A Hierarchical Solve-and-Merge Framework for Multi-Objective Optimization

Christine L. Mumford
School of Computer Science
5 The Parade
Cardiff, CF24 3AA
C.L.Mumford@cs.cardiff.ac.uk

Abstract- This paper presents *hierarchical solve-and-merge (HISAM)*: a two-stage approach to evolutionary multi-objective optimization. The first stage involves a simple genetic algorithm working on a number of isolated subpopulations, each using its own uniquely weighted linear scalarizing function to encourage it to focus on a different region of the Pareto space. At the second stage, the best solutions from stage one are passed to a Pareto-based hierarchy, where the solution set is judged on Pareto dominance and further improved. Preliminary results for large knapsack problems with 2 - 4 objectives are highly competitive with those obtained using other methods. Furthermore, the HISAM implementation has a fast execution time.

1 Introduction

The application of evolutionary algorithms (EAs) to problems with multiple objectives is currently a very active area of research, and many successful algorithms have been developed. Most of these approaches use a global fitness function to bias the choice of parents for breeding, emulating approaches popular in the single objective case, such as tournament selection or the roulette wheel. Unlike their single objective counterparts, however, multi-objective EAs have to take account of several criteria simultaneously, and this makes the design of a suitable scalar fitness function much more of a challenge. Most researchers base their fitness functions for multi-objective EAs on dominance conditions that exist between individuals within the population, for example NSGA-II (non-dominated sorting genetic algorithm II) [2], SPEA (strength Pareto evolutionary algorithm) [15] and SPEA2 (improved SPEA) [14]. Additionally, fitness functions for some EAs (notably NSGA-II and SPEA2) also incorporate a density component, encouraging the solution set to be more evenly spread in Pareto objective space, by inhibiting selection from the denser regions and favoring it from the sparser regions. Indeed, another popular multi-objective EA, the PESA algorithm (Pareto envelope-based selection algorithm) [3], relies solely on density information in the computation of its fitness function. In PESA inferior members of the population are quickly deleted leaving only non-dominated individuals to become the parents of the next generation.

Fitness functions for multi-objective EAs based on dominance or density are examples of *Pareto-based selection techniques* [4]. Alternatively, it is possible to calculate fitness functions using a simple weighted combination of the multiple objective values. This approach, taken by Jaszkiewicz [5, 6] in his MOGLS algorithm (multiple objective local genetic search), has produced excellent results for the multi-objective knapsack problem (MKP). In MOGLS, a random set of weights is generated at the start of each iteration, and this ensures a good coverage of the Pareto space.

The SEAMO algorithm (Simple Evolutionary Algorithms for Multi-objective Optimization) [11, 13] and its successor, SEAMO2 [10], have a different approach to reproduction, and do not need to maintain global information on fitness, dominance or density relationships within the population. In SEAMO and SEAMO2, parents are simply chosen from a uniform distribution, and progression of the genetic search relies solely on decisions taken locally, after an offspring has been produced. A strong offspring will replace an inferior parent (or occasionally another individual) and a weak offspring will be allowed to die without ever entering the population.

The present study extends the work of Mumford [9], in which a hierarchical framework of SEAMO2 algorithms, working on separate subpopulations, produced convincing improvements over a comparable implementation with a single SEAMO2 implementation. However, in the earlier work no attempt was made to focus the subpopulations and each could freely operate over the entire Pareto space. This would seem to be inefficient. The new approach addresses this weakness by adding an extra stage (stage one) to the bottom of the hierarchy (see Figure 1). The function of stage one is to focus different subpopulations onto separate regions of the Pareto space. Stage one is implemented using multiple copies of a simple genetic algorithm, each copy running on its own independent subpopulation and optimizing a differently weighted "MOGLS style" linear scalarizing function. At the second stage of the new two-stage approach, the best solutions from stage one are passed to the hierarchy of SEAMO2 implementations for further improvement. Here the solution sets are judged on Pareto dominance, rather than on the linear scalarizing function used in stage one.

The new approach easily outperforms both MOGLS and SEAMO2, for solution quality and run time, on a suite of large multi-objective knapsack problems (MKP) with 2 - 4 objectives. Recent studies have established MOGLS as a front-runner in terms of solutions quality on the MKP [1, 6]. The new HISAM algorithm not only produces better solutions than MOGLS, but does so much faster.

A final motivation for the present work is to correct a known weakness present in the SEAMO and SEAMO2 algorithms: there is no specific mechanism to ensure an even spread of solutions across the Pareto front. Solution sets produced by multi-objective EAs are usually judged on the basis of three criteria: solutions should be (1) of good quality, (2) widely spread and (3) evenly spread. Although the SEAMO algorithms include specific mechanisms to im-

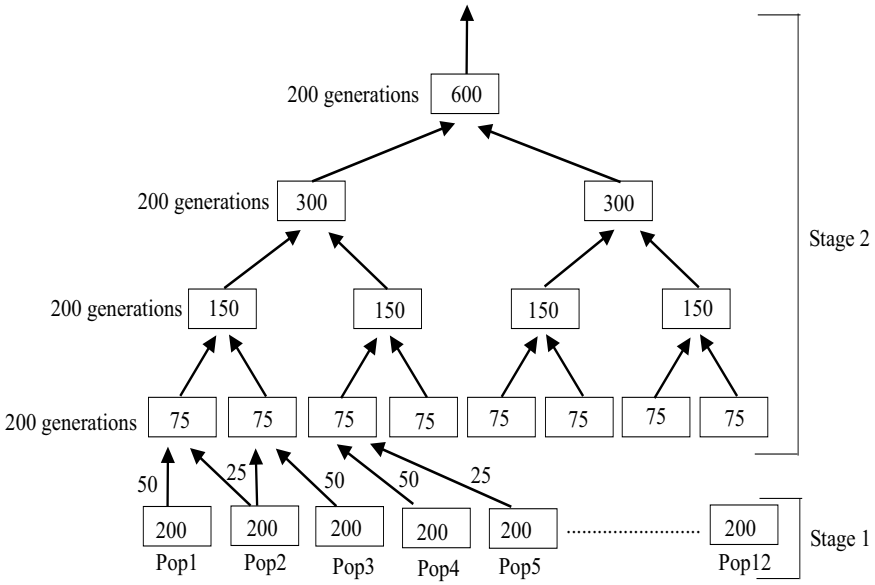

Figure 1: The two-stage framework of HISAM

prove the quality and range of the solutions set, they lack control over the evenness of spread.

2 Test Data Used

The test problems used in this paper are MKPs taken from [15]. Instances with 500 and 750 items are used, with 2, 3 and 4 objectives, denoted as kn$n.m$, where n is the number of items and m the number of knapsacks (objectives).

The 0-1 multi-objective knapsack problem 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 $S \subseteq O$ 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 nor its profit is fixed, and will probably have different values in each knapsack.

3 The Two-Stage Framework for HISAM

In the first stage of the two-stage process, the Pareto space is discretized into a predetermined number of regions using linear weighted scalarizing functions. Simple single objective GAs are then run on independent subpopulations covering each region. Following execution of the single objective stage, the best individuals from each subpopulation are collected and passed to a hierarchical Pareto-based algorithm known as the *hierachical-balanced algorithm*, *HBA*, described in [9]. Figure 1 gives an example implementation of the two-stage framework. Here we start stage one with twelve independent subpopulations each consisting of 200 individuals, and each using a differently weighted linear scalarizing function. After the twelve single objective GAs have executed for 200 generations, each subpopulation will contribute its 50 best individuals to the $12 \times 50 = 600$ (total) population at the level above. The hierarchical-balanced algorithm then runs for a total 800 generations, executing 200 generations at each of the four levels. There are eight subpopulations, each consisting of 75 individuals, at the base level of stage two. These merge in pairs to give four subpopulations of 150 individuals at the next level up. Subsequently, these also merge in pairs to give two subpopulations each of 300, and finally a single population of 600 is produced at the root level.

3.1 Discretizing the Space at level one

Weighted linear scalarizing functions provide the simple GA, at stage one, with its single objective values. A different combination of weights is supplied to each of subpopulation to spread the focus into different regions of the Pareto space. Weighted linear scalarizing functions are defined in Equation 1.

$$S(f(\mathbf{x})) = \sum_{j=1}^{j=m} \lambda_j f_j(\mathbf{x}) \tag{1}$$

where λ_j is the weight applied to the value of j^{th} objective, $f_j(\mathbf{x})$. The weight vectors are initially set as fixed sum integer combinations, such that $\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_m = k, 1 < k < \infty$. For example, with a two objective case, it is possible to split the Pareto space into 6 separate regions using the fixed integer sum $k = 5$. The resulting weight vectors are: $(0, 5), (1, 4), (2, 3), (3, 2), (4, 1)$ and $(5, 0)$. For a given value of k, it is relatively simple to generate all possible integer weight combinations for a general problem with m objectives. In this way the number of regions assigned is easily varied. The weights in the weight vectors are finally scaled with respect to the average values of the individual

objectives. (Although this is probably not strictly necessary for the problem instances considered here, as range of profits in the various knapsacks are very similar.)

Figure 2 illustrates the discontinuities frequently observed following the execution of stage one of HISAM. It also show the effect of applying a simplified stage two in place of the hierarchical balanced algorithm. The instance Kn500.2 is used for this demonstration, and the Pareto space is discretized into six regions at stage one. We use subpopulation sizes of 200 for this experiment, and run the simple GA for 500 generations on each of the six subpopulations.

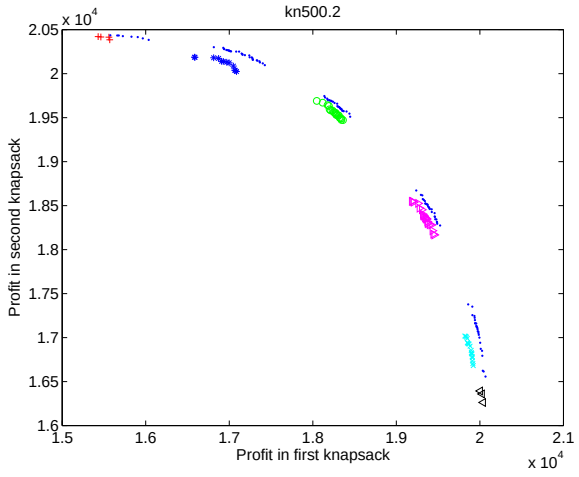

Figure 2: Single run of stage 1, followed by a simplified stage 2

The lower broken trace in Figure 2 represent the non-dominated outputs produced by the six subpopulations at the end of stage one. The upper broken trace in Figure 2 represents the output following the application of the simplified second stage. Stage two simply runs a single copy of SEAMO2 on the entire accumulated population of $6 \times 200 = 1200$ for 500 generations. The discontinuities evident in the lower trace, are still apparent in the upper trace. Clearly, the simplified second stage has failed to smooth them out. Either we need to use many more subpopulations at stage one, or we apply a more sophisticated approach at stage two. We shall see later that the hierarchical-balanced algorithm provides a highly effective smoothing mechanism at stage two.

3.2 The Representation and Genetic Framework

The single objective and multi-objective GAs, that form the basis of the HISAM framework, both operate on a simple a steady-state principle. The algorithms sequentially step through the entire population (or subpopulation), selecting every individual in its turn to serve as a first parent, and pairing 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 (usually one of its parents), or it will die, depending on the outcome of the replacement strategy used.

Procedure *SIMPLE GA*
Begin
Generate N random individuals {N is the population size}
Evaluate the fitness function for each population member and store it
Repeat
For each member of the population
This individual becomes the first parent
Select a second parent at random
Apply crossover to produce single offspring
Apply a single mutation to the offspring
Evaluate the objective function produced by the offspring
if the offspring is better than weaker parent
(and not a duplicate)
Then the offspring replaces it in the population
else it dies
Endfor
Until stopping condition satisfied
End

Figure 3: Algorithm 1 The simple single-objective GA

The representation chosen for HISAM on multiple knapsack problems is order-based and uses a first fit decoder to convert a permutation of objects into a packing for the knapsacks [8]. Briefly, a first-fit decoder packs objects taken from the permutation list in sequence, until an object is reached that would violate a constraint if it was packed. This object will be skipped over and the next one tried. For example, assume we are given an order-based sequence of (1, 5, 2, 4, 3, 6) for a six object problem. A first-fit decoder will attempt to pack object 1, then object 5, then object 2 etc., until it reaches an object that, if packed, would exceed the capacity constraint of at least one of the knapsack. If this occurs, for example, when object 4 is tried, the first-fit decoder would skip over object 4 and try object 3, then 6.

Cycle Crossover (CX) [12] is used as the recombination operator for HISAM, and a simple mutation operator swaps two arbitrarily selected objects within a single permutation list. The order-based representation scheme was chosen for HISAM because SEAMO2, upon which HISAM is based, produced better results using an order-based approach than it did when other methods were tried in a recent comparative study, [1].

The simple single objective GA, outlined in Figure 3, operates independently on each subpopulation at stage one. Using a single scalar value, consisting of the weighted sum of the multiple objectives (see Section 3.1), it is possible to concentrate each subpopulation onto a different region of the Pareto space. Parents are chosen by uniform selection, as described above, and progress of the genetic search is dependent on the replacement strategy, in which an offspring will replace its weaker parent if and only if it has a better value for its weighted linear scalarizing function.

The SEAMO2 algorithm, outlined in Figure 4, is used as a basis for the hierarchical-balanced algorithm at the second stage of the optimization framework. Here copies of SEAMO2 operate independently on each subpopulation in the hierarchy. SEAMO2 works in a very similar manner to the simple GA, and the parents are selected in exactly the same manner. The replacement strategy in SEAMO2, how-

Table 1: Population parameters for experimental runs

Problem	Algorithm						
	HISAM					SEAMO2	
	stage 1			stage 2			
	# subpops	subpopsize	generations	popsize	generations	popsize	generations
kn500.2	12	200	200	600	800	960	1,000
kn750.2	12	250	250	600	1000	1080	1,250
kn750.3	15	250	300	750	1200	1350	1,500
kn750.4	35	250	400	1,750	1600	3150	2,000

```
Procedure SEAMO2
Parameters: N – population size, stopping condition
Begin
Generation of the initial set of solutions:
    Generate N random individuals
    Evaluate the objective vector for each individual
            and store it
    Record the bestSoFar for each objective function
Main loop
    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 offspring
            Evaluate the objective vector
                produced by the offspring
            If offspring objective's vector
                improves on any bestSoFar
                Then it replaces a parent
            Else if offspring is a duplicate
                Then it dies
            Else if offspring dominates either parent
                Then it replaces it
            Else if offspring is neither dominated by
                 nor dominates either parent
                Then it replaces another individual
                 that it dominates at random
            Otherwise it dies
        End for
    Until stopping condition is satisfied
    Print all non-dominated solutions in the final population
End
```

Figure 4: Algorithm 2 A basic framework for SEAMO2

ever, is based on Pareto dominance, and it is not as straightforward to determine which individuals are better than their parents when we are dealing with more than one objective. It can be seen by examining the pseudocode in Figure 4 that new offspring are evaluated according to the following criteria in SEAMO2:

1. Does offspring dominate either parent?
2. Does offspring produce a global improvement on any Pareto components?
3. Does offspring have a mutually non-dominating relationship with both its parents?
4. Is offspring a duplicate?

When offspring are evaluated in relation to their parents, the first parent is always considered before the second, with the offspring replacing the first parent if it passes the necessary "test". It is only if the offspring is not considered good enough to replace the first parent that it is evaluated in relation to the second parent.

When there is a mutually non-dominating relationship between parents and offspring, the new offspring may replace another population member. In this situation, a random search will be initiated in an attempt to locate an individual that is dominated by the new offspring. This process will sequentially test and then remove individuals from consideration until a suitable individual is found, or until the population is exhausted without discovering such an individual, whichever occurs first. The new offspring will die if it fails to dominate any member of the population.

At all levels in the hierarchy, a strict 'deletion of duplicates' policy is implemented, preventing offspring identified as duplicates from entering the population. This is to to help promote genetic diversity and prevent premature convergence. For the simple GA as well as for SEAMO2, pairwise comparisons identify new offspring that have identical counterparts already present in the current population, and these offspring are allowed to die. For speed and simplicity it is the values of the multiple objectives that are used for comparison (even at the single objective stage), rather than the genotypic values (i.e. the permutation lists of objects). Experimental evidence presented in [10] demonstrates that the pairwise comparisons required to implement the deletion of duplicates is efficient and does not increase the run time for large multiple knapsack problems. The time required for the pairwise comparisons is compensated by reductions in number of long permutation lists copied into the population.

4 Experimental Design

The new hierarchical solve-and-merge algorithm is evaluated using two benchmarks: 1) SEAMO2 and 2) MOGLS. SEAMO2 is an obvious choice, given that stage two of the new approach is based on this algorithm. MOGLS is chosen as the second benchmark because it has produced superior results to other state-of-the-art algorithms on multiobjective knapsack problems [1, 6], and is hard to beat. Comparisons are made on the basis of each algorithm performing equal numbers of objective function evaluations. 30 replicated runs are collected for each set of experiments. All three algorithms were implemented by the present author.

Population and subpopulation sizes for HISAM and SEAMO2 are given in Table 1, together with the numbers of generations allocated to the runs. Parameters for MOGLS

will be dealt with later, because it does not operate on the basis of a fixed population in quite the same way as the other approaches.

The entries in Table 1 will now be explained in a little more detail. It is probably helpful to refer back to Figure 1 which shows the precise architecture for HISAM applied to kn500.2. The table specifies the number of subpopulations, the subpopulation sizes, and the number of generations used at stage one of HISAM, for all the problem instances. Only total population sizes and generations are specified for stage two, on the other hand. For the purposes of this study, the basic architecture shown in Figure 1, with four levels at stage two, is used on all four problems. Thus, copies of the SEAMO2 algorithm operate on eight subpopulations at the lowest level of stage two, then these undergo pairwise merging to produce four subpopulations at the next level up, then two at the level above that, and finally a single panmictic population is produced at the top of the tree. The eight subpopulations at the lowest level of stage 2 will each contain 75 individuals for kn500.2 and kn750.2, 93 or 94 for kn750.3 and 218 or 219 for kn750.4. Total populations at stage two of 600, 750 or 1,750 are derived by accumulating the best 50 individuals from each of the subpopulations from stage one, i.e. $12 \times 50 = 600$, $15 \times 50 = 750$ and $35 \times 50 = 1,750$. Selection of these 50 individuals from each subpopulation at stage one is carried out in two phases. In the first phase all the non-dominated solutions are isolated from the subpopulation, and in the second phase the non-dominated solution set is expanded or reduced, as necessary, by randomly adding or removing solutions, so that each subpopulation contributes exactly 50 individuals.

The stand-alone SEAMO2 algorithm, used as a benchmark, is allowed to run for exactly the same number of generations in total as the HISAM algorithm. Furthermore, the population size for SEAMO2 is adjusted to ensure the algorithm performs exactly the same number of objective function evaluations as HISAM on each problem. For example, SEAMO2 uses a population size of 960 for kn500.2, because this is the average size of the HISAM population taken over the two stages,($960 = \frac{(12\times200\times200+600\times800)}{1000}$).

Unlike SEAMO2 and most other multi-objective EAs, MOGLS does not operate on a fixed size population. Instead it maintains a long list of current solutions, CS, and stores an archive, PP of potentially non-dominated solutions. Following guidelines given in [6], maximum sizes of CS are set at 4,000, 5,000, 6,000 and 7,000 for kn500.2, kn750.2, kn750.3 and kn750.4 respectively.

While SEAMO2 and HISAM are implemented with order-based representations, as described previously, MOGLS uses a bit string representation with repair, exactly as described in [6]. MOGLS relies on the repair mechanism to drive the local search phase of the algorithm. Furthermore, the order-based approach produced very poor results for MOGLS in a recent study [1]. To give each algorithm a fair change in the contest, each is allowed to use the representation and operators that seem to suit it best.

5 Performance Measures

Assessing the relative quality of non-dominated set of points produced by different multi-objective algorithms is a challenge. Simple two dimensional plots of the approximate Pareto sets can provide a useful visualization when problems are limited to two objectives. However, this approach lacks rigor and, in any case, it is of no use for problems with many objectives. Of the multitude of metrics from the literature which attempt to express the solution quality as a single number, the $\mathcal{S}$ metric of Zitzler and Thiele[15] has been recommended in [7] as a good all-round choice, provided the problem involves relatively few dimensions, and the non-dominated sets are not overlarge. For this reason we have adopted $\mathcal{S}$ as the main metric for the the present study. In addition, we use simple 2D plots, where appropriate, to give a visual interpretation of the results and we also compare the run times of the algorithms.

5.1 The $\mathcal{S}$ Metric

The $\mathcal{S}$ metric is a measure of the size of the dominated space, measured in the number of dimensions determined by the number of objectives. Let $X' = (x_1, x_2, \ldots, x_l) \subseteq X$ be a set of l solution vectors. The function $\mathcal{S}(X')$ gives the volume enclosed by the union of the polytypes $p_1, p_2, \ldots, p_l$, where each p_i is formed by the intersection of the hyperplanes arising out of x_i, along with the axes. For each axis in the objective space, there exists a hyperplane perpendicular to that axis and passing through the point $(f_1(x_i), f_2(x_i), \ldots, f_k(x_i))$. In the two dimensional case, each p_i represents a rectangle defined by the points $(0, 0)$ and $(f_1(x_i), f_2(x_i))$. In this study, the size of the dominated space is quoted as a percentage of the reference volume between the origin and an upper approximation of the ideal point taken from [6].

6 Results

Figure 5 compares single runs of HISAM with SEAMO2 and MOGLS on kn500.2 and kn750.2. Clearly the new approach produces superior results to MOGLS, as the curve for HISAM lies above the curve for MOGLS. Unfortunately, the results for SEAMO2 are a little difficult to distinguish on the graphs, particularly for kn500.2. On careful examination, however, the SEAMO2 results can just be seen bunched up in a small area on the graph for kn750.2, forming the top curve. This indicates that the quality is good for SEAMO2, but the spread is poor.

The boxplots in Figure 6 compare the amount of dominated space, $\mathcal{S}$ [15] produced for the 30 runs of the three algorithms on kn500.2, kn750.2, kn750.3 and kn750.4. These plots provide very strong evidence in favour of the HISAM.

Finally, Table 2 shows very clearly that HISAM has a very much faster run time than either SEAMO2 or MOGLS. MOGLS is particularly slow because of its frequent need to re-evaluate all the members of CS, the current list of solutions, with respect to their weighted linear scalarizing functions. This occurs every time a new random weight

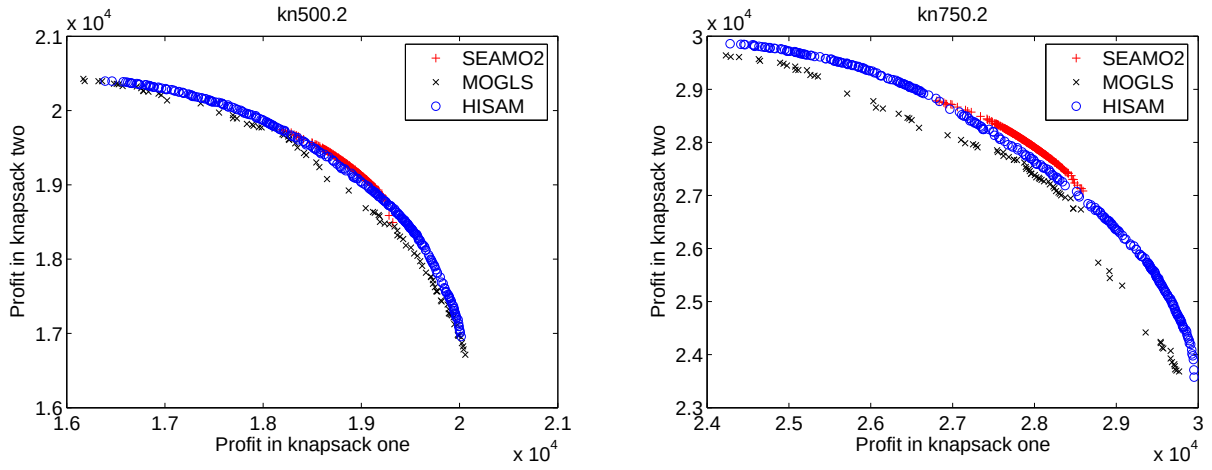

Figure 5: Example runs, comparing HISAM with MOGLS and SEAMO2 on kn500.2 and kn750.2

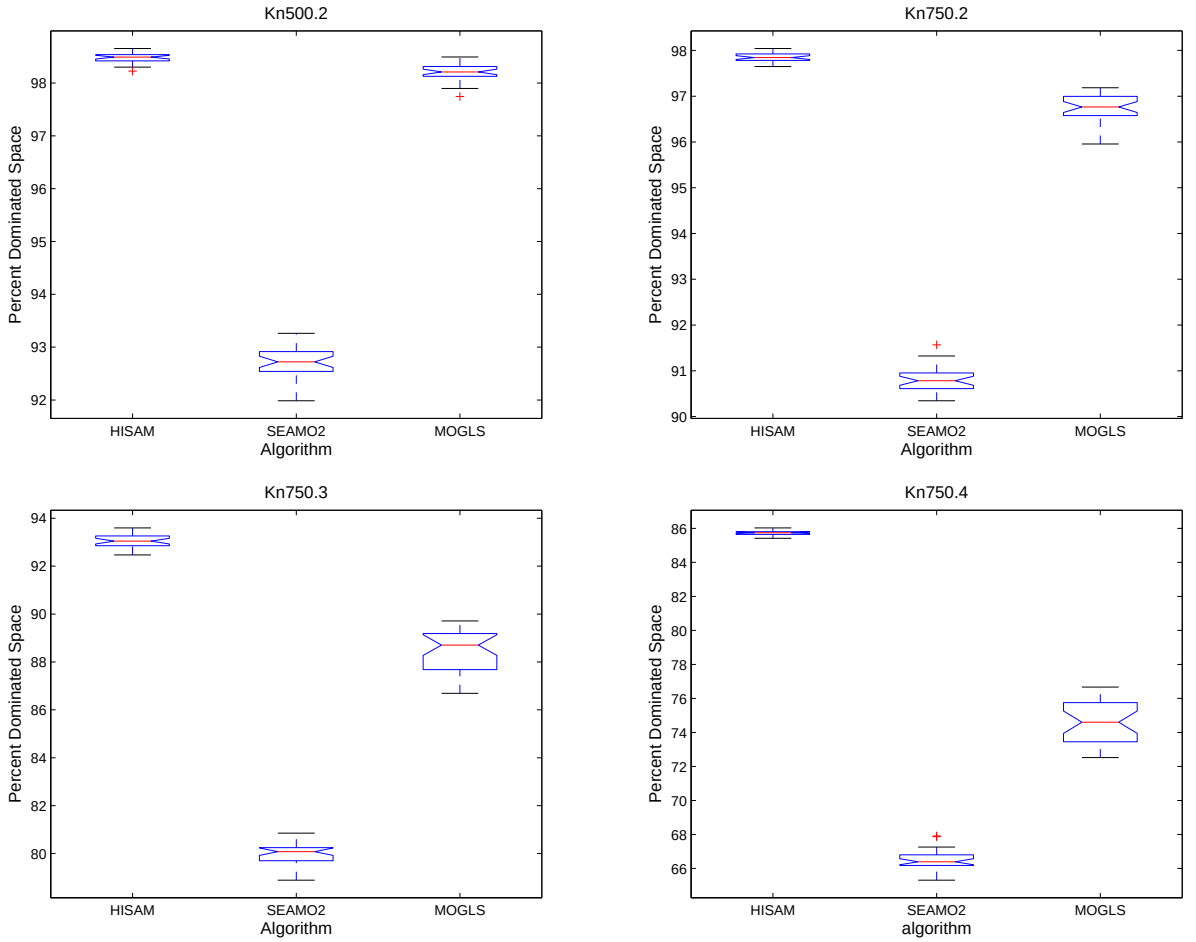

Figure 6: Box plots showing amount of dominated space

Table 2: Comparing run times of the multi-objective algorithms (in seconds)

Algorithm	kn500.2	kn750.2	kn750.3	kn750.4
HISAM	59	105	207	1,097
SEAMO2	139	254	572	4,208
MOGLS	2,918	8,955	14,085	45,099

vector is generated. Also note that HISAM runs faster than SEAMO2, the algorithm upon which HISAM is based. This is probably due to the reduced time required for checking the population for duplicates (run time = $O(N^2)$, for each iteration of the population, where N is the population, or subpopulation size).

7 Conclusion

A new fast and effective two-stage, evolutionary multi-objective framework called *hierarchical solve-and-merge (HISAM)* has been presented. At stage one (level 1) a simple single objective genetic algorithm operates concurrently on a discretized Pareto space, using linear weighted scalarizing functions on independent subpopulations. Stage two (level 2 and above) is made up of hierarchy consisting of multiple copies of the SEAMO2 algorithm. Each level in the stage two hierarchy, $l > 1$, is seeded from the previous level, $l-1$, by the merging of pairs of subpopulations, until finally, a single panmictic population is produced at the top of the hierarchy. The single objective stage at the very bottom level seeds the Pareto-based hierarchy with groups of high quality solutions, each group focussed towards a different region of Pareto space. The role of the Pareto-based hierarchy is to simultaneously improve the results while smoothing out any discontinuities.

The new approach is tested on large multi-objective knapsack problems (MKP) with 2 - 4 objectives. Experimental results, conducted on on an evaluation for evaluation basis, demonstrate that a much wider, and better spread set of solutions can be obtained using a hierarchical approach than is possible using an equivalent Pareto-based algorithm on a single panmictic population. In addition, it is clear that the HISAM algorithm runs much faster than its standalone panmictic SEAMO2 counterpart. HISAM also compares favorably with MOGLS (multi-objective genetic local search), in terms of solution quality and run time. MOGLS was selected as a benchmark because it outperformed all its competitors (including SPEA2) on multi-objective knapsack instances in some recent studies [1, 6]. The encouraging results for HISAM presented here suggest that it could be a useful addition to the armory of best-performing multi-objective evolutionary algorithms. Further work is needed, however, to extend its application beyond multi-objective knapsack instances and establish its potential on a wider range of problems. Future plans include a coarse-grained parallel implementation of HISAM.

Bibliography

[1] G. Colombo and C. L. Mumford. comparing algorithms, representations and operators for the multi-objective knapsack problem. CEC 2005 (to appear).

[2] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature – PPSN VI*, number 1917 in LNCS, pages 849–858, Berlin, 2000. Springer.

[3] J. K. D.W. Corne and M. Oates. The pareto envelope-based selection algorithm for multiobjective optimisation. In *Parallel Problem Solving from Nature – PPSN VI*, number 1917 in LNCS, pages 839–848, Berlin, 2000. Springer.

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

[5] A. Jaskiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 137:50–71, 2002.

[6] A. Jaskiewicz. On the performance of multiple objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, 2002.

[7] J. D. Knowles and D. W. Corne. On metrics for comparing non-dominated sets. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC2002)*, pages 711–716, Honolulu, Hawaii, 2002.

[8] C. L. Mumford. Comparing representations and recombination operators for the multi-objective 0/1 knapsack problem. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC2003)*, pages 854–861, Canberra, Australia, 2003.

[9] C. L. Mumford. A hierarchical approach to multi-objective optimization. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC2004)*, pages 1944–1951, Portland, Oregon, 2004.

[10] C. L. Mumford. Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In *Proceedings of the 2004 Genetic an Evolutionary Computation Conference (GECCO)*, pages 1389–1400, Seattle, Washington, USA, 2004.

[11] C. L. Mumford-Valenzuela. A simple approach to evolutionary multi-objective optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications*, chapter 4, pages 55–79. Springer Verlag, London, 2005.

[12] I. M. Oliver, D. J. Smith, and J. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Genetic Algorithms and their Applications:Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987.

[13] C. L. Valenzuela. A simple evolutionary algorithm for multi-objective optimization (seamo). In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC2002)*, pages 717–722, Honolulu, Hawaii, 2002. (C.L. Valenzuela is now known as C.L. Mumford).

[14] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

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