A Simple Evolutionary Algorithm for Multi-Objective Optimization (SEAMO)

Christine L. Valenzuela
Department of Computer Science
Cardiff University
United Kingdom

Abstract - A simple steady-state, Pareto-based evolutionary algorithm is presented that uses an elitist strategy for replacement and a simple uniform scheme for selection. Throughout the genetic search, progress depends entirely on the replacement policy, and no fitness calculations, rankings, subpopulations, niches or auxiliary populations are required. Preliminary results presented in this paper show improvements on previously published results for some multiple knapsack problems.

I. INTRODUCTION

Multi-objective optimization problems involve the simultaneous optimization of several (often competing) objectives, and usually there is no single optimal solution. Instead, multi-objective optimization problems tend to be characterized by a set of alternative solutions, each of which must be considered equivalent in the absence of further information regarding the relative importance of each of the objectives in the solution vectors. Such a solution set is called the *Pareto-optimal set*, and the objective values in the set are located at the *Pareto front*. 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 for 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, rather than on a set of viable alternatives. When a set of viable alternatives is required, it is essential to incorporate the concept of Pareto dominance into the EA. This is usually achieved by using some form of Pareto-based selection. Pareto-based fitness assignment was first proposed by Goldberg [6], the idea being to assign equal probability of reproduction to all non-dominated individuals. The method consisted of assigning rank 1 to all non-dominated individuals and removing them from contention, then finding a new set of non-dominated individuals, ranked 2, and so on.

Fonseca and Fleming [5] have proposed a slightly different scheme for fitness assignment, whereby an individual's rank corresponds to the number of individuals in the current population by which it is dominated. Tournament selection based on Pareto dominance has also been proposed (see Horn, Nafpliotis, and Goldberg [8]). However, Pareto-based ranking does not, on its own, guarantee that the Pareto set is uniformly sampled. Indeed, finite populations tend to converge to a single solution, due to stochastic errors in the selection process, known as *genetic drift*. The addition of fitness sharing [6] and niches to the multi-objective domain was implemented by (Fonseca and Fleming [5]) and (Horn, Nafpliotis and Goldberg [8]) in an attempt to reduce the effect of genetic drift. In addition, Zitzler and Thiele [13] and Corne, Knowles and Oates [3] have introduced an element of elitism into their EAs in the form of auxiliary or archive populations. An external population used in this way comprises non-dominated individuals copied from the regular population during the genetic search. Individuals can be selected for mating from either population. A simpler approach to elitism is used in [2]. Here the population is divided into a number of subpopulations or *tribes* and fitness is evaluated on the basis of dominance relationships. Each new pair of offspring that are generated replace the weakest individuals in their tribe. No fitness sharing, niche calculations or other elaborate diversity maintaining routines are used by this algorithm.

The present paper proposes an even simpler approach than [2]. It disposes of all selection mechanisms based on fitness values and instead uses a straightforward uniform selection procedure (i.e. each population member has an equal chance of being selected). Thus no dominance ranking is required. In the new approach improvements to the population and progress of the genetic search depend entirely upon a replacement strategy that follows a few simple rules:

1. parents can be replaced only by their own offspring,
2. offspring can only replace parents if the offspring are superior – thus the scheme is elitist,
3. duplicates in the population are deleted.

This evolutionary algorithm depends on rules 1 and 3 to maintain diversity and prevent premature convergence and on rule 2 to ensure that the best solutions are not lost. As with other EAs, finding good solutions in the first place depends largely on the quality of the representation, the choice of genetic operators and the various settings for the EA search parameters, such as population size, crossover and mutation rates etc.. The EA itself is elistist, thus no archive population is needed.

II. THE 0-1 MULTIPLE KNAPSACK PROBLEM

The 0-1 multiple knapsack problem (0-1 MKP) is a generalization of the 0-1 simple knapsack problem, and is a well known member of the NP-hard class of problems. In the simple knapsack problem, a set of objects $O = \{o_1, o_2, o_3, ..., o_n\}$ and a knapsack of capacity C are given. Each object o_i has an associated profit p_i and weight w_i. The objective is to find a subset $S \subseteq 0$ such that the weight sum over the objects in S does not exceed the knapsack capacity and yields a maximum profit. The 0-1 MKP involves m knapsacks of capacities $c_1, c_2, c_3, ..., c_m$. Every selected object must be placed in all m knapsacks, although neither the weight of an object o_i nor its profit is fixed, and will probably have different values in each knapsack. The present study is confined to problems involving two knapsacks, i.e. $m = 2$.

III. THE EVOLUTIONARY ALGORITHM

A. *The Representation and Decoder*

Several approaches have been suggested for representing solutions to knapsack problems for EAs. Michalewicz [9] identifies three classes: algorithms based on penalty functions, algorithms based on repair methods, and algorithms based on decoders. The main challenge with the knapsack problem is to ensure that the EA does not waste vast amounts of its time in generating illegal solutions with over-full knapsacks.

In the present paper, solutions are represented as simple permutations of the objects to be packed. A decoder then packs the individual objects, one at a time, starting at the beginning of the permutation list, and working through. For each object that is packed, the decoder checks to make sure that none of the weight limits is exceeded for any knapsack. Packing is discontinued as soon as a weight limit is exceeded for a knapsack, and when this is detected the final object that was packed is removed from all the knapsacks. Thus, each knapsack contains exactly the same objects as required, and each solution that is generated is a feasible solution. Cycle crossover [10] is used as the recombination operator, and the mutation operator swaps two arbitrarily selected objects within a single permutation list. Cycle crossover was selected as the recombination operator because it transmits absolute positions of objects in the permutation lists from the parents to the offspring. Neither edge based nor order based operators would seem to be appropriate here, for a set membership problem such as this.

B. *SEAMO*

The Simple Evolutionary Algorithm for Multi-objective Optimization (SEAMO), is outlined in Figure 1.

Procedure *SEAMO*
begin
  Generate N random permutations {N is the population size}
  Evaluate the objective vector for each structure and store it
  Record the *best-so-far* for each objective function
  Repeat
    For each member of the population
      This individual becomes the first parent
      Select a second parent at random
      Apply crossover to produce offspring
      Apply a single mutation to the offspring
      Evaluate the objective vector produced by offspring
      If offspring's objective vector improves on any *best-so-far*
        Then it replaces one of the parents
                 and *best-so-far* is updated
      Else If offspring dominates one of the parents
        Then it replaces it
                 (unless it is a duplicate, then it is deleted)
    Endfor
  Until stopping condition satisfied
  Print all non-dominated solutions in the final population
End

Fig. 1. Algorithm 1 A Simple Evolutionary Algorithm for Multi-objective Optimization

For the purposes of the present, preliminary study, the four test problems of Zitzler and Thiele [13] with two knapsacks are used to demonstrate the viability of the new approach. The test problems can be obtained from:

http://www.tik.ee.ethz.ch/zitzler/testdata.html

and consist of 100, 250, 500 or 750 objects. Restricting the test-bed to two knapsacks means that solutions can be plotted using standard 2D graphics, and their quality easily visualized.

When two knapsacks are used, SEAMO's objective vector for each population member contains two values: the profits associated knapsack 1 and knapsack 2 respectively. The idea is to breed a diverse population of solu-

tion pairs that is as close to the Pareto front as is possible. The dual aims pursued during the search process are: (1) to move the current solutions in the population ever closer to the Pareto front, and (2) to extend the diversity of the solution set by improving on the individual global best profits for knapsack 1 and knapsack 2. Improvements in both (1) and (2) are achieved by the replacement strategy used in SEAMO, and not by the selection process.

The selection procedure for SEAMO is very simple and does not rely on fitness calculations or dominance relationships. Each individual in the population serves as the first parent once, and the second parent is then selected at random (uniformly). Objective values and dominance relationships are only considered at the replacement stage, and it is here, rather than during selection, that the pressure for improvement is applied.

To be more precise, the replacement of a parent by its offspring is considered whenever an offspring is deemed to be superior to that parent. This idea, called *pre-selection* when it was first suggested in [1], was originally used for EAs with scalar objective functions. The technique easily extends to multi-objective optimization, however. In the present study, the superiority test is applied first of all to the first parent, and then to the second parent if that fails. Usually superiority is measured as a dominance relationship, i.e. if an offspring dominates its parent, it replaces it in the population. 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 improved coverage of the Pareto set, the dominance condition is relaxed whenever a new global best value is discovered for the total profit in either of the knapsacks. Care has to be taken, however, to ensure that the global best value for the other knapsack is not lost when a dominance condition is relaxed. As a final precaution the solution vector for a dominating offspring is compared with all the solution vectors in the current population before a final decision is made on replacement. If the solution vector produced by the offspring is duplicated elsewhere in the population, the offspring dies and does not replace its parent. The deletion of duplicates helps maintain diversity in the population and thus helps to avoid the premature convergence of the population to identical sets of a small number of solution vectors. The final action of SEAMO is to save all the non-dominated solutions from the final population to a file.

IV. RESULTS

Results are presented for four multiple knapsack problems taken from Zitzler and Thiele [13]. These were chosen because the previous authors have already carried out an extensive set of comparisons for various EAs and shown that their algorithm, the strength Pareto evolutionary algorithm (SPEA), is superior to others such as the vector evaluated genetic algorithm (VEGA) [11], Hajela's and Lin's genetic algorithm (HLGA) [7], the niched Pareto genetic algorithm (NPGA) [8] and the non-dominated sorting genetic algorithm (NSGA) [12]. In addition, Zitzler and Thiele have calculated optimal Pareto sets for three of the problems using integer linear programming together with the epsilon-constraint method.

The results produced using the SEAMO algorithm are compared graphically with SPEA and with the Pareto optimal values (where available). For all the experiments in section IV-A the total number of evaluations is the same for SEAMO and SPEA. The results for SPEA were obtained from the Zitzler and Thiele web site (URL given earlier) and comprise solution sets for 30 replicate runs (held in 30 separate files) for each of the problems. Important parameter settings are given in Table I for three of the four test problems.

For the purposes of the present study, the 30 solution sets for the replicate runs of SPEA were combined for each problem and all non-dominated solutions were then extracted from the combined files. Thus the total numbers of evaluations required to produce the final non-dominated solutions sets for each problem are given by:

$$(population\ size) \times (number\ of\ generations) \times 30$$

A. *Initial Comparative studies*

TABLE I
PARAMETER SETTINGS FOR SPEA.

Number of objects	Number of knapsacks	Population size	Number of generations
250	2	150	500
500	2	200	500
750	2	250	500

In the first set of experiments the performance of SEAMO and SPEA are compared on the 500 and the 750 object two knapsack problems, using the same population sizes and numbers of generations for SEAMO that were used for SPEA (see Table I). The results for 30 replicate runs for each problem are combined for SEAMO and the non-dominated solutions extracted in exactly the same way as was done for the SPEA results. Individual runs of SEAMO took about 12 seconds each for the 500

objects problem and about 23 seconds for the 750 objects problem, on a Pentium III processor with 128 Mb of RAM. The plots for these experiments are shown in Figures 2 and 3.

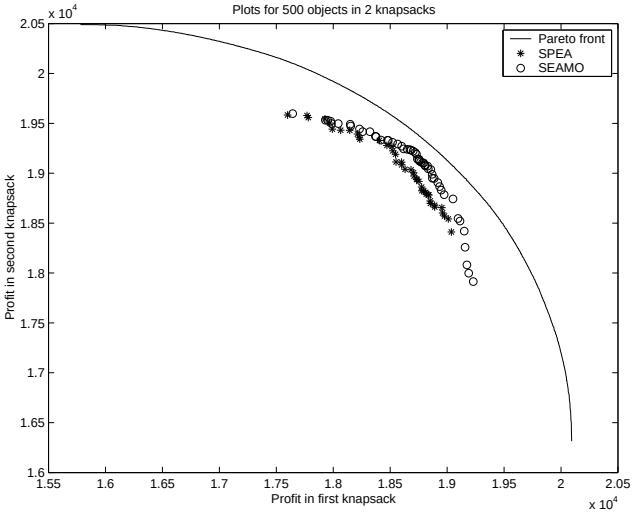

Fig. 2. Non-dominated solutions from 30 replicate runs of SEAMO and SPEA for the 2 knapsack problem with 500 objects

Clearly, the results for SEAMO are better than those for SPEA, and this is particularly noticeable for the 750 object problem.

For the second set of experiments the total number of evaluations is kept the same but single runs of SEAMO generate the non-dominated solutions instead of replicate runs. Each run, in this set of experiments, consists of 30 times more evaluations than were used for the same problem in the previous set of experiments. The purpose of these experiments is to see whether the results improve when more effort is put into running the individual EAs. For the 500 object problem the total number of evaluations is $200 \times 500 \times 30 = 3,000,000$, and for 750 objects, $250 \times 500 \times 30 = 3,750,000$.

Figure 4 shows the results of running SEAMO on the 500 object, two knapsack problem. For the first SEAMO run, a population of 200 was used and the EA was run for 15,000 generations. For the second run, SEAMO used a population of 500 for 6,000 generations, and in the final run the population size was 1,000 and SEAMO was run for 3,000 generations.

Figure 5 illustrates the runs for SEAMO on the 750 object, two knapsack problem. Once more the total number of evaluations is the same for all experiments. This time SEAMO population sizes of 250, 500 and 1,000 were tried and run for 15,000, 7,500 and 3,750 generations, respectively.

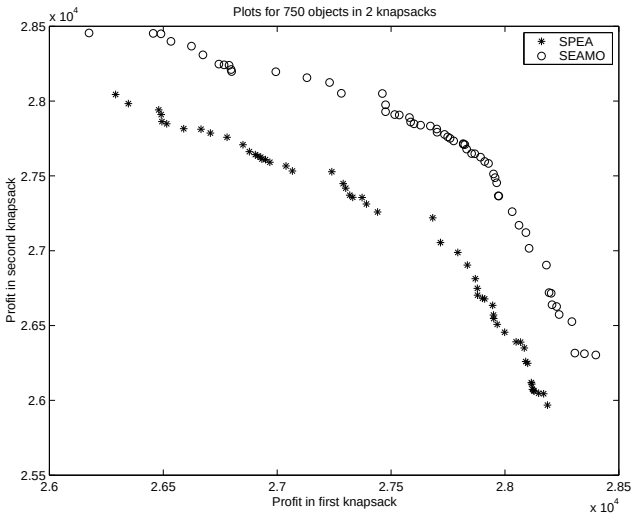

Fig. 3. Non-dominated solutions from 30 replicate runs of SEAMO and SPEA for the 2 knapsack problem with 750 objects

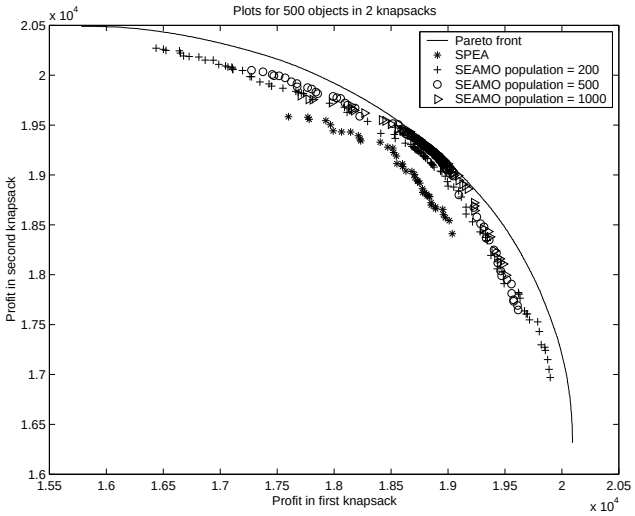

Fig. 4. SEAMO results for the 2 knapsack problem with 500 objects using various population sizes

Clearly, the results for SEAMO are better using single, longer runs rather than the 30 replicate runs. The new results are closer to the Pareto front and they are also more diverse. Results for long runs with small populations tend to be more diverse than results for short runs with large populations. However, the results where larger populations are used seem to be closer to the Pareto front than the results for the smaller populations. The SPEA results are included in the two diagrams for comparison

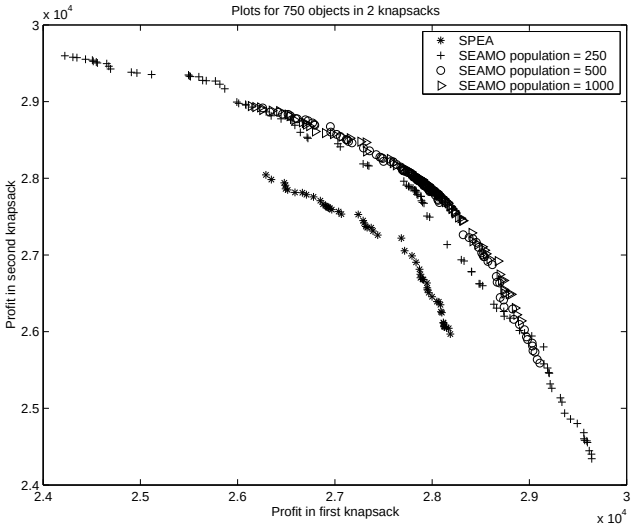

Fig. 5. SEAMO results for the 2 knapsack problem with 750 objects using various population sizes

as before.

B. *Further Experimental Results*

For the final set of experiments SEAMO is run on all four test problems, using an (arbitrarily) large population of 10,000. In each case the EA is halted after 40 generations have elapsed with no replacements being made in the population. The purpose of these final runs is to show what SEAMO is capable of. Once more the SPEA results are included in all the plots to aid visualization, although in this case it should be remembered that SEAMO used massive numbers of evaluations in order to obtain the results in these final plots. The run times for these problems vary between half an hour for the 100 object problems and about 26 hours for the 750 object problem (about 50,000 generations) on a Pentium III processor with 128 Mb RAM. The results are presented in Figure 6.

Clearly, increasing the number of evaluations for SEAMO improves the results for the 500 and 750 object problems. In particular the results for these problems lie on a smoother curve. SEAMO results for the 100 and 250 object problem are not so impressive, and barely improve on the SPEA results obtained using far fewer evaluations.

V. CONCLUSIONS

A new evolutionary algorithm, SEAMO, has been presented for multi-objective optimization. It is much simpler than other approaches and has produced some promising results for the multiple knapsack problem, outperforming many state-of-the-art Pareto-based EAs compared in [13]. The algorithm uses a steady-state elitist strategy for replacement and a simple uniform scheme for selection. Throughout the genetic search, progress depends entirely on the replacement policy, and no fitness calculations, rankings, sub-populations, niches or auxiliary populations are required.

Establishing the new approach as a general technique for multi-objective optimization, however, requires its extension beyond knapsack problems to other domains. Work in progress includes the application of SEAMO to the function optimization problems described in [14]. To solve these problems the author has replaced the permutation representation, operators and decoder used for the multiple knapsack problem with a string of real variables, and appropriate recombination and mutation operators. Further plans include comparing SEAMO with the latest Pareto-based EAs such as SPEA2 [14], PESA [3] and NSGA-II [4] for multiple knapsack and function optimization problems. Initial studies suggest SEAMO may need to undertake more objective function evaluations, in order to equal or surpass the very best results obtained elsewhere. Direct comparisons of computational effort are difficult, however, given the relative simplicity of SEAMO and the more elaborate calculations undertaken by competing EAs. Given that the routines in SEAMO are computationally cheap, the approach is surely worthy of serious consideration and further investigation.

References

[1] D. J. Cavicchio, Adaptive Search Using Simulated Evolution, Ph.D. dissertation, University of Michigan, Ann Arbor, 1970.

[2] N. Chakraborti, R. Kumar, and D. Jain, A study of the continuous casting mold using a pareto-converging genetic algorithm. *Applied Mathematical Modelling*, vol. 25. pp 287–297, Elsevier, 2000.

[3] D.W. Corne, J. D. Knowles, and M.J. Oates, The Pareto envelope-based selection algorithm for multiobjective optimization. *Parallel Problem Solving from Nature – PPSN VI*, Lecture Notes in Computer Science 1917, pp. 839–848, Springer, 2000.

[4] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, 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, pp. 849–858, Springer, 2000.

[5] C. M. Fonseca and P. J. Fleming, Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416–423, Morgan Kaufmann, 1993.

[6] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[7] P. Hajela and C.-Y. Lin, Genetic search strategies in multicriterion optimal design, *Structural Optimization*, Volume 4, pp. 99–107, New York: Springer, 1992.

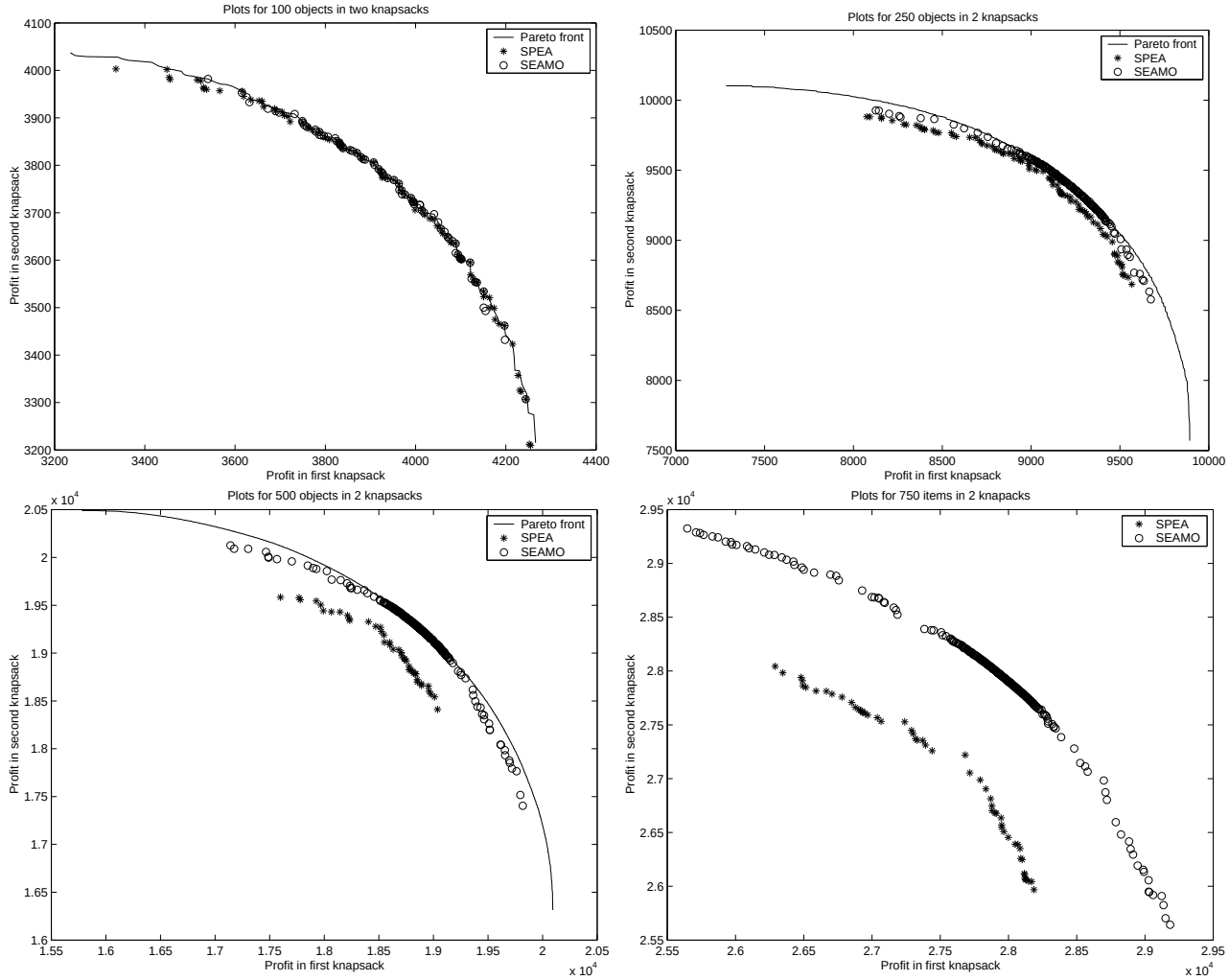

Fig. 6. SEAMO results for the four 2 knapsack problems using populations of 10,000

[8] J. Horn, N. Nafpliotis, and D. E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation*, Volume 1, pp. 82–87, IEEE Press, 1994.

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolutionary Programs*, Third, revised and extended edition, Springer, 1996.

[10] I. M. Oliver, D. J. Smith and J.R.C. Holland, 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*, pp. 224–230, 1987.

[11] J. D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, pp. 93–100, 1985.

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

[13] E. Zitzler and L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Transactions on Evolutionary Computation*, 3(4), pp. 257–271, 1999.

[14] E. Zitzler M. Laumanns and L. Thiele, 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.