Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 169 (2006) 875–897
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

The influence of the fitness evaluation method on the performance of multiobjective search algorithms

E.K. Burke, J.D. Landa Silva *

Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science and Information Technology, University of Nottingham, Nottingham NC8 1BB, UK

Accepted 13 August 2004
Available online 12 October 2004

Abstract

In this paper we are concerned with finding the Pareto optimal front or a good approximation to it. Since non-dominated solutions represent the goal in multiobjective optimisation, the dominance relation is frequently used to establish preference between solutions during the search. Recently, relaxed forms of the dominance relation have been proposed in the literature for improving the performance of multiobjective search methods. This paper investigates the influence of different fitness evaluation methods on the performance of two multiobjective methodologies when applied to a highly constrained two-objective optimisation problem. The two algorithms are: the Pareto archive evolutionary strategy and a population-based annealing algorithm. We demonstrate here, on a highly constrained problem, that the method used to evaluate the fitness of candidate solutions during the search affects the performance of both algorithms and it appears that the dominance relation is not always the best method to use.

Keywords: Multiobjective metaheuristics; Fitness evaluation; Space allocation; Dominance relation

1. Introduction

Many real-world problems have more than one objective and sometimes more than one decision-maker is involved in selecting an appropriate solution. At present, three broad typical approaches can be identified in the literature to deal with multiple objectives [30,54]:

(1) Optimising one objective at a time while imposing constraints on the other objectives.
(2) Combining all objectives into a single objective.
(3) Optimising all objectives simultaneously.

* Corresponding author.
E-mail addresses: ekb@cs.nott.ac.uk (E.K. Burke), jds@cs.nott.ac.uk (J.D. Landa Silva).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2004.08.028

Frequently, in the first two approaches, preferences for the objectives are established a priori while, in the last one, no preference information is considered or is available before the search. In terms of the number of solutions needed, it may be that only one solution is required or that a set of solutions should be presented to the decision-makers so that one of the solutions can be chosen. In the last case, this set of solutions should represent a trade-off among the different objectives. It is also commonly required that this set of solutions be as diverse as possible. Such diversity may be in terms of the solution space, the objective space or both, depending upon the problem domain.

Multiobjective methodologies have been applied to a wide range of application areas which include personnel scheduling (e.g. [12,25,31]), university timetabling (e.g. [9,15,50]), production scheduling (e.g. [4,5]), engineering optimisation (e.g. [6,55,57]), and many others. Various different methods have been applied to tackle multiobjective optimisation problems over the years. Among these, the classical methods (also called traditional methods in some of the literature) include weighting approaches, goal programming, constraint methods, the Tchebycheff method and others [7,54]. In recent years, multiobjective metaheuristics have received considerable attention. For example, adaptations of local search methods including simulated annealing and tabu search have been proposed [19,29,32,46,58]. Moreover, there is a growing interest in applying evolutionary algorithms to solve multiobjective optimisation problems [18,22,27,64]. Several surveys on the application of metaheuristics to multiobjective problems are available in the literature [17,24,41,59]. Also, there are several studies that focus on measuring and comparing the performance of different algorithms for multiobjective optimisation [51,56,60,63,65]. In fact, a considerable amount of the available literature on the application of multiobjective metaheuristics concentrates on the competition between different approaches to produce the best results for a set of test problems. As noted by Purshouse and Fleming, "*predictably, the existence of alternative algorithms has instigated a degree of algorithmic competition into the EMO (Evolutionary Multi-objective) arena*" [51, p. 2]. The results obtained from these comparisons are not always very conclusive because the performance of each algorithm is very dependent on the parameter settings. For example, Purshouse and Fleming showed that by tuning their multiobjective genetic algorithm, they could achieve results that were much better compared against other evolutionary multiobjective algorithms that had been shown to outperform their algorithm in previous studies carried out by other researchers (see [51] for full details). Although the competition between different multiobjective algorithms is an interesting research direction, in order to understand the functioning of metaheuristics it is also important to study their performance and identify those features that make them succeed or fail in different problem domains.

In this paper, we contend that one of the issues that should be investigated is the effect of the method used for evaluating the fitness of candidate solutions on the performance of multiobjective algorithms. The evaluation of solutions in optimisation problems can be considered as a separate process from the algorithm itself. However, in this paper we consider the solution evaluation method as an integral part of the multiobjective metaheuristic. We investigate how different fitness evaluation methods affect the performance of two multiobjective approaches. Two of these evaluation methods are (1) aggregate the objectives and directly compare the obtained scalar value or (2) compare their vectors containing all the objective values using the dominance relation (see Section 3.2). It is noted that some recent multiobjective algorithms use aggregation of objectives to assign fitness to solutions while others apply the dominance relation. It has been claimed by some authors that the use of aggregating functions to guide the search in multiobjective optimisation is not adequate mainly because of the difficulty of setting appropriate weights for each objective and instead they suggest that the dominance relation should be used [18,22]. Other researchers suggest that the use of the dominance relation is not well-suited for approaches that employ local search [32]. Furthermore, Kokolo et al. discuss whether non-dominated solutions represent good approximations to optimality or not [39], while

Knowles et al. suggest that by the 'multi-objectivization' of single objective optimisation problems the number of local optima can be reduced in an attempt to provide more freedom to explore by the searching mechanism [36]. Also, relaxed forms of the dominance relation have been proposed as a means of improving the performance of some multiobjective methods [13,39,43,48]. Although arguments have been put forward in favor of either aggregating functions or the dominance relation for multiobjective optimisation, there has been no previous study to investigate the influence that different fitness evaluation methods may have on the performance of multiobjective approaches. This paper attempts to present such an investigation in order to have an insight into the circumstances in which one or another method for fitness evaluation is more or less appropriate. Results are reported for a highly constrained combinatorial optimisation problem with two objectives: the space allocation problem which is a variant of the class of knapsack problems with assignment constraints [21,34]. Three ways of dealing with the two objectives are compared: an aggregation of the objectives, the dominance relation and a relaxed form of this dominance relation. The results presented here show that although the aim is to generate sets of non-dominated solutions, the use of methods such as the aggregation of objectives or relaxed forms of dominance can improve the performance of multiobjective methods. It is observed that this is particularly true for the highly constrained problem being studied in this paper.

It should be noted that the aim of this paper is to investigate the influence of the evaluation method on the performance of the given algorithms on the two-objective space allocation problem. We aim to identify any relationships between the search strategy used and the problem domain in order to better understand the functioning of metaheuristic approaches for multiobjective optimisation. Section 2 provides details of the two-objective space allocation problem and the test instances used here. Section 3 describes the three fitness evaluation methods compared in this paper while Section 4 describes the two algorithms used for this purpose. Experiments are described and results presented in Section 5 and further discussion and final remarks are given in Section 6.

2. The two-objective space allocation problem

2.1. Description and formulation

The space allocation problem is used for the experiments in this paper. In this problem, the aim is to allocate a set of n entities into a set of m areas of space in such a way that all the existing hard constraints are satisfied and two minimisation objectives are accomplished. These objectives are (1) to minimise the amount of space misused and, (2) to minimise the violation of soft constraints. Examples of the type of (hard or soft) constraints that can occur in this problem are:

Be located in—a particular entity can be allocated only in a subset of the available areas.
Be adjacent to—entities should be allocated in adjacent areas.
Be away from—entities should be allocated away from each other or away from certain areas.
Be together with—entities should be allocated in the same area.
Not sharing—entities should not share the allocated area.
Be grouped with—two or more entities should be allocated as a group using adjacent areas, shared areas or areas close to each other.

This problem is formulated as follows:

$m =$ number of available areas of space
$n =$ number of entities to allocate
$h =$ number of hard constraints where $z(k)$ is assigned value, "*true*"
$s =$ number of soft constraints where $z(r)$ is assigned value, "*true*"
$c(i) =$ capacity or size of area i
$w(j) =$ space requirement of entity j
$x(i, j) = 1$ if entity j is assigned to area i, 0 otherwise

Minimise:

$$F(x) = \mathrm{F1}(x) + \mathrm{F2}(x) \tag{1}$$

subject to:

$$\sum_{j=1}^{m} x(i,j) = 1 \quad \text{for } j = 1,2,\ldots,n, \tag{2}$$

$$z(k) = \text{true} \quad \text{for } k = 1,2,\ldots,h, \tag{3}$$

where

$$\text{F1}(x) = \sum_{i=1}^{m} (\text{WP}(i) + \text{OP}(i)), \tag{4}$$

$$\text{F2}(x) = \sum_{j=1}^{s} \text{SCP}(r). \tag{5}$$

For a solution x, the amount of space misuse and the penalty due to the violation of soft constraints are given by F1(x) and F2(x). The penalties due to the space wasted and the space over used in the ith area are given by WP(i) and OP(i) respectively, while SCP(r) is the penalty applied when the rth soft constraint is not satisfied.

For the ith area, there is space wastage if:

$$c(i) > \sum_{j=1}^{n} (w(j) \cdot x(i,j)), \tag{6}$$

and the penalty is given by

$$\text{WP}(i) = c(i) - \sum_{j=1}^{n} (w(j) \cdot x(i,j)). \tag{7}$$

For the ith area, there is space over used if:

$$c(i) < \sum_{j=1}^{n} (w(j) \cdot x(i,j)), \tag{8}$$

and the penalty is given by

$$\text{OP}(i) = 2 \cdot \left(\sum_{j=1}^{n} w(j) \cdot x(i,j) - c(i) \right). \tag{9}$$

The problem formulated above is motivated by the problem of allocating office space in academic institutions and it can be seen as a variant of the class of knapsack problems [44]. Specifically, this is a variant of the bin-packing problem with varying bin capacities in which additional constraints limit the allocation of entities to bins [34]. Other constrained variants of knapsack type problems have also been investigated in the literature [21]. Moreover, space allocation problems similar to the one formulated above occur in forest and geographic management. For example, an important resource allocation problem in geographic research is the goal of optimising the allocation of land use to each of the cells forming an area of land (e.g. [2,45]. In forest management, the maximal covering location problem consists of selecting reserve areas that maximise the coverage of species with the aim of having enough forest areas for the conservation of all protected species (e.g. [16]). The optimal utilisation of physical space is a concern in many other scenarios that range from industrial and commercial environments (e.g. [8,28,62]) to computer systems [53]. For a more detailed description of the space allocation problem considered in this paper see [14,40]. A brief discussion about why this problem is considered as a two-objective optimisation problem in this paper is provided in the next section.

2.2. The conflicting objectives

Using the dominance relation when dealing with a multiobjective optimisation problem makes sense only if the objectives are partially or totally conflicting [52]. If the objectives are uncorrelated or reinforce each other, it is often adequate to combine all of them into a single scalar value and approach the problem as a single-objective one. More than two objectives could be considered in the space allocation problem described above. In fact, it can be argued that this problem is an 8-objective optimisation problem, i.e. the satisfaction of each of the six types of constraints listed in Section 2.1 plus the minimisation of space wastage and space overuse.

Sets of experiments were carried out in order to investigate the conflicting nature of the objectives in the problem considered here (full details are presented in [40]). For each test problem, eight sets of ten runs were executed using a simple iterative best improvement algorithm. In each set of ten runs, one of the eight objectives was subject to the search process, i.e. only the value of that objective was used to assign fitness to solutions while the values of the other seven objectives were traced to observe their response. Since, in each set of runs

one of the objectives is subject to minimisation, it is possible to calculate the correlation between that objective and the others. A positive correlation is an indication that the two objectives are reinforcing each other or moving together, i.e. improvements in one objective are associated with improvements in the other. A negative correlation is an indication of the conflict between two objectives, i.e. improvements in one objective are associated to detriments in the other. A correlation value close to zero is an indication of the two objectives being unrelated or not affecting each other. The correlation values obtained in each set of ten runs were averaged for each pair of objectives.

The results obtained in our experiments showed that in general: (1) the minimisation of space misuse (wastage and overuse) was in conflict with the satisfaction of soft constraints and that, (2) the satisfaction of one type of soft constraint was not in conflict with the satisfaction of another type of soft constraint. These results allowed us to conclude that, at least on the test instances used here, not all the eight objectives are conflicting. Therefore, we grouped the eight objectives into two conflicting objectives: the *minimisation of space misuse* and the *minimisation of soft constraints violation*. It should be noted that the conflicting nature of the objectives will depend very much on the constraints that exist in each particular problem instance and hence, an analysis similar to the one described here would be appropriate in order to illustrate the multiobjective nature of the problem in hand.

2.3. Measuring diversity

In some problems, diversity in the solution space is more important than diversity in the objective space. For example, it may be that after obtaining a set of diverse solutions with respect to the objective space, the actual solution structures are very similar. Therefore, it is important to establish which type of diversity is aimed at or whether both are equally important in the problem domain. In the context of the space allocation problem described above, diversity in the solution space is very important since it is desirable to obtain a set of solutions representing different allocations. In this paper, the diversity in the solution space is measured by representing each solution by an n-dimensional vector $Y = [y_1, y_2, \ldots, y_n]$ where y_j is the area of space to which the jth entity has been allocated, i.e. each y_j can have a value between 1 and m. The similarity between solutions is measured by comparing their corresponding vectors. Two solutions are completely non-similar if they differ in all the n values of their vectors. The non-similarity in the solution space for a population of size p is calculated as follows:

$$V_p = \frac{\sum_{j=1}^{n} \left(\frac{D_j - 1}{p-1} \right)}{n} \cdot 100. \tag{10}$$

D_j is the number of different values in the jth position for all p vectors. V_p denotes the percentage of non-similarity for the population and it is used as a measure of diversity here. For a population of p allocations, V_p is an indication of how similar the p n-dimensional vectors are. For example, if $V_p = 100\%$ it means that the jth entity for $j = 1, \ldots, n$ is allocated to a different area of space in each of the p solutions, i.e. all the solutions are totally different. If $V_p = 0\%$ it means that the jth entity for $j = 1, \ldots, n$ is allocated to the same area of space in each of the p solutions, i.e. all the solutions are identical.

2.4. Test instances

The three instances of the space allocation problem summarised in Table 1 were used in the experiments described in this paper. These instances have been taken from real data sets from two British Universities. The instance **nott1** is the largest and most constrained one with respect to hard and soft constraints. The **nott1b** instance is a variant of the previous problem which also contains a considerable number of constraints but which has less areas of space available. This makes the efficient utilisation of space more difficult to achieve. The **trent1** instance contains many hard constraints of the **not sharing** type only. More details of these test problems can be found in [40] and at `http://www.cs.nott.ac.uk/~jds/research/spacedata.html`.

Table 1
Test instances of the two-objective space allocation problem used in the experiments described in this paper (n is the number of entities and m is the number of areas of space)

Constraints	nott1		nott1b		trent1	
	$n = 158$	$m = 131$	$n = 104$	$m = 77$	$n = 151$	$m = 73$
	Hard	Soft	Hard	Soft	Hard	Soft
Not sharing	100	58	46	58	80	71
Be allocated in	0	35	0	9	0	19
Be adjacent to	5	15	4	10	0	5
Be away from	6	14	1	2	0	0
Be together with	0	20	0	20	0	36
Be grouped with	0	10	0	9	0	0
Total	263		159		211	

3. Assigning fitness to solutions in multiobjective optimisation

We need to establish the way in which the various objectives will be handled in order to assign fitness to candidate solutions during the search and therefore decide which solutions will survive and which ones will be discarded. Three ways of doing this are investigated and studied here: an aggregating function, the dominance relation and a relaxed form of the dominance relation. These methods are described below for the two-objective problem studied here but they can be extended in the obvious way when more than two objectives exist.

3.1. Aggregation of objectives

With aggregating functions, the two objective values are combined into a single scalar value as shown below:

$$F(x) = w_1 \cdot \mathrm{F1}(x) + w_2 \cdot \mathrm{F2}(x). \quad (11)$$

In this approach, weights (w_1 and w_2) can be incorporated in order to establish preference between the objectives. Selecting an adequate set of weights that reflects this preference demands prior knowledge of the decision-makers' criteria. This is not always possible and also the preferences may vary which makes this selection process more complicated. Here, both weights are set to 0.5, indicating no particular preference for one of the objectives. With the aggregating function, the solution with the smaller value of $F(x)$ is preferred or considered better.

3.2. Pareto dominance

In Pareto dominance, the solution fitness is represented using a two-dimensional vector containing the values of $\mathrm{F1}(x)$ and $\mathrm{F2}(x)$. A solution x with a fitness vector V is preferred over the solution x' with fitness vector U if and only if the vector V dominates the vector U. Two types of dominance are:

Strong dominance. A vector $V = [v_1, v_2]$ *strongly dominates* the vector $U = [u_1, u_2]$ if V is better than U in both objectives.
Weak dominance. A vector $V = [v_1, v_2]$ *weakly dominates* the vector $U = [u_1, u_2]$ if V is better than U in at least one of the objectives and is as good as U in the other one.

Dasgupta et al. refer to these two types of dominance as *strict dominance* and *loose dominance* respectively [20]. In both types of dominance, if neither V dominates U nor U dominates V, then both vectors are said to be *incomparable* and no solution is clearly preferred over the other. In this paper, strong dominance is used to distinguish a dominated solution from a non-dominated one, i.e. only solutions that are *strongly dominated* are rejected. This means that solutions that are *weakly dominated* are also considered because of the interest in obtaining diversity in the solution space. In the following, *strong dominance* is referred to as dominance. A solution x is said to be *non-dominated* with respect to a set of solutions S if there

is no solution in S that dominates x. The *Pareto-optimal front* in multiobjective optimisation is the set of all non-dominated solutions in the whole solution space [18,22,54]. When there is no knowledge of the localization of the Pareto-optimal set, the obtained set is referred to as the non-dominated set found. In the instances of the test problem used in this paper, there is no knowledge of the localization or shape of the Pareto-optimal front.

3.3. Relaxed Pareto dominance

Relaxed forms of Pareto dominance have been proposed by researchers as a means to improve the performance of multiobjective optimisers. For example, Kokolo et al. suggested the use of α-dominance for dealing with what they call dominance resistant solutions, i.e. solutions that are fairly inferior quantitatively but solutions that dominate them are scarcely found [39]. This variant of dominance establishes lower and upper bounds for tradeoffs between the objectives. In α-dominance small detriments in one of the objectives are considered acceptable if this leads to an attractive improvement in the other objective.

Fig. 1 illustrates the concept of α-dominance for a two-objective minimisation problem and it also compares it to the other two evaluation methods considered here: dominance and aggregation of objectives. Solutions in regions B, C and D all α-dominate solution x. In region C for example, β_{uv} represents the maximum detriment permitted in objective u given the minimum improvement β_{vu} in objective v. In region D, β_{vu} and γ_{uv} are defined in a similar way. Solution x is dominated by all solutions in region B while solution x dominates all solutions in region A. When using the aggregation of objective values, a line that splits the objective space into two regions is drawn. All the solutions above the line are considered worse than x and all solutions below the line are considered better that x. A line at 45 degrees of inclination is used here according to equal weight values for the two objectives but different slopes will reflect different preferences. In α-dominance, given an optimisation problem with k objectives, the relation between β_{vu} and γ_{uv} for each pair of

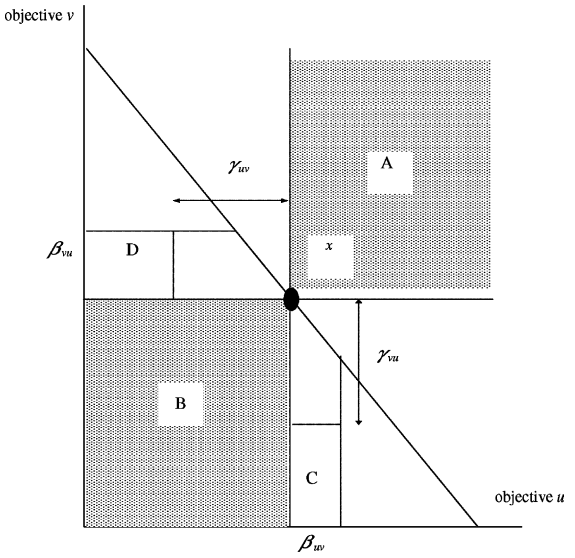

Fig. 1. Aggregating function, dominance relation and α-dominance.

objectives $u \neq v$ represents the relation between the detriment permitted in the objective v and the improvement obtained in the objective u. For the formal definition of α-dominance see [39]. A similar form of relaxed dominance called ϵ-dominance was recently suggested by Laumanns et.al. to implement better archiving strategies that allows us to overcome the difficulty of multiobjective evolutionary algorithms to converge towards the Pareto-optimal set and maintain a wide diversity in the population at the same time [43]. The ϵ-dominance has also been proposed to implement specialised archiving mechanisms for multiobjective search algorithms [33,48].

In some sense, the relaxed forms of dominance (α-dominance and ϵ-dominance) are similar to establishing preferences among the objectives using weights in an aggregating function. In both cases, a detriment in one or more of the objectives is permitted in an attempt to widen the search by accepting not only dominating solutions. The different perspectives in viewing candidate solutions affects the way in which surviving solutions are selected. An algorithm may find it difficult to discover feasible solutions that dominate the current one(s). This is particularly true in highly constrained combinatorial optimisation problems like

the one considered here. Then, by accepting α-dominating (or ϵ-dominating) solutions or solutions for which the aggregated value is better, it is possible to provide the algorithm with a wider view of the potential ways to approach the Pareto-optimal front.

The relaxed form of dominance implemented in this paper follows the same principle as α-dominance and ϵ-dominance but it is slightly different. Let x be the current solution and x' be a candidate solution with fitness vectors given by $V=(v_1, v_2)$ and $U=(u_1, u_2)$ respectively. If the first objective in the candidate solution is better than in the current solution, i.e. $u_1 < v_1$ then the corresponding *gain* or improvement proportion is calculated as $gain=(v_1-u_1)/v_1$. The candidate solution x' is considered to be better than the current solution x if the detriment proportion in the other objective is at most *gain*, i.e. if $u_2 < v_2 \times (1+gain)$. This calculation is modified in the obvious way in the case $u_2 < v_2$. That is, it can be seen that the relaxed dominance is equivalent to the following relation: [1]

$$\frac{u_1}{v_1}+\frac{u_2}{v_2}<2. \tag{12}$$

4. The two multiobjective approaches tested

4.1. Justification

The two algorithms used in this investigation are: an amended version of the population-based hybrid annealing algorithm previously proposed by the authors in [11] and the $(1+1)$-Pareto archived evolutionary strategy proposed by Knowles and Corne [35]. The first one is a hybrid approach that has been developed as a result of the previous research carried out by the authors into investigating metaheuristic approaches to the space allocation problem. When this algorithm is applied to the two-objective space allocation problem, it is observed that better non-dominated fronts are produced if the aggregation of objectives or the relaxed concept of dominance is used instead of the dominance relation to assign fitness to solutions during the search. In order to investigate whether this behavior is due to the search strategy used by the algorithm or due to the problem domain, a multiobjective search algorithm that has been well-studied in the specialised literature was also implemented and tested. The $(1+1)$-Pareto archived evolutionary strategy is a modern multiobjective optimisation technique that is simple to implement, has been tested across a range of problems and is considered to be competitive with other modern multiobjective evolutionary algorithms [37,56].

The two approaches above are alike in the sense that both evolve solutions based on self-adaptation, i.e. the current solution is modified by mutation or local search and no recombination is used. Algorithms like these are often referred to as trajectory-based methods. The population-based hybrid annealing algorithm has been tested on other instances of the space allocation problem while the $(1+1)$-Pareto archived evolutionary strategy is an approach that has been applied to many other multiobjective optimisation problems but not to the one tackled in this paper so the effect of the fitness evaluation method can be further investigated without being biased by the algorithm design. Also, the authors's previous experience is that the recombination of solutions in this highly constrained problem almost always produces infeasible solutions [10,40]. Of course this only means that good crossover operators or repairing heuristics need to be designed and therefore the applicability of recombination-based multiobjective evolutionary algorithms to this problem can be considered in the future. Then, since both algorithms use local search as the main operator to generate candidate solutions, they can be readily applied to the highly constrained two-objective space allocation problem. Brief descriptions of these two algorithms are given below.

4.2. The population-based hybrid annealing algorithm

The approach described next is based on the simulated annealing algorithm but it also incorpo-

[1] Thanks to the anonymous referee for suggesting this equation.

rates elements of other metaheuristics such as tabu search and evolutionary algorithms [47]. Simulated annealing is a local search approach that attempts to improve the current solution and in which the acceptance of worse solutions is controlled by an annealing schedule [1]. Initially, a high temperature (corresponding to a high acceptance probability) is set. As the search continues, this temperature (and hence the acceptance probability) is reduced gradually. When the temperature equals zero (corresponding to zero acceptance probability), only better solutions are accepted.

The hybrid approach described here is presented in Pseudocode 1. A previous version of this algorithm was presented in [11] and the modification incorporated here is in the co-operative local search mechanism described later in Section 4.2.2. The algorithm evolves a population of individuals using the local search heuristic H_{LS}, a mutation operator and a common annealing schedule for the whole population. The heuristic H_{LS} incorporates a mechanism to induce co-operation among the individuals in the population (see sections below). The comparison between the current solution x_c and a candidate solution $x_{c'}$ in steps 6.2 and 6.3 of Pseudocode 1 to establish which one is better depends on the fitness evaluation method used, which is the main topic of discussion in this paper. In a typical simulated annealing implementation, the initial temperature is usually set to a value which enables a considerable proportion of solutions to be accepted in the first iterations even if these solutions are non-improving [1]. However, note that in the algorithm presented here the initial acceptance probability ρ is set to 0. That is, the algorithm behaves as a pure iterative improvement algorithm for η iterations (step 4) and then the acceptance probability is set to 1 (step 5). After that, this probability ρ is decreased at each interval of η iterations (step 6.4). Then, re-heating occurs when this probability is raised again from 0 (or near 0) to 1 (step 6.3b) and it takes place if after η iterations with acceptance probability equal to zero, no candidate solution x'_c has been found that is better than the current solution x_c. Due to this probability of accepting non-improving solutions (step 6.3a), the solutions in P_C can eventually be worsened as the algorithm evolves. Therefore, P_B archives the best solution found by each of the individuals in the population while P_C maintains the current solution for each individual. Note that although co-operation between individuals is implemented (described in Section 4.2.2), P_B stores the best solution found by each individual regardless of the rest of the population and there is no guarantee that all solutions in P_B be non-dominated. Therefore, an additional archive P_{ND} of the non-dominated solutions found during the search is maintained. Note that the different evaluation methods (aggregation of objectives, dominance and relaxed dominance) only refer to steps 6.2 and 6.3 of the algorithm in Pseudocode 1. The set P_{ND} is always updated using the dominance relation described in Section 3.2. The algorithm parameters used in the experiments described in this paper are given in Section 5.1.

Pseudocode 1. The Population-based Hybrid Annealing Algorithm.

1. Generate the initial current population of solutions P_C
2. Copy P_C to the population of best solutions P_B
3. Set acceptance probability $\rho \leftarrow 0$, cooling factor $0 < \alpha < 1$, decrement step η, re-heating step φ, and re-heating counter $\tau \leftarrow 0$ (η, φ and τ are a number of iterations)
4. For η iterations, apply the local search heuristic H_{LS} to each individual in P_C
5. Set $\rho \leftarrow 1$
6. For each X_C in P_C and its corresponding X_B in P_B,
   - 6.1. Generate a candidate solution X'_C using H_{LS}
   - 6.2. If X'_C is better than X_C, then $X_C \leftarrow X'_C$
     - (a) If X'_C is better than X_B, then $X_B \leftarrow X'_C$
   - 6.3. If X'_C is not better than X_C, then
     - (a) if $\rho > 0$ and a random generated number in the normal distri-

bution [0, 1] is smaller than ρ, then $X_C \leftarrow X'_C$

(b) if $\rho \approx 0$ (in our setting, if $p < 0.0001$), then $\tau \leftarrow \tau + 1$ and if $\tau \geqslant \varphi$, then $\rho \leftarrow 1$ and $\tau = 0$

6.4. If (*iterations* mod η) = 0, then $\rho \leftarrow \alpha \cdot \gamma$

6.5. If X'_C is non-dominated with respect to P_{ND}, update P_{ND}

7. Go to Step 8 if no individual has achieved further improvement for η iterations, otherwise go to Step 6
8. Apply the mutation operator to each individual in P_C
9. If stopping criterion has not been satisfied, go to Step 6

4.2.1. The local search heuristic

The heuristic H_{LS} shown in Pseudocode 2 employs four neighbourhood structures or moves to carry out local search: *allocate*, *relocate*, *swap* and *interchange*. These structures are described below together with their respective size in terms of the number of entities to allocate (n) and the number of available areas of space (m).

- Allocate an unallocated entity to an area. $|N_A| = n \cdot m$.
- Relocate an entity to a different area. $|N_R| = n \cdot (m - 1)$.
- Swap the areas between two entities. $|N_S| = n \cdot (n - 1)/2$.
- Interchange the allocated entities between two areas. $|N_I| = m \cdot (m - 1)/2$.

Pseudocode 2. The local search heuristic H_{LS}. This heuristic selects the type of move or neighbourhood structure and then explores the selected neighbourhood to find a move. The number of *attempts* refers to the number of previously consecutive failed (i.e. no accepted) moves. The value *maximum attempts* refers to the maximum number of failed attempts permitted.

1. If all n entities are allocated then
   1.1. Select the move type at random: *relocate*, *swap* or *interchange*
2. If not all n entities are allocated then
   2.1. If the number of *attempts* > *maximum attempts* permitted then
      2.1.1. If the previous selected move type was *allocate* then select a move between *relocate*, *swap* and *interchange* at random
      2.1.2. If the previous selected move type was not *allocate* then select the *allocate* move
      2.1.3. Set the number of *attempts* equal to zero.
   2.2. If the number of *attempts* < *maximum attempts* permitted then
      2.2.1. If the previous selected move type was not *allocate* then select a move between *relocate*, *swap* and *interchange* at random
3. Explore the neighbourhood and return a move of the selected type

To select the type of move, the heuristic H_{LS} takes into account the current state of the allocation and the history of success in applying each type of move. The type that is undertaken in each iteration is determined by the number of allocated entities and the number of prior failed attempts to find a feasible move of the selected type. That is, if all entities are allocated in the current solution, only the moves *relocate, swap* and *interchange* are explored. In the case that not all entities are allocated, a certain number of maximum attempts normally set to $n/10$ (decided by preliminary experimentation) is given to either of the three move types. For example, suppose that in the current solution there are still 5 unallocated entities from a total of 200 in the allocation problem. If after 20 failed attempts, none of these entities have been successfully allocated, the algorithm examines the feasibility of modifying the solution using the *relocate, swap* and *interchange* moves up to a maximum of 20 failed attempts. The number of failed modification attempts is set to zero when a move has been accepted by the driving metaheuristic.

4.2.2. Co-operative local search

The heuristic H_{LS} selects the best from a subset of candidate solutions in the neighborhood of the current solution. Since a set of individuals is evolved, memory structures are implemented in order to take advantage of the collective searching process. Two matrices M_T and M_A of size $n \times m$ are used and in both of them the cell (j, i) corresponds to the allocation of the jth entity to the ith area, for $j = 1, \ldots, n$ and $i = 1, \ldots, m$. The matrix M_T stores those (entity, area) pairs that will be considered as tabu for a number of iterations while the matrix M_A stores those (entity, area) pairs that will be considered attractive during the search. The tabu matrix M_T is updated each time a move suggested by the heuristic H_{LS} produces a detriment in the fitness of the current solution while the attractive matrix M_A is updated each time the move produces an improvement. Updating a cell in M_T refers to setting the value in the cell to the value of the *current iteration* + *tenure* so that a move involving the (entity, area) pair corresponding to that cell is set as tabu for *tenure* number of iterations. Preliminary experiments showed that a *tenure* value of around n and which is kept constant throughout the search produced good results. Updating a cell M_A refers to incrementing the value in the cell in one unit, i.e. $M_A(j, i) = M_A(j, i) + 1$. In each type of move, the cells that are updated are the ones corresponding to the (entity, area) pairs after implementing the move. For example, if the 6th entity is relocated from the 2nd to the 4th area, the cell $M_A(6, 4)$ is incremented by 1 if the move produced a better solution but if the move generated an inferior solution, the cell M_T $(6, 4)$ is set to *current iteration* + *tenure*.

The tabu matrix acts as the short-term memory component while the attractive matrix acts as the long-term memory component. Since both matrices store (entity, area) pairs, this mechanism can be regarded as a way of memorising parts of allocations or genes that come from bad solutions (M_T) or good solutions (M_A). The two matrices are shared by all individuals in the population as a mechanism for interchanging local search information. Within the local search heuristic H_{LS} the moves in M_T are avoided while the moves in M_A are used when no suitable move can be found for an individual. The local search heuristic H_{LS} used in the population-based hybrid annealing algorithm also searches until a feasible solution is found. Again, previous work on this problem showed that using various neighbourhood structures works well [10,40].

4.2.3. The mutation operator

The mutation operator in this algorithm (step 8 in Pseudocode 1) disturbs the solutions in a controlled way in order to explore other areas of the solution space. In the context of the space allocation problem, this disruption consists of removing from their assigned area, those allocated entities that contribute the most to the total penalty. This operation releases the space assigned to those entities so that new possibilities of allocating them can be explored. A maximum number of entities to be unallocated is determined according to the size of the problem instance. The allocated entities are sorted in decreasing order of their associated penalty, i.e. the violation degree of the soft constraints associated to each of them. Then, starting from the most penalised one, entities are unallocated up to the maximum specified. Once the current allocation is disrupted in this way, the algorithm reallocates the unallocated entities using the heuristic H_{LS} until all entities are allocated again. The purpose of this heavy mutation operator is to modify the current allocation after the algorithm gets stuck but this modification is directed so that only bad parts of the solution (penalised entities) are disturbed.

4.3. The Pareto archive evolutionary strategy

Several variants of the Pareto archived evolutionary strategy have been proposed but this paper refers to the (1 + 1)-Pareto archived evolutionary strategy [35]. The pseudocode for this algorithm is given in Pseudocode 3.

Pseudocode 3. The (1 + 1) Pareto archived evolutionary algorithm. This algorithm was proposed by Knowles and Corne [35]. For each of the k objectives in a problem, the space is divided into l bisections. That is, the objective space is divided

into $2^{l \times k}$ regions. The number of solutions in each region is recorded as an indication of how crowded the region for each solution is.

1. Generate initial random solution x_c and add it to the archive
2. Repeat until a termination criterion has been reached
   - Mutate x_c to produce x'_c and evaluate x'_c
   - If x_c dominates x'_c then discard
   - Else if x'_c dominates x_c, $x_c \leftarrow x'_c$ and add x'_c to the archive
     - Else if x'_c is dominated by any member of the archive, then discard x'_c
       * Else apply test (x_c, x'_c, archive) to determine which becomes the new current solution and whether to add x'_c to the archive

This algorithm starts with one initial solution x_c and in each iteration, one candidate solution x'_c is generated by means of mutations. For the problem domain considered here, when a mutated solution is infeasible, successive mutations are tried until a feasible solution is generated. This is a very fast operation and it worked well in this implementation. An external archive (of limited size) is maintained to collect non-dominated solutions.

Pseudocode 4. Subprocedure test (x_c, x'_c, archive).

If the archive is not full then
- Add x'_c to the archive
- If x'_c is in a less crowded region of the archive than x_c, accept x'_c as the new current solution
- Else maintain x_c as the current solution

Else
- If x'_c is in a less crowded region of the archive than x for some member x of the archive
  - Add x'_c to the archive, and remove a member of the archive from the most crowded region
  - If x'_c is in a less crowded region of the archive than x_c, accept x'_c as the new current solution
  - Else maintain x_c as the current solution
- Else
  - If x'_c is in a less crowded region of the archive than x_c, accept x'_c as the new current solution
  - Else maintain x_c as the current solution

An adaptive grid that divides the objective space is used to evaluate how crowded the region in which each solution lies is. The candidate solution is discarded if it is dominated by the current solution or any other solution in the external archive. The candidate solution is added to the archive and becomes the new current solution if it dominates the old current solution. If none of them dominates the other, the decision on which solution becomes the current solution and whether to add or not the candidate solution to the archive is made using the *Subprocedure test* (x_c, x'_c, archive) shown in Pseudocode 4.

5. Experiments and results

5.1. Experimental settings

Solutions are initialised with the following procedure: one unallocated entity is selected at random, then a free area is also selected at random to allocate the entity. If no free area is available, another area is selected at random from those that are already occupied by another entity or entities regardless of the amount of space available. This process continues until all entities are allocated. Next, the solution is fixed to satisfy the hard constraints that are violated at that point. This fixing procedure moves entities to areas so that the hard constraints are satisfied regardless of the space utilisation and soft constraint objective values (F1 and F2).

For each test instance and each fitness evaluation method (aggregation of objectives, dominance

and relaxed dominance) 10 repetitions of the experiments (which we describe next) were executed. An initial population of size 20 was generated as described above. The population-based hybrid annealing algorithm was executed for a number of maximum solutions evaluations denoted by *eval*. Since the Pareto archived evolutionary strategy evolves a single solution, one run of the algorithm corresponds to 20 executions for *eval*/20 solution evaluations, one with each of the 20 initial solutions. That is, the same initial population was used in each set of runs comparing the three evaluation methods in the two algorithms, i.e. 10 different populations were generated and in total 90 runs were executed for each algorithm. Below, we discuss the offline and the online results obtained in these experiments. The offline set of non-dominated solutions for each algorithm is obtained after merging all the sets produced in each of the 10 experiment repetitions and filtering this set to eliminate all dominated solutions. An online set of non-dominated solutions is obtained by each algorithm in each of the 10 experiment repetitions. That is, for each algorithm there are ten online sets and one offline set of non-dominated solutions.

For the population-based hybrid annealing algorithm, the parameters were set as follows: $|P_C| = |P_B| = 20$, $\alpha = 0.95$, $\eta = n$ and $\varphi = 10 \cdot n$. The number of maximum solution evaluations *eval* was set to 100,000, 80,000 and 50,000 for the **nott1**, **nott1b**, and **trent1** test instances respectively. The number of non-dominated solutions in the external archive P_{ND} was limited to 30 in both algorithms although in some cases fewer solutions were obtained in the final set. In the rest of this paper, the population-based hybrid annealing algorithm and the Pareto archived evolutionary strategy are referred to as PBAA and PAES respectively for simplicity. The above experiments were carried out on a PC with a 700 MHz processor and 132MB of RAM running under the Windows 2000 operating system. The algorithms were coded using MS Visual C++ version 6.0. In these conditions, the execution time for each run was similar for the two algorithms (PBAA and PAES). The times for each test instance were: **nott1** around 120 seconds, **nott1b** around 70 seconds and **trent1** around 135 seconds.

5.2. The offline non-dominated sets

For each set of 10 runs corresponding to the same triplet (algorithm, problem instance, fitness evaluation method) the offline non-dominated sets were collected and these are presented in Figs. 2–4. It is observed from Fig. 2 that for the **nott1** problem, the non-dominated sets obtained with both algorithms using the relaxed dominance and the aggregating function are better than those sets produced using the standard dominance relation. For both algorithms, the relaxed dominance clearly produces better results than the dominance relation. Also for both algorithms, a considerable section of the front obtained using the relaxed dominance is dominated by the front obtained using the aggregating function with the exception of a few solutions at the top end of these fronts. That is, using the aggregating function seems to benefit the performance of the algorithms in finding more solutions with low violation of soft constraints (small values of F2) but none of the solutions obtained have values of space misuse (F1) as low as some of the solutions obtained using the relaxed dominance relation.

For the problem **nott1b**, Fig. 3 shows that the non-dominated sets obtained using the standard dominance and the relaxed dominance are comparable for the two algorithms. That is, none of these two fitness evaluation methods appears to clearly outperform the other. With PBAA some of the solutions obtained using dominance have better space utilisation while with the PAES many solutions obtained using relaxed dominance are better with respect to the satisfaction of soft constraints. It is noticeable that for both algorithms, none of the solutions obtained using the aggregating function is dominated by solutions produced with the other two fitness evaluation methods. However, as in the **nott1** problem, using the aggregating function produces solutions that excel with respect to F2 but solutions with very low values of F1 are not found. Fig. 4 shows that for the **trent1** problem, the comparison between the non-dominated sets obtained using the standard dominance and the aggregating function is very tight. In the case of PBAA the aggregating method outperforms the dominance relation with respect to

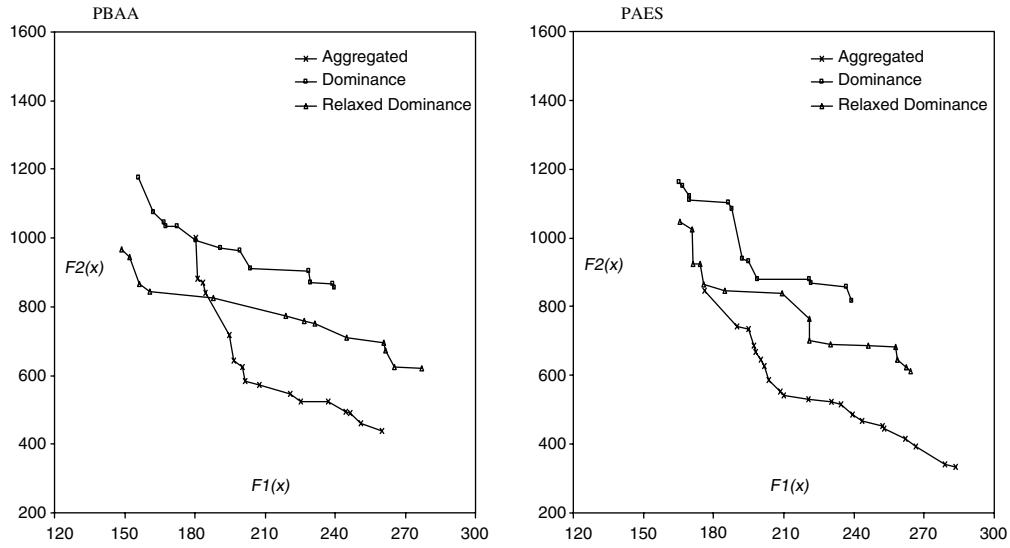

Fig. 2. Offline non-dominated solutions obtained by the two algorithms with each evaluation method for the test instance **nott1**.

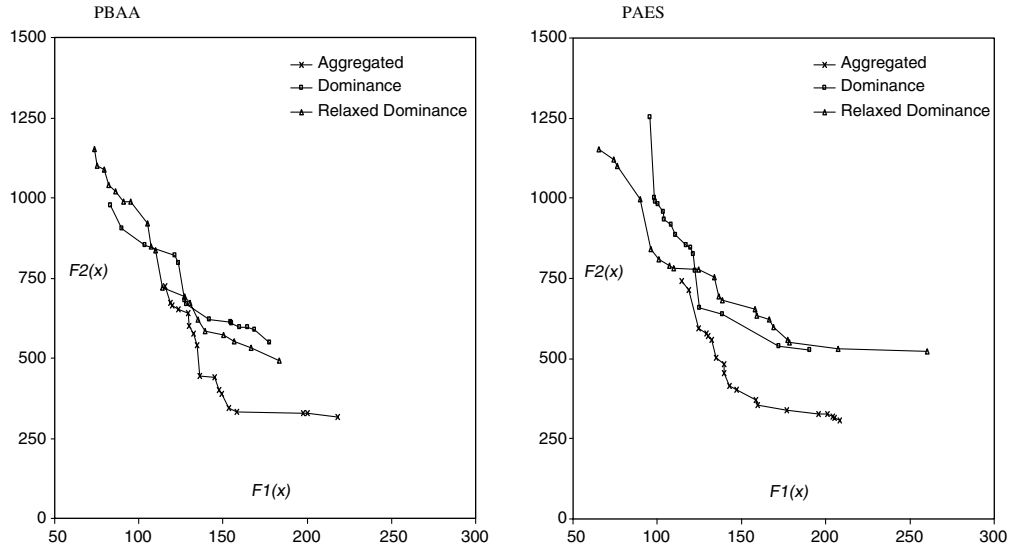

Fig. 3. Offline non-dominated solutions obtained by the two algorithms with each evaluation method for the test instance **nott1b**.

the solutions in the bottom half of the front, i.e. solutions with low values of soft constraint violation (F1). But in the case of the PAES, using the dominance relation generates a few solutions that dominate a section in the middle of the front produced with the aggregating method. Note that the results obtained using the relaxed dominance are very poor for both algorithms. Only a few solutions in the top end of the front produced with the relaxed form of dominance are competitive with those produced by the other two evaluation methods.

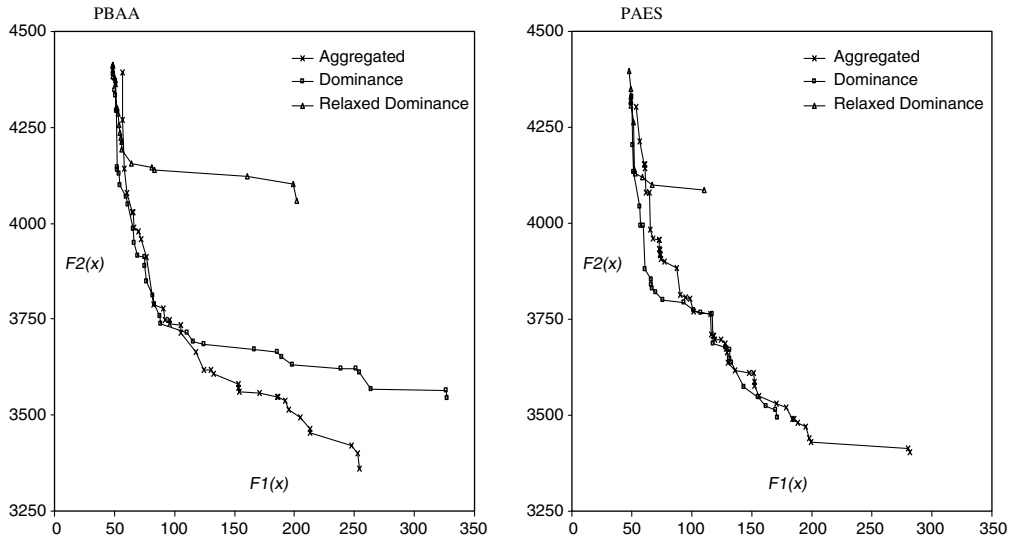

Fig. 4. Offline non-dominated solutions obtained by the two algorithms with each evaluation method for the test instance **trent1**.

It seems that when the relaxed dominance is used in problem **trent1**, both algorithms have difficulty in finding solutions with low values of soft constraint violation (F2). One of the reasons for this behavior might be the level of compromise established between improvement in one of the objectives and the corresponding detriment in the other. That is, as described in Section 3.3, the detriment permitted in objective u cannot be greater that the *gain* obtained in objective v. However, it may be that this level of acceptance is not adequate if improvements in one of the objectives are more difficult to achieve than for the other objective. Perhaps smaller or greater levels of detriment in one objective should be considered as acceptable after obtaining an improvement in the other objective. This is further investigated later in Section 5.5 of this paper.

5.3. The online non-dominated sets

With respect to the online performance, the non-dominated populations obtained in the runs using the same algorithm on the same test instance but with the three different fitness evaluation methods were compared by using the metric proposed by Zitzler et al. [63]. This metric was selected because it directly compares the quality of two non-dominated sets, it is not required to know the Pareto optimal front and it is simple to compute (various metrics are described in [3,26,38,49]). This metric is described by Eq. (13), where A, B are sets of non-dominated vectors.

$$C(A,B) = \frac{|\{b \in B; \exists a \in A : a \preceq b\}|}{|B|}. \tag{13}$$

With the above metric, a value of $C(A, B) = 1$ indicates that all solutions in set B are dominated by at least one solution in set A while a value of $C(A, B) = 0$ indicates that no solution in set B is dominated by a solution in set A. In our experiments, ten values of $C(\mathrm{D}, \mathrm{A})$, $C(\mathrm{A}, \mathrm{D})$, $C(\mathrm{D}, \mathrm{RD})$, $C(\mathrm{RD}, \mathrm{D})$, $C(\mathrm{A}, \mathrm{RD})$ and $C(\mathrm{RD}, \mathrm{A})$ were computed for each set of runs comparing the three fitness evaluation methods using the same algorithm and test problem. The average of these 10 values is presented in Table 2 for each test instance and each algorithm. By observing the comparison between the aggregating function results and the two other evaluation methods, it can be seen that in general, the aggregating function helps the approach to obtain the best results in the case of both algorithms or at least it is as competitive as the relaxed dominance. Only for

Table 2
Comparison of the online performance using the three evaluation methods: A is the aggregating function, D is the dominance relation, and RD is the relaxed dominance

	PBAA			PAES		
	nott1	**nott1b**	**trent1**	**nott1**	**nott1b**	**trent1**
C(D,A)	0.13	0	0.39	0.08	0.17	0.28
C(A,D)	0.99	0.76	0.23	0.96	0.81	0.32
C(D,RD)	0	0.58	0.94	0	0.51	0.92
C(RD,D)	1	0.49	0.14	0.98	0.64	0.21
C(A,RD)	0.65	0.54	0.97	0.77	0.62	0.96
C(RD,A)	0.41	0.43	0.10	0.26	0.35	0.17

the PBAA method on the **trent1** instance is the average coverage $C(D, A)$ slightly better than the average coverage $C(A, D)$. When comparing the results obtained with standard dominance and relaxed dominance, it is clear that for the **nott1** instance the relaxed dominance is better. In the case of the **nott1b** instance both strategies appear to be comparable across the ten runs. However, as mentioned above, in the **trent1** instance the performance of both algorithms when using the relaxed dominance is very poor and beaten clearly by the standard dominance too. The following section presents and discusses results in terms of the population diversity.

5.4. Results on diversity

Table 3 shows the results with respect to the diversity V_p (Eq. (10)) of the non-dominated sets obtained in the experiments described above. Again, for each set of 10 runs corresponding to the same triplet (algorithm, problem instance, fitness evaluation method), the values of V_p were averaged and these are shown as the online results in Table 3. The values of V_p were also calculated for the offline populations collected after each set of 10 runs and these are shown as the offline results in the same table. It can be observed that with respect to the online performance, both algorithms obtain non-dominated sets with very similar diversities for the three fitness evaluation methods in the three test problems. In all cases, the relaxed dominance helps both algorithms to achieve slightly more diverse populations but the difference with the other methods is almost insignificant. In the case of the offline non-dominated sets, although the results obtained with the three fitness evaluation methods are still very similar, greater differences between the diversity values obtained can be observed. For example, the aggregating function benefits PBAA in problems **nott1** and **nott1b** and PAES in problem **nott1b**. The relaxed dominance method favors PBAA in the **trent1** problem and PAES in the **nott1** problem. The standard dominance relation helps PAES to obtain a slightly more diverse offline population in problem **trent1**. In general it can be said, from these results, that none of the three fitness evaluation methods seems to be clearly more beneficial than the others with respect to the population diversity that the two algorithms achieve. In the next section, further experiments are carried out in order to investigate the reasons why the relaxed dominance appears to adversely affect the performance of both algorithms in the **trent1** instance as noted in Section 5.2.

Table 3
Results on diversity V_p for the online and offline non-dominated sets obtained with each algorithm when using the three different fitness evaluation methods: A is the aggregating function, D is the dominance relation, and RD is the relaxed dominance

		PBAA			PAES		
		nott1	**nott1b**	**trent1**	**nott1**	**nott1b**	**trent1**
Online (average)	A	71.3	75.7	81.9	71.2	75.7	82.9
	D	72.1	76.9	81.5	73.6	75.9	81.8
	RD	72.5	78.2	84.4	73.8	77.5	83.6
Offline	A	32.2	53.8	32.0	28.1	48.8	30.5
	D	27.0	39.1	32.8	29.7	30.5	33.6
	RD	26.3	34.6	40.0	32.5	32.3	23.5

5.5. Compromise between objectives in relaxed dominance

As described in Section 3.3, in the relaxed dominance relation used here, the detriment proportion acceptable in one of the objective values cannot be greater than the *gain* or improvement proportion obtained in the other objective value. If improvements for one of the objectives are more difficult to achieve than for the other, then the above compromise may not be as beneficial as thought. This appears to be the case in the **trent1** problem instance as revealed in the experiments and results presented next. Given the results obtained with the relaxed dominance as evaluation method in the **trent1** problem, it was decided to carry out more experiments with different levels of compromise between the two objectives. Consider the current and candidate solutions x and x' with fitness vectors $V[v_1, v_2]$ and $U = [u_1, u_2]$ respectively. Four levels of trade-off between the two objectives were considered:

Relaxed dominance. In this case the compromise is set as described in Section 3.3. In the three cases below *gain* is calculated as before.

Relaxed dominance variant A. Here, a greater detriment proportion is permitted in F1 given an improvement in F2. That is, when $u_2 < v_2$ then x' is considered better than x if $u_1 < v_1 \cdot (1 + 10 \cdot gain)$. When $u_1 < v_1$, the detriment permitted in v_2 is as before.

Relaxed dominance variant B. In this case, a greater detriment proportion is permitted in F2 given an improvement in F1. That is, when $u_1 < v_1$ then x' is considered better than x if $u_2 < v_2 \cdot (1 + 10 \cdot gain)$. When $u_2 < v_2$, the detriment permitted in v_1 is as before.

Relaxed dominance variant C. Now, the detriment proportion permitted in F2 given an improvement in F1 is less than in the previous case. That is, when $u_1 < v_1$ then x' is considered better than x if $u_1 < v_2 \cdot (1 + 5 \cdot gain)$. When $u_2 < v_2$, the detriment permitted in v_1 is as before.

The variant A refers to the case in which an improvement in the satisfaction of soft constraints (F2) is more desirable and therefore more detriment in space misuse (F1) is permitted. The other two variants reflect the case in which the improvement in space misuse (F1) is considered more attractive and the detriment permitted in the soft constraints satisfaction (F2) is greater. Sets of runs were executed as described in Section 5.1 but using only the above four variants of the relaxed dominance relation on the **trent1** instance. The results (offline non-dominated sets) of these experiments are presented in Fig. 5. It is clear that the level of compromise between the objectives has an influence on the performance of both algorithms when solving the **trent1** instance. Among the levels of compromise considered here, the best results are obtained when greater detriments in the satisfaction of soft constraints (F2) are allowed given an improvement in the space misuse (F1). This can be interpreted in two ways. It may be that improvements in F1 are difficult to achieve so they are highly welcomed regardless of the detriment caused in F2. The other possibility is that improvements in F2 are the ones which are difficult to achieve so that this objective is permitted to deteriorate sometimes in order to find improvements later in the search. In order to find out which of the above possibilities is occurring here, counters were maintained for the number of times in which the combination of improvement in one objective and detriment in the other led to the candidate solution being considered to be better. The results given next correspond to the relaxed dominance variant B (the one producing better results).

In the case of PBAA, out of the total number of times in which an improvement in at least one of the objectives was achieved, F1 was improved for 70% of the times and F2 was improved for 32% (the sum is greater than 100% since sometimes both objectives are improved). Out of the number of times in which F1 was improved, in 30% of these the detriment in F2 was acceptable and the new solution considered to be better than the current one. Out of the number of times in which F2 was improved, in 35% of these the detriment in F1 was acceptable and the new solution considered to be better than the current one. For PAES the results are as follows: out of the total number of times in which an improvement in at least one of the objectives was achieved, F1 was improved 61% of the times and F2 was improved 41% of the times. Out of the number of times in which

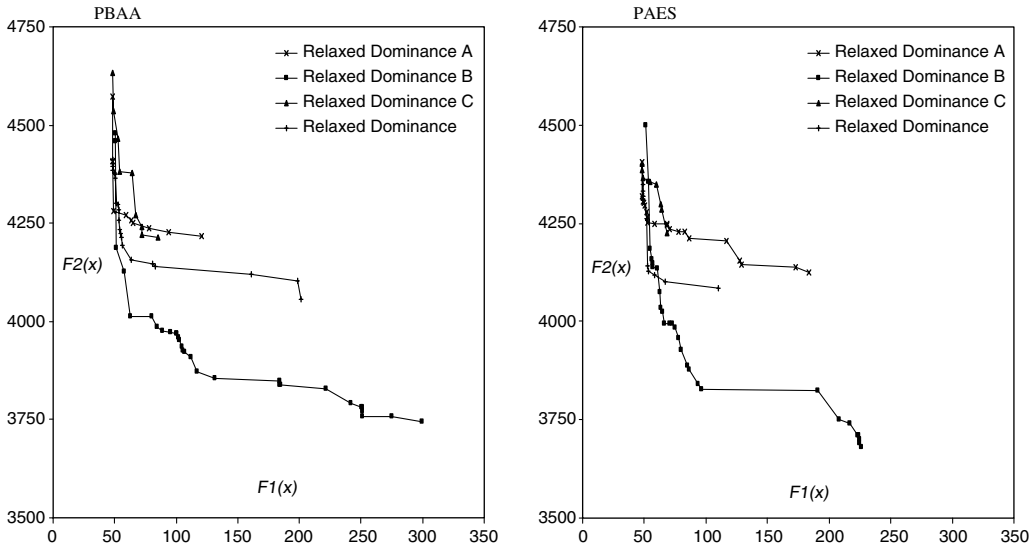

Fig. 5. Offline non-dominated solutions obtained by the two algorithms using the four variants of the relaxed dominance relation for the test instance **trent1**.

F1 was improved, in 43% of these the detriment in F2 was acceptable and the new solution considered to be better than the current one. Out of the number of times in which F2 was improved, in 35% of these the detriment in F1 was acceptable and the new solution considered to be better than the current one. The above results suggest that finding candidate solutions with lower values of soft constraint violation (F2) than the current solution is more difficult in general. It seems that by relaxing the acceptance of solutions with higher values of F2 in the **trent1** problem, the algorithms are provided with a wider view and these solutions may lead to better ones later on in the search. Finally, Fig. 6 compares for the **trent1** instance, the offline non-dominated sets obtained with the relaxed dominance variant B and the other two evaluation methods (shown in Fig. 4). Although the non-dominated sets obtained with both algorithms using the relaxed dominance variant B are much better than the ones obtained with the original relaxed dominance, still the two other fitness evaluation methods help to obtain better results in both algorithms. In the next section, more results are presented in an attempt to investigate the effect of the fitness evaluation method on the evolution of the objective values.

5.6. The evolution of objective values

To investigate the effect of the fitness evaluation method on the evolution of the objectives, the values of $\mathrm{F1}(x)$, $\mathrm{F2}(x)$ and $F(x) = \mathrm{F1}(x) + \mathrm{F2}(x)$ for each individual x in the P_{B} populations of PBAA were recorded. The same was done for the current solution in PAES. Only a sample of the results are presented here, but the graphs shown below are typical of the observations made in all the runs of the experiments for both algorithms and the three test problems. Figs. 7–9 show for the **nott1** instance and the PBAA, the evolution of $\mathrm{F1}(x)$, $\mathrm{F2}(x)$ and $F(x)$ for one individual in P_{B} when each of the evaluation methods was used. As expected, the values of $\mathrm{F1}(x)$ or $\mathrm{F2}(x)$ when using the aggregating function are sometimes worsened in favor of improving the aggregated value but frequently that detriment is temporal and the previous value is recovered or improved later on in the process. Similar observations can be made when the relaxed dominance is used to evaluate solution fitness. This, of course, cannot happen when using the standard dominance relation since the candidate solution is accepted only if at least one of the objectives is improved without worsening the other.

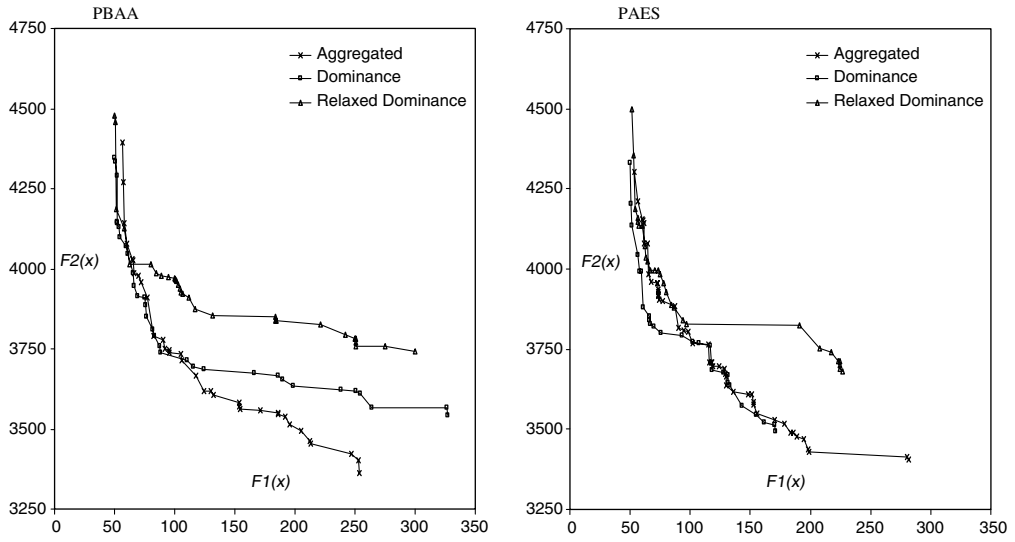

Fig. 6. Offline non-dominated solutions obtained by the two algorithms with each evaluation method for the test instance **trent1**.

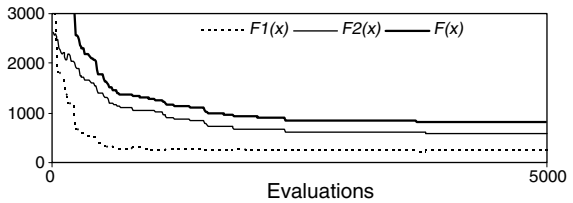

Fig. 7. Evolution of the objective values for one individual in P_B of PBAA during a typical run using the aggregating function.

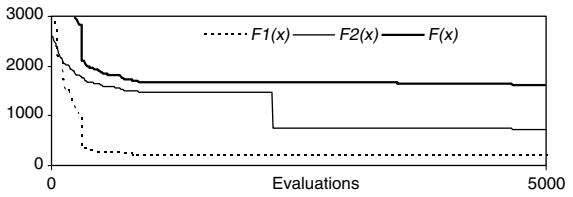

Fig. 8. Evolution of the objective values for one individual in P_B of PBAA during a typical run using the standard dominance.

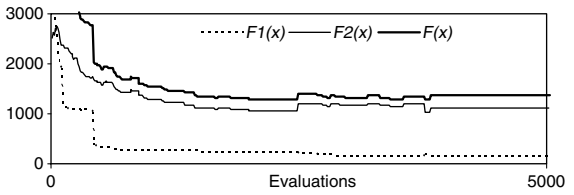

Fig. 9. Evolution of the objective values for one individual in P_B of PBAA during a typical run using the relaxed dominance.

6. Further discussion and final remarks

There is an increasing interest by researchers in various fields on the application of metaheuristics to multiobjective optimisation problems. Most of the published research on this subject has been focused on the development of new algorithms or extending existing single-objective methods toward multiobjective approaches. As noted in the introduction, a considerable number of papers report on the comparison between multiobjective optimisers on test and real-world problems. It is also fundamental to investigate the reasons why metaheuristics for multiobjective optimisation succeed or fail in certain problem domains to gain a better understanding of their functioning in order to design more effective and efficient approaches. Although some research has been published on the effect that some strategies have on the performance of some metaheuristics for multiobjective optimisation (e.g. [23,42,48]), we believe that more research on this subject is required. The research

presented in this paper aims to be a contribution to the better understanding of the mechanisms and conditions that influence the performance of multiobjective search algorithms. The subject of study here has been the effect of the method used to assign fitness to solutions on the performance of multiobjective methods. The fitness evaluation methods considered here were: the aggregation of objectives, the dominance relation and a relaxed form of this dominance relation. Arguments can be found in the literature both in favor and against the use of aggregating functions or the use of dominance within metaheuristics for multiobjective optimisation. For example:

- Some researchers have expressed the view that Pareto-based evolutionary algorithms are more suitable for multiobjective optimisation than local search methods using aggregation of objectives [18,22] while other researchers have shown that approaches that use local search and aggregating functions are suitable for dealing with various multiobjective optimisation problems [19,29,32,46,58].
- Knowles proposed and evaluated several approaches for multiobjective optimisation based on a form of local search: mutation operators and using the dominance relation to evaluate solutions [37].
- Jaszkiewick said "... *Pareto ranking is not well suited for hybridization with local search*" and found that weighted linear functions had better ability than Tchebycheff functions in finding potential non-dominated solutions within a genetic local search algorithm [32, p. 54].
- Knowles et al. suggested that using the dominance relation can be beneficial even in single-objective optimisation for reducing the number of local optima [36].
- Kokolo et al. illustrated the difficulty that approaches using dominance selection may exhibit in finding Pareto optimal solutions and suggested the use of α-dominance (relaxed dominance) [39].
- Some researchers have used ϵ-dominance (similar to α-dominance) to implement better archiving strategies that attempt to help multiobjective evolutionary algorithms to converge towards the optimal Pareto front and maintain a wide diversity in the population at the same time [33,43,48].
- The use of subcost guided search was proposed by Wright to deal with compound-objective timetabling problems [61]. An improvement of a sub-cost (objective) is preferred even if the overall cost or solution fitness is not improved at all or it is worsened. The hope is that the detriment suffered will be repaired later in the process since the improvement in one aspect of the solution (the subcost) enables a kind of guided diversification towards promising areas of the solution space.

The above points show the various opinions (some of these conflicting) that researchers have expressed when referring to the fitness evaluation method used when implementing algorithms for multiobjective optimisation. In relation to this, we have demonstrated in this paper that, although an approximation to the Pareto optimal set is the aim here, the (standard) dominance relation is not the best method to assign fitness to solutions. The results from the experiments described in this paper suggest that the performance of the multiobjective metaheuristics investigated here is very much influenced by the method used to evaluate the fitness of solutions during the search process. The test problem used here is a highly constrained combinatorial optimisation problem and the existence of constraints seems to be a reason for the difference observed in the performance of both algorithms when using different fitness evaluation methods. It is apparent that if it is more difficult to achieve improvements in one of the objectives (F2 here) than in the other (F1 here), then a compromise that allows detriments in the objectives should be made so that the algorithms are provided with better mechanisms to explore other areas of the solution space. In terms of both online and offline performance, the inferiority of the dominance evaluation method is evident. Between the aggregating function and the relaxed dominance it seems that the first one helps to achieve better values of F2 while with respect to F1 the relaxed dominance benefits the most. It also appears that the relaxed dominance evaluation method

helps to achieve a better coverage of the intended compromise surface. However, the distance between the obtained non-dominated fronts and the intended compromise surface is shorter when using the aggregating method. In terms of diversity in the solution space for the obtained sets, the three methods seem to be competitive with each other but a small inferiority with respect to the dominance relation can be observed. In terms of the computation time, we did not observe a significant difference in any of the runs as a result of using the different fitness evaluation methods considered here.

In multiobjective optimisation non-dominated solutions are sought, but there is a question about the circumstances (problem domain and search strategy) under which the use of the dominance relation to identify improvement during the search is the best alternative. As shown in this paper, sometimes it might be more beneficial to use a combination of objectives or a relaxed definition of dominance. Considering these alternative ways for assessing solutions during the search in Pareto optimisation is worthwhile. We have a few suggested directions for further research. It would be interesting to investigate if the observations made here hold for other multiobjective approaches and other problem domains. The space allocation problem formulated here is just one example of many other similar problems that refer to the optimisation of space resource utilisation. Therefore, we also suggest the investigation of the issues studied in this paper on similar problems like land allocation [2,16,45], shelf space allocation [62], multiple knapsack problems [21] and others. Another issue worth investigating is the use of different fitness evaluation methods within the same population. For example, some solutions in the population can be evaluated using dominance while others could use an aggregated function and others might employ relaxed dominance or perhaps even other methods.

Acknowledgment

We are grateful for the valuable comments and suggestions from the anonymous referees.

References

[1] E. Aarts, J. Korts (Eds.), Simulated Annealing and Boltzman Machines, Wiley, 1998.
[2] J.C.J.H. Aerst, G.B.M. Heuvelink, Using simulated annealing for resource allocation, International Journal of Geographic Information Science 16 (2002) 571–587.
[3] K.H. Ang, G. Chong, Y. Li, Preliminary statement on the current progress of multi-objective evolutionary algorithm performance measurement, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, 2002, pp. 1139–1144.
[4] P. Bagchi Tapan, Pareto-optimal solutions for multi-objective production scheduling problems, in: Proceedings of the 1st International Conference on Evolutionary Multi-criterion Optimization (EMO 2001)Lecture Notes in Computer Science, vol. 1993, Springer, 2001, pp. 458–471.
[5] M. Basseur, F. Seynhaeve, E.G. Talbi, Design of multi-objective evolutionary algorithms to the flow-shop scheduling problem, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, 2002, pp. 1151–1156.
[6] A. Baykasoglu, S. Owen, N. Gindy, A taboo search based approach to find the pareto optimal set in multiple objective optimisation, Engineering Optimization 31 (1999) 731–748.
[7] V. Belton, T.J. Stewart, Multiple Criteria Decision Analysis—An Integrated Approach, Kluwer Academic Publishers, 2002.
[8] J.A. Bland, Space-planning by ant colony optimisation, International Journal of Computer Applications in Technology 12 (6) (1999) 320–328.
[9] E.K. Burke, Y. Bykov, S. Petrovic, A multicriteria approach to examination timetabling. The practice and theory of automated timetabling III, Selected papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000)Lecture Notes in Computer Science, vol. 20792001, Springer, pp. 118–131.
[10] E.K. Burke, P. Cowling, J.D. Landa Silva, Three methods to automate the space allocation process in UK Universities. The practice and theory of automated timetabling III, Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000), Lecture Notes in Computer Science, vol. 20702000, Springer, pp. 254–273.
[11] E.K. Burke, P. Cowling, J.D. Landa Silva, Hybrid population-based metaheuristic approaches for the space allocation problem, in: Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001), IEEE Press, 2001, pp. 232–239.
[12] E.K. Burke, P. De Causmaecker, S. Petrovic, G. Vanden Berghe, A multi criteria meta-heuristic approach to nurse scheduling, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, 2002, pp. 1197–1202.

[13] E.K. Burke, J.D. Landa Silva, Improving the performance of multiobjective optimisers by using relaxed dominance, in: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), 2002, pp. 203–207.
[14] E.K. Burke, D.B. Varley, Space allocation: An analysis of higher education requirements. The practice and theory of automated timetabling II, Selected papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT 1997)Lecture Notes in Computer Science, vol. 14081998, Springer, pp. 20–33.
[15] M.P. Carrasco, M.V. Pato, A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem, The Practice and Theory of Automated Timetabling III, Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000)Lecture Notes in Computer Science, vol. 20792001, Springer, pp. 3–17.
[16] R.L. Church, D.M. Stoms, F.W. Davis, Research selection as a maximal covering allocation problem, Biological Conservation 76 (1996) 105–112.
[17] C.A. Coello Coello, A comprehensive survey of evolutionary based multiobjective optimization techniques, Knowledge and Information Systems 1 (3) (1999) 269–308.
[18] A. Coello Coello Carlos, A. Van Veldhuizen David, B. Lamont Gary, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, 2002.
[19] P. Czyzak, A. Jaszkiewicz, Pareto simulated annealing—A metaheuristic for multiple-objective combinatorial optimization, Journal of Multi-Criteria Decision Analysis 7 (1) (1998) 34–47.
[20] P. Dasgupta, P.P. Chakrabarti, S.C. DeSarkar, Multiobjective heuristic search—An introduction to intelligent search methods for multicriteria optimization, Computational Intelligence–Vieweg, 1999.
[21] M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi, F.S. Salman, Approximation algorithms for the multiple knapsack problem with assignment restrictions, Journal of Combinatorial Optimization 4 (2) (2000) 171–186.
[22] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
[23] K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, Evolutionary Computation 7 (3) (1999) 205–230.
[24] M. Ehrgott, X. Gandibleux, A survey and annotated bibliography of multiobjective combinatorial optimization, OR Spectrum 22 (4) (2000) 425–460.
[25] W. El-Moudani, C.A. Nunes Cosenza, M. de Coligny, F. Mora Camino, A bi-criterion approach for the airlines crew rostering problem, Proceedings of the 1st International Conference on Evolutionary Multi-criterion Optimization (EMO 2001)Lecture Notes in Computer Science, vol. 19932001, Springer, pp. 486–500.
[26] A. Farhang-Mehr, S. Azarm, Minimal sets of quality metrics, in: Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol. 2632, Springer, 2003.
[27] C.M. Fonseca, P. Fleming, E. Zitzler, K. Deb, L. Thiele (Eds.), Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Lecture Notes in Computer Science, vol. 2632, Springer, 2003.
[28] R.L. Francis, L.F. McGinnis Jr., J.A. White, Facility Layout and Location: An Analytical Approach, Prentice-Hall, 1992.
[29] X. Gandibleux, A. Freville, Tabu search based procedure for solving the 0–1 multi-objective knapsack problem: The two objectives case, Journal of Heuristics 6 (3) (2000) 361–383.
[30] Ambrose Goicoechea, R. Hansen Don, Lucien Duckstein, Multiobjective Decision Analysis with Engineering and Business Applications, Wiley, 1982.
[31] A. Jaszkiewicz, A Metaheuristic Approach to multiple objective nurse scheduling, Foundations of Computing and Decision Sciences 22 (3) (1997) 169–183.
[32] A. Jaszkiewicz, Genetic local search for multi-objective combinatorial optimization, European Journal of Operational Research 137 (1) (2002) 50–71.
[33] H. Jin, M.L. Wong, Adaptive diversity maintenance and convergence guarantee in multiobjective evolutionary algorithms, in: Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003), IEEE Press, 2003, pp. 2498–2505.
[34] H. Kellerer, U. Pferschy, Cardinality constrained bin-packing problems, Annals of Operations Research 92 (1999) 335–348.
[35] J. Knowles, D.C. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, Evolutionary Computation 8 (2) (2000) 149–172.
[36] J.D. Knowles, R.A. Watson, D.W. Corne, Reducing local optima in single-objective problems by multi-objectivization, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)Lecture Notes in Computer Science, vol. 19932001, Springer, pp. 269–283.
[37] J.D. Knowles, Local-search and hybrid evolutionary algorithms for pareto optimization, PhD Thesis, Department of Computer Science, University of Reading, UK, 2001.
[38] J. Knowles, D. Corne, On metrics for comparing non-dominated sets, in: Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002), IEEE Press, 2002, pp. 711–716.
[39] I. Kokolo, K. Hajime, K. Shigenobu, Failure of pareto-based MOEAs: Does non-dominated really mean near to optimal? in: Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001), IEEE Press, 2001, pp. 957–962.
[40] J.D. Landa Silva, Methaeuristic and multiobjective approaches for space allocation, PhD Thesis, School of Computer Science and Information Technology, University of Nottingham, UK, 2003.

[41] J.D. Landa Silva, E.K. Burke, S. Petrovic, An introduction to multiobjective metaheuristics for scheduling and timetabling, X. Gandibleux, M. Sevaux, K. Sorensen, V. T'kindt (Eds.), Metaheuristic for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, vol. 535, Springer, pp. 91–129.
[42] M. Laumanns, E. Zitzler, J. Thiele, On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)Lecture Notes in Computer Science, vol. 19932001, Springer, pp. 181–196.
[43] M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining convergence and diversity in evolutionary multiobjective optimization, Evolutionary Computation 10 (3) (2002) 263–282.
[44] S. Martello, P. Toth, Knapsack Problems—Algorithms and Computer Implementations, Wiley, 1990.
[45] K.B. Matthews, Applying genetic algorithms to multiobjective land-use planning, PhD Thesis, The Robert Gordon University, Aberdeen, UK, 2001.
[46] F. Menczer, M. Degeratu, W.N. Street, Efficient and scalable pareto optimization by evolutionary local selection algorithms, Evolutionary Computation 8 (2) (2000) 223–247.
[47] Z. Michalewicz, D.B. Fogel, How to Solve It: Modern Heuristics, Spinger, 2000.
[48] S. Mostaghim, J. Teich, The role of ϵ-dominance in multiobjective particle swarm optimization methods, in: Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003), IEEE Press, 2003, pp. 1764–1771.
[49] T. Okabe, Y. Jin, B. Sendhoff, A critical survey of performance indices for multi-objective optimisation, in: Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003), IEEE Press, 2003, pp. 862–869.
[50] S. Petrovic, Y. Bykov, A multiobjective optimisation technique for exam timetabling based on trajectories. The practice and theory of automated timetabling IV, Selected Papers from the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2002)Lecture Notes in Computer Science, vol. 27402003, Springer, pp. 179–192.
[51] R.C. Purshouse, P.J. Fleming, The multiobjective genetic algorithm applied to benchmark problems—An analysis. Technical Report No. 796, Department of Automatic Control and Systems Engineering, University of Sheffield, UK, 2001.
[52] R.C. Purshouse, P.J. Fleming, Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation, Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization (EMO 2003)Lecture Notes in Computer Science, vol. 26322003, Springer, pp. 16–30.
[53] D. Romero, A. Sanchez-Flores, Methods for the one-dimensional space allocation problem, Computers and Operations Research 15 (5) (1990) 465–473.
[54] E. Steuer Ralph, Multiple Criteria Optimization: Theory, Computation and Application, Wiley, 1986.
[55] A. Suppapitnarm, A. Seffen, G.T. Parks, P.J. Clarkson, A simulated annealing algorithm for multiobjective optimisation, Engineering Optimization 33 (1) (2000) 59–85.
[56] K.C. Tan, T.H. Lee, E.F. Khor, Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons, in: Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001), IEEE Press, 2001, pp. 979–986.
[57] R. Taprabata, K.M. Liew, A swarm metaphor for multiobjective design optimization, Engineering Optimization 34 (2) (2002) 141–153.
[58] E.L. Ulungu, J. Teghem, P.H. Fortemps, D. Tuyttens, MOSA method: A tool for solving multiobjective combinatorial optimization problems, Journal of Multicriteria Decision Analysis 8 (1999) 221–236.
[59] D.A. Van Valdhuizen, G.B. Lament, Multiobjective evolutionary algorithms: Analyzing the state-of-the-art, Evolutionary Computation 8 (2) (2000) 125–147.
[60] D.A. Van Valdhuizen, G.B. Lament, On measuring multiobjective evolutionary algorithms performance, in: Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), IEEE Press, 2000, pp. 204–211.
[61] M. Wright, Subcost-guided search—Experiments with timetabling problems, Journal of Heuristics 7 (2001) 251–260.
[62] M.H. Yang, W.C. Chen, A study on shelf space allocation and management, International Journal of Production Economics 60–61 (1999) 309–317.
[63] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evolutionary Computation 8 (2) (2000) 173–195.
[64] E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (Eds.), Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Lecture Notes in Computer Science, vol. 1993, Springer, 2001.
[65] J.B. Zydallis, D.A. Van Valdhuizen, G.B. Lamont, A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II, Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)Lecture Notes in Computer Science, vol. 19932001, Springer, pp. 226–240.