Local search enabled extremal optimisation for continuous inseparable multi-objective benchmark and real-world problems

Randall, Marcus; Lewis, Andrew; Hettenhausen, Jan; Kipouros, Timoleon

Published in:
2014 International conference on computational science

DOI:
10.1016/j.procs.2014.05.175

Published: 01/01/2014

Document Version:
Publisher's PDF, also known as Version of record

Procedia Computer Science

Volume 29, 2014, Pages 1904–1914

ICCS 2014. 14th International Conference on Computational Science

Local Search Enabled Extremal Optimisation for Continuous Inseparable Multi-objective Benchmark and Real-World Problems

Marcus Randall[1,2], Andrew Lewis[2], Jan Hettenhausen[2], and Timoleon Kipouros[3]

[1] Department of Informatics
Bond University, QLD 4229, Australia
Ph: +61 7 55953361
Email: mrandall@bond.edu.au
[2] Institute for Integrated and Intelligent Systems
Griffith University, Brisbane, Australia
[3] Department of Engineering
University of Cambridge, Cambridge, United Kingdom

Abstract
Local search is an integral part of many meta-heuristic strategies that solve single objective optimisation problems. Essentially, the meta-heuristic is responsible for generating a good starting point from which a greedy local search will find the local optimum. Indeed, the best known solutions to many hard problems (such as the travelling salesman problem) have been generated in this hybrid way. However, for multiple objective problems, explicit local search strategies are relatively under studied, compared to other aspects of the search process. In this paper, a generic local search strategy is developed, particularly for problems where it is difficult or impossible to determine the contribution of individual solution components (often referred to as *inseparable problems*). The meta-heuristic adopted to test this is extremal optimisation, though the local search technique may be used by any meta-heuristic. To supplement the local search strategy a diversification strategy that draws from the external archive is incorporated into the local search strategy. Using benchmark problems, and a real-world airfoil design problem, it is shown that this combination leads to improved solutions.

Keywords: local search, diversification, extremal optimisation, multi-objective optimisation, airfoil design

1 Introduction

Multi-objective, as opposed to single objective, optimisation requires that the search algorithm delivers a variety of trade-off solutions. These are often referred to as the attainment surface,

doi:10.1016/j.procs.2014.05.175

with the aim that this closely approximates the actual Pareto front (or optimal set of points). However, finding this optimal set of points is difficult for meta-heuristic search algorithms (such as genetic algorithms, particle swarm optimisation, differential evolution etc.) as they are ostensibly coarse grain optimisers, designed to determine good trajectories in search space. Local search, on the other hand, is designed to deliver refined solutions (i.e., solutions which are optimal given all immediate neighbours).

While greedy local search strategies are regularly integrated into meta-heuristics for single objective problems (the travelling salesman (TSP) being the classical example), this is done less for multi-objective problems. Either it becomes the basis of the main search algorithm (such as an algorithm like Pareto Archived Evolution Strategy (PAES) [13]) or it becomes part of the selection mechanics of the search algorithm. An example of the latter is Genetic Local search [1, 11, 12] in which, at points outside the main genetic algorithm, a restricted (greedy) form of crossover is used.

One of the issues for multi-objective local search is to determine the definition of a locally optimal solution in this context. As an example, Gomez-Meneses, Randall and Lewis [9] added a local search to a multi-objective EO (discussed next) used to solve discrete problems. To deal with all the objectives, the algorithm simply locally optimises an objective chosen at random at each iteration. Of course, in multi-objective space a solution refined by this technique is unlikely to be locally optimal in a multi-objective sense. This paper attempts to present a multi-objective aware local search algorithm that is further supplemented by a diversification strategy. The latter, when activated, reintroduces known good solutions from the archive to restart stagnated search.

In order to test the ideas of multi-objective local search and diversification, a simple meta-heuristic is needed. Ideally, this should be a meta-heuristic that has few features that may potentially confuse the effect of these two concepts. Extremal optimisation is such a technique. Extremal optimisation [2, 3, 4] is a nature-inspired strategy for solving combinatorial and continuous optimisation problems. Canonically, it manipulates a single solution rather than a population of solutions. The main characteristics of the technique are that it does not converge and, at each iteration, a probabilistic worst solution component's value is changed to a random value. Solution components are the building blocks of the solution, an example being a city placed between a preceding and subsequent city in a chain that forms a TSP tour. In the original version of the EO algorithm, at each iteration the component whose fitness was worst would be replaced by another solution component value generated at random. In essence, however, this choice of always selecting the worst component to modify led to a search that was too greedy, and consequently its performance was poor. Like other meta-heuristic algorithms, an element of randomness (in the form of the probabilistic selection of solution components to change) was introduced. This became known as $\tau-$EO [4]. Components are ranked from worst (rank 1) to best (rank n). The parameter τ and the rank control the selection probability for each solution component [3]. This is achieved using Equation 1.

$$P_i \propto i^{-\tau} \qquad 1 \leq i \leq n \tag{1}$$

Where:

i is the rank of the component,

P_i is the probability ($P_i \in [0,1]$ when normalised) that component i is chosen and

n is the number of components.

Algorithm 1 shows the mechanics of a single $\tau-$EO iteration. Note that this algorithm assumes that there is only one objective function that is separable (i.e., each solution component's fitness can be evaluated independently). Due to the sorting, this is an $O(n^2)$ algorithm in the worst case.

Algorithm 1 A single $\tau-$EO iteration.

```
Rank the solution components in $x$ from worst to best according to their adverse effect on the objective function $f(x)$
$i$ = Select a component from $x$ using roulette wheel selection on $P$
Assign $x_i$ a random (legal) value
$c$ = Evaluate $f(x)$
if $c$ is better than $Best$ then
    $Best = c$
end if
end
```

The remainder of the paper is organised as follows. Section 2 describes the general concepts of multi-objective optimisation along with the two classes of problems that will be studied. These are the benchmark instances of Zitzler et al. [15] along with an airfoil design problem. Section 3 firstly briefly summarises the mechanics of EO and develops the notion of generic local search with diversification for these types of spaces. Finally, Sections 4 and 5 give the computational results and conclusions respectively.

2 Multi-Objective Optimisation and Test Problems

This section presents a brief overview of multi-objective optimisation as well as an introduction to the test problems and related algorithms that have been used to solve them.

2.1 Multi-objective Optimisation

For a thorough discussion of multi-objective optimisation and its relation to evolutionary metaheuristics algorithms, the reader is referred to Coello Coello [6] and Deb [7].

Multi-objective optimisation, by definition, involves the simultaneous optimisation of more than one objective. While each objective requires minimisation or maximisation, this discussion (without loss of generality), will consider objectives that need to be minimised. The minimisation of objectives is commonly expressed according to Equation 2.

$$\text{Minimise:} \overrightarrow{f}(\overrightarrow{x}) = \big(f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_m(\overrightarrow{x})\big) \tag{2}$$

Where:

$\overrightarrow{f}$ is the set of objective functions,

$\overrightarrow{x}$ is the solution vector and

m is the number of objective functions

Rather than a single optimal solution being produced, a number of trade-off solutions are generated instead. The key question becomes which solutions should become part of this trade-off (attainment) surface. This is solved by the process of Pareto dominance ranking. A decision vector $\overrightarrow{x_1}$ dominates $\overrightarrow{x_2}$ (denoted $\overrightarrow{x_1} \prec \overrightarrow{x_2}$) if it is as good on all objectives, and better on one or more objectives. Over a number of iterations, the optimisation algorithm will produce a set of decision vectors which dominate other solutions that have been generated, i.e., solutions will be generated that dominate previously non-dominated solutions. The aim is to produce an attainment surface that is as close as possible to the optimal surface (also known as the *Pareto Front*).

2.2 Test Problems

Two sets of problems will be used in this study: benchmark problems and a real-world case study.

The ZDT problems [15] are well-known, and often used to test new algorithms. Chen, Lu and Yang [5] proposed a population-based non-tau version of EO. At each iteration of the algorithm, for each solution, n mutations occur on the n solution components (i.e., one mutation per solution component). The n new solutions are then ranked using dominance ranking, and the solution with rank 0 (i.e., the solution component being the best change, so, by implication the worst solution component to begin with) is always chosen to be changed. If this were a τ version of the algorithm, components with ranks other than 0 would have the opportunity to be changed. If it is the case that more than one component is ranked at 0, a diversity preservation mechanism is invoked. This means that the resultant solution that crowds members in the external archive the least, is chosen. Across the five test problems it performs well in comparison to standard techniques such as NSGA-II.

A real-world continuous problem that can be modelled and implemented in a very similar way to the benchmark problems above is that of designing 2D airfoil sections [14]. The design of the case study closely resembles challenges encountered in a real world compressor blade design problem that is still under active investigation [10]. For that study, a reference airfoil was used, with symmetric top and bottom surfaces (no camber) and a maximum thickness of 12% of the width of the airfoil. Additional thickness constraints are applied at 25% and 50% of the chord. The particular configuration used here features two objectives, based on the lift and drag coefficients. Due to the symmetry, the reference airfoil design has relatively poor performance characteristics compared to an optimised, non-symmetric airfoil. The resulting optimisation problem features a substantial number of invalid points within the search space. They can, amongst other reasons, be caused by physically impossible shapes produced by the parametrisation, violation of thickness constraints or non-convergence of any aspect of the simulation. The optimisation problem becomes a free form deformation (FFD) problem that has a vector of eight variables in which the lift objective is to be maximised[1] while drag is to be minimised [14]. The eight variables represent four FFD control point horizontal and vertical pairs which define the exact shape of the airfoil section. Lattarulo, Kipouros and Parks [14] report that versions of tabu search, the alliance algorithm and NSGA-II have been used to solve this problem. According to them, the alliance algorithm is capable of producing the most robust designs.

[1]This objective is often minimsied by multiplying its value by -1 [10].

3 Applying Extremal Optimisation with Local Search and Diversification

A central tenet of extremal optimisation is that the contributions of individual components need to be assessable in order to ultimately determine which component will be changed in an iteration. However, for black box and inseparable problems (such as those being solved in this paper) this is not possible. Instead, at each iteration each component, in turn, is assigned a random value. The objective function values returned as a result of these new solutions can then be dominance ranked. From this, the solution components are ranked (with those that lead to non-dominated solutions ranked first) from best to worst. A component is then selected to be changed.

To supplement EO and help refine its solution, this paper adds local search and diversification components to the implementation.

3.1 Local Search

Local search is a greedy activity that is designed to take a solution to the nearest local optimum and stop. This is essential for meta-heuristic search, as they are designed to create search space trajectories and strategies, using various paradigms (such as biological metaphors), rather than fine-grained optimal solutions. For single objective optimisation, it simply becomes a matter of progressively accepting better and better solutions until further improvement is not possible. This becomes more complex for multi-objective problems. One could either optimise each of the objectives (in a round robin or random manner such as Gomez-Meneses et al. [9]) or holistically. By holistic is meant some form of dominance ranking is used. In the simplest form, if a new solution is generated from an existing solution, it will replace that solution if it dominates it on all objectives. Algorithm 2 shows how this can be achieved, particularly for inseparable problems.

Algorithm 2 The general local search algorithm.

```
repeat
  for $i$=1 to $num\_ls\_points$ do
    Assign a random solution component a random value
    Evaluate the solution's objectives
    if this solution dominates the current solution then
      Add this solution to a candidate list
    end if
  end for
  if the candidate list is not empty then
    Randomly choose one of the solutions from the candidate list to become the new solution
  end if
until no further improvement is possible (i.e. the candidate list is empty)
```

This algorithm is applied each time a feasible solution is produced. For the ZDT problems, this will be all the time as they are unconstrained problems. However, for the airfoil problem, some solutions will produce physically impossible designs, and hence are considered infeasible.

3.2 Diversification

Applying local search gives a convenient means of detecting whether the search is stuck in a local optimum or not. If the local search algorithm (above) cannot find a new non-dominated point on its first iteration, one can reasonably assume that the search may be stagnating at that point. At this stage, three options exist; do nothing and hope that the EO processes can overcome the local optimum; replace the current solution with a random solution or; replace the current solution with a known good solution. In the latter case, a collection of known good points is the external archive. Using a random member of the archive to replace the current solution would be appropriate in this instance.

The issue of balance also needs to be taken into consideration. Diversifying too frequently and not allowing EO sufficient opportunity to pursue its own search trajectory, may lead to poor quality solutions. Therefore, diversification needn't be activated all the time in these circumstances, but governed by a probability. This becomes a parameter of the process.

4 Computational Experiments

The computing platform used to perform the experiments is a 3GHz Pentium 4 based PC. Each problem instance is run across ten random seeds. The experimental programs are coded in the C language and compiled with `gcc`. The ZDT test problems [15] are used as well as the open source Xfoil evaluator [8] for the airfoil problems.

The main questions that are being asked in this study are whether local search and/or diversification help EO to find improved solutions. To this end, for the benchmark problems the key parameters that need to be tested are a) the number of test points that are used for local search (denoted *num_ls_points* in Algorithm 2 and b) the probability that diversification will be activated if local search cannot make an initial improvement to a solution (hereafter referred to as *divers*). The following sets are used for the above two parameters respectively: $\{10, 50, 100\}$ and $\{0, 0.2, 0.5, 0.8, 1\}$. The zero in the latter set ensures that diversification is not used (and local search is treated in isolation) and the one means that it will always be activated.

Table 1 shows the results of this testing across the five ZDT test problems, initially using no local search. This will serve as a baseline to the local search and diversification techniques. Hypervolume is used as optimal hypervolumes are known and, as such, quantitative comparisons can be made. All results are expressed as relative percentage deviations (RPDs) from the optimal hypervolume value [15]. Given a maximum archive size of 100, these are: 120.662137 (ZDT1), 120.328881 (ZDT2), 128.775955 (ZDT3), 120.662137 (ZDT4), 117.51495 (ZDT6).

Table 1: Results without using local search and diversification.

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Min	0.0023	0.0077	0.00776	0.2409	0.2496
Med	0.003	0.0082	0.0111	0.3387	0.264
Max	0.0045	0.0097	0.03184	0.466	0.2923

The most noticeable aspect of Table 1 is that the results for ZDT4 and ZDT6 are relatively poor. ZDT4 is a notoriously difficult (multimodal) problem, its difficulty being acknowledged in the original paper [15] in which it was introduced. The authors observed that introducing elitism was the only way they were able to improve results for a number of evolutionary algorithms

(EAs) on this problem. The candidate list used in the local search implicitly implements elitism, enhancing the ability of the algorithm with local search to find good solutions. ZDT6 has a non-uniform distribution of solutions, with more far away from the Pareto front than close to it. This also caused all the EAs tested in the original paper difficulty. By implementing local search, the modified algorithm is given a mechanism to move away from the densely distributed, mediocre solutions to the more sparsely distributed good solutions by explicitly searching to improve individual solutions. The local search is merit driven, rather than being based on a population consensus. Table 2 shows the local search and diversification results.

Table 2: The local search and diversification results.

		10			50			100		
		Min	Med	Max	Min	Med	Max	Min	Med	Max
0	ZDT1	0.0026	0.0031	0.0042	0.0017	0.0026	0.0069	0.0013	0.0024	0.0048
	ZDT2	0.0057	0.0067	0.0089	0.0044	0.0052	0.0072	0.0029	0.0036	0.005
	ZDT3	0.0046	0.0079	0.03	0.023	0.0297	0.0379	0.0069	0.0291	0.0452
	ZDT4	0.0853	0.164	0.2035	0.0445	0.0612	0.0683	0.019	0.0313	0.0419
	ZDT6	0.2293	0.237	0.2537	0.1664	0.1911	0.2168	0.1606	0.1703	0.1825
0.2	ZDT1	0.0007	0.0016	0.0022	0.0009	0.0014	0.0031	0.0009	0.0018	0.0037
	ZDT2	0.0005	0.0019	0.0105	0.0002	0.0013	0.0072	0	0.0024	0.0085
	ZDT3	0.0023	0.0062	0.0286	0.0019	0.009	0.0402	0.002	0.0265	0.0378
	ZDT4	0.001	0.002	0.003	0.0009	0.0026	0.0044	0.0011	0.0017	0.0042
	ZDT6	0.0968	0.108	0.1177	0.0843	0.097	0.109	0.0817	0.0904	0.101
0.5	ZDT1	0.0007	0.0014	0.0024	0.0005	0.0014	0.0037	0.0008	0.0014	0.0036
	ZDT2	0.0005	0.0021	0.0043	0.0003	0.0011	0.0042	0	0.0015	0.0052
	ZDT3	0.0011	0.0056	0.0365	0.0011	0.0262	0.0355	0.0015	0.0138	0.0259
	ZDT4	0.0008	0.002	0.0033	0.0007	0.002	0.0039	0.0005	0.0011	0.0048
	ZDT6	0.0857	0.0939	0.0985	0.0713	0.0849	0.0932	0.0732	0.0816	0.0873
0.8	ZDT1	0.0011	0.0014	0.0024	0.001	0.0014	0.0057	0.0008	0.0014	0.0136
	ZDT2	0.0003	0.0015	0.0034	0.0007	0.0012	0.0032	0	0.0018	0.0098
	ZDT3	0.0019	0.0105	0.0503	0.0013	0.008	0.0531	0.001	0.0265	0.0447
	ZDT4	0.001	0.002	0.0035	0.0006	0.0016	0.0046	0.0005	0.0017	0.0053
	ZDT6	0.094	0.103	0.1091	0.0717	0.0805	0.0986	0.0706	0.0798	0.0869
1	ZDT1	0.0012	0.0021	0.0033	0.0007	0.0017	0.0054	0.0004	0.003	0.0105
	ZDT2	0.0031	0.0035	0.0094	0.0018	0.0023	0.0072	0.0003	0.0014	0.0034
	ZDT3	0.0031	0.0106	0.054	0.0016	0.0037	0.0267	0.0011	0.004	0.0261
	ZDT4	0.0022	0.0183	0.0397	0.0046	0.0096	0.0159	0.0009	0.0022	0.0094
	ZDT6	0.0966	0.1205	0.1383	0.0647	0.0774	0.1036	0.0531	0.0693	0.0849

To properly make sense of the contents of Table 2, some statistical analysis needs to be undertaken. The key questions are:

1. Does the application of local search have a positive effect?
2. Does the addition of diversification to local search help the latter?
3. Which parameter combination performs best?

As the data are not normally distributed, non-parametric statistics will be used – in particular the Kruskal-Wallis test with $\alpha = 0.05$. The combination of each value of *num_ls_points* and *divers* gives 15 different groups. Considering the runs without local search and diversification gives another group. Statistical testing revealed a significant difference between the groups on RPD. Post-hoc analysis using Scheffé's method revealed that groups $(100, 0.5)$ and $(50, 0.8)$, where the first part of the ordered pair represents *num_ls_points* and the latter *divers*, performed the best. Individual analyses on each of the two variables showed that:

- There was no significant difference in terms of the number of local search points. It seems that as long as there are some (as opposed to none when local search is not activated), significantly better results will be obtained.
- There was a significant difference between values for *divers*, indicating that the level of this variable can have significant impact on results. The value of 0.5 was found to be best.

Figure 1 shows graphically the Pareto graphs for the five benchmark problems with the 'LS' versions showing the attainment surface generated using the (100,0.5) parameter combination. It may be argued that the local search version of EO has a number of extra function evaluations, and therefore must perform better as a result. To test this, EO without local search and diversification was run for five times the previous number of iterations. The results of this are given in Table 3. It is evident that the same pattern as for no local search with fewer iterations is present. This is particularly evident again for ZDT4 and ZDT6. The same statistical analysis reveals that the local search/diversification enabled EO was still significantly better, even though they take less computational time.

Table 3: Results without using local search and diversification for 500000 iterations.

	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Min	0.002249	0.00672	0.006258	0.147959	0.242569
Med	0.002731	0.007835	0.009882	0.239465	0.247855
Max	0.004029	0.009483	0.033693	0.346831	0.257016

Taking all of the above into consideration, in terms of the questions posed, both local search and diversification decidedly help EO to find better attainment surfaces.

4.1 Airfoil Design Results

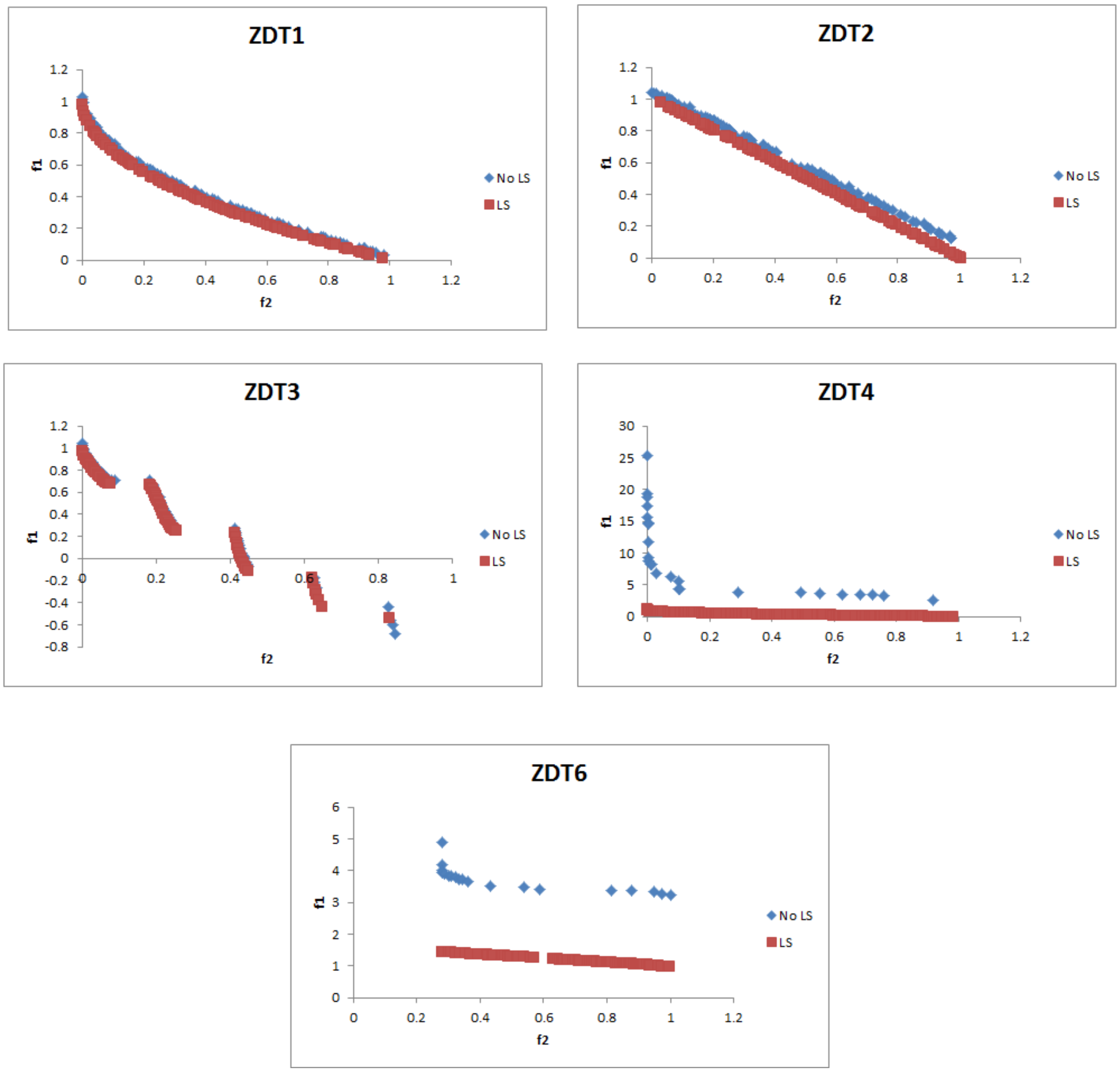
Some preliminary results for the Airfoil design problem, outlined earlier in the paper, are presented. In order to allow for some comparison to previous work, each solver was allowed 6000 function evaluations [10]. Each evaluation can take up to one minute of CPU time. As such, only a limited number of runs could be attempted. The best parameter set form the previous experiments, (100,0.5), will be used.

Figure 2 shows the final attainment surfaces for EO with and without local search/ diversification. It shows that while the non local search/diversification method can sometimes produce improved attainment surface points, the other version has a much greater coverage and uniformity. It may be hypothesised that by expending function evaluations on refinement of solutions, leading to "filling in" and extending the attainment surface, the hybrid algorithm necessarily evolved the surface *as a whole* to a lesser degree than the unmodified algorithm achieved, in places. In fact, these results compare very favourably with those reported by Hettenhausen et al. [10]. It is also evident that some form of hybrid between the two variations should be trialled. One can imagine this taking the form of the EO (or other controlling heuristic) applying local search on a sporadic basis, rather than all the time.

5 Conclusions

Local search, coupled with a diversification strategy, has been shown in this paper to be an effective supplement for multi-objective optimisation based on extremal optimisation. The

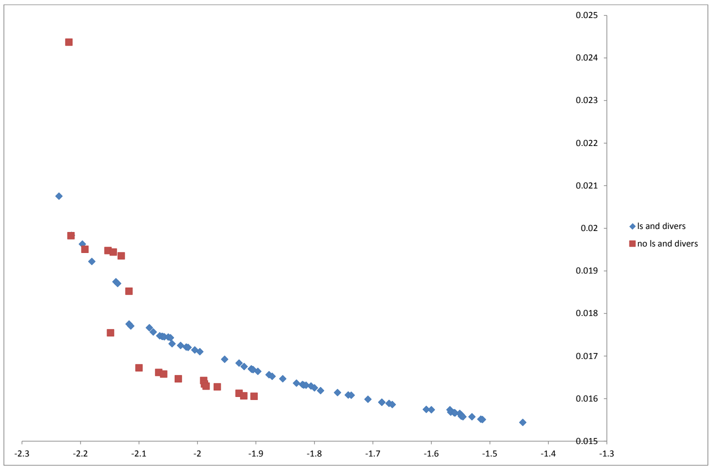
Figure 1: Pareto graphs for the five test problems showing the advantage of including local search, particularly for the challenging problems ZDT4 and ZDT6.



greedy local heuristic balances the more explorative meta-heuristic base algorithm.

Given this demonstration of concept, a much wider range of multi-objective problems needs to be tested. The problems in this paper are effectively based on black-box or inseparable functions. However, a number of separable problems, particularly combinatorial problems, exist as multi-objective problems. The generalised assignment, quadratic assignment and travelling salesman problems are examples. It will be interesting to see if any computational advantage can be leveraged out of this separability. Additionally, the model of the diversification strategy could be refined. One way to do this is to make the *divers* parameter self-tuning, as the value of it appears to be dependent on problem. Beyond this, another research direction will be to

Figure 2: Attainment surfaces of the airfoil design problem showing the greater extent and coverage obtained using local search and diversification.



modify extremal optimisation itself by adding a population component to it.

References

[1] J. Arroyo and V. Armentano. Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167(3):717–738, 2005.

[2] S. Boettcher and A. Percus. Extremal optimization: Methods derived from co-evolution. In *Proceedings of the Genetic and Evolutionary and Computation Conference*, pages 825–832. Moran Kaufmann, 1999.

[3] S. Boettcher and A. Percus. Nature's way of optimizing. *Artificial Intelligence*, 119:275 – 286, 2000.

[4] S. Boettcher and A. Percus. Extremal optimization: An evolutionary local search algorithm. In H. Bhargava and N. Ye, editors, *Computational Modeling and Problem Solving in the Networked World*, Interfaces in Computer Science and Operations Research, pages 61–77. Kluwer Academic Publishers, 2003.

[5] M. Chen, Y. Lu, and G. Yang. Multiobjective optimization using population-based extremal optimization. *Neural Computing and Applications*, 17:101–109, 2008.

[6] C. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1:269–308, 1999.

[7] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2002.

[8] M. Drela. Xfoil: An analysis and design system for low reynolds number airfoils. In *Low Reynolds number aerodynamics*, pages 1–12. Springer, Berlin, Heidelberg, 1989.

[9] P. Gomez-Meneses, M. Randall, and A. Lewis. A hybrid multi-objective extremal optimisation approach for multi-objective combinatorial optimisation problems. In *Proceedings of the Congress on Evolutionary Computation*, pages 292–299, 2010.

[10] J. Hettenhausen, A. Lewis, M. Randall, and T. Kipouros. Interactive multi-objective particle swarm optimisation using decision space interaction. In *Proceedings of the Congress of Evolutionary Computation*, pages 3411–3418, 2013.

[11] H. Ishibuchi and T. Murada. A multi-objective genetic local search algorithms and its application to flow shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics - Part C:Applications and Reviews*, 28, 1998.

[12] A. Jaszkiewicz. Genetic local search for multi-objective combinatorial optimization. *European Journal of Operational Research*, 137, 2002.

[13] J. Knowles and D. Corne. The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 98–105, 1999.

[14] V. Lattarulo, T. Kipouros, and G. Parks. Application of the multi-objective alliance algorithm to a benchmark aerodynamic optimization problems. In *Proceedings of the Congress of Evolutionary Computation*, pages 3182–3189, 2013.

[15] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.