Simple Population Replacement Strategies for a Steady-State Multi-Objective Evolutionary Algorithm

Christine L. Mumford

School of Computer Science, Cardiff University
PO Box 916, Cardiff CF24 3XF, United Kingdom
`christine@cs.cardiff.ac.uk`

Abstract. This paper explores some simple evolutionary strategies for an elitist, steady-state Pareto-based multi-objective evolutionary algorithm. The experimental framework is based on the SEAMO algorithm which differs from other approaches in its reliance on simple population replacement strategies, rather than sophisticated selection mechanisms. The paper demonstrates that excellent results can be obtained without the need for dominance rankings or global fitness calculations. Furthermore, the experimental results clearly indicate which of the population replacement techniques are the most effective, and these are then combined to produce an improved version of the SEAMO algorithm. Further experiments indicate the approach is competitive with other state-of-the-art multi-objective evolutionary algorithms.

1 Introduction

Multi-objective optimization problems are common in the real world and involve the simultaneous optimization of several (often competing) objectives. Such problems are characterized by optimum sets of alternative solutions, known as *Pareto sets*, rather than by a single optimum. Pareto-optimal solutions are *non-dominated solutions* in the sense that it is not possible to improve the value of any one of the objectives, in such a solution, without simultaneously degrading the quality of one or more of the other objectives in the vector.

Evolutionary algorithms (EAs) are ideally suited to multi-objective optimization problems because they produce many solutions in parallel. However, traditional approaches to EAs require scalar fitness information and converge on a single compromise solution, so need to be adapted if a set of viable alternatives is required for multi-objective optimization. Like their single objective counterparts however, most multi-objective EAs focus the genetic search on the selection stage, and use a fitness function to bias the choice of parents for breeding, favoring the 'better individuals'. In a multi-objective context, fitness functions are usually based either on a count of how many contemporaries in the population are dominated by a particular individual, or alternatively, on a count of by how many contemporaries the individual is itself dominated. This technique, known as Pareto-based selection, was first proposed by Goldberg [3], and is favored, in one form or another, by most researchers (for example see [1, 2, 10, 11]). In contrast, SEAMO (a Simple Evolutionary Algorithm for Multi-objective Optimization, [7, 9]) uses uniform selection and thus does not need any fitness functions to bias the selection of parents. Instead progression of the genetic search relies entirely on a few simple rules for replacing individuals with newly generated offspring in a steady-state environment. The implementation of these rules usually requires nothing more complicated than a simple 'who shall live and who shall die' decision, based on the outcome of a straight comparison between the solution generated by an offspring with those produced by its parents (or other population members). Despite its simplicity, SEAMO has produced some very good results in earlier studies [7, 9].

The present study explores a range of simple population replacement strategies for a steady-state multi-objective EA, based on the SEAMO framework. Its purpose is twofold:

- to discover the best strategies
- and use them to improve the original SEAMO algorithm.

The evolutionary strategies are developed and compared using the multiple knapsack problem (MKP) as a testbed. The instances chosen are kn500.2 and kn750.2 of [10], consisting of 500 and 750 items, respectively, in two knapsacks. The best strategies are finally combined to produce an improved version of the SEAMO algorithm, and its performance is compared to other state-of-the-art multi-objective EAs, on various multi-objective functions.

2 A Basic Framework for the SEAMO Algorithm

The SEAMO framework, outlined in Figure 1, illustrates a simple steady-state approach, which sequentially selects every individual in the population to serve as the first parent once, and pairs it with a second parent that is selected at random (uniformly). A single crossover is then applied to produce one offspring, and this is followed by a single mutation. Each new offspring will either replace an existing population member, or it will die, depending on the outcome of the chosen replacement strategy. This paper will investigate different replacement strategies for lines 10 – 13 in Figure 1.

2.1 The Original SEAMO Algorithm

In the original SEAMO algorithm, an offspring is evaluated using the following criteria:

1. Does offspring dominate either parent?
2. Does offspring produce any global improvements on any Pareto components?

```
 Procedure SEAMO
1. Begin
2.     Generate $N$ random individuals {$N$ is the population size}
3.     Evaluate the objective vector for each population member and store it
4.     Repeat
5.         For each member of the population
6.             This individual becomes the first parent
7.             Select a second parent at random
8.             Apply crossover to produce single offspring
9.             Apply a single mutation to the offspring
10.            Evaluate the objective vector produced by the offspring
11.            if offspring qualifies
12.                Then the offspring replaces a member of the population
13.            else it dies
14.        Endfor
15.    Until stopping condition satisfied
16.    Print all non-dominated solutions in the final population
17.End
```

Fig. 1. Algorithm 1 A basic framework for SEAMO

On the basis of this 'superiority test', the offspring will replace one or other of its parents, if it is deemed to be better.

On average an offspring will have 50 % genetic material in common with each parent, and, for this reason, parental replacement is favored in SEAMO in the hope that it will encourage the maintenance of genetic diversity within the population and thus help avoid premature convergence. One purpose of the current study is to put assumptions like this to the test, and also try some alternative strategies.

In more detail, the superiority test applied in the original SEAMO algorithm progresses as follow. To start with, a new offspring is compared with its first parent, and replaces that parent in the population if it dominates it, provided that the offspring is not a duplicate, in which case it dies immediately (the deletion of duplicates is explained see later in the present section). Any offspring that fails the first test, and thus does not dominate its first parent, is next compared with its second parent. Similar to before, a non-duplicate, dominating offspring will replace its second parent in this situation. If an offspring fails to dominate either parent, however, it will usually die at this stage. The replacement of population members by dominating offspring ensures that the solution vectors move closer to the Pareto front as the search progresses. To additionally ensure an improved range of coverage, the dominance condition is relaxed whenever a new global best value is discovered for any of the individual components of the solution vector (i.e. for improved maximum profits in individual knapsacks). Care has to be taken, however, to ensure that global best values for other components (i.e. maximum profits in other knapsacks) are not lost when a dominance condition is relaxed. Ensuring that global best components are not lost is straightforward if multi-objective optimization is restricted to two components in the solution vector, as is the case in this paper: whenever an offspring produces an improved global best for either of the components, if the global best for the second component happens to occur in one of the parents, the offspring will simply replace the other parent. One weakness with the replacement strategies applied in the original SEAMO algorithm is that offspring that neither dominate nor are dominated by their parents will usually die immediately and their potential is wasted.

To complete the description of the original SEAMO algorithm, an explanation of the 'deletion of duplicates' policy is now given. A simple way to help promote genetic diversity is avoid the propagation of genetic duplicates through the population. Thus, before a final decision is made on replacement of a parent, a dominating offspring is compared with every individual in the current population, and if the offspring is duplicated elsewhere in the population, the offspring dies and does not replace its parent. For speed and simplicity it is the phenotypic values of the offspring that are compared to those of other population members (i.e. the values of the Pareto vectors) rather than the genotypic values (i.e. the permutation lists of items). Ideally, the genotypes should be compared, but due to the lengths of the permutation lists, this would be very time consuming.

3 Experimental Design

An order-based representation with a first fit decoder is used for the MKP, and Cycle Crossover (CX) [8] is used as the recombination operator. A simple mutation operator swaps two arbitrarily selected objects within a single permutation list. The representational scheme was chosen because it produced the best results in a recent comparative study, [5]. The reader is referred to the earlier work for full details.

In all the experiments that follow, each strategy is tested by 30 replicate runs, initialized with different random seeds. 2D plots are obtained by combining all 30 results files, for each experiment, and extracting the non-dominated solutions from the combined results. 2D plots give a good fast visual indication the solution quality, spread and range of the approximate Pareto sets produced by the competing strategies. Additionally, some performance metrics are used to compare the improved SEAMO approach with other state-of-the-art EAs.

4 Simple Strategies: Replacing a Population Member with a Dominating Offspring

When using a steady-state evolutionary algorithm, a decision has to be made each time that a new offspring is created, whether that offspring will live or die. If it is allowed to live, one has to determine, which population member to replace. In the SEAMO framework no selective pressure is applied when choosing parents, so if the population is to improve over time, new individuals entering the population need to be generally superior to those individuals that they are replacing. In the first set of experiments we shall compare three simple strategies that replace a current population member with an offspring that dominates that individual:

1. offspring replaces a population member that it dominates at random
2. offspring replaces a parent that it dominates
3. offspring replaces a parent if it dominates either parent, otherwise it replaces a population member that it dominates at random.

To implement the first strategy, and part of the third, the population is sampled without replacement until a suitable candidate for replacement is found, or until the whole population has been exhausted. In the latter case the offspring will be allowed to die. The pseudocode is given below.

```
11.             Repeat
12a.                Select population member at random without replacement
12b.                If offspring dominates selected individual
12c.                    Then offspring replaces it in the population; **quitloop**
12d.             Until all members of population are tried
13.         {offspring dies if it does not replace any member of the population}
```

The second strategy is implemented by testing the offspring with the first parent and then the second parent, in the way described in the earlier section for the original SEAMO algorithm. An offspring will replace a parent that it dominates.

The third strategy is a combination of the first two. A new offspring will replace a parent if it dominates either of them. When this is not the case the offspring will replace a population member that it dominates at random. If it fails to dominate any individual, it dies. For each strategy, we assess the effect that deleting duplicates has on the results. We use population sizes of 200 and 250 for kn500.2 and kn750.2 respectively, and stop the runs after 500 generations have elapsed.

4.1 Results for the Simple Strategies

Table 1. Average run times of experiments in seconds

Problem	1a	1b	2a	2b	3a	3b
kn500.2	19	19	9	9	19	19
kn750.2	31	32	15	15	31	32

Figure 2 summarizes the results for replacement strategies 1, 2 and 3 on kn500.2 and kn750.2. For each trace the non-dominated solutions are extracted from the combined results of 30 replicated experiments. Clearly strategy 3 appears to be the most successful. Figure 3 indicates that failing to delete duplicates

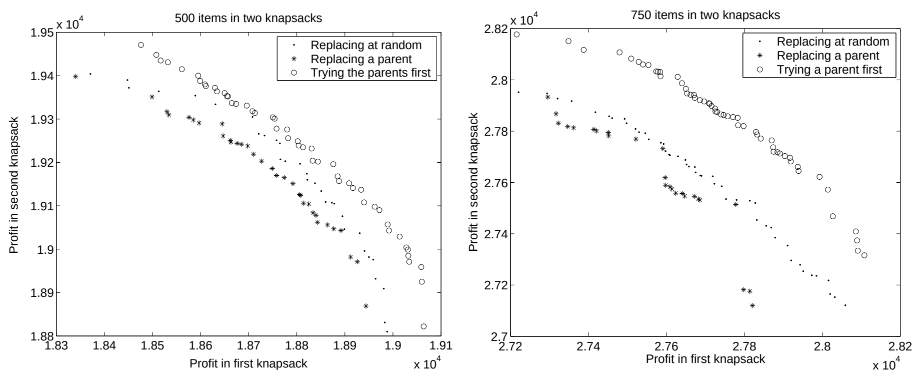

Fig. 2. Comparing replacement strategies with duplicates deleted

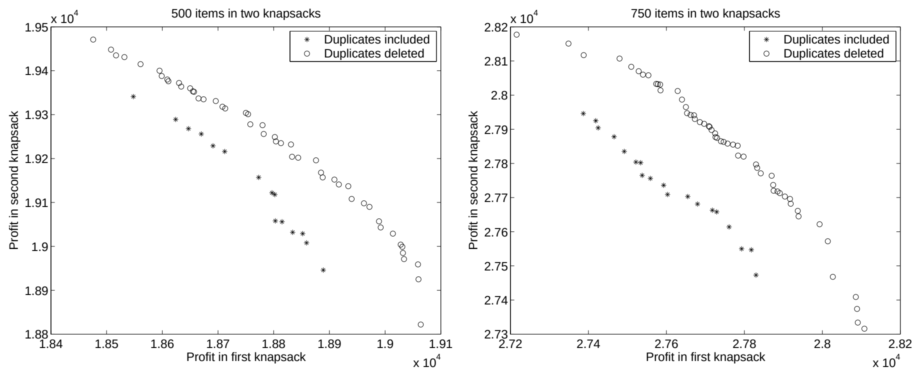

Fig. 3. Examining the effect the deleting duplicates has on the results produced by strategy 3

has a serious deleterious effect on the results for strategy 3. (A similar pattern was observed for strategies 1 and 2.)

Table 1 compares the average run times, on a 1.5 GHz PC laptop for the three strategies. For experiments 1a, 2a and 3a phenotypic duplicates are allowed, but in 1b, 2b and 3b, the duplicates are deleted. From table 1 it would appear that including a routine to test and exclude phenotypic duplicates, does not add to the run time of the EA. Although this may seem counter-intuitive, closer examination reveals that, as a direct result of deleting the duplicates, fewer new offspring genotypes are copied into the population, and copying permutation lists 500 or 750 item long is indeed a lengthy business. In the next section we will try improving on strategy 3. Phenotypic duplicates will be deleted in all future experiments.

5 Further Strategies

As discussed in Section 2.1, replacing parents with their offspring is likely to more successfully preserve genetic diversity than replacing arbitrary members of the population with the offspring of other individuals. Nevertheless, replacement strategy 3 will frequently maintain offspring that are dominated by both of their parents. Perhaps it would make better sense if such individuals were allowed to die? Strategy 4 will investigate the following:

Replacement Strategy 4

1. **if** offspring dominates either parent it replaces it
2. **else if** offspring is neither dominated by nor dominates either parent it replaces another individual that it dominates at random
3. **otherwise** it dies

Strategy 4 differs from strategy 3 by killing off offspring that are dominated by both parents. Unlike the simpler strategy 2, though, strategy 4 will maintain offspring that are neither dominated by nor dominate their parents, provided a weaker candidate can be found elsewhere in the population. The loss of such offspring is a weakness of the original SEAMO algorithm. Unfortunately, occasional loss of a non-dominated individual will occur, even applying stategy 4, if a weaker individual cannot be found. This is inevitable when maintaining a constant population size.

In the original SEAMO algorithm, dominance rules are relaxed when new global best components appear in the objective vectors, and an offspring is then allowed to replace one of its parents (or occasionally another individual) whether it dominates that parent or not. This approach tends to increase the range of values in the solution set. Strategy 5 extends strategy 4 to incorporate new global best components. The precise mechanism is outlined below:

Replacement Strategy 5

1. **if** offspring harbors a new best-so-far Pareto component
   (a) it replaces a parent, if possible
   (b) **else** it replaces another individual at random
2. **else if** offspring dominates either parent it replaces it
3. **else if** offspring is neither dominated by nor dominates either parent it replaces another individual that it dominates at random
4. **otherwise** it dies

(Note: Condition 1 (b) in strategy 5 is not needed for problems with only two objectives, but is required for three or more.) The parameters for population sizes and the number of generations are the same as set previously in section 4

5.1 Results for Strategies 4 and 5

Strategies 3, 4 and 5 are compared in Figure 4. Clearly, strategy 5 produces the best results, as they are much more widely spread. An additional set of experiments confirmed that SEAMO using strategy 5 (SEAMO2) is able to produce better results than the original SEAMO algorithm (see Figure 5).

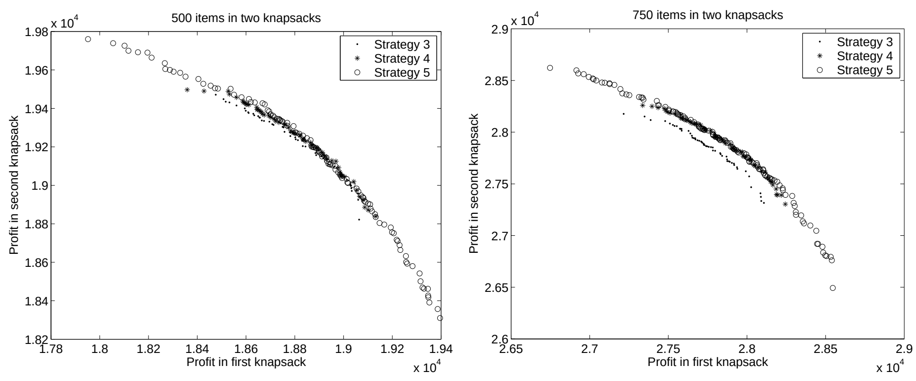

Fig. 4. Comparing strategies 3, 4 and 5

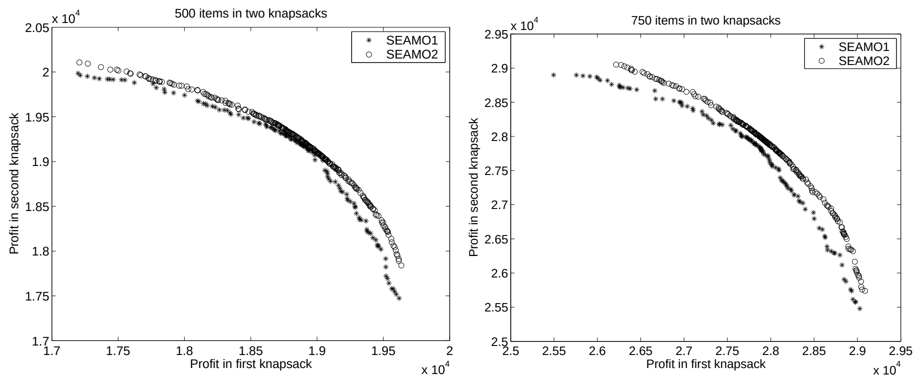

Fig. 5. Comparing SEAMO with strategy 5 (SEAMO2) with the original SEAMO (SEAMO1)

6 Comparing SEAMO Using Strategy 5 with Other State-of-the-art EAs

A final set of experiments compares the performance of SEAMO using strategy 5 (i.e.SEAMO2) with NGSA2 (a fast elitist non-dominated sorting genetic algorithm) [2], PESA (the Pareto envelope-based seletion algorithm) [1], and SPEA2 an improved version of SPEA (the strength Pareto evolutionary algorithm) [10]. The test problems used are kn750.2, plus four continuous functions, SPH-2, ZDT6, QV, and KUR, [11]. The results for PESA, NSGA2 and SPEA2, were obtained from [11].

For kn750.2 a population of 250 is used and 30 replicate runs collected for SEAMO2, as previously. However, the experiments in this section are allowed to run for 1920 generations, to make results comparable with those in [11]. The parameters for the continuous function experiments (domain size, population size, and number of evaluations etc.) are as given in [11]. For each algorithm

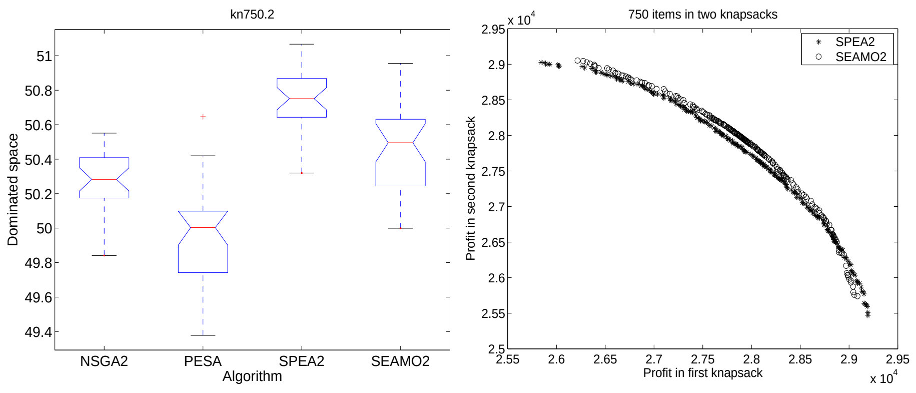

Fig. 6. Comparing SEAMO2 with SPEA2

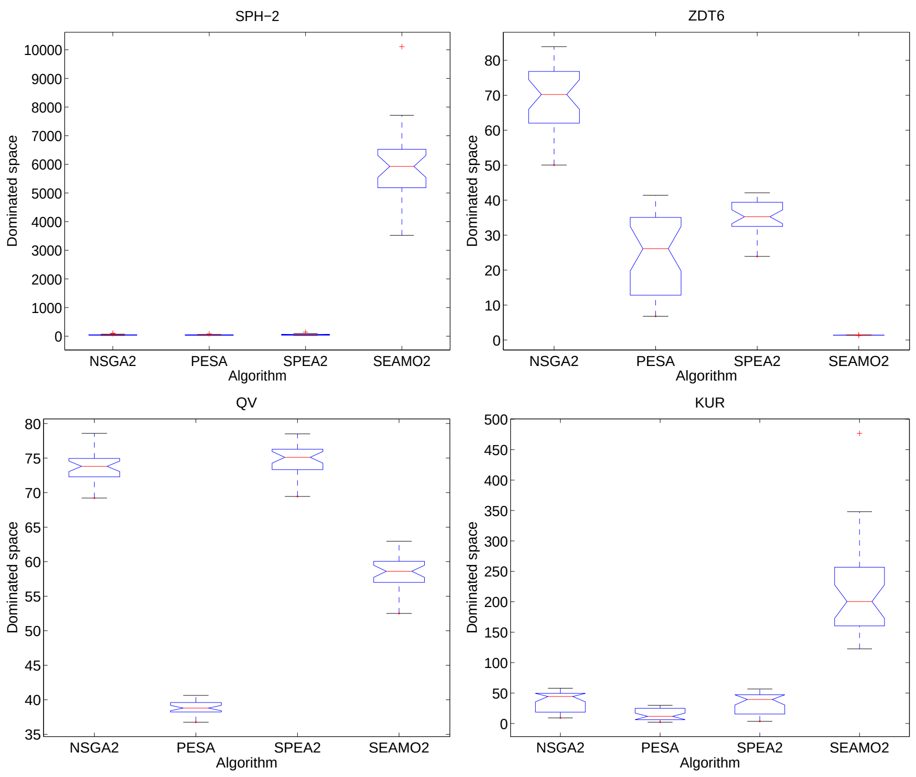

Fig. 7. Comparing SEAMO2 with NSGA2, PESA, and SPEA2

on each function, 30 replicate runs are collected, each run consisting of 10,000 generations on populations of 100. The continuous test problems are all specially contrived minimization problem of two objectives and 100 variables. For all of the continuous functions the solutions are coded as real vectors of length 100 for SEAMO2, and one-point crossover acts as the recombination operator. The mutation operator is based on the non-uniform mutation described on page 111 of [4]. For full details of the implementation of non-uniform mutation, the interested reader is referred to [7].

An important feature of SEAMO algorithms is their deletion of duplicates, designed to help maintain diversity and prevent premature convergence. For the knapsack problem and other combinatorial problems, where the objective functions can take on only limited number of discrete values, phenotypic duplicates are easily identified as individuals with matching solution vectors. With continuous functions, however, exact duplicates are likely to be rare. For this reason, values for component objective functions x_i and x'_i of $\mathbf{x}$ and $\mathbf{x}'$, respectively, are deemed to be equal if and only if $x_i - \epsilon \leq x'_i \leq x_i + \epsilon$, where ϵ is an error term, which is set at $0.00001 \times x_i$ for the purpose of these experiments.

SEAMO2 is compared with its competitors using the two metrics, $\mathcal{S}$ and $\mathcal{C}$, described in [10]. For the purpose of the $\mathcal{S}$ metric, the minimization problems, SPH-2, ZDT6, QV and KUR, have been transformed into maximization problems by replacing the Pareto values with their reciprocals. Furthermore, all the $\mathcal{S}$ hypervolumes have been scaled as percentages of suitable reference values for ease of tabulation. The reference values are 1.649e+009, 40, 500, 1 and 0.002 for kn750.2, SPH-2, ZDT6, QV and KUR respectively.

Figure 6 compares the performance of the various algorithms on kn750.2 The boxplots on the left show the spread of dominated space produced by the 30 replicate runs collected for each algorithm, and the 2D plot on the right compares the non-dominated solutions produced by SEAMO2 and SPEA2 directly. From the boxplots it is clear that SPEA2 and SEAMO2 are the leaders with SPEA2 performing a little better than SEAMO2. However, the 2D plots suggest that the solution quality produced by SEAMO2 is slightly better than that of SPEA2. (Further evidence for is provided by the coverage metric). Figure 7 gives the boxplots showing the dominated space obtained from the experiments with the continuous functions. Clearly SEAMO2 performs extremely well, with respect to this metric, on SPH-2 and KUR, not so well in QV and very poorly indeed on ZDT6. (Note: a high average of 5907 was obtained for SEAMO2 on SPH-2, distorting the plots for SPH-2 in Figure 7. This distortion seems to be an unfortunate feature of the transformation process used to convert functions from a minimization to maximization, and does not reflect superiority on a scale suggested by this result.)

Table 2 gives the average values for $\mathcal{C}$ = Coverage $(A \succeq B)$ (the number of points in set B that are weakly dominated by points in set A). The standard deviations are given in brackets. Table 2 shows a very strong performance for SEAMO2 on kn750.2, SPH-2, and KUR, and a performance comparable with

Table 2. Average values (and standard deviations) for Coverage ($A \succeq B$)

Coverage ($A \succeq B$)						
Algorithm		Test problems				
A	B	kn750.2	SPH-2	ZDT6	QV	KUR
SEAMO2	NSGA2	73.5 (20.0)	85.5 (14.1	0 (0)	36.9 (11.8)	93.1 (8.9)
	PESA	69.4 (19.4)	88.0 (9.5)	0 (0)	52.1 (11.5)	89.6 (16.8)
	SPEA2	72.5 (13.1)	81.4 (13.4)	0 (0)	35.0 (11.7)	93.4 (7.4)
NSGA2	SEAMO2	11.7 (15.5)	0 (0)	97.7 (0.3)	35.5 (15.7)	0.2 (0.8)
PESA		10.8 (11.8)	0 (0)	96.9 (1.4)	0.23 (0.6)	0.15 (0.8)
SPEA2		9.7 (9.4)	0 (0)	97.7 (0.3)	33.6 (19.7)	0(0)

NSGA2 and SPEA2 on QV. Notably, SEAMO2 performs very poorly on ZDT6 for coverage as well as for hypervolume.

To summarize, Figures 6 and 7 and Table 2 show that SEAMO2 outperforms its competitors on SPH-2 and KUR for both metrics and additionally outperforms the other EAs on kn750.2 and QV (marginally) for Coverage ($A \succeq B$). SEAMO2 performs very poorly on ZDT6, however.

Some caution is required in interpreting the results in this section. For the knapsack problems SEAMO2 uses a different representation scheme to that of its competitors, and slightly different mutation and recombination operators are used by SEAMO2 on the continuous problems. The results for SEAMO2 are, nevertheless, encouraging. Perhaps the performance of the other EAs could be improved with some changes to the representations and operators.

7 Conclusions and Future Work

This paper explores some simple evolutionary strategies for an elitist, steady-state Pareto-based multi-objective evolutionary algorithm. It validates the approach developed earlier for the SEAMO algorithm and also produces some improvements. The paper demonstrates experimentally that simple population replacement strategies coupled with the deletion of duplicates can produce excellent results, without the need for dominance ranking or global fitness calculations. Furthermore, the results clearly indicate that, despite its simplicity, the SEAMO approach is competitive with other state-of-the-art multi-objective evolutionary algorithms. Since the original submission of the present paper, further work has produced encouraging results for some hierarchical versions of the SEAMO2 algorithm, [6]. However, even these improvements have failed to lift performance on the ZDT6 continuous function. Work in progress is focussed on parallel implementations and also on improving the performance of the algorithm on non-uniformly spread functions, such as ZDT6.

References

1. Corne D W, Knowles J D, and Oates M J: The Pareto envelope-based selection algorithm for multiobjective optimization. *Parallel Problem Solving from Nature – PPSN VI*, Lecture Notes in Computer Science 1917 (2000) 839–848, Springer.
2. Deb K, Agrawal S, Pratap A, and Meyarivan T: A fast elitist non-dominated sorting genetic algorithm for mult-objective optimization: NSGA-II, *Parallel Problem Solving from Nature – PPSN VI*, Lecture Notes in Computer Science 1917 (2000) 849–858, Springer.
3. Goldberg D E: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).
4. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996).
5. Mumford C L (Valenzuela): Comparing representations and recombination operators for the multi-objective 0/1 knapsack problem, *Congress on Evolutionary Computation (CEC)* Canberra Australia (2003) 854–861.
6. Mumford C L (Valenzuela): A hierarchical approach to multi-objective optimization, *Congress on Evolutionary Computation (CEC)* Portland, Oregon (2004) (to appear).
7. Mumford-Valenzuela C L: A Simple Approach to Evolutionary Multi-Objective Optimization, In *Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications*, edited by Ajith Abraham, Lakhmi Jain and Robert Goldberg. Springer Verlag (2004) London.
8. Oliver I M, Smith D J, and Holland J R C: A study of permutation crossover operators on the traveling salesman problem, *Genetic Algorithms and their Applications:Proceedings of the Second International Conference on Genetic Algorithms* (1987) 224–230.
9. Valenzuela C L: A simple evolutionary algorithm for multi-objective optimization (SEAMO), *Congress on Evolutionary Computation (CEC)*, Honolulu, Hawaii (2002) 717–722.
10. Zitzler E and Thiele L: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Transactions on Evolutionary Computation*, 3(4) (1999) 257–271.
11. Zitzler E, Laumanns M, and Thiele L: SPEA2: Improving the strength Pareto evolutionary algorithm, TIK-Report 103, Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, {zitzler, laumanns, thiele}@tik.ee.ethz.ch.(2001) (Data and results downloaded from: http://www.tik.ee.ethz.ch/zitzler/testdata.html)