The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization

David W. Corne, Joshua D. Knowles, Martin J. Oates

School of Computer Science, Cybernetics and Electronic Engineering
University of Reading, Reading RG6 6AY, UK
D.W.Corne, J.D.Knowles, M.J.Oates@reading.ac.uk

Abstract. We introduce a new multiobjective evolutionary algorithm called PESA (the Pareto Envelope-based Selection Algorithm), in which selection and diversity maintenance are controlled via a simple hyper-grid based scheme. PESA's selection method is relatively unusual in comparison with current well known multiobjective evolutionary algorithms, which tend to use counts based on the degree to which solutions dominate others in the population. The diversity maintenance method is similar to that used by certain other methods. The main attraction of PESA is the integration of selection and diversity maintenance, whereby essentially the same technique is used for both tasks. The resulting algorithm is simple to describe, with full pseudocode provided here and real code available from the authors. We compare PESA with two recent strong-performing MOEAs on some multiobjective test problems recently proposed by Deb. We find that PESA emerges as the best method overall on these problems.

1 Introduction

Following seminal work on multiobjective evolutionary algorithms (MOEAs) such as the Niched Pareto Genetic Algorithm (Horn et al., 1994; Horn and Nafpliotis, 1994), and the Non-Dominated Sorting method (Srinivas and Deb, 1994), more recent work on MOEAs has resulted in improved techniques which provide fast and effective approximations to the Pareto frontier for a variety of benchmark problems. These new methods include SPEA (Strength Pareto Evolutionary Algorithm –Zitzler and Thiele, 1999), and PAES (Pareto Archived Evolution Strategy – Knowles and Corne, 2000). Both PAES and SPEA have been shown to outperform sophisticated versions of NPGA and NDS on a variety of benchmark problems, while various other modern MOEAs exist which have been shown to perform well on particular applications (eg: Fonseca and Fleming, 1995; Parks and Miller, 1998), but have not yet been systematically compared against other modern MOEAs on a common set of test problems.

Somewhat removed from the MOEA research community, researchers in multiple criteria decision making (MCDM) and operations research communities have also worked on multiobjective optimization over the years, and produced a variety of local-search based multiobjective techniques These include, for example, Czyzak and Jaszkiewicz (1998), Gandibleux et al. (1996), and Hansen (1996; 1997). Cross comparison of techniques between these communities and the MOEA community has not yet been done to any significant extent, although it seems clear from the results reported in Zitzler and Thiele (1999) and Knowles and Corne (2000) that the MOEA community would be best represented by either SPEA or PAES in such a comparative study.

Here we introduce a new multiobjective evolutionary algorithm called PESA (the Pareto Envelope-based Selection Algorithm), which incorporates ideas from each of SPEA and PAES, and which performs quite well in comparison to both of these new methods. In fact, PESA seems to be the overall best on the suite of test functions used in this paper.

Like SPEA, PESA uses a small 'internal population' and a (usually) larger 'external population'. The external population is actually the *archive* which stores the current approximation to the Pareto front, and the internal population are new candidate solutions vying for incorporation into the archive. Like PAES, PESA implicitly maintains a hyper-grid division of phenotype space which allows it to keep track of the degree of crowding in different regions of the archive. However, unlike both PAES and SPEA, selection in PESA is based on this crowding measure (in PAES, selection is trivial – there is just a single current solution, as in local search, which is the parent for a mutation; in SPEA, selection is based on a sophisticated and ingenious 'strength' measure). However, like both PAES and SPEA, replacement (deciding what must leave the archive if it becomes over-full) is also based on a crowding measure.

Each of SPEA, PAES and PESA are compared on six test functions from Deb (1998), and results are analysed using a sophisticated statistical comparison method based on ideas from Fonseca and Fleming (1995a). Both SPEA and PESA have already been found to outperform NDS and NPGA on these functions (Zitzler et al, 1999; Knowles and Corne, 1999). We find that each algorithm turns out to be the winner on at least one of the test functions, but PESA is the overall best performer.

The remainder of the paper is organized as follows. In Section 2 we describe the PESA algorithm, and in Section 3 we describe the test functions. Experimental design is discussed in Section 4 and Results given and discussed in Section 5. Section 6 indicates our conclusions and notes for further work.

2 The PESA Algorithm

Apart from standard parameters such as crossover and mutation rates, PESA has two parameters concerning population size, and one parameter concerning the hyper-grid crowding strategy. A high-level description of the PESA algorithm is as follows, in which the two population based parameters are P_I (the size of the internal population, IP) and P_E (the maximum size of the archive, or 'external population', EP).

1. Generate and evaluate each of an initial 'internal' population (IP) of P_I chromosomes, and initialise the 'external' population (EP) to the empty set.
2. Incorporate the non-dominated members of IP into EP.
3. If a termination criterion has been reached, then stop, returning the set of chromosomes in EP as the result. Otherwise, delete the current contents of IP, and repeat the following until P_I new candidate solutions have been generated:
   - With probability p_C, select two parents from EP, produce a single child via crossover, and mutate the child. With probability $(1-p_C)$, select one parent and mutate it to produce a child.
4. return to Step 2.

In the 'archive incorporation' step (step 2), the current set of new candidate solutions (IP) are incorporated into the archive one by one. A candidate may enter the archive if it is non-dominated within IP, and if is not dominated by any current member of the archive. Once a candidate has entered the archive, members of the archive which it dominated (if any) will be removed. If the addition of a candidate renders the archive over-full (its size temporarily becomes P_E+1), then a current member of EP is removed. The choice of which member is removed will be detailed later.

The selection of a parent in PESA, within step 3, is based on the degree of crowding in different regions of the archive. This is simply illustrated by Figure 1. In the figure, a number of points are shown in the phenotype space of a two-objective minimization problem. The circles are non-dominated points, and hence might currently be in the PESA archive. The squares are dominated by members of the archive, but they may be points in the current internal population.

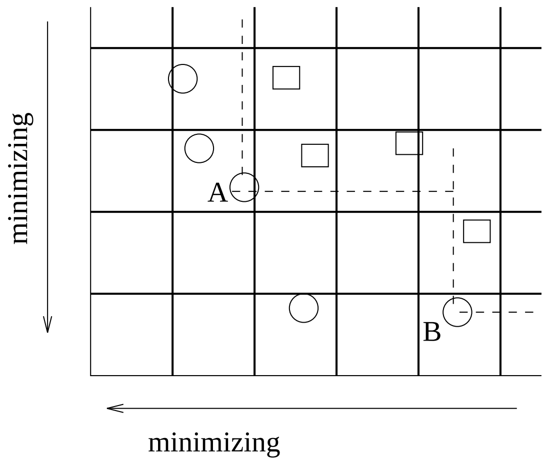

Fig. 1. PESA's crowding strategy

The crowding strategy in PESA works by forming an implicit hyper-grid which divides (normalised) phenotype space into hyper-boxes. In Figure 1, this is illustrated by the thick horizontal and vertical lines; the problem is two-dimensional and hence these hyper-boxes are simply squares. Each chromosome in the archive is associated with a particular hyper-box in phenotype space, and has an attribute called a 'squeeze factor', which is simply the total number of other chromosomes in the archive which inhabit the same box. For example, the squeeze factor of chromosome A in Figure 1 is 2, and the squeeze factor of chromosome B is 1. The squeeze factor is used for selective fitness. For example, when PESA uses binary tournament selection, two chromosomes are taken at random from the archive, and the one with the lowest squeeze factor is chosen (breaking ties randomly), hence orienting search effort towards areas of the emerging Pareto frontier which currently have little representation in the population.

The squeeze factor is also used for archive update. As indicated above, the incorporation of an internal population member into the archive may lead to the the archive size temporarily exceeding the maximum size E_P. One solution must therefore be removed from the archive. The choice is made by first finding the maximal squeeze factor in the population, and removing an arbitrary chromosome which has this squeeze factor.

We can now more easily distinguish between the selection and archive-update strategies of PAES, SPEA, and PESA¡ again with reference to Figure 1. As we have seen, PESA uses the squeeze factor for both of these tasks. PAES uses a hyper-grid strategy, but only for archive update. In the selection step of PAES, since PAES is actually a hillclimbing algorithm, selection is only between two candidates: the current solution, and a mutant. If the archive is full, squeeze factor is indeed used in PAES, and if the mutant has a lower squeeze factor than the current then it will become the new current solution. To some extent PAES is therefore similar to PESA in employing squeeze factor in both selection and archive update, however the fact that PAES is a local search method and PESA is a population based technique render them fundamentally different algorithms.

SPEA, on the other hand, uses a novel selection strategy in which a 'strength' is associated with each member of the archive, based on the number of chromosomes in the internal population which it dominates. Each member of the internal population is also given a strength value, based on summing the strengths of the archive members which dominate it. This is illustrated by the dashed lines in Figure 1. Chromosome A clearly has a higher strength value than chromosome B, since it dominates more members of the internal population than B. In SPEA, this means that B will have a higher selective fitness than A (fitness for selection is based on minimal strength value).

Archive update in SPEA is also quite different to the hyper-grid strategy. In SPEA, a simple agglomerative clustering algorithm is used based on phenotypic distance, which prunes an archive of $k > E_P$ chromosomes into a an archive of E_P chromosomes. It first produces E_P clusters of chromosomes from the overfull archive. The chromosomes in each cluster which are nearest to its centre then become the E_P chromosomes in the pruned archive.

3 The Test Problems T1–T6

Deb (1998) gives a procedure for designing tunable test functions for multiobjective optimisation, involving a range of characteristics which may or may not be present to varying degrees in the Pareto surface. In particular, Deb argues that key aspects of a multiobjective search landscape which would cause difficulty for an optimizer are, among others, discontinuities in the Pareto front, non-uniform distribution of solutions along the Pareto front, and deception. The test function design scheme produces test functions which vary in these aspects.

Six test functions, $\mathcal{T}_1$-$\mathcal{T}_6$, designed using Deb's scheme, were used in a comparison of the performance of eight different MOEAs by Zitzler et al (1999). $\mathcal{T}_1$ has a convex Pareto front but no particular difficulty characteristics; $\mathcal{T}_2$ has a non-convex Pareto front; $\mathcal{T}_3$ has many discontinuities in the Pareto front; $\mathcal{T}_4$ is highly multimodal and has 21^9 Pareto fronts; $\mathcal{T}_5$ is a deceptive problem, and $\mathcal{T}_6$ has a non-uniformly distributed search space with solutions non-uniformly distributed along the Pareto front.

Each is a two-objective problem defined on m parameters, in which both objectives are to be minimized. In five of the problems the parameters x_i were coded as a binary string decoded such that $x_i \in [0, 1]$. The remaining function ($\mathcal{T}_5$) also employed a binary chromosome but this time unitation was used to evaluate each of the parameters. The experiments in this paper employ identical functions to those presented in Zitzler et al (1999) and are coded onto chromosomes using identical numbers of bits to represent each parameter.

The results of Zitzler et al (1999) indicated that on three of the test functions, $\mathcal{T}_1 - \mathcal{T}_3$ and $\mathcal{T}_6$, SPEA (Zitzler and Thiele, 1999) generates solution sets which consistently dominate *all* of the other algorithms tested. On test function $\mathcal{T}_4$ SPEA is clearly superior to all other algorithms, although it no longer consistently beats two of the algorithms, namely NSGA (Srinivas and Deb, 1994), and SOEA (a single objective EA run 100 times with a different randomly chosen linear combination of the objectives). On $\mathcal{T}_5$, SPEA and NSGA perform very similarly, with SOEA performing slightly better according to the measures used by Zitzler et al (1999).

In summary, Zitzler's study indicated that SPEA seemed to be the best algorithm overall (of the eight tested) on Deb's test functions. In this study, we hence use SPEA as our main comparative algorithm, but also use PAES. We therefore compare the performance of SPEA against two rival techniques, one of which has been found best on the Deb functions when compared with a wide range of other multiobjective approaches.

4 Experimental Design

4.1 Experiments

Our experiments sought to determine the relative quality of PESA, SPEA and PAES on the Deb test functions. Guided by real-world considerations, we were also very interested in speed. That is, we were interested in the development of the approximations to the Pareto front over time. In some applications (for example, design) the optimisation method must develop as good as possible an approximation to the Pareto tradeoff surface, but without any real-time or near real-time processing constraints. In other applications, for example optimal multiobjective control, good approximations to the Pareto front must be produced very quickly.

To compare the algorithms along these lines we therefore performed three sets of experiments, to different time limits, of 1,000, 5,000, 20,000 evaluations. In each set of experiments, 20 trial runs of PESA, PAES, and SPEA were performed on each of the six test functions. Parameters were set as described in Table 1.

Crossover rate	0.7 in PESA and SPEA; not used in PAES
Crossover method	uniform crossover in PESA and SPEA; not used in PAES
Mutation rate	bit-flip mutation set to $1/L$ where L is chromosome length
Populations	archive 100 in all algorithms, IP size 10 in PESA and SPEA
Chromosome lengths	900 in $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, 300 in $\mathcal{T}_4$ and $\mathcal{T}_6$, 80 in $\mathcal{T}_5$
Hyper-grid size	32×32 grid in PESA and PAES, not used in SPEA

Table 1. Parameter settings

In the next section we summarise the statistical comparison method used to analyse the results within a set of experiments.

4.2 Statistics

Given the results of 20 or more trial runs for each algorithm, we compare the performance of two or more multiobjective optimisers using a method proposed originally by Fonseca and Fleming (1995a) which we have implemented with certain extensions. When comparing two multiobjective algorithms (A and B), this method essentially returns two numbers: the percentage of the Pareto frontier on which algorithm A conclusively beats algorithm B (based on a Mann-Whitney U test at the 95% confidence level), and the percentage of the Pareto frontier on which algorithm B conclusively beat algorithm A.

Typically, for example, two very good MOEAs with similar performance on a problem might yield a result like [3.7, 6.1], indicating that each algorithm was definitely better than the other in small regions of the space, but they performed similarly well on the majority of the Pareto frontier. A clear indication that one algorithm is superior to another, however, is given by a comparison result such as [58.3, 2.2], or [100, 0.0].

In a comparison of $k > 2$ algorithms, the comparison code performs pairwise statistical comparisons, as before, for each of the $k(k-1)/2$ distinct pairs of algorithms. The results then show, for each algorithm, on what percentage of the space we can be statistically confident that it was unbeaten by any of the other $k-1$ algorithms, and on what percentage of the space it beat *all* $k-1$

algorithms. For example, in Table 2, we can see that, on problem $\mathcal{T}_1$, PESA was conclusively better than both PAES and SPEA on more than half of the Pareto Tradeoff surface, and was only bettered (in this case by PAES) on 28% of this frontier. SPEA performed particularly poorly in this case, being conclusively outperformed by the other methods on all but about 1% of the Pareto surface.

5 Results and Discussion

Table 2 summarises all results for the set of experiments in which each trial run was allowed just 1,000 fitness evaluations. The best performing algorithm for each problem has its table entries highlighted in bold; when there is little difference between the best two (or all three), both sets of entries are highlighted in bold.

	Problem	PAES	SPEA	PESA
unbeaten (beats all)	$\mathcal{T}_1$	44.7 (28.0)	0.9 (0.0)	**72.0 (54.4)**
unbeaten (beats all)	$\mathcal{T}_2$	0.0 (0.0)	1.8 (0.0)	**100 (98.2)**
unbeaten (beats all)	$\mathcal{T}_3$	0.0 (0.0)	**98.9 (0.0)**	**100 (1.1)**
unbeaten (beats all)	$\mathcal{T}_4$	24.6 (0.0)	18.2 (0.0)	**100 (57.2)**
unbeaten (beats all)	$\mathcal{T}_5$	0.9 (0.0)	**100 (36.8)**	63.2 (0.0)
unbeaten (beats all)	$\mathcal{T}_6$	16.3 (0.0)	**100 (0.0)**	**100 (0.0)**

Table 2. Comparison of PAES, SPEA and PESA at 1,000 evaluations

As Table 2 shows, PESA was clearly the best method on three of the functions, and joint best with SPEA on a further two. On the one remaining function it achieved the second-best performance. SPEA is clearly best on just one function, and joint best with PESA on two. PAES is the worst performer here, being clearly worst on three of the test functions, and second or joint second best on the remaining three.

In Table 3, we can see the results when trials were set to run for 5,000 evaluations. PESA is now clearly best on two of the six test functions, and joint best with SPEA on a further two. It is second best on the remaining two test functions. Given the greater time limit available, PAES starts to improve its comparative performance, now being clearly best on one function and joint best on two. SPEA seems to lose out to PAES somewhat with the increased time limit, now being clearly best on one function, but worst or joint worst on the remaining five.

Table 4 indicates the results when trials were given a full 20,000 evaluations each. PESA is now best or joint best on five of the six test functions. In the other case it is second best, beating SPEA, although rather a poor second to PAES. PAES is clearly best in one case and joint best in another, while SPEA is joint-best in two cases, second best in two cases, and worst in the remaining two cases.

All of these results are summarised in Table 5, in which we show the rank for each algorithm at each evaluation time limit, on each problem. For example, the

	Problem	PAES	SPEA	PESA
unbeaten (beats all)	$\mathcal{T}_1$	**94.3 (76.5)**	0.0 (0.0)	23.5 (5.7)
unbeaten (beats all)	$\mathcal{T}_2$	0.0 (0.0)	0.0 (0.0)	**100 (100)**
unbeaten (beats all)	$\mathcal{T}_3$	**66.2 (21.7)**	23.7 (0.0)	**78.3 (21.1)**
unbeaten (beats all)	$\mathcal{T}_4$	**96.1 (0.0)**	40.5 (0.0)	**100 (0.0)**
unbeaten (beats all)	$\mathcal{T}_5$	0.0 (0.0)	**100 (24.0)**	76.0 (0.0)
unbeaten (beats all)	$\mathcal{T}_6$	15.5 (4.6)	0.0 (0.0)	**95.4 (84.5)**

Table 3. Comparison of PAES, SPEA and PESA at 5,000 evaluations

	Problem	PAES	SPEA	PESA
unbeaten (beats all)	$\mathcal{T}_1$	**99.6 (98.7)**	0.0 (0.0)	1.2 (0.2)
unbeaten (beats all)	$\mathcal{T}_2$	0.5 (0.0)	41.4 (2.4)	**97.6 (58.6)**
unbeaten (beats all)	$\mathcal{T}_3$	47.9 (9.1)	37.4 (0.0)	**90.9 (18.3)**
unbeaten (beats all)	$\mathcal{T}_4$	**99.9 (0.0)**	**100 (0.0)**	**100 (0.0)**
unbeaten (beats all)	$\mathcal{T}_5$	0.0 (0.0)	**100 (2.3)**	**97.7 (0.0)**
unbeaten (beats all)	$\mathcal{T}_6$	11.2 (8.5)	0.1 (0.0)	**97.5 (88.7)**

Table 4. Comparison of PAES, SPEA and PESA at 20,000 evaluations

entry '1/3/3' for SPEA under $\mathcal{T}_3$ indicates that for the shorter time limit it was the best algorithm (indicated by the '1'), but then became third-best (ie: worst) at the moderate time limit of 5,000 evaluations (indicated by the first '3'), and also at the longest time limit of 20,000 evaluations (indicated by the second '3').

	Problem	PAES	SPEA	PESA
rank (at 1k/5k/20k evals)	$\mathcal{T}_1$	2/**1**/**1**	3/3/3	**1**/2/2
rank (at 1k/5k/20k evals)	$\mathcal{T}_2$	3/3/3	2/3/2	**1**/**1**/**1**
rank (at 1k/5k/20k evals)	$\mathcal{T}_3$	3/**1**/2	**1**/3/3	**1**/**1**/**1**
rank (at 1k/5k/20k evals)	$\mathcal{T}_4$	2/**1**/**1**	3/3/**1**	**1**/**1**/**1**
rank (at 1k/5k/20k evals)	$\mathcal{T}_5$	3/3/3	**1**/**1**/**1**	2/2/**1**
rank (at 1k/5k/20k evals)	$\mathcal{T}_6$	3/2/2	**1**/3/3	**1**/**1**/**1**

Table 5. Summary of PAES, SPEA, PESA comparisons on Functions T1–T6

With reference to Table 5, PAES clearly seems to be a slow starter, with poor comparative performance at short time limits, but quite able to compete with other methods at longer time limits. To some extent this would indicate that PAES is best not used for (near) real-time applications. However, the time complexity of PAES algorithm is much more favourable than population-based multiobjective optimisers (Knowles and Corne, 2000), so this conclusion may not be fair. The conclusion stands, however, in the case of applications where fitness evaluation is the bottleneck.

There is no clear pattern to the performance of SPEA as we increase the time limit, except for the fact that its rank generally worsens from the fast to the moderate time limit as PAES 'catches up' with it. PESA, on the other hand, seems to successfully hold its own at all time limits. reference to table 5 seems to suggest it is generally the best of the three algorithms compared here on these test functions.

It is instructive to now consider performance in terms of the problem characteristics. $\mathcal{T}_\infty$ is not a particularly difficult problem (lacking multimodality, deception, and so forth), and this was the only problem on which PESA was not best or joint best at the longest time limit. It seems reasonable to conclude that this is because the hillclimbing strategy of PAES (the clear winner on this problem) is particularly suitable here. the problem has none of the characteristics that the sophisticated aspects of PESA or SPEA (such as use of crossover, and selection from a population) are designed to address. PAES therefore makes aggressive and fruitful use of its time on this problem, while PESA and SPEA essentially waste much of their search effort. This intuition seems to be confirmed by the fact that one of the problems on which PAES performed particularly badly was $\mathcal{T}_5$, which is the deceptive problem in the suite.

6 Conclusion

We have described the Pareto Envelope-based Selection Algorithm, and compared its performance with two recent strong-performing multiobjective optimisers on a suite of test functions devised by Deb (1998). Comparative performance was measured using a sophisticated statistical comparison technique, and performance was compared in respect of three separate time limits, reflecting the varying needs for solution speed in real-world applications. We found that PESA generally outperforms both SPEA and PAES on these functions. It was never the worst of the three, and tended to perform best or joint best (with SPEA), whether solutions were needed quickly, moderately quickly, or without stringent time constraints.

PAES and SPEA are both modern multiobjective optimisation methods which have previously been found to outperform a wide range of classical methods on a wide range of problems, The relative performance of PESA demonstrated here therefore suggests that PESA may be well qualified to join these two methods in the current set of 'best performers' in the multiobjective evolutionary algorithm community. However, results on a limited set of test functions must always be regarded as tentative, and hence much further work is needed on further problems to better assess the value of PESA.

Acknowledgments

The authors are grateful to British Telecommunications Plc for financial support of the second and third authors.

References

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis 7*, 34–47.

Deb, K. (1998). Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical Report CI-49/98, Department of Computer Science, University of Dortmund.

Fonseca, C. M. and Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation 3*, 1–16.

Fonseca, C. M. and Fleming, P. J. (1995a). On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In Voigt, H-M., Ebeling, W., Rechenberg, I. and Schwefel, H-P., editors, *Parallel Problem Solving From Nature - PPSN IV*, pages 584–593, Springer.

Gandibleux, X., Mezdaoui N. and Freville, A. (1996). A tabu search procedure to solve multiobjective combinatorial optimization problems. In Caballero, R. and Steuer, R., editors, in *Proceedings volume of Multiple Objective Programming and Goal Programming '96*, Springer-Verlag.

Hansen, M. P. (1996). Tabu Search for Multiobjective Optimization : MOTS. Presented at *MCDM '97*, Cape Town, South Africa, Jan 6-10.

Hansen, M. P. (1997). Generating a Diversity of Good Solutions to a Practical Combinatorial Problem using Vectorized Simulated Annealing. Submitted to *Control and Cybernetics*, August 1997.

Horn, J., Nafpliotis, N., Goldberg, D.E. (1994). A niched Pareto genetic algorithm for multiobjective optimization, in *Proceedings of the First IEEE Conference on Evolutionary Computation IEEE World Congress on Computational Intelligence, Volume 1*, pages 67–72. Piscataway, NJ: IEEE Service Centre.

Horn, J. and Nafpliotis, N. (1994). Multiobjective Optimization Using The Niched Pareto Genetic Algorithm. IlliGAL Report 93005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Champaign.

Knowles, J. D. and Corne, D. W. (1999). Local Search, Multiobjective Optimisation and the Pareto Archived Evolution Strategy, in *Proceedings of the Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pp. 209–216.

Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172.

Parks, G. T., Miller, I. (1998). Selective Breeding in a Multiobjective Genetic Algorithm. In *Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V)*, pages 250–259. Springer.

Srinivas, N., Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation, 2(3)*, 221–248.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation, 2(4)*, 257–272.

Zitzler, E., Deb, K. and Thiele, L. (1999) Comparison of multiobjective evolutionary algorithms: empirical results. Technical report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich.