

Benchmarking a New Memetic Algorithm Framework for Pareto Multiobjective Optimization

Joshua D. Knowles and David W. Corne
School of Computer Science, Cybernetics and Electronic Engineering
University of Reading, Reading RG6 6AY, UK
j.d.knowles@reading.ac.uk, d.w.corne@reading.ac.uk
<http://www.rdg.ac.uk/~ssr97jdk>

Abstract.

Memetic algorithms (MAs) are, at present, amongst the most successful approximate methods for combinatorial optimization. In this paper, a memetic algorithm framework for problems involving two or more distinct objectives is presented. The new framework specifies the use of Pareto selection in both the local-search and population-based phases, in contrast to current MAs for multiobjective optimization, which use objective aggregation. A simple memetic algorithm based on the framework, called M-PAES, is also described. Verification of M-PAES is carried out by testing it on a suite of 0/1 knapsack problems, involving two, three and four objectives. On each problem instance, comparison is made between M-PAES, the strength Pareto evolutionary algorithm (SPEA) of Zitzler and Thiele, and another memetic algorithm, the random directions multiple objective genetic local searcher (RD-MOGLS) of Jaskiewicz. The results indicate that the new memetic algorithm framework has significant potential; M-PAES performs well compared to both SPEA and RD-MOGLS on the knapsack problems.

Keywords: memetic; genetic local-search; evolutionary; multiobjective optimization; Pareto

Received xxx

Revised xxx

Accepted xxx

1. Introduction

In many problem domains, including operations research, machine learning, task planning and design or engineering, it is common practice to formulate problems so that a single, scalar measure, or objective function, must be maximized. In reality, however, many problems are not adequately represented in this way, and instead there are really multiple (often conflicting) goals. In *multiobjective* optimization [5, 7] the different attributes to be optimized are viewed as distinct from one another, so that the quality of solutions is described by a vector of objective values. Thus, multiobjective optimization *problems* (MOPs) represent a more general class than single-objective problems, and one which applies to a broad range of applications.

In most multiobjective problems it is not possible to find solutions that are simultaneously optimal on all attributes, and one must accept a solution that represents an acceptable trade-off of the different goals. In fact, if we could *a priori* quantify the relative importance of each of the goals then we could weight each of them accordingly and aggregate them to give a single, scalar objective function. This would allow traditional optimization methods to be applied, and a solution or solutions that optimize the scalarized objective function could be found. But another (and increasingly popular) possibility is to perform Pareto optimization [9], in which the different objectives are maintained as separate and incommensurable measures of quality. Pareto optimization implies not one but a *set* of optimal points in objective space, the Pareto front, each representing a different trade-off of the objectives. In possession of the Pareto front, solutions can be selected in light of the knowledge of the different trade-offs possible, leading to a more informed choice.

Of course, many optimization problems cannot be solved exactly, and instead an approximate method — one that does not guarantee to find all optimal solutions — must be used. One such method is evolutionary computing — now a very well-established and popular technique. The basic idea is to evolve a population of potential solutions to a problem by repeated phases of evaluation, selection and variation. The range of applications of this general approach is vast, and in recent years, evolutionary algorithms (EAs) have been applied more and more to multiobjective problems. In particular, the possession of a population, seems to make EAs very attractive for use in multiobjective problems, where a number of solutions approximating the Pareto set are often required.

As a whole, some of the most influential algorithms [8, 15, 20, 30, 32, 34, 44] put forward in evolutionary multiobjective optimization (EMOO) [3] indicate the progression in selection schemes, diversity-preserving mechanisms, and the proper use of elitism that has occurred in the field since it began in the eighties. But evolutionary methods are not the only approximate methods that have been applied to multiobjective problem solving.

In recent years, in parallel to the development of multiobjective EAs (MOEAs), there has been a growing research effort in the use of metaheuristics within the field of multiple criteria decision making (MCDM) — a branch of operations research. Algorithms based on both tabu search and simulated annealing have been put forward [4, 13, 14, 17, 33, 37]. Most of these algorithms do not have a population but store the best solutions discovered during a local search process. Rather than using Pareto ranking, weighted metrics are used to aggregate the objectives into a single score to be used in the acceptance function [39].

Whether the algorithms devised and investigated in the MCDM field are

more or less effective than MOEAs remains an open question: Very few studies that attempt to directly measure and compare the performance of MOEAs with algorithms based on tabu search [13, 14, 17] or simulated annealing [4, 33, 37], on problems of sufficient variety, have been carried out. But, despite the lack of communication between the two fields, there does seem to be some convergence of the approaches taken by them. For example, our own work on (1+1)-PAES [23, 25, 26] shows that an algorithm that employs only local search moves may be competitive with many of the most respected MOEAs, on a range of problems. PAES goes some way to bridging the gap between local search and population based methods, and is unique amongst local search algorithms in its direct use of the partial ordering of solutions in its selection mechanism. Similarly, work by Czyżak and Jaskiewicz [4] also takes inspiration from both multiobjective camps, with a novel population-based approach to multiobjective simulated annealing. Their algorithm uses utility functions to obtain a single score from the vector of objective values, but exploits population information to adjust the direction of the utility function to direct progress in a direction approximately perpendicular to the current Pareto front. The technique inherently encourages an even spread of solutions, as well.

Memetic algorithms (MAs) (also called genetic local search algorithms or hybrid genetic algorithms) provide a further framework in which to combine and improve on the population-based MOEAs and the local-search algorithms in the multiobjective domain. Proposals for multiobjective MAs have already been put forward. The first of these, was devised by Ishibuchi and Murata [21], and the basic idea was extended by Jaskiewicz [22], who introduced a variant based on simulated annealing, and another variant in which mating was restricted. These approaches all use randomly selected weight vectors to aggregate the objectives, and combine this with a standard selection function, rather than performing selection based directly on the partial ordering of solutions induced by the so-called Pareto methods. The use of local-search heuristics within a population-based approach has also been applied to the optimization of spacecraft trajectories for Mars missions [18].

In this paper, a general framework for a multiobjective MA, is put forward. Its selection mechanism is based directly on the partial ordering of solutions, in contrast to the other multiobjective MAs put forward to date. Then, a simple MA based on the framework, using a modified version of (1+1)-PAES as the local improvement heuristic, is presented. Other possible instantiations of the general framework are also discussed. The simple MA, M-PAES, is then tested using a suite of 0/1 knapsack problems possessing 2, 3 and 4 objectives. A direct comparison is then made between the performance of the new memetic algorithm proposed, and three existing approaches: Our own local search method, (1+1)-PAES; the strength Pareto evolutionary algorithm (SPEA) [41, 43, 44]; and, the RD-MOGLS algorithm of Jaskiewicz. To make the comparison, each algorithm is run 30 times on each instance of the 0/1 knapsack problem. Statistical methods are then used to give a single measure of relative quality as judged between pairs of algorithms.

The remainder of the paper is organised as follows: The general multiobjective optimization problem is defined in Section 2. Terminology relating to partial ordering relationships and scalarizing functions are also given. In Section 3, methods for selection in multiobjective metaheuristics are reviewed, and we briefly describe the SPEA and RD-MOGLS algorithms. The framework for a new multiobjective memetic algorithm is put forward in Section 4. Then, in

Section 5 the M-PAES algorithm is described. The experimental method used for verifying the algorithm, and comparing its performance is discussed in Section 6. The parameter choices made for each algorithm, including two versions of SPEA, RD-MOGLS, and a benchmark single objective EA, are given in this section. Results are presented and analysed in Section 7. Finally, we summarize the paper and draw conclusions in the last section.

2. Multiobjective optimization terminology

A general (unconstrained) multiobjective optimization problem (MOP) can be defined as

$$\begin{aligned}
 \text{"maximize"} \quad & \vec{z} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})) \\
 \text{subject to} \quad & \vec{e}(\vec{x}) = (e_1(\vec{x}), e_2(\vec{x}), \dots, e_m(\vec{x})) \leq 0 \\
 \text{and} \quad & \vec{x} = (x_1, x_2, \dots, x_n) \in X \\
 & \vec{z} = (z_1, z_2, \dots, z_k) \in Z
 \end{aligned} \tag{1}$$

where \vec{x} is a decision vector or solution, and X is the decision space i.e. the set of all expressible solutions. The objective function $\vec{f}(\vec{x})$ maps X into \mathbb{R}^k , where $k \geq 2$ is the number of objectives. The vector $\vec{z} = \vec{f}(\vec{x})$ is the objective vector or point. The image of X in objective space is the set of all attainable points, Z . The constraints $\vec{e}(\vec{x}) \leq 0$ determine the set of feasible solutions. The term *maximize* appears in quotation marks because, in general, there does not exist a single solution that is maximal on all objectives. Therefore, one may seek to find a set of solutions $X^* \subseteq X$, the Pareto optimal set, with the property that:

$$\begin{aligned}
 \forall \vec{x}^* \in X^* \quad & \nexists \vec{x} \in X \text{ such that } \vec{x} \succ \vec{x}^* \\
 \text{where } \vec{x} \succ \vec{x}^* \quad & \iff \forall i \in \{1, \dots, k\} \quad z_i = f(x_i) \geq z_i^* = f(x_i^*) \\
 & \wedge \exists i \in \{1, \dots, k\} : z_i > z_i^*
 \end{aligned} \tag{2}$$

where $\vec{x} \succ \vec{x}^*$ is read as \vec{x} is *superior* to \vec{x}^* , and the corresponding objective vector (or point) \vec{z} is said to *dominate* \vec{z}^* . Conversely, $\vec{x} \prec \vec{x}^*$ is read as \vec{x} is *inferior* to \vec{x}^* , and the corresponding point \vec{z} is said to be *dominated* by \vec{z}^* . Similarly, a pair of mutually nondominated points are defined as $\mathbf{x} \sim \mathbf{x}' \iff \mathbf{x} \not\prec \mathbf{x}' \wedge \mathbf{x}' \not\prec \mathbf{x}$. Solutions in the Pareto optimal set are also known as efficient, admissible or non-inferior solutions. Their corresponding points in objective space are termed nondominated and when plotted in objective space, form the *Pareto front*.

A set of solutions $A \subseteq X$ is said to be an *internally efficient* set if

$$\forall \vec{a} \in A \quad \nexists \vec{b} \in A \text{ such that } \vec{b} \succ \vec{a} \tag{3}$$

We can see from this definition that if $A = X$ then the internally efficient set is the Pareto optimal set defined in (2). However, with $A \subset X$ the set need not be the Pareto optimal set. Approximations to the Pareto optimal set, generated by a search process will be such internally efficient sets (since there is no point in returning solutions that are known to be dominated), but will usually not be Pareto optimal.

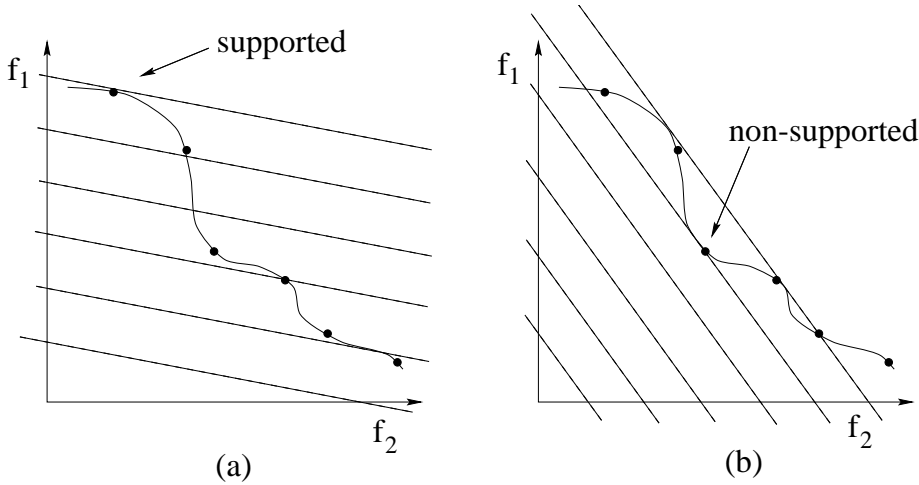


Fig. 1. Supported and non-supported solutions in the Pareto front.

It is possible to convert the original MOP into a parameterized SOP by performing a weighted sum of the objectives, thus:

$$\begin{aligned}
 & \text{maximize} && S_{ws}(\vec{z}, \vec{\lambda}) = \sum_{i=1}^k \lambda_i \cdot z_i \\
 & \text{subject to} && \vec{z} \in Z, \\
 & && \sum_{i=1}^k \lambda_i = 1, \\
 & \text{and} && \forall i, i \in \{1, k\} \lambda_i \geq 0
 \end{aligned} \tag{4}$$

Every point in objective space that is a maximum of S_{ws} , given a weight vector $\vec{\lambda}$, is a nondominated point. The corresponding solution is, therefore, Pareto optimal and is called a *supported* efficient solution. However, there can exist nondominated points that are not a maximum of S_{ws} for *any* possible weight vector $\vec{\lambda}$. These points necessarily lie in a concave region of the Pareto front (Figure 1). The corresponding solutions are termed, *non-supported* efficient solutions. To make it possible to obtain every Pareto-optimal solution, including the non-supported efficient solutions, a non-linear scalarizing function can be employed. One such function is the λ -weighted Tchebycheff metric defined as:

$$\begin{aligned}
 & \text{minimize} && S_T(\vec{z}, \vec{\lambda}) = \max(\lambda_i \cdot z_i), i \in \{1, k\} \\
 & \text{subject to} && \vec{z} \in Z, \\
 & && \sum_{i=1}^k \lambda_i = 1, \\
 & \text{and} && \forall i, i \in \{1, k\} \lambda_i \geq 0
 \end{aligned} \tag{5}$$

It is clear that both methods given above, can be used when the relative importance weights of the various objectives are known *a priori*, and the goal is to find one or more solutions that maximize the particular weight vector, $\vec{\lambda}$. However, it is now common to use these scalarizing functions to approximate the whole Pareto front (in a single algorithm run) by systematically varying the parameters during the search process. For more information on scalarizing functions see [39].

In this paper, only methods for approximating the entire Pareto front are considered. However, it is noted that there are many other methods for performing multiobjective optimization, and the interested reader is referred to [3, 38] for further information.

3. Acceptance functions and selection mechanisms in multiobjective metaheuristics

Algorithms for finding approximate solutions to intractable optimization problems rely on very general properties of search spaces in order to work. These methods are thus termed *metaheuristics* because they are more generally applicable than heuristics; the latter being any method helpful in searching a specific problem space. The general form of a search metaheuristic such as an evolutionary algorithm or simulated annealer is very simple: solutions are initially generated randomly, evaluated, and stored. Following this, new solutions are constructed using the stored solution(s). The new solutions are used to update the store, and the process is repeated until some stopping criterion is met [36].

What makes the difference between the particular metaheuristics is: 1. The methods used to construct new solutions from those already stored; and 2. The criteria for solutions to be placed into the store (acceptance function), or for using particular solutions already stored (selection mechanism). Usually, 2. is based on measuring the relative quality of solutions as judged by the (scalar) objective function. Solutions with higher relative quality are preferred. However, in Pareto optimization, only a partial ordering of solutions is available, resulting in three alternative methods for carrying out selection [19]: 1. Consider just one objective in isolation, each time a solution is evaluated for selection. This method is known as criterion selection. 2. Utilise the partial ordering directly to perform selection. This is Pareto selection. 3. Scalarize the different objectives, using a parameterized function, and vary the parameters such that a diverse set of solutions may be found. This is called scalarizing selection.

3.1. Criterion selection

The first, pioneering work in the EMOO field was Schaffer's vector evaluated GA (VEGA) [32]. Its selection scheme worked by building up the intermediate population in fractions. Each fraction of the intermediate population was selected from the current generation using a different component of the objective vector to assign fitness. VEGA worked well, although it has a tendency to favour extreme solutions to the detriment of solutions that represent a compromise of the different objective attributes, particularly when the shape of the Pareto front is concave. Two other EAs that used criterion selection: a tournament selection method based on comparing pairs of individuals on one chosen objective was proposed by Fourman in 1985 [11], and, in 1991, Kursawe devised a method based on deleting the worst-performing fraction of the population according to one objective at a time [27]. Both methods are discussed in more detail in [9]. Criterion selection has not gained much popularity since these early algorithms because other methods of selection seemed to exhibit better behaviour. However, some new methods based on different forms of criterion selection have been recently devised [12, 35].

```

Initialize:    $P = \emptyset$ 
              Generate  $\mathbf{x} \in X$  randomly
               $P \leftarrow P \cup \mathbf{x}$ 
Main Loop:    $\mathbf{x}' \in X \leftarrow \text{mutate}(\mathbf{x})$ 
              if ( $\mathbf{x}' \succ \mathbf{x}$ )
                   $P \leftarrow P \cup \mathbf{x}'$ ,  $\mathbf{x} \leftarrow \mathbf{x}'$ 
              else if ( $\mathbf{x} \sim \mathbf{x}'$ ) {
                   $i = 0$ ,  $replace = TRUE$ 
                  while ( $i < |P|$ ) {
                      if ( $\mathbf{x}' \succ \mathbf{x}_i \in P$ )
                           $P \leftarrow P \cup \mathbf{x}'$ ,  $P \leftarrow P \setminus \mathbf{x}_i$ 
                      else if ( $\mathbf{x}_i \succ \mathbf{x}'$ )
                           $replace = FALSE$ ,  $i = |P|$ 
                       $i++$  }
                  if ( $replace == TRUE$ )
                       $\mathbf{x} \leftarrow \mathbf{x}'$ 

```

Fig. 2. PAES-like Pareto ranking local search. P is the set of internally efficient solutions.

3.2. Pareto selection

Selection mechanisms based on Pareto dominance relationships have been the most prevalent in the population-based algorithms used in the EMOO community. In 1989, Goldberg [15] first suggested an elegant method of ranking a population of solutions, based on their mutual dominance relations. His non-dominated sorting method effectively ‘peeled off’ Pareto fronts one at a time from the population, assigning maximum fitness to all those solutions in the first layer, and progressively less fitness to successive layers. The nondominated sorting method was implemented in an algorithm, NSGA, by Srinivas and Deb [34] in 1994. The NSGA has since been one of the most popular and successful methods in EMOO and fuelled interest in the field. Similar methods of selection, based on Pareto ordering of solutions, have been devised by Fonseca and Fleming [8], Horn and Nafpliotis [20], and others. More recently, Zitzler and Thiele [44] proposed a method of exploiting co-evolution to perform fitness assignment in an elitist EA.

These population-based algorithms all exploit the knowledge in the whole (or a sample) of the population in order to perform selection. This is achieved by performing many dominance comparisons between solutions in order to compute fitness. The advantage of such an approach is that the current store of solutions is being used efficiently to drive the search towards the Pareto front in all directions simultaneously. This is because newly generated solutions are judged by their dominance relationships with all other solutions. Commonly, speciation methods are also used to ensure that an even and diverse spread of points across the internal Pareto front is achieved, further increasing the efficiency and efficacy of these methods. The disadvantage of this approach is the high computational cost of performing so many comparisons, particularly when large populations are involved, or there are many objectives. However, this criticism is often dismissed using arguments that, in most real-world applications, the cost of comparisons is irrelevant compared to the time expended in evaluating solutions.

In single-point search methods, there has been little use of Pareto selection. This is because only two solutions, the current and candidate (mutant) solution, are compared at each step. Frequently, pairs of solutions ($\mathbf{x}, \mathbf{x}' \in X$) will be *nondominated* with respect to each other. That is, neither is better than the other on all objectives. Thus, it is not possible to judge accurately which of the

- Initialization - Repeat S times:
 - Generate a random weight vector $\vec{\lambda}$.
 - From a randomly generated solution x , perform local search, using a scalarizing function $S(\vec{z}, \vec{\lambda})$, to obtain x' .
 - Add x' to the current set of solutions, CS .
 - Update the potentially efficient set PE with x' .
- Main Loop - Repeat until some stopping criterion is met:
 - Generate a random weight vector $\vec{\lambda}$.
 - Select the $N \leq S$ best solutions from CS , using the measure $S(\vec{z}, \vec{\lambda})$, to form a temporary population TP .
 - Repeat N times:
 - * Do crossover on a pair of uniformly randomly selected parents from TP .
 - * Do local search from the offspring x to form x' using $S(\vec{z}, \vec{\lambda})$, and update PE with x' .
 - * If x' is better than worst member of TP , then add to CS and TP , deleting worst member of TP .
 - * Update set PE with x' .

Fig. 3. RD-MOGLS

current and mutant solutions is better, and much of the selection pressure is lost. However, if a *comparison set* of the best (nondominated) solutions found during the search is maintained, and used to aid in judging solution quality in these undecidable cases, then a very high selection pressure can result. This is the basis of the local search method, (1+1)-PAES, introduced by Knowles and Corne [25, 26]. The PAES technique has the advantage that one particular search direction is not favoured in the local search, so solutions spread across the whole Pareto front can be found from a single search. Pseudocode for a very simple, PAES-like local searcher is given in Figure 2. In fact, in (1+1)-PAES, efficiency is ensured by storing only a limited number of internally efficient solutions in P , and a technique for encouraging diversity is also employed.

3.3. Scalarizing selection

The advantage of using scalarizing functions is that standard selection or acceptance mechanisms can be used unchanged. This has led to the great popularity of the approach in the operations research and multi-criteria decision making (MCDM) community. Adaptations of both tabu search [13, 14, 17], and simulated annealing [4, 33, 37] have used these methods. Most of the methods store the internally efficient solutions found but they are not used further in the search. In most of the algorithms, a random weighting of the scalarizing function is simply chosen at each step, and the new solution generated is accepted or rejected

according to its utility on this measure compared with the current solution. This method does not take advantage of the information from the previously found solutions, nor does it direct the search in any particular direction. These factors seem to suggest that such an approach would be inefficient compared to the latest elitist Pareto MOEAs, but there is little empirical evidence of this fact, to date.

Some researchers argue that the use of such scalarizing vectors more naturally allows the preferences of the decision maker to be used to guide the direction(s) of the search towards the region(s) of interest (see for example [4]). This may be true, although good methods for directing the search towards desirable regions of the moving Pareto front within Pareto ranking methods are now available [6, 31]. Also, the use of randomly selected utility functions in the absence of such preference information, as used in many of the MCDM algorithms, seems wholly unsatisfactory.

Examples of the use of scalarizing functions are few in population-based Pareto optimization approaches. Hajela and Lin's [16] proposed a GA using weighted-sum aggregation of objectives. In their method, the parameters of the weight vector were encoded on the chromosome, thus cleverly providing an implicit mechanism for maintaining diversity in objective space. Later, Bentley and Wakefield [1] put forward and tested a number of different weighting schemes for providing a sub-set of Pareto-optimal solutions. But these methods have not caught on, and in the case of [16], there is now some evidence that this approach is less efficient than some pure Pareto GAs [42]. More recently, however, memetic algorithms for multiobjective optimization put forward by Ishibuchi and Murata [21] and Jaskiewicz [22], have also used linear scalarizing functions. The latter, the random directions multiple objective genetic local search (RD-MOGLS) is described next.

The RD-MOGLS algorithm is based heavily on the genetic local search algorithm put forward by Ishibuchi and Murata. In both approaches a weighted scalarizing function u is drawn at random at each iteration. After selection and recombination, the offspring produced is improved with respect to the *same* utility function u , using a local search. A set of potentially efficient solutions is updated at the end of the local search phases, and in both approaches all potentially efficient solutions can take part in future iterations of the algorithm. However, there are differences between the algorithms, too. The key difference is that RD-MOGLS uses a small temporary population TP that is used to restrict mating at the selection stage. Also, the main population, CS , has variable size, and is organised as a queue, with older solutions being eventually discarded. An outline of the RD-MOGLS algorithm is given in Figure 3. The algorithm uses either weighted linear (Equation 4) or weighted Tchebycheff (Equation 5) scalarizing functions. Ishibuchi and Murata specified the use of a weighted linear function only.

4. A new framework for a multiobjective MA

It is clear from the previous section that many methods for performing Pareto optimization are already available. However, the most recently developed MAs have used scalarizing methods rather than Pareto selection for identifying and exploiting the solutions already found. However, there are a number of arguments against this approach. In the schemes put forward [21, 22] the weighting

	Pareto selection	Linear aggregation
Advantages	Proven to be efficient as selection mechanism in GA populations.	Fast and easy method for use in local search.
	Searches all directions simultaneously.	Easily parallelizable.
	Can find non-supported solutions.	
Disadvantages	Less suited to parallelization.	Not well-suited as selection mechanism in GA populations.
	More difficult to use with local search.	Efficient use of information in the current population?
		Cannot easily find non-supported solutions.

Table 1. Advantages and disadvantages of Pareto selection versus Linear aggregation

parameters specifying the direction of search are chosen at random without regard to the solutions that have already been found. In many cases, it will not be sufficient to select weights in all directions with equal probability. Rather, because some areas of the Pareto front are easier to obtain than others, some form of adaptive scheme is required. This is well catered for in Pareto selection methods that invariably use some form of diversity maintenance technique aimed at giving more opportunity to solutions whose corresponding points in objective space exist in unpopulated areas of the nondominated front.

Also, the current MA schemes perform local search on the offspring in the same direction as was used for selecting the parents. This unidirectional searching means that a solution generated during the local search phase may be immediately rejected even though it dominates one or more of the best solutions already found, and may be an excellent solution when evaluated according to a *different* weighting vector. In Pareto selection methods, there is more efficiency because solutions are not usually discarded if they dominate solutions in the discovered Pareto front. Effectively, Pareto selection methods search in all directions at once. More *comparisons* are required in Pareto selection methods, but in many optimization problems minimizing the number of objective function evaluations necessary to achieve a certain level of result is more important than minimizing the overhead of the algorithm.

Some empirical evidence is also given in [42] that weighted aggregation methods do not seem as effective as Pareto selection, in population-based approaches. This result relates to both the ability to maintain a spread of good solutions, in both convex and concave Pareto fronts, and the proximity of the discovered Pareto front to the global Pareto front that the algorithms achieve, given equal numbers of function evaluations. While it is not obvious how Pareto selection can be used with single point local-search methods, our algorithm, PAES [25], is a simple solution to this problem that has been shown to work well [23, 26]. Scalarizing methods may yet have an advantage in parallel MA implementations, however, because local search phases could be independently run on different processors with no need for communication between them.

We give a summary of the possible advantages and disadvantages of Pareto selection compared with linear aggregation in Table 1.

The new framework for a memetic algorithm, based entirely on Pareto selection, is given below.

1. Generate a population P of solutions using some method.
2. Store the internally efficient solutions from P in a global archive G .
3. Repeat until a stopping criteria is satisfied:
 - (a) For each candidate solution c in P :
 - i Initialize a local archive of solutions H .
 - ii Perform a local search from c . At each step of the local search:
 - Generate c' from c .
 - Compare c' with one or more solutions from H to determine whether to accept c' ($c \leftarrow c'$) or maintain c .
 - Update H with c' if applicable.
 - Update G with c' if applicable.
 - iii Place improved solution c back in P .
 - (b) Perform recombination of solutions in P to obtain P' :
 - i Select parents from $P \cup G$.
 - ii Recombine to form offspring c .
 - iii Compare c with one or more solutions from G to determine whether to accept c ($P' \leftarrow P' \cup c$), or reject it.
 - iv Update G with c if applicable.
 - (c) Update population: $P \leftarrow P'$.
4. Return global archive G of unique internally efficient solutions.

This framework is deliberately loosely-specified. The key defining features of the framework are: There are separate single-point (local search) and population-based search phases; all comparisons to determine acceptance/rejection of solutions are Pareto comparisons; the archives G and H store internally efficient solutions; H is initialized for each local search phase; H is used to determine the acceptance of local-search moves; G is updated for every new solution generated; in the population-based phase, G is used to determine acceptance of the offspring.

The method used to generate solutions in the local search phase can be any operator based on perturbing a *single* solution. The recombination method can be any operator based on generating solutions using information from two or more solutions. The initialization of the archive H can be achieved using any solution already stored in P or G and should encourage an effective local improvement of the current solution c .

No particular form of diversity maintenance has been specified, either. However, it would necessary to use some form of ‘niching’ or ‘crowding’ in either objective space or parameter space to ensure a good spread of points/solutions. In our MA, M-PAES, an adaptive grid algorithm described in [26] is used.

Elitism is inherent in the framework, and it can be controlled in several ways. Each time a solution is compared with either the local archive H , or the global archive G , to determine whether to accept it or not, a sample of solutions

from the archive is used. By adjusting the size of the sample, the degree of elitism can be controlled. The acceptance function used is not defined in the framework, either. What is specified is that the comparisons used to accept or reject solutions are Pareto comparisons. Possible acceptance functions include some form of tournament acceptance method, or a simulated annealing style acceptance function where solutions may be accepted even if dominated by a small number of members of the archive, depending on a temperature parameter.

It is expected that the multiobjective MA framework will be most effective when very good local search heuristics are available for the particular multiobjective problem. For example, the Lin-Kernighan method for solving TSP problems could be used within this framework, quite easily. However, to test its general applicability we will first use the framework to define a very generally applicable memetic algorithm: M-PAES, that does not incorporate any special operators.

5. M-PAES

The memetic-PAES algorithm (M-PAES) is shown in pseudocode in Figure 4. It is based on the local search multiobjective algorithm, (1+1)-PAES [25], and the framework given above. The archiving of solutions in M-PAES is a little more complicated than in (1+1)-PAES. Recall that at the heart of PAES is a procedure for maintaining a finite sized archive of internally efficient solutions. The solutions in the archive are representative of the best nondominated solutions found by the algorithm as it searches the space. The solutions in the archive serve a dual purpose in (1+1)-PAES: as a memory of the solutions found during the run for presentation at the end; and as a comparison set to aid in estimating the dominance rank of new candidate solutions. In order that these same jobs are performed in M-PAES, two archives are required. This is because each local search phase needs to be partially independent of the global search being performed by the algorithm as a whole. Thus we have a global archive G that maintains a finite set of internally efficient solutions found, and a local archive H that is used as the comparison set in each of the local search phases. At the beginning of a local search phase, H is cleared and filled with solutions from G which do not dominate the candidate solution c . The archive H is then used as in (1+1)-PAES to improve c , i.e. H is maintained and used as a comparison set, while G is continually updated but plays no part in the estimation of the quality of new solutions.

The PAES local search procedure used by M-PAES to improve solutions in P is almost the same as the basic (1+1)-PAES algorithm. However, it differs in the way that termination of the procedure is determined. Termination may be invoked when either of two conditions are fulfilled: (1) If the maximum number of local search moves l_opt is exceeded. (2) If the maximum number of local search fails l_fails is exceeded. To achieve (2), the variable $\#fails$, initially zero, is incremented every time the mutant is dominated by the current solution. It is reset to zero every time a move occurs i.e. when the mutant is accepted as the new current solution. Hence, $\#fails$ effectively counts the number of potentially detrimental moves between improving moves. If this number exceeds the threshold l_fails , the local search is stopped. The local search procedure $PAES(c, G, H)$ is shown in Figure 5.

In the recombination phase, parents are randomly selected from the union of the post-local search population, and the global archive. The resultant child

```

Generate initial population  $P$  of  $n$  random solutions and evaluate
Place each nondominated member of  $P$  in a global archive  $G$ 
Do
  For(each candidate solution  $c \in P$ )    %% local search phase
    Set the current local archive  $H = \emptyset$ 
    Fill  $H$  with any solutions from  $G$  that do not dominate  $c$ 
    Copy the solution  $c$  from  $P$  into  $H$ 
    Perform local search using procedure  $PAES(c, G, H)$ 
    Replace improved solution  $c$  back into population  $P$ 
  End For
Set intermediate population empty:  $n_i = 0$ ,  $P' = \emptyset$ 
Do    %% recombination phase
  Set # recombination trials  $r = 0$ 
  Do
    Randomly choose two parents from  $P \cup G$  and recombine to form  $c$ 
    Compare  $c$  with the solutions in  $G$ 
    Update  $G$  with  $c$  as necessary
     $r++$ 
    While ((( $c$  is dominated by  $G$ )  $\vee$  ( $c$  is in more crowded grid
    location than both parents))  $\wedge$  ( $r < recomb\_trials\_max$ ))
      If ( $c$  is dominated by  $G$ )
        Discard  $c$  and use binary tournament to select a new solution  $c$  from  $G$ 
      Endif
    Place offspring  $c$  into intermediate population  $P'$ ,  $n_i++$ 
  While ( $n_i < n$ )
  Update population:  $P \leftarrow P'$ 
While (stopping criterion is not satisfied)
Return global archive  $G$  of unique internally efficient solutions
    
```

Fig. 4. The M-PAES Algorithm.

```

While((#fails <  $l\_fails$ )  $\wedge$  (#moves <  $l\_opt$ ))
  Mutate  $c$  to produce  $m$  and evaluate  $m$ 
  If ( $c$  dominates  $m$ ) discard  $m$ , #fails++
  Else if ( $m$  dominates  $c$ )
    Replace  $c$  with  $m$ , add  $m$  to  $H$ , #fails = 0
  Else if ( $m$  is dominated by any member of  $H$ ) discard  $m$ 
  Else apply  $test(c, m, H)$  to determine which becomes the new
  current solution and whether to add  $m$  to the archive
  Archive  $m$  in  $G$  as necessary
  #moves++
End while
    
```

 Fig. 5. The $PAES(c, G, H)$ procedure.

is accepted only if it is nondominated with respect to the entire global archive, and it resides in a less crowded region (grid location [26]) than at least one of its parents. If it dominates any member of G it is naturally accepted too. However, solutions that are dominated by member(s) of G , or that reside in crowded regions are rejected. In this case two new parents are selected again and recombination is applied once more. The procedure is repeated until either a child is accepted or a threshold number of recombinations $recomb_trials_max$ is exceeded. In the latter case, a solution is selected by binary tournament, from the global archive, to join the intermediate population P' . The recombination strategy is, as a whole, extremely elitist, following the general form of the (1+1)-

```

If the archive is not full
  Add  $m$  to the archive
  If ( $m$  is in a less crowded region of the archive than  $c$ )
    Accept  $m$  as the new current solution
  Else maintain  $c$  as the current solution
Else
  If ( $m$  is in a less crowded region of the archive than  $x$  for
    some member  $x$  on the archive)
    Add  $m$  to the archive, and remove a member of the archive from
    the most crowded region
    If ( $m$  is in a less crowded region of the archive than  $c$ )
      Accept  $m$  as the new current solution
    Else maintain  $c$  as the current solution
  Else
    If ( $m$  is in a less crowded region of the archive than  $c$ )
      Accept  $m$  as the new current solution
    Else maintain  $c$  as the current solution

```

Fig. 6. Pseudocode for $test(c, m, archive)$.

PAES algorithm employed in the local search phase. In the development of M-PAES, early versions did not have the facility of repeatedly rejecting children of recombination. However, we found that this weakened the effectiveness of the elitism inherent in (1+1)-PAES and so the recombination phase was made more stringent in later versions.

6. Experimental Method

The M-PAES algorithm is tested on a suite of multiobjective 0/1 knapsack problems. The problems are taken from a recent paper [44] by Zitzler and Thiele (ZT), in which the general ability of their strength Pareto evolutionary algorithm (SPEA) was demonstrated. In [44], the performance of SPEA on these problems was compared with eight other evolutionary algorithms (EAs). Four of the algorithms, each a well-known multiobjective EA, as well as two versions of SPEA, were run 30 times with different random seeds on each of the problems, for 500 generations using the same population sizes¹. The nondominated sets generated from each of the runs were used to make a statistical comparison of the algorithms tested.

The findings of the ZT study are that SPEA is superior to each of the other MOEAs on all of the knapsack problems. However, also included in the set of eight algorithms tested in [44], are two single-objective EAs that use weighted-sum aggregation of the objectives. The relative performance of SPEA and these algorithms is not clear-cut. Hence, in the first part of our comparative study we select as benchmarks, the data sets from the SPEA runs and those of the more powerful of the two single-objective algorithms, SO-5. The other algorithms are not considered. As additional comparators, we generated our own data sets for the (1+1)-PAES algorithm, and an enhanced setup of SPEA, on the knapsack problems.

¹ In the case of SPEA, an internal and an external population exist. The sizes of these were chosen to provide a fair comparison with the other MOEAs in the study.

The second part of the study is a comparison with the memetic algorithm, RD-MOGLS. The same problem instances are used, and once again we also compare performance with (1+1)-PAES acting as a benchmark indicating a basic, good level of performance. The parameter settings for all of the six algorithms that we compared are described in Section 6.2.

6.1. Multiobjective 0/1 Knapsack Problems

An excellent general text on knapsack problems is [28]. In it, the standard 0/1 knapsack problem is described, including the history of methods for solving it, its numerous applications, and its links with fundamental integer programming problems. The multiobjective version of the problem is not described, but it has become a favourite problem in multiobjective combinatorial optimization (see [13]). The following text taken directly from [40] defines the problem:

Given a set of n items and a set of k knapsacks, with

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i, \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i, \\ c_i &= \text{capacity of knapsack } i, \end{aligned}$$

find a vector $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$, such that the capacity constraints

$$e_i(\mathbf{x}) = \sum_{j=1}^n w_{i,j} \cdot x_j \leq c_i \quad (1 \leq i \leq k) \quad (6)$$

are satisfied and for which $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_k(\mathbf{x}))$ is maximum, where

$$f_i(\mathbf{x}) = \sum_{j=1}^n p_{i,j} \cdot x_j \quad (7)$$

and $x_j = 1$ if and only if item j is selected.

Zitzler and Thiele generated nine instances of the problem altogether, of differing combinations of size (number of items), and number of objectives (knapsacks). At the time of writing, the problems are available from an Internet website². In the following experiments, we employ the same chromosome encoding and constraint handling techniques as described in [44], and no additional heuristics for use with the knapsack problems are employed. This allows for a direct comparison between our results and those published by Zitzler and Thiele.

6.2. Parameter Choices

The problem of setting parameters in comparative studies of algorithm performance is a serious one. Our philosophy in this study is that each algorithm be run with settings, found through some experimentation, to provide near-best performance for that algorithm. It is not possible, or even desirable, to use exactly the same parameter settings for each algorithm, as they differ considerably.

The study comprises two different sets of data, one where M-PAES is compared with the results from the ZT study, and the other, where we implemented

² <http://www.tik.ee.ethz.ch/~zitzler>

an algorithm, RD-MOGLS, ran it, and collected the results ourselves. In the former, parameter control is obviously not possible, although we did supplement the results from the ZT study with two further algorithms: an SPEA implementation of our own; and (1+1)-PAES. The parameters of these algorithms, like those of RD-MOGLS, were set empirically to give good performance.

To reduce some of the burden of testing different parameter settings, we do keep core parameters/conditions constant across those algorithms that we have control over. Further details of the parameter choices made for each of the algorithms tested are given below. The data from the setup of SPEA used in [44] is referred to as SPEA(ZT). Our own setup of SPEA is labelled SPEA(KC).

SO-5

The SO-5 data comes from one of two single-objective EAs used in [44]. Unlike the other algorithms considered, these single-objective EAs were run 100 times per test problem, each run optimizing toward a different randomly chosen linear combination of the objectives. The resultant internally efficient solutions among all those generated in the runs form the tradeoff front achieved by the algorithm. The two algorithms both employed equal population sizes to their multiobjective rivals, and differed only in that one (SO-1) was run for 100 generations, and the other (SO-5) was run for 500 generations in every single of the 100 runs used to form the nondominated front. Thus, in the case of SO-5, one hundred times as many function evaluations as in the other MOEAs in the ZT study were performed in order to generate the (single) set of internally efficient solutions. Zitzler and Thiele did not perform the whole process thirty times to give thirty different data sets, but instead just used the same set repeatedly in the statistical analysis carried out. We follow this approach, using ZT's data sets. Hence, where statistical information is given in relation to the SO-5 algorithm, it should be noted that, in fact only one data set for this algorithm is being used, in contrast to all the other algorithms in this study for which 30 runs were performed.

As with the other algorithms in the ZT study, one-point crossover was used. The mutation probability and crossover rate were fixed at 0.01 and 0.8 respectively, as for SPEA(ZT).

SPEA(ZT)

The setup of SPEA is described fully in [44]. The study was designed to show that SPEA could clearly outperform the other MOEAs tested, even with very conservative choices of parameters. Thus the authors kept the external population quite small - 1/5 of the population size of the other MOEAs in the study.

It is important to note that the data sets for SPEA(ZT) record the off-line performance of the algorithm. That is, all of the internally efficient solutions returned in a run were recorded. With the exception of SO-5 the other algorithms in this study are judged using the on-line performance. That is, only solutions stored in the external population (or archive) at the end of the run are recorded.

SPEA(KC)

In [44], the authors chose to run the algorithms for a fixed number of generations and increase population size with the size and number of objectives of the knapsack problem being tackled. To make direct comparison possible, we choose to use the same number of function evaluations as ZT, but do not deem

it necessary to employ equal population sizes. In fact, the total number of evaluations max_evals used by each of the algorithms in this study is the same for a given knapsack problem. Table 2 lists the value of max_evals for each knapsack problem.

SPEA requires two population sizes, N and N' to be set. ZT selected to use $N = 4/5$ and $N' = 1/4$ of the size of population used by the other GAs in their study. Experiments performed by us show that the performance of SPEA on these knapsack problems is improved significantly when population sizes of $N = 1/5$ and $N' = 4/5$ are used, for the same total number of function evaluations.

Our experiments also indicate that changing the crossover type from one-point to uniform improves the performance of SPEA on the knapsack problems. Thus, SPEA(KC) employs uniform crossover. A fixed per-bit mutation rate $p_m = 0.01$ is used, as in [44]. No experiments in which p_m was varied were undertaken by us. Since no other parameters need to be set for SPEA, we believe that SPEA(KC) is close to the best setup of SPEA possible for the problems tackled.

(1+1)-PAES

With (1+1)-PAES, very few parameters must be set. The archive size was set equal to the external population size N' of SPEA, so that the same number of solutions is returned by each algorithm. Similarly, the number of evaluations is set in accordance with the total number performed by SPEA.

The mutation rate p_m was set to $4/L$ (where L is the number of bits in the chromosome) for all problems. This setting follows our principle of using the best setting for the particular algorithm.

The number of bisections of the objective space l used in the adaptive grid algorithm for maintaining diversity [26] was set according to the number of objectives of the problem. The values $l = 5$, $l = 4$, $l = 3$ were used for the 2, 3, and 4 objective multiple knapsack problems, respectively.

M-PAES

The total number of evaluations, number of bisections of objective space, l , and per-bit mutation rate used in M-PAES are as for (1+1)-PAES. The population size N , was set equal to the internal population of SPEA(KC). The two archives were sized equally, to match the external population of our setup of SPEA. Thus, the same number of solutions are returned by SPEA(KC), M-PAES and (1+1)-PAES. The recombination operator was uniform crossover, in line with SPEA(KC), and the mutation rate was set to $4/L$, as with PAES.

In M-PAES, three more parameters must be set. These are the number of crossover trials, the maximum number of local moves l_opt , and the maximum number of consecutive failing local moves l_fails . Choices that give good general performance were found to be $l_opt = 50$, $l_fails = 20$, and $cr_trials = 25$. However, it was found that increasing the number of crossover trials for the 3 and 4-objective problems increased performance further. A list of the best parameter selections found is given in Table 2. The results presented in Figure 8 and Table 4 are for these settings.

RD-MOGLS

The parameters chosen for RD-MOGLS are given in Table 3. The initial and temporary population sizes were derived empirically from a few test runs of RD-MOGLS. The size of PE was set equal to the equivalent nondominated *archive*,

Knapsack problem	Parameter			
	l_fails	l_opt	cr_trials	max_evals
2-250	20	100	25	75000
2-500	20	100	25	100000
2-750	20	100	25	125000
3-250	20	50	100	100000
3-500	5	20	125	125000
3-750	20	50	150	150000
4-250	20	50	125	125000
4-500	20	50	150	150000
4-750	5	20	150	175000

Table 2. Parameter settings used in the M-PAES algorithm for the various multiple objective knapsack problems. The same total number of function evaluations max_evals , shown for each problem, was used in M-PAES, SPEA(KC) and (1+1)-PAES.

initial population size S :	100
temporary population size N :	20
efficient solution set size $ PE $:	100
local search fails l_fails :	5
weight vector parameter n :	100

Table 3. Parameter settings for RD-MOGLS.

used in M-PAES and (1+1)-PAES. There must also be a stopping criterion for each of the local search phases. This detail is omitted in [22], so we choose to end each local search phase when l_fails consecutive moves do not improve the solution. This is similar to the method used in M-PAES. Once again, l_fails was set empirically. As with M-PAES, the mutation rate $p_m = 4/L$, and the recombination operator was uniform crossover.

To obtain solutions distributed across the whole Pareto front, RD-MOGLS selects a weight vector at random at each step. We follow the same procedure used by Jaskiewicz for doing this, which was put forward in [2], and gives a maximally dispersed set of weight vectors. A random weight vector is generated in which each individual weight takes on one of the following values $\{\frac{m}{n}, m = 0, \dots, n\}$, where the fixed parameter n is any positive integer, and the value of m is uniformly randomly distributed in $0, \dots, n$. The number of unique weight vectors this results in is equal to:

$$\binom{k+n-1}{n} = \frac{(n+k-1)!}{n!(k-1)!} \quad (8)$$

Following Jaskiewicz, we use a value of $n = 100$, in all experiments, giving respectively 101, 5151, and 176,851 weight vectors for the 2, 3, and 4 objective problems.

7. Results

7.1. Performance Metrics

As in previous research, we measure the performance of the algorithms tested using a statistical comparative assessment technique adapted from [10]. We refer the reader to [24, 25] for a complete description of our implementation of the technique and a discussion of its advantages and disadvantages.

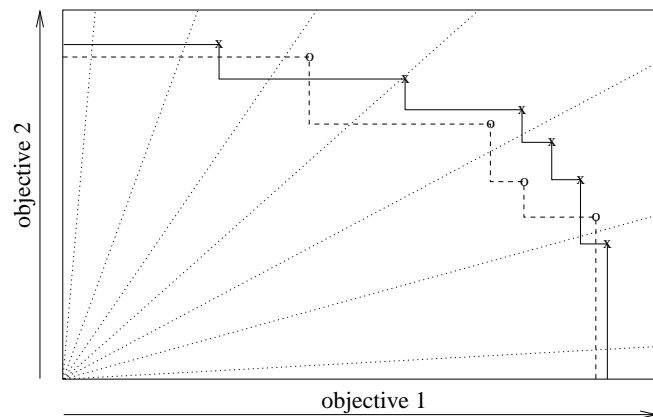


Fig. 7. A collection of two sets of nondominated vectors. Each set of vectors defines an *attainment surface*, dividing the objective space. The attainment surfaces can be sampled using a number of angled sampling lines, as shown.

Here, the reader need only understand the following points. 1. Any set of objective vectors can be viewed as defining a surface (an *attainment surface*) in objective space, dividing the space into a dominated region and a nondominated region (Figure 7). 2. A collection of runs of an algorithm will thus generate a collection of such surfaces. 3. The collection of attainment surfaces can be sampled at various points using lines, angled in the direction of increasing value in each objective, that intersect the surfaces (Figure 7). 4. The intersection points along each of sampling lines gives a univariate distribution which can be analysed using standard non-parametric statistical tests. 5. When the collections of attainment surfaces come from different algorithms, statistical inferences as to which algorithm's distribution of surfaces is 'better', along each sample line, can be made. Using these points, we are able to input *collections* of individual runs from each of a pair of algorithms, and from these, present results in two ways: First, as a pair of numbers [a,b] indicating, respectively, the percentage of the space where algorithm A outperforms algorithm B, and the percentage of the space where algorithm B outperforms algorithm A. The percentages returned are the result of a set of Mann-Whitney U tests [29] performed on each sample line on the collections of data, at a given confidence level (we choose 95%). Second, for two-objective problems, we can plot surfaces representing the median, best, or worst surface that the algorithm returns. Here, we plot only the median surface of the three algorithms tested, on the two-objective problems. In all cases approximately 500 lines were used to sample the attainment surfaces.

7.2. Analysis - Part 1

The results of the first part of the study, where M-PAES is compared with the algorithms in the ZT study are shown in Figure 8 and Table 4. In Figure 8 the median surfaces generated by each algorithm are plotted for the two-objective problems. A number of observations can be made from these plots. First, our setup of (1+1)-PAES gives very similar levels of performance to the setup of SPEA used by Zitzler and Thiele on the three problems. This observation is in keeping with

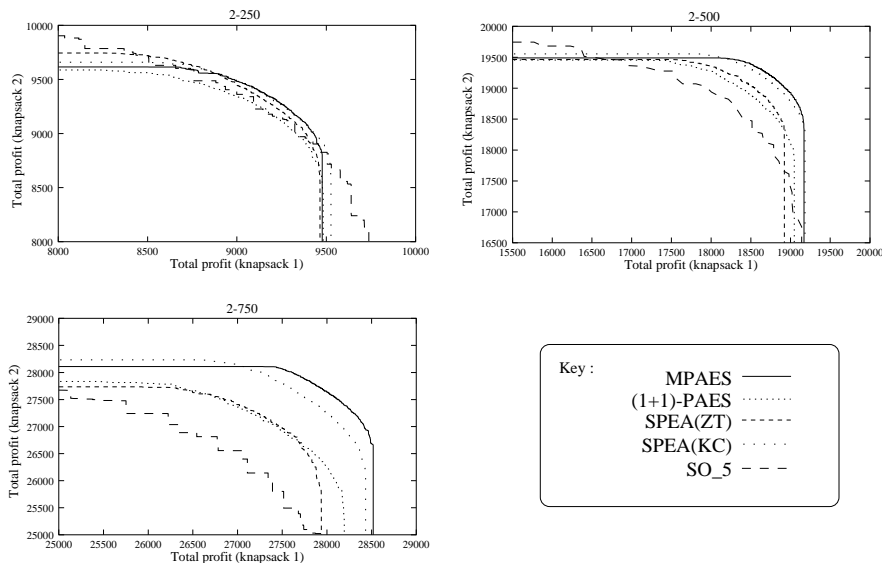


Fig. 8. Median surfaces calculated from 30 runs of each algorithm, on the two-objective knapsack problems.

Instance	Algorithm			
	1+1-PAES	SPEA(ZT)	SPEA(KC)	SO-5
2-250	[87.2, 0]	[61.6, 24.5]	[28.1, 21.8]	[64.3, 27.5]
2-500	[100, 0]	[100, 0]	[80.9, 7.2]	[92.8, 3.6]
2-750	[100, 0]	[100, 0]	[95.5, 0]	[100, 0]
3-250	[100, 0]	[69.1, 18.7]	[53.6, 26.6]	[44.4, 51.8]
3-500	[100, 0]	[93.7, 0]	[67.9, 20.8]	[74.5, 21.7]
3-750	[100, 0]	[100, 0]	[84.2, 3.9]	[94.2, 3.6]
4-250	[100, 0]	[46.1, 31.6]	[32.9, 43.4]	[10.0, 85.9]
4-500	[100, 0]	[92.5, 3.7]	[63.3, 21.7]	[35.5, 56.6]
4-750	[100, 0]	[100, 0]	[82.9, 7.4]	[71.7, 25.4]

Table 4. The results of testing M-PAES against the algorithms shown, using our statistical techniques [26]. The knapsack problems have 2, 3 or 4 objectives and 250, 500 or 750 items, as indicated.

previous research where (1+1)-PAES was compared with SPEA [23]. Second, in all cases SPEA(KC) outperforms SPEA(ZT). Third, M-PAES is very competitive with SPEA(KC) and clearly outperforms both (1+1)-PAES and SPEA(ZT). In the largest of the three problems, M-PAES generates a median surface that is clearly superior to the median surface of any of the other algorithms. On the smaller problems, M-PAES fails to generate solutions as far towards the extremes of the objective space as either SO-5 or SPEA(KC), but has generated a median surface that dominates these algorithms in the region where the two objectives trade off most rapidly with each other.

The statistical results for all the problems are summarised in Table 4. Comparisons between M-PAES and each of the algorithms are presented only. Thus, the statistic [100, 0] in the upper left entry in the table means that M-PAES gives a better distribution of surfaces over 100% of the combined nondominated front than (1+1)-PAES on the 250 item, 2 knapsack problem. Again, several

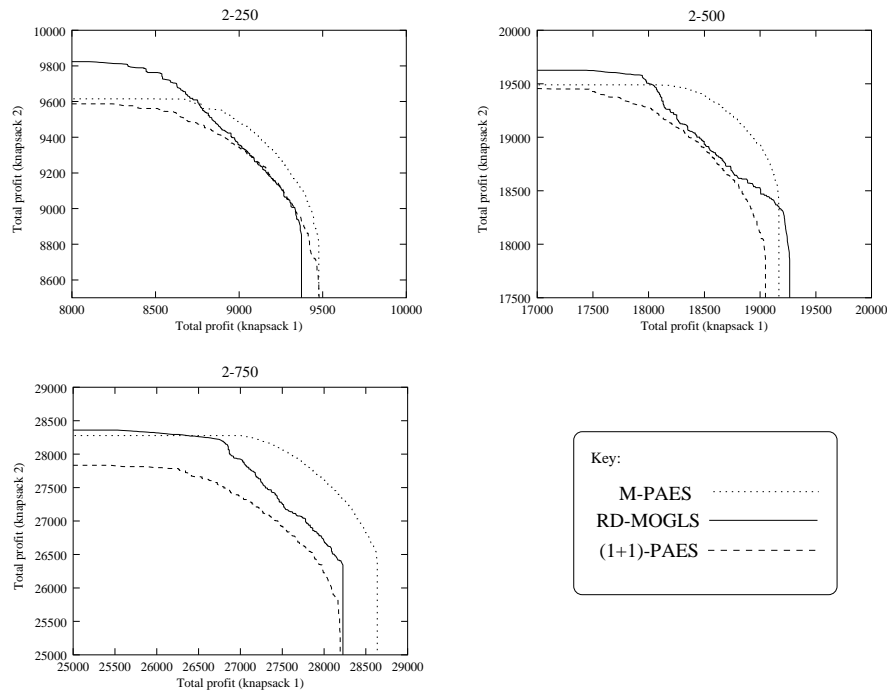


Fig. 9. Median surfaces calculated from 30 runs of the three algorithms, M-PAES, RD-MOGLS, and (1+1)-PAES, on the 2-objective knapsack problems.

observations from these results can be made. First, the first three rows of the table verify that M-PAES performs well on the two-objective problems, as suggested by the plots in Figure 8. Its relative performance increases as the number of items increases. This is true, not only on the 2-objective problem but on all the problems presented. However, as the number of objectives increases, the performance of SPEA(KC) and SO-5 increase relative to M-PAES, so that M-PAES is outperformed by SO-5 on the two smaller four-objective knapsack problems. SPEA(KC) only outperforms M-PAES on one problem, the smallest of the 4-objective knapsack problems, and only by a small margin, according to our statistical analysis.

7.3. Analysis - Part 2

Results collected from the second part of the study are shown in Figure 9 and Table 5. In all three of the plots in Figure 9, the median surface generated by RD-MOGLS extends beyond the surface generated by M-PAES in at least one of the objectives. However, the M-PAES algorithm generates a median surface that dominates the surface of RD-MOGLS over a larger portion of the tradeoff front. With increasing problem size (items), the M-PAES algorithm's performance improves relative to RD-MOGLS. Both M-PAES and the baseline algorithm (1+1)-PAES give smoother, more convex median surfaces than RD-MOGLS.

The statistical results for all the problems are summarised in Table 4. Comparison is made between M-PAES and RD-MOGLS only. Thus, the statistic

Knapsacks	Items		
	250	500	750
2	[65.4, 28.2]	[49.4, 0]	[61.5, 0]
3	[72.7, 25.1]	[89.8, 0]	[100, 0]
4	[100, 0]	[100, 0]	[100, 0]

Table 5. The results of testing M-PAES against RD-MOGLS using our statistical techniques [26] and 30 independent runs of each algorithm. The knapsack problems have 2, 3 or 4 objectives and 250, 500 or 750 items, as indicated.

[65.4, 28.2] in the upper left entry in the table means that M-PAES gives a better distribution of surfaces than RD-MOGLS over 65.4% of the tradeoff front, and vice-versa, RD-MOGLS gives a better distribution than M-PAES on 28.2%, on the 250 item, 2 knapsack problem. Again, several observations from these results can be made: The first row of the table verifies that M-PAES performs well on the two-objective problems, as suggested by the plots in Figure 8. Its relative performance increases as the number of items increases. This is true, not only on the 2-objective problem but also the 3-objective problems. Clearly, as the number of objectives increases, the relative performance of M-PAES increases compared to RD-MOGLS, with the results showing that the distribution of solutions found from the runs of M-PAES show statistically significant superiority to those of RD-MOGLS on 100% of the sampling lines used to probe the Pareto fronts.

8. Conclusion

Multiobjective combinatorial optimization (MOCO) problems often demand the application of approximate methods for their solution. So, memetic algorithms, which have proved to be very effective on a number of single-objective combinatorial problems, may offer an ideal approach to MOCO. However, dealing with multiple objectives is not straight forward: One must decide how to guide selection, given a vector of objective values. In this paper, the key methods of selection used in Pareto optimization were reviewed. Pareto selection seems to offer certain advantages not offered by either criterion selection or aggregation methods: it is able to efficiently use the solutions found in the multiobjective search, in order to find new ones, and when combined with provisions for maintaining diversity, it ensures that an even spread of solutions are found across the whole Pareto front.

Although Pareto selection has been applied successfully in many evolutionary algorithms, the memetic algorithms so far proposed have used aggregation selection. In this paper, we put forward a new memetic algorithm framework based on Pareto selection. The key elements of the framework were identified, and its flexibility was also indicated. The strength of the new approach may not become apparent until it is used with local-improvement heuristics that would give it a strong advantage over traditional genetic algorithms. However, a generic memetic algorithm, M-PAES, based on the framework was constructed and its performance was compared with other state-of-the-art approaches.

The efficacy of M-PAES was verified on a set of nine multiobjective 0/1 knapsack problems. Two separate studies were conducted. In the first, the results from runs of M-PAES were compared with the results gathered in a published study by Zitzler and Thiele. The results sets from the study were supplemented by also

running our own implementation of the strength Pareto evolutionary algorithm (SPEA), as well as (1+1)-PAES. The results indicate that M-PAES performs better than the local-search algorithm, (1+1)-PAES (on which it is based), on all of the problem instances. Compared with SPEA, the performance of M-PAES is similar, for a setup of each algorithm empirically derived to give near-best performance. Indeed, M-PAES appears to be superior on some problem instances, although comparison between these very different algorithms is difficult.

In the second study, M-PAES was compared with another multiobjective MA, the random directions multiple objective genetic local search (RD-MOGLS) algorithm of Jaszkiwicz, using (1+1)-PAES as a baseline, once more. Both algorithms work well, generating results that are better than (1+1)-PAES, produced. However, on the experiments carried out, M-PAES, was found to be superior overall. The RD-MOGLS algorithm seemed capable of generating solutions over a wider range in each of the objectives on the smaller 2-objective problems, but as the size of the problem and the number of objectives were increased, M-PAES exhibited superior performance according to our statistical measures.

On the negative side, the setting of parameters in the M-PAES algorithm is difficult at present. It appears to be more sensitive to them than SPEA is. Some investigation into mechanisms for controlling the parameters would be necessary for M-PAES to become a useful general purpose optimizer.

Nonetheless, the findings indicate that further investigation into the use of memetic algorithms in multiobjective combinatorial optimization is warranted. The M-PAES algorithm only represents one instance of the new memetic algorithm framework put forward, and it is anticipated that when good local-search heuristics are available for a particular problem, algorithms based on the framework will have an advantage over standard evolutionary approaches.

Acknowledgments

The first author would like to express his gratitude to BT Labs Plc. for the continuing sponsorship of his Ph.D.

References

- [1] P. J. Bentley and J. P. Wakefield. Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, Part 5, pages 231–240. Springer Verlag London Limited, London, June 1997. (Presented at the 2nd On-line World Conference on Soft Computing in Design and Manufacturing (WSC2)).
- [2] Pedro Castro Borges and Michael Pilegaard Hansen. A basis for future successes in multiobjective combinatorial optimization. Technical Report IMM-REP-1998-8, Institute of Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, Denmark, 1998.
- [3] Carlos A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, August 1999.
- [4] P. Czyzak and A. Jaszkiwicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.
- [5] Kalyanmoy Deb. Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 8, pages 135–161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

- [6] Kalyanmoy Deb. Multi-Objective Evolutionary Algorithms: Introducing Bias Among Pareto-Optimal Solutions. KanGAL report 99002, Indian Institute of Technology, Kanpur, India, 1999.
- [7] Matthias Ehrgott and Xavier Gandibleux. An Annotated Bibliography of Multi-objective Combinatorial Optimization. Technical Report 62/2000, Fachbereich Mathematik, Universität Kaiserslautern, Kaiserslautern, Germany, 2000.
- [8] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kaufman Publishers.
- [9] Carlos M. Fonseca and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.
- [10] Carlos M. Fonseca and Peter J. Fleming. Nonlinear System Identification with Multiobjective Genetic Algorithms. In *Proceedings of the 13th World Congress of the International Federation of Automatic Control*, pages 187–192, San Francisco, California, 1996. Pergamon Press.
- [11] M. P. Fourman. Compaction of symbolic layout using genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum, 1985.
- [12] L. M. Gambardella, Eric Taillard, and Giovanni Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, F. Glover, and M. Dorigo, editors, *New Ideas in Optimization*, pages 63–76. McGraw-Hill, London, UK, 1999.
- [13] Xavier Gandibleux and Arnaud Fréville. Tabu Search Based Procedure for solving the 0/1 Multiobjective knapsack Problem: the two objective case. *Journal of Heuristics*, Accepted 1998. (to appear).
- [14] Xavier Gandibleux, Nazik Mezdaoui, and Arnaud Fréville. A tabu Search Procedure to Solve Multiobjective Combinatorial Optimization Problems. In R. Caballero and R. Steuer, editors, *Proceedings volume of MOPGP'96*, pages 291–300, Berlin, 1996. Springer-Verlag.
- [15] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
- [16] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.
- [17] M. P. Hansen. Tabu Search in Multiobjective Optimisation: MOTS. In *Proceedings of MCDM'97*, Cape Town, South Africa, January 1997.
- [18] J. W. Hartmann, V. Coverstone-Carroll, and S. N. Williams. Optimal interplanetary spacecraft trajectories via a pareto genetic algorithm. *Journal of the Astronautical Sciences*, 46(3):267–282, 1998.
- [19] Jeffrey Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, 1997.
- [20] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niche Pareto Genetic Algorithm. Technical Report IlliGAL Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.
- [21] Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.
- [22] Andrzej Jaskiewicz. Genetic local search for multiple objective combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznań University of Technology, December 1998.
- [23] J. D. Knowles and D. W. Corne. Local Search, Multiobjective Optimization and the Pareto Archived Evolution Strategy. In B. et al. McKay, editor, *Proceedings of Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems*, pages 209–216, Ashikaga, Japan, November 1999. Ashikaga Institute of Technology.
- [24] Joshua D. Knowles and David W. Corne. Assessing the Performance of the Pareto Archived Evolution Strategy. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 123–124, Orlando, Florida, July 1999.
- [25] Joshua D. Knowles and David W. Corne. The Pareto Archived Evolution Strategy: A New

- Baseline Algorithm for Multiobjective Optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105, Washington, D.C., July 1999. IEEE Service Center.
- [26] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
- [27] Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.
- [28] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley and Sons Ltd., Chichester, UK, 1990.
- [29] William Mendenhall and Robert J. Beaver. *Introduction to Probability and Statistics - 9th edition*. Duxbury Press, International Thomson Publishing, Pacific Grove, CA, 1994.
- [30] Geoffrey T. Parks and I. Miller. Selective Breeding in a Multiobjective Genetic Algorithm. In A. E. Eiben, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature — PPSN V*, pages 250–259, Amsterdam, Holland, 1998. Springer-Verlag.
- [31] Ian C. Parmee, Dragan Cvetković, Andrew H. Watson, and Christopher R. Bonham. Multiobjective Satisfaction within an Interactive Evolutionary Design Environment. *Evolutionary Computation*, 8(2):197–222, Summer 2000.
- [32] J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.
- [33] Paolo Serafini. Simulated Annealing for Multi Objective Optimization Problems. *Multiple Criteria Decision Making - Expand and Enrich the Domains of Thinking and Application*, pages 283–292, 1994.
- [34] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.
- [35] Ricardo Szmit and Amnon Barak. Evolution strategies for a parallel multi-objective genetic algorithm. In Darrell Whitley et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, pages 227–234, 2000.
- [36] Éric. D. Taillard, Luca M. Gambardella, Michel Gendreau, and Jean-Yves Potvin. Adaptive memory programming: A unified view of meta-heuristics. Technical Report IDSIA-19-98, IDSIA, Lugano, Switzerland, 1998. Also published in *EURO XVI Conference Tutorial and Workshop Reviews booklet* (semi-plenary session), Brussels, July.
- [37] E. L. Ulungu, J. Teghem, P. H. Fortemps, and D. Tuytens. MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.
- [38] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.
- [39] A. P. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *MCDM Theory and Application, Proceedings Hagen/Königswinter 1979*, Lecture Notes in Economics and Mathematical Systems 177, pages 468–486, Berlin, 1980. Springer-Verlag.
- [40] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
- [41] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriestrasse 35, CH-8092 Zurich, Switzerland, December 1999.
- [42] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.
- [43] Eckart Zitzler and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.
- [44] Eckart Zitzler and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.