pareto-optimal solutions in the presence of non-convex search spaces. The multiobjective EPSOC method described in this paper (EPSOC-MO) falls into the last category.

EAs applied to multi-objective optimisation need to address two main problems:

- How is fitness assignment and selection performed?
- How is a diverse population maintained, to aid search space exploration?

Fitness assignment can conceptually be divided into a number of approaches (Zitzler 2002):

- Aggregation-based
- Criterion-based
- Pareto-based

A great deal of recent work has concentrated on the use of pareto-based selection. EPSOC-MO does not make explicit use of pareto dominance, selection being made according to rankings based on single objective function values, and their combination. It does not do this, however, by aggregation of the objectives.

In approach, EPSOC-MO can be seen as a hybrid of criterion-based and Pareto-based fitness assignment and selection. Fitness is determined by objective function values in turn, a criterion-based assignment. But selection, for extinction, is tempered by consideration of ranking against all objectives, an implicit Pareto-based approach.

Diversity of the population is addressed in EPSOC-MO by the same means as for its use for single objective optimisation, i.e. achieving a self-organised critical state through operations of mutation and extinction. Of the common methods (Zitzler 1999):

- Fitness sharing
- Restricted mating
- Isolation by distance
- Overspecification
- Reinitialization
- Crowding

the method of species extinction in EPSOC is closest to a form of reinitialization, though of a carefully selected section of the population.

EPSOC operates on an ordered population, ranked by fitness according to the objective function. However, for it to be used for multi-objective optimisation it would not be possible to sort the population according to two or more objectives simultaneously. EPSOC-MO, instead maintains an ordered set for each objective, with a mapping into the original population. An example for two objectives is illustrated in Figure 1.

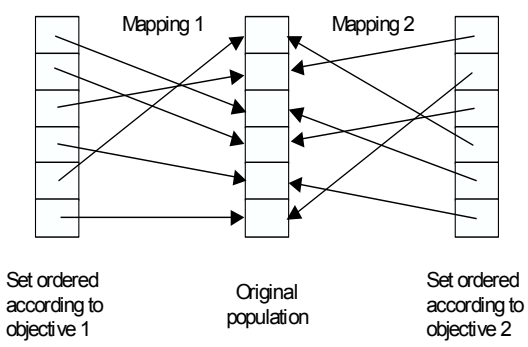

Figure 1: Ordered set mapping for multi-objective EPSOC

Referring to the outline of the EPSOC algorithm in Section 3.1, half of the *nbad* members selected come from the worst members of the first ordered set, and the remainder from the worst members of the second ordered set, *with the proviso* that those chosen from the first set not be among the elite of the second set and *vice versa*.

To implement this restriction, a set of reverse mappings, from the original population back into the ordered sets, was also maintained. Their use is illustrated in Figure 2. A member chosen from the first set, for example, would be mapped into the original population, and then into the second set. It would only be chosen for extinction if it were not in the elite of either set. For the example shown in Figure 2, the chosen member from set 1 would be rejected since it is in the elite of set 2. Similar restrictions are imposed on the choice of nearest neighbours at step 3 of the algorithm. For a problem involving more objectives, chosen members would be tested against the elite of each set in turn. The elites of each set were chosen so that their sum was the size of the elite of the single objective implementation of EPSOC.

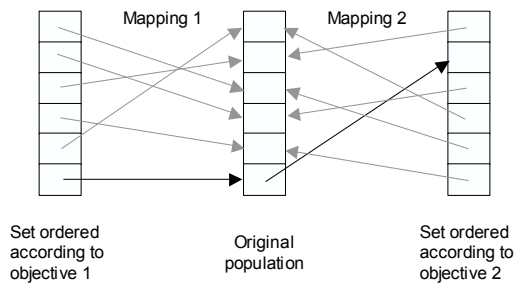

Figure 2: Use of reverse mappings in multi-objective EPSOC

The use of archives to preserve the Pareto-optimal set is commonplace in multi-objective evolutionary algorithms (MOEAs) (Zitzler 2002) and remains an area of active research (Laumanns 2002). Most algorithms use the archive as a form of "non-volatile" storage, serving only to preserve Pareto-optimal approximations. Only a few use the archive as a dynamic element in the algorithm (Knowles 1999).

In contrast, in EPSOC-MO the ranked sets play an integral role in the operation of the algorithm. These sets not only preserve the approximations to the Pareto-optimal set, but also mediate between objectives during the search. In this regard they can be considered as playing a role in the decision-making process; in effect *decision making during search* but without the interaction of the Decision Maker. In a sense, through these sets EPSOC-MO maintains multiple, "virtual" archives.

5 A Case Study: Multi Objective Radio Frequency Design

A simple experiment was constructed on a data set from a case study in radio-frequency simulation. In this problem, a ceramic bead must be designed to minimise distortion of the radiation pattern of a mobile telecommunications handset during testing (Lewis 2000). The objective function value, which is computed by a FDTD full-wave analysis of the cable structure, is a measure of transmission strength through the bead at 1 GHz. The dataset for the case study, of which isosurfaces for a particular value are shown in Figure 3, is quite complex and rich in structure. It contains 298 local minima. The

entire dataset is available as part of the Nimrod/O optimisation toolset (Abramson 2000)

A multiobjective version of this original problem can be derived by formulating a second objective for this test case. In this new scenario, not only is gain to be optimised, but the length of the optimal bead must also be minimised. This is in some sense a realistic goal, in that a shorter, more compact component might be desirable in practice. Ten runs of EPSOC-MO were performed using these objectives, each run with a population of 64 and a maximum iteration limit set to 500. The length of the bead was permitted in the range 35mm to 60mm and the range of the gain values to be minimised was from –39.85 to 48.52.

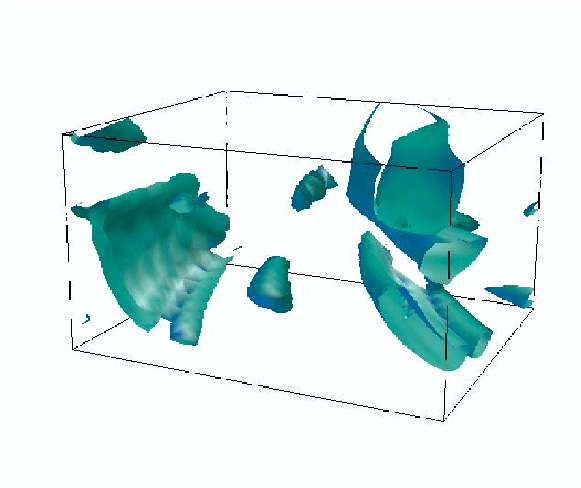

Figure 3: Isosurfaces of the gain for the ceramic bead test case

Gain and length values for the points returned at the top of the set ordered by gain for each run are shown in Table 1. The median gain for this set of points is –22.35, compared with a median of 34.99 for the whole dataset. The median length of these returned points is 48.1, compared with a median of 47.5 for the whole dataset.

Gain	Length
-12.27	50.07
-26.98	48.44
-27.55	51.77
-14.35	41.37
-20.41	41.71
-30.42	51.68
-7.73	41.34
-35.57	51.97
-10.89	42.98
-24.29	47.79

Table 1: Gain and length of "best" points from 10 runs

The entire decision space was scanned at integer intervals, and the results are plotted in objective space in Figure 4 to illustrate the distribution of solutions in objective space, and give some indication of the location of Pareto-optimal solutions. It should be noted that the test case uses these sample points and linear interpolation between them to provide objective function values, i.e. these are extremal points. The "best" returned values are highlighted. By inspection, it can be seen that the returned points approach the Pareto-optimal front of minimal gain values combined with minimal length.

The top-ranked points in the gain-ranked virtual archive in any particular run also approach the Pareto-optimal front. The top ten points from a representative run are shown in Figure 5, superimposed on the same parameter sweep data as Figure 4.

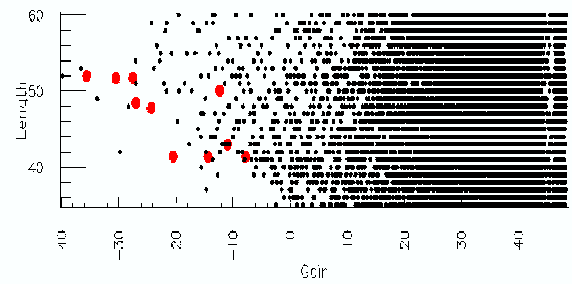

Figure 4: Multi-objective test case objective space sampling with "best" points from 10 runs

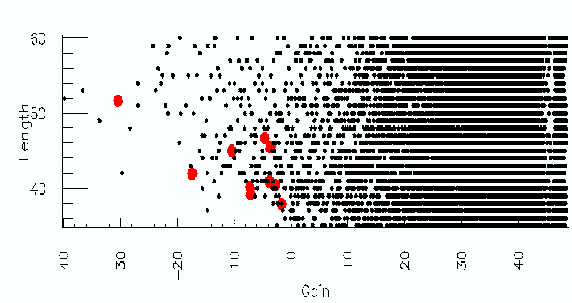

Figure 5: Multi-objective test case objective space sampling with archive points from a single run

In order to compare the effect of optimising for multiple objectives, another 10 runs were performed, the only change being the removal of the second objective function (relating to the length). The median gain of the set of points returned from these runs was –29.97, and the median length of the optimised beads was 49.0. The length of the returned beads was purely a product of the location of significant minima in the dataset, of which there are two dominant, one in a region of parameter space corresponding to a length of 52mm which also contains the global minimum, and another at length 42mm. The returned points, superimposed on the parameter sweep data, are shown in Figure 6. It can be seen that returned points are clustered at higher gain values, unlike the points in Figure 4 which are more spread along the Pareto-optimal front.

The median gain returned for the multi-objective test case is neither near that returned for the single-objective, gain-only case, nor near the median of the entire dataset.

It lies between and demonstrates EPSOC-MO is capable of returning a compromise solution, independent of an external Decision Maker supplying preference information after the search.

The median length returned for the multi-objective case lay close to the median for the entire dataset. It could be concluded that behaviour of EPSOC-MO was neutral toward the length objective, but evidence from the single-objective test case indicates this value may also be a compromise between lengths for two solutions of attractive gain in different regions of parameter space.

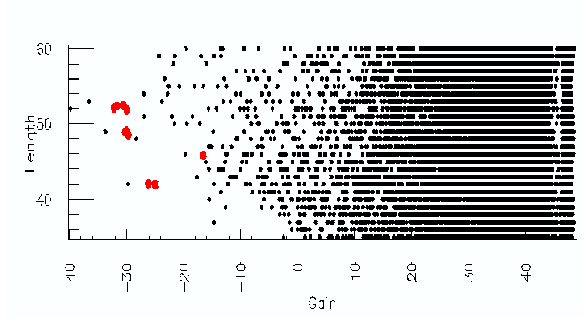

Figure 6: Test case objective space sampling with "best" points from 10 runs, optimised for gain only

6 Conclusion

An Evolutionary Programming algorithm using concepts of Self-Organised Criticality, EPSOC, has been described and a method for its use for multiobjective optimisation outlined. The algorithm uses multiple, "virtual" archives both to store optimal results and also to mediate between objectives during selection. A test of the implemented algorithm on a challenging test case with a highly nonlinear objective has demonstrated its ability to return close approximations to Pareto-optimal solutions.

These are preliminary results, and by no means exhaustive. Several questions remain as to the efficacy of EPSOC for multi-objective optimisation, not the least being the scalability of the method, but the results returned so far are quite promising. Further work is necessary to compare EPSOC-MO with other, widely-used MOEAs, and to demonstrate its behaviour on standard test problems

Bibliography

Abramson D., Lewis, A. and Peachey, T., (2000) "Nimrod/O: A Tool for Automatic Design Optimization", *The 4th International Conference on Algorithms & Architectures for Parallel Processing (ICA3PP 2000)*, Hong Kong.
http://www.csse.monash.edu.au/~davida/nimrodo/

Alander, J.T., (1995) "An Indexed Bibliography of Genetic Algorithms in Computer Aided Design", Report 94-1-CAD, Department of Information Technology and Pro-duction Economics, University of Vaasa, Finland.

Bak P. and Sneppen K. (1993) "Punctuated equilibrium and criticality in a simple model of evolution", *Phys. Rev. Let.* **71**, pp. 4083-4086.

Bak, P. (1996) "How Nature Works", Springer-Verlag, New York.

Bäck, T. (1996) "Evolutionary Algorithms in Theory and Practice", Oxford University Press, New York, NY.

Coello Coello, C.A., (1998) "An Updated Survey of Evolutionary Multiobjective Optimization Techniques" State of the Art and Future Trends", Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, México.

Coello Coello, C.A., (1999) "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques", *Knowledge and Information Systems*, **1**(3), pp. 269-308.

Grefenstette, J.J., (1984) "GENESIS: A system for using genetic search procedures", In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pp. 161-165.

Holland, J.H. (1975) "Adaptation in natural and artificial systems", University of Michigan Press, Ann Arbor, MI.

Knowles, J.D. and Corne, D.W., (1999) "The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation", In *1999 Congress on Evolutionary Computation*, pp. 98-105, Washington, DC.

Krink, T., and Thomsen, R., (2001) "Self-Organized Criticality and Mass Extinction in Evolutionary Algorithms", *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, Seoul, Korea.

Laumanns, M., Thiele, L., Deb, K. and Zitzler, E., (2002) "Archiving with guaranteed convergence and diversity in multi-objective optimization", In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 439-447.

Lewis, A., Saario, S., Abramson, D. and Lu, J., (2000) "An Application of Optimisation for Passive RF Component Design", *Conference on Electromagnetic Field Computation*, Milwaukee, June 4-7th.

Lewis, A., Abramson, D., and Peachey, T., (2003) "An Evolutionary Programming Algorithm for Automatic Engineering Design", *Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM 2003)*, Czestochowa, Poland.

Schwefel, H.-P. (1965) “Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik”, Diplomarbeit, Technische Universität Berlin.

Van Veldhuizen, D.A., (1999) “Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations”, PhD Thesis, Air Force Institute of Technology, Air University, USA.

Van Veldhuizen, D.A., and Lamont, G.B., (2000) “Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art”, *Evolutionary Computation*, pp. 125-147, 8(2).

Zitzler, E., (1999) “Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications”, D.Tech.Sci. Dissertation, Swiss Federal Institute of Technology, Zurich, Switzerland.

Zitzler, E., Deb, K., and Thiele, L., (2000) “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”, *Evolutionary Computation,* pp. 173-195, 8(2).

Zitzler, E., (2002) “Evolutionary Algorithms for Multiobjective Optimization”, EUROGEN-2001, Evolutionary Methods for Design, Optimisation and Control, Barcelona, Spain.