An Introduction to Multiobjective Metaheuristics for Scheduling and Timetabling

J.D. Landa Silva, E.K. Burke, S. Petrovic

Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science and IT, University of Nottingham, UK
{jds|ekb|sxp}@cs.nott.ac.uk
http://www.asap.cs.nott.ac.uk

Abstract. In many real-world scheduling problems (eg. machine scheduling, educational timetabling, personnel scheduling, etc.) several criteria must be considered simultaneously when evaluating the quality of the solution or schedule. Among these criteria there are: length of the schedule, utilisation of resources, satisfaction of people's preferences and compliance with regulations. Traditionally, these problems have been tackled as single-objective optimisation problems after combining the multiple criteria into a single scalar value. A number of multiobjective metaheuristics have been proposed in recent years to obtain sets of compromise solutions for multiobjective optimisation problems in a single run and without the need to convert the problem to a single-objective one. Most of these techniques have been successfully tested in both benchmark and real-world multiobjective problems. However, the number of reported applications of these techniques to scheduling problems is still relatively scarce. This paper presents an introduction to the application of multiobjective metaheuristics to some multicriteria scheduling problems.

1 Introduction

Scheduling is the arrangement of entities (people, tasks, vehicles, lectures, exams, meetings, etc.) into a pattern in space-time in such a way that constraints are satisfied and certain goals are achieved [120]. Constructing a schedule is the problem in which time, space and other (often limited) resources have to be considered in the arrangement. The constraints are relationships among the entities or between the entities and the patterns that limit the construction of the schedule. Constraints can be classified as *hard* or *soft*. *Hard constraints* must not be violated under any circumstances. Solutions which satisfy such constraints can be called *feasible*. It is desirable to satisfy as many *soft constraints* as possible but if one of them is violated, a penalty is applied and the solution is still considered to be feasible. In practice, the scheduling activity can be regarded as a search problem for which it is required to find any feasible schedule or as an optimisation problem for which the *best* feasible schedule is sought. The *best* solution is often defined to be the one with the lowest penalty (for violation of the soft constraints). In real-world problems, expressing the conditions that make a schedule more preferable than another and incorporating this information into an automated system, is not an easy task. In addition, the combinatorial nature of these problems implies exploring huge search spaces [93, 123] and human intervention is often necessary to bias the search towards promising regions.

The class of scheduling problems includes a wide variety of problems such as machine scheduling, events scheduling, personnel scheduling and many others (eg. see [10, 14, 96, 120]). Many real world scheduling problems are multiobjective by nature, i.e. several objectives should be achieved simultaneously (eg. see [4, 55, 92, 113, 114]). Examples of such objectives are: minimise the length of the schedule, optimise the utilisation of the available resources, satisfy the preferences of human resources (personnel scheduling), minimise the tardiness of orders (production scheduling), maximise the compliance with regulations (educational timetabling) and there are many others. Over the years, there have been several approaches used to deal with the various objectives in such problems. Traditionally, the most common approach has been to combine the multiple objectives into a single scalar value by using weighted aggregating functions according to the preferences set by the decision-makers and then, to find a solution that satisfies these preferences [9, 87, 113]. However, in many real scenarios involving multiobjective scheduling problems, it is preferable to present various compromise solutions to the decision-makers, so that the most adequate schedule can be chosen. Although this can be achieved by performing the search several times using different preferences each time, another approach is to generate the set of compromise solutions in a single execution of the algorithm. The latter strategy has attracted the interest of researchers for investigating the application of Pareto optimisation techniques to multiobjective scheduling problems (eg. [4, 5, 13, 71, 89]). The aim in Pareto optimisation (which is discussed in some detail below) is to find a set of compromise solutions that represent a good approximation to the Pareto optimal front [100, 107]. In recent years, the number of algorithms proposed for Pareto optimisation has increased tremendously mainly because multiobjective optimisation problems exist in almost any domain (eg. see [55, 60, 77, 110, 125]).

Voss et al. describe a metaheuristic as *"an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions"* [118]. Metaheuristics include tabu search [65], simulated annealing [1], variable neighbourhood search [67], genetic algorithms [84], neural networks [98], ant colony optimisation [52] and many others (see also [2, 42, 64, 118]). Many metaheuristics that were first applied to solve single-objective optimisation problems have also been extended to multiobjective variants. Among these, multiobjective evolutionary algorithms have received particular attention because some researchers argue that these methods are well suited to deal with multiobjective optimisation problems [41, 50]. Also, some multiobjective metaheuristics based on local search, such as simulated annealing and tabu search have been proposed recently (eg. [8, 48, 61, 66, 75, 108, 115]). In the context of single-objective combinatorial optimisation problems and in particular scheduling problems, it is often the case that local search is incorporated into evolutionary algorithms in order to improve the results obtained with these methods (eg. [33, 16, 30, 40, 99]). Such methods are sometimes called memetic algorithms. This appears to be true also in the multiobjective case given the evidence reported by researchers in the field (eg. [16, 53, 62, 71, 72, 74, 76, 109]). Although there are a considerable number of proposed algorithms for Pareto optimisation, the number of reported applications of these techniques to multiobjective scheduling problems is still relatively scarce. This is particularly true for event scheduling (timetabling) and personnel scheduling (rostering) problems, for which the majority of the recent publications still consider the use of aggregating functions to combine the multiple criteria into a single value (eg. see [18, 25, 105]).

This paper is organised as follows. Section 2 gives an introduction to concepts in multicriteria decision-making and multiobjective optimisation. This work seeks to present a brief (but not exhaustive) overview of the recent (from 1996 onwards) reported literature on multiobjective scheduling and timetabling. We concentrate in particular on the application of metaheuristic approaches. The modeling of multiobjective scheduling and timetabling problems is outside the scope of this paper and the reader is referred to the relevant literature when appropriate. Nevertheless, some descriptions of multiobjective scheduling and timetabling problems are discussed in order to facilitate the understanding of the approaches we consider. An introduction to machine scheduling problems is given in Sect. 3 while a description of educational timetabling problems and a discussion of their multiobjective nature are presented in Sect. 5. One aim of this paper is to identify the strategies that have been successful in the multiobjective optimisation (using metaheuristics) of some multicriteria scheduling problems. Therefore, Sect. 4 and Sect. 6 describe some of the multiobjective metaheuristics that have been proposed to tackle machine scheduling problems and educational timetabling problems respectively. Also, some applications of multiobjective metaheuristics to personnel scheduling are described in Sect. 7. Another aim here, is to identify promising research directions that may be interesting to explore in order to strengthen the application of modern multiobjective metaheuristics to these and related problems. This is done in Sect. 8. Finally, remarks are presented in Sect. 9.

2 Multicriteria Decision-Making and Multiobjective Optimisation

2.1 Introduction

The general multiobjective combinatorial optimisation problem can be formulated as follows:

$$\text{Minimise or Maximize } F(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \text{ s.t. } x \in S \ . \tag{1}$$

where x is a solution, S is the set of feasible solutions, k is the number of objectives in the problem, $F(x)$ is the image of x in the k-objective space and each $f_i(x)$ $i = 1, \ldots, k$ represents one (minimisation or maximisation) objective.

In many problems, the aim is to obtain the optimal arrangement of a group of discrete entities in such a way that the additional requirements and constraints (if they exist) are satisfied [93, 98]. If the problem is a multiobjective one, various criteria exist to evaluate the quality of solutions and there is an objective (minimisation or maximisation) attached to each of these criteria [114]. It is often the case that some of the criteria are in conflict, i.e. an improvement in one of them can only be achieved at the expense of worsening another. Moreover, some of the criteria may be incommensurable, i.e. the units used to measure the compliance with each of the criteria are not comparable at all. The incommensurability of criteria adds to the difficulty of the problem because the aggregation or comparison of different objectives is not straightforward. Let us illustrate some of these issues using a timetabling problem as example. For an examination timetable, two of the criteria (among others) that may be used to express the quality of the schedule are its length and the satisfaction of students' preferences (eg. see [16]). The objectives would be to produce the shortest schedule possible and to satisfy most of the requests from students respectively. These objectives are conflicting because students usually prefer to have the longest time possible between exams and this of course, implies a longer schedule. The length of the schedule is expressed in number of timeslots and this metric may not be the most appropriate to indicate the level of compliance with the preferences of students. To express the degree at which the schedule satisfies the students' requests, other aspects such as the spread and balance of the schedule and the location of difficult exams within the schedule would be more adequate.

2.2 Search and Decision-Making

The first decision that has to be made when dealing with a multiobjective optimisation problem is on how to combine the search and the decision-making processes. This can be done in one of three ways [107]:

Decision-making and then search (*a priori* approach). The preferences for each objective are set by the decision-makers and then, one or various solutions satisfying these preferences have to be found.

Search and then decision-making (*a posteriori* approach). Various solutions are found and then, the decision-makers select the most adequate. The solutions presented should represent a trade-off between the various objectives.

Interactive search and decision-making. The decision-makers intervene during the search in order to guide it towards promising solutions by adjusting the preferences in the process.

Another important decision is how to evaluate the quality of solutions, because the conflicting and incommensurable nature of some of the criteria makes this process more complex. Also here, there are several alternatives [41]:

Combine the objectives. This is one of the "classical" methods to evaluate the solution fitness in multiobjective optimisation. It refers to converting the multiobjective problem into a single-objective one by combining the various criteria into a single scalar value. The most common way of doing this is by setting weights to each criterion and add them all together using an aggregating function.

Alternating the objectives. This is another "classical" approach. It refers to optimising one criterion at a time while imposing constraints on the others. The difficulty here is on how to establish the ordering in which the criteria should be optimised, because this can have an effect on the success of the search.

Pareto-based evaluation. In this approach, a vector containing all the objective values represents the solution fitness and the concept of *dominance* is used to establish preference between solutions [107]. A solution x is said to be non-inferior or non-dominated if there is no other solution that is better than x in all the criteria. Suppose two distinct vectors $V = (v_1, v_2, \ldots, v_k)$ and $U = (u_1, u_2, \ldots, u_k)$ containing the objective values of two solutions for a k-objective minimisation problem, then:

- V *strictly dominates* U if $v_i < u_i$, for $i = 1, 2, \ldots, k$.
- V *loosely dominates* U if $v_i \leq u_i$, for $i = 1, 2, \ldots, k$ and $v_i < u_i$, for at least one i.
- V and U are *incomparable* if neither V (strictly or loosely) dominates U nor U (strictly or loosely) dominates V.

Minimisation is considered here mainly because most of the scheduling problems are of this type (minimise processing time, minimise soft constraints violation, minimise schedule length, etc.), but the above definition is altered in the obvious way for the case of maximisation problems. It is important to note that, using strict or loose dominance can have an effect on how the search is performed. This is because if a solution x_1 is strictly dominated, it means that it is outperformed by the other solution x_2 in all criteria. But, if the solution x_1 is loosely dominated it means that it is outperformed in some of the criteria but it is as good as x_2 in at least one of them. Then, finding a new solution that strictly dominates the current one may be more difficult than finding a solution that loosely dominates it. This is particularly true in some combinatorial problems in which the connectedness of the search space is such that some solutions are more difficult to reach from the current one. Examples of such problems are the spanning tree problem and the shortest path problem (see [56]). Also, given the set of solutions in the neighbourhood $N(x)$ of a solution x, some of the solutions in that set will (strictly or loosely) dominate x while others will be (strictly or loosely) dominated by x. However, it is true that the set of solutions in $N(x)$ that loosely dominate x is a superset of the set of solutions in $N(x)$ that strictly dominate x. Therefore, by using loose dominance, it is more likely that attractive (dominating) neighbouring solutions can be visited during the neighbourhood search. However, using loose dominance could be inappropriate in those cases in which the solution space contains too many loosely dominating solutions because the search algorithm would spend too much time visiting these solutions.

2.3 Pareto Optimisation

When the aim is to obtain a set of compromise (non-dominated) solutions (search and then decision-making), these solutions should represent a good approximation to the Pareto optimal front. The Pareto optimal front is the set of all non-dominated solutions in the multiobjective space [107]. Pareto optimisation refers to finding the Pareto optimal front or a set that represents a good approximation to that front. Pareto optimisation is appealing because in most multiobjective optimisation problems there is no such single-best solution and it is also very difficult to establish preferences among the criteria before the search. Even when this is possible, it may be that these preferences change and therefore having a set of solutions eases the decision-making process. A problem may have several objectives but we usually consider it to be multiobjective if the criteria are in conflict. Two objectives can be considered to be in conflict if the complete satisfaction of one of them prevents the complete satisfaction of the other. If any improvement in one of the objectives induces a detriment on the other, then the objectives can be said to be strictly conflicting [4]. It has expressed that even if the conflicting nature of the criteria is not proved, Pareto-based metaheuristics would be able to find the ideal solution that is the best in all criteria [58].

Another important aspect to consider is how to evaluate the quality of the obtained non-dominated front. This is a multicriteria problem on its own because several aspects have to be considered to determine how good the obtained front is. Among these aspects there are [50]: 1) the number of non-dominated solutions obtained, 2) the closeness between the obtained front and the Pareto optimal front (if known) and 3) the coverage of the Pareto front, i.e. the spread and distribution of the non-dominated solutions. Several methods have been proposed to evaluate the quality of the obtained non-dominated front in Pareto optimisation and assess the performance of multiobjective optimisers (see [79, 127]). Since the Pareto optimal front is defined with respect to the objective space, it is common that most of the metrics proposed are also defined with respect to this space. One aspect that is frequently overlooked, is the diversity of the obtained front with respect to the solution space. In fact, when researchers report on the quality of the obtained non-dominated sets, they do not usually provide information about the diversity of the solutions in the solution space. This is extremely important, because although the obtained non-dominated solutions may be well spread and distributed over the front in the objective space, it may be that either the solutions are also structurally different (diverse) or very similar between them. Considering diversity in the solution space when assessing the quality of the obtained front becomes even more important in real-world multiobjective combinatorial optimisation problems. This is because the similarity among solutions directly relates to how different the arrangement of the discrete entities is between the solutions. For example, consider the problem of creating an examination timetable where the two criteria used to evaluate the quality of solutions are the lenght of the timetable and the satisfaction of student's preferences. Then, the decision-makers may require solutions that are:

Similar in structure and in objective values. Schedules that are very similar and although each of them is non-dominated, perhaps the decision-makers are interested in a certain part of the trade-off surface. For example, they may prefer a set of similar timetables that have a short lenght and the satisfaction of student's preferences is high enough.

Similar in structure but very different in objective values. The schedules are very similar but the decision-makers want solutions from all over the trade-off surface. In this case, a set of solutions representing a wide range of trade-off between the lenght of the timetable and the satisfaction of student's preferences is required. However, the decison-makers would like the timetables to be similar.

Diverse in structure and in objective values. Solutions from all over the trade-off surface are required, but the schedules must be very different in structure (i.e. timetables that do not look too similar).

Diverse in structure but similar in objective values. The decision-makers require schedules of certain similar quality with respect to the trade-off between objectives but they want to see solutions that actually represent very different schedules. For example, the decison-makers may want timetables that satisfy most of the students' preferences and have a lenght within a given range. However, they would like these timetables not to be very similar (perhaps to discuss the implications of implementing them).

Large multiobjective combinatorial optimisation problems are particularly difficult to tackle. One reason for this, is that the size of the search space grows exponentially as the problem size increases, making impracticable the application of exact optimisation algorithms [55, 93, 98]. Also, in many multiobjective combinatorial optimisation problems there is no notion of the localization and shape of the Pareto optimal front [114]. Considering the fact that many real-world combinatorial optimisation problems are also highly constrained, the scenario is even more complex. Many real-world scheduling problems are examples of combinatorial optimisation problems that involve multiple criteria and almost always are highly constrained.

3 Machine Scheduling Problems

3.1 Introduction

Machine scheduling refers to problems where a set of jobs or tasks have to be scheduled for processing in one or more machines [96]. Each job or task consists of one or more operations (sub-tasks) and usually, a number of additional constraints must also be satisfied. Examples of such constraints are precedence relations between the jobs and limited availability of resources (eg. workforce, machine processing time, materials, etc.). Machine scheduling problems arise across a range of applications. This is perhaps the class of scheduling problems that has attracted the most attention from researchers and practitioners in this area. Two important types of machine scheduling problems are shop scheduling

[11, 96, 113] and project scheduling [14]. In shop scheduling, a set of jobs have to be processed through a number of machines while project scheduling is more concerned with the execution of activities within a project (shop scheduling problems can be modeled as special cases of project scheduling problems, see [14]).

3.2 Some Types of Machine Scheduling Problems

Shop scheduling problems are common in many applications such as industrial production and multiprocessor computer systems. A notation which is commonly used to formulate shop scheduling problems is based on three fields: $\alpha|\beta|\gamma$. In this notation, α describes the machine environment, i.e. the structure of the problem. The field β describes the constraints in the problem and other processing conditions. The third element γ describes the criteria to be optimised. There are many configurations of shop scheduling problems and hence, many different mathematical formulations. Below, we illustrate the use of the above notation with a few well-known configurations of shop scheduling problems. For a more detailed presentation of this notation, including precise models and formulations of other problem configurations, refer to [96, 113]. The following notation is of relevance here:

- n is the number of jobs or tasks.
- m is the number of machines available.
- $p(i, j)$ is the time that takes to process job j on machine i.
- $d(j)$ is the due date of job j, i.e. the committed completion time.
- $c(j)$ is the completion time of job j, i.e. the time taken to finish the job.
- $e(j)$ is the earliness of job j, i.e. how much time the job was completed before the due date, $e(j) = max(0, d(j) - c(j))$.
- $l(j)$ is the lateness of job j, i.e. the delay on the completion of the job with respect to the due date, $l(j) = c(j) - d(j)$.
- $t(j)$ is the tardiness of job j, i.e. the time that the job is actually completed late, $t(j) = max(0, c(j) - d(j))$.
- $r(j)$ is the release date of job j, i.e. the earliest time at which the processing of the job can begin.
- $Cmax$ is the makespan or total completion time which is equal to the completion time of the last job, $Cmax = \max(c(j))$ for $j = 1 \ldots, n$.

The characteristics that define the problem structure (α) include:

- single machine *vs.* multiple machines,
- whether the sequence of operations within the jobs are fixed,
- identical *vs.* different machines,
- existence or not of parallel machines, etc.

Among the constraints that can exist (β) there are [11, 83, 116]:

- pre-emption allowed or not, i.e. whether the processing of jobs can be interrupted and resumed,

– splitting allowed or not, i.e. whether the operations in a job can be split in several parts,
– waiting times between the operations in the same job are permitted or not,
– whether special processing conditions (due dates, setup times, removal times, etc.) are specified or not and if these are deterministic or stochastic,
– availability or resources is limited or not, fixed or flexible, etc.
– whether the capacity input and output buffer are finite,
– consideration of material handling operations or not,
– fixed or dynamic arrival of jobs, etc.

The criteria (γ) used to evaluate the quality of the schedule include:

– minimum total completion time or makespan $Cmax$,
– maximum earliness $Emax = \max(e(j))$ for $j = 1 \ldots, n$,
– maximum lateness $Lmax = \max(l(j))$ for $j = 1 \ldots, n$,
– maximum tardiness $Tmax = \max(t(j))$ for $j = 1 \ldots, n$,
– the total number of late jobs (i.e jobs for which $t(j) > 0$), etc.

Most of the research reported in the literature is focused on the single objective case of shop scheduling problems, in which the makespan should be minimised. Some researchers have investigated machine scheduling problems from a multiobjective perspective (eg. [4, 113]) but the amount of literature in this area is still scarce compared to the single-objective case.

Four of the most well-known types of shop scheduling problems are the single-machine problem, the flowshop problem, the jobshop problem and the openshop problem, which are briefly described below.

Single-Machine Scheduling. This is the simplest case of machine scheduling problems, in which the set of n jobs have to be processed in a single machine. The problem is to find the sequencing of jobs that optimises the given criteria. For example, $1|d_j|Lmax$ denotes a single machine configuration in which the jobs have a due date and the criterion used to evaluate the quality of the schedule is the maximum lateness.

Flowshop Scheduling. There are n jobs or tasks that have to be processed in each of the m machines, i.e. each job consists of m steps or operations. The processing of each job is carried out in the same sequence through the processing stages, i.e. from the first to the last machine. After the processing of the job is finished in machine i the job joins the queue in machine $i+1$. Then, each machine i is used to process step i of each job. The problem is to find the sequence in which the jobs should be processed so that the given objectives are achieved. For example, $F_m|p(i, j) = p(1, j)|Cmax$ denotes a flowshop configuration in which each job has equal processing times for all its operations and the objective is to minimise the makespan.

Jobshop Scheduling. This is a more general case of the flowshop scheduling problem, in which the sequencing of each job through the machines is not necessarily identical. As in a flowshop, there are also n jobs consisting of m operations and m machines are available. The sequence of operations within each job are predefined and fixed. For example, $J_m|d_j|Cmax$ denotes a jobshop configuration in which all jobs have a due date and the objective is to minimise the makespan.

Openshop Scheduling. The openshop is a more general case of the jobshop scheduling problem. As before, there are n jobs consisting of m steps to be processed in m machines. The sequencing of each job through the machines can be different and finding the optimal sequencing for each of the n jobs is also part of the problem. Since the sequence of steps within each job has to be determined in addition to the jobs processing schedule, the search space is even larger than in the jobshop scheduling problem. For example, $O_3|pmtn, r_j|(Cmax + Lmax)$ denotes a 3-machine openshop configuration in which pre-emption ($pmtn$) is allowed, all jobs have a release date and the criteria used to evaluate the quality of the schedule is a sum of the makespan and the maximum lateness.

4 Multiobjective Approaches for Machine Scheduling

4.1 Introduction

Heuristic techniques are applied to obtain an acceptable schedule in a reasonable amount of processing time. Reviews of some of the specialised heuristics for job scheduling problems can be found in [11, 83, 96, 116]. Almost every type of metaheuristic has been applied to machine scheduling problems (see [11, 83, 116]). However, the design of efficient search operators, selection of adequate solution representations, tuning of parameters, etc. is still an art. When applying metaheuristics to machine scheduling problems, researchers have found that it is essential to incorporate knowledge about the problem domain, constraint-handling techniques, specialised operators and local search heuristics in order to obtain good results (eg. [54, 69, 90, 116, 117]). In this paper we are concerned with the application of multiobjective metaheuristics.

Most of the reported applications of multiobjective metaheuristics to multi-criteria machine scheduling consider two or three objectives and many have concentrated on flowshop scheduling problems. A literature survey on multicriteria scheduling problems up to 1995 is available in [92]. More recently, T'kindt and Billaut provided a good framework on multicriteria scheduling for any researcher and practitioner interested in this field [113]. In their book, the authors describe relevant concepts and ideas in the fields of multicriteria decision-aid and scheduling. They provide notations, formulations and a topology for single-criterion and multicriteria scheduling problems. They also describe several algorithms (exact and heuristic) for these problems.

4.2 Measuring the Effectiveness of Local Search

Marett and Wright presented a study on the application of three techniques based on local search to multiobjective flowshop scheduling problems [85]. Although their aim was not Pareto optimisation, we decided to include their work in this introductory paper because they made interesting observations regarding the effect of the complexity of these problems on the performance of local search heuristics. Since almost all the proposed multiobjective metaheuristics for scheduling include some form of local search, the results presented by Marett and Wright are of relevance to us. They assessed the performance of a simple descendent method, a tabu search technique and a simulated annealing algorithm according to the complexity of various multicriteria flowshop problems. They considered the following four criteria: total setup time (tst), total setup cost (tsc), total holding time (tht) and total late time (tlt). For each of these criteria, the minimisation of the corresponding cost value was taken as the objective. Test problems with 4 (all of the above criteria), 3 (tst, tsc and tht), 2 (tst and tsc) and 1 (each of the above criteria) objectives were created. All problems had 30 jobs and 3 machines. In the problems with more than 1 objective, a weighted sum of the cost values for each criterion was used as the total solution cost. For each criterion, a weight was set for each of the three machines (i.e. 12 weights in total) in order to produce total costs of the same order of magnitude in all test problems. Marett and Wright assumed problems with more objectives to be more complex (and hence harder to solve) than problems with a smaller number of objectives. One neighbourhood structure was used in the three techniques investigated: the swap or exchange of two jobs. The neighbourhood sampling was carried out in a systematic fashion using a set order without replacement. At the start of each algorithm, the order in which the neighbours are generated is made random and the whole ordering has to be used before it could be reused, even if any move to a new solution has been made by the method. They observed different performances of the three techniques for different degrees of problem complexity (assumed as explained above). But in general, they noted that exploring not all but a subset of neighbours produced much better results, an observation that was also made in [90] for single-objective flowshop problems.

Marett and Wright also proposed two metrics to measure the complexity of a combinatorial optimisation problem: *the mean steepest descent length* and *the first autocorrelation*. The first metric is a measure of the number of complete neighbourhoods that need to be examined before a local optimum is found. They made an estimation of this metric for each problem by executing a repeated steepest descent heuristic until a thousand descents had been made. The second metric is based on a random walk through the solution space and then observing the rate of improvements made. An estimate of the κ^{th} correlation is given by

$$r_\kappa = \frac{\sum_{t=1}^{q-\kappa}(y_t - \bar{y})(y_t + \kappa - \bar{y})}{\sum_{t=1}^{q}(y_t - \bar{y})^2} \quad . \tag{2}$$

where y_1 , $y_2 \ldots$,y_q are the successive values of the q solutions visited during the random walk and $\bar{y}$ is their average. Then, r_κ measures the correlation between the total cost of the current solution and the cost of the current solution κ moves ago. The authors used only the first autocorrelation r_1 in their experiments.

Marett and Wright recognised that it was not completely clear how the above metrics should vary with the complexity of the problem. However, they observed that a high value of *the mean steepest descent* implies that very few local minima exist and therefore, it would be hard for a neighbourhood search algortihm to find them. If this value is small, it is an indication of the existence of too many local optima, and the neighbourhood search technique would find difficult to identify the good ones. For *the first autocorrelation*, a value close to 1 is an indication of the existence of large *plateaux* in the solution space with few good solutions which are difficult to find. A value close to 0 implies that the solution space looks like a very *spiky* surface with lots of mountains and valleys. Then, the search algorithm would find it difficult to uncover any structure in the solution space. Marett and Wright proposed to use these metrics to obtain an indication of how difficult it is to carry out local search, and use this information to select the most appropriate local search technique to tackle each particular multiobjective problem.

4.3 Multiobjective Genetic Algorithms

Murata et al. proposed a multiobjective genetic algorithm for the flowshop problem with two and three objectives [91]. The criteria considered were: makespan, total tardiness and total flow time. Before selection, a vector of weights is generated at random and all the individuals in the population are evaluated using that vector. Then, two individuals are selected according to a probability function before applying the genetic operators to produce one offspring. A secondary population of non-dominated solutions is maintained and some individuals from this elite population are copied to the next generation. The randomly generated weights aim to specify different search directions towards the Pareto optimal front. In addition to the secondary population, elite individuals with respect to each of the k objectives are maintained. The two-point crossover and the shift mutation were used because they observed that these worked well in their previous work [90]. For the two-objective case, the weights were generated (evenly distributed over the interval [0,1]) according to (3) and the solution fitness calculated using (4) as shown below, where $N_{selection}$ is the number of selection steps in each generation of the algorithm, i.e. $N_{selection}$ individuals are produced in each generation.

$$w_1 = \frac{i-1}{N_{selection}-1} \text{ and } w_2 = 1 - w_1 \text{ for } i = 1, 2, \ldots, N_{selection} \ . \tag{3}$$

$$f(x) \ = \ w_1 f_1(x) + w_2 f_2(x) \ . \tag{4}$$

For the k-objective case, Murata et al. proposed to generate the weights and calculate the solutions fitness according to (5) and (6) respectively, where rnd_i and rnd_j are non-negative random numbers.

$$w_i = \frac{rnd_i}{\sum_{j=1}^{k} rnd_j} \text{ for } i = 1, 2, \ldots, k \ . \tag{5}$$

$$f(x) = w_1 f_1(x) + \ldots + w_k f_k(x) \ . \tag{6}$$

Then, k weights are generated in each of the $N_{selection}$ selection steps to choose a pair of parents for recombination. Murata et al. found that their approach with variable weights was capable of approximating the Pareto optimal set in non-convex fronts and produced better results than the vector evaluated genetic algorithm (VEGA) [104]. The VEGA algorithm is considered to be among the first genetic algorithms in which the concept of dominance was implemented for the evaluation and selection of individuals. In each generation, a group of individuals is selected according to one of the k objectives in the problem until k groups are formed. That is, each group of individuals excels in one of the k criteria. Then, the k groups are shuffled together and the genetic operators are applied to produce the new population.

4.4 Extensions to the Multiobjective Genetic Algorithm

The multiobjective genetic algorithm described above, was later hybridised with local search in [69] and applied to multiobjective flowshop scheduling problems in [68]. The new version used the strategy of specifying different random search directions for each selection of parents according to (5). But now, after each offspring is generated using the genetic operators, local search is applied to the new individual in order to improve it. The mutation operator was also used to explore the neighbourhood in the local search phase. The same vector of weights generated to select the parents was used to guide the local search and if no parents exist (an initial generated solution), random weights are used. The number of neighbours explored during the local search was a subset of the whole neighbourhood as suggested in [90] as a way of controlling the computation time spent by the local search. The elitist strategy was slightly modified so that the local search is also applied to some randomly selected individuals from the elite population. The authors compared their approach against the VEGA and against a genetic algorithm with fixed weights and found that the proposed algorithm outperformed these two methods. Ishibuchi and Murata also carried out experiments to assess the dependence of the their hybrid algorithm to parameters such as the number of neighbours examined in the local search, the number of non-dominated solutions copied from the secondary population and the multipliers used for the normalisation of objectives. From their results, they concluded that the algorithm was sensitive to these parameters.

The above multiobjective genetic local search algorithm was extended to a multiobjective cellular genetic local search algorithm [88]. In a cellular algorithm, each individual resides in a cell of a spatially structured space. A different weight vector is assigned to each cell so that for a k-objective problem the space is structured in a k-objective weight space. This cellular structure used by Murata et al. is similar to the concept used in the Pareto archived evolutionary strategy (PAES) of Knowles and Corne for diversity and niching [78]. The PAES algorithm uses an adaptive grid that divides the objective space to evaluate how much crowded the region in which each solution lies is (see [78] for full details).

Later, Murata et al. proposed a proportional weight specification method and incorporated it into the multiobjective genetic algorithm and into the cellular variant in order to examine the effect of this new mechanism on the performance of these algorithms in multiobjective flowshop scheduling problems [89]. The weights were generated systematically (not randomly as before) in order to allocate cells of uniformly distributed weight vectors. The distance between cells with weights $w = (w_1, w_2, \ldots, w_k)$ and $v = (v_1, v_2, \ldots, v_k)$ is measured with the Manhattan distance given by (7) and the neighbourhood of a cell is given by (8), where D is a predefined distance that is set as a parameter of the algorithm.

$$\text{distance}\ (w, v) = \sum_{i=1}^{k} |w_i - v_i| \ . \tag{7}$$

$$\text{neighbourhood}(w) = \{v | \text{distance}(w, v) \leq D\} \ . \tag{8}$$

To generate an individual in a cell, two parents are selected from its neighbourhood. The fitness during the selection of the neighbours is calculated using the weighted vector of the cell to which the individual is being generated. The cellular structure restricts the genetic operations to be performed on individuals that are not too far away. Murata et al. applied their algorithm to flowshop scheduling problems with two and three objectives. They compared their new cellular multiobjective genetic algorithm against their previous multiobjective genetic algorithm with random weights and with weights generated by the new proposed mechanism. They observed that the new weights generating method improved the performance of the algorithms and also found that the level of restriction in the genetic operations (D, the distance for neighbouring solutions) in the cellular approach had an effect on the performance of the algorithm. Later, Ishibuchi et al. modified the multiobjective genetic local search algorithm by selecting only good individuals for applying the local search phase instead of applying it to all the offspring [71]. In this new version of the algorithm, the authors addressed the two difficulties that they found when hybridising genetic algorithms with local search: how to specify the objective function and how to establish the balance between local search and genetic search. The two modifications proposed in the new version of their algorithm were:

1. Only a few good offspring are selected for applying local search.
2. The local search direction is specified according to the localization of the solution in the objective space.

Basically, they modified the step for selecting individuals for local search. A random vector of weights is generated and then, using tournament selection with replacement, one solution from the population is selected and added to the local search pool. Once this pool is complete, a number NLS of solutions are selected from this set for applying local search. The local search direction of each solution is specified by the weighted vector used in the selection of that solution when constructing the local search pool. The new population of solutions is composed by the improved NLS solutions and the other non-selected solutions in the local search pool. Ishibuchi et al. compared this version with their previous ones: without local search and applying local search to all offspring and found that the new version was more effective. They also compared the new proposed version against the strenght Pareto evolutionary algorithm (SPEA) [126] and the improved non-dominated sorting genetic algorithm (NSGA-II)[51]. Ishibuchi et al. observed that their modified algorithm was competitive with these two contemporary algorithms in terms of solution quality. In terms of computation time efficiency, their algorithm was better. They also analysed the effect of the number of solutions selected for local search on the performance of the algorithm and they noted that it was necessary to tune this parameter in order to obtain better results. In summary, the authors proposed the specification of an appropriate search direction for the local search by using tournament selection and the application of local search to only good solutions as additional strategies for establishing a good balance between local search and genetic search.

In [72] Ishibuchi et al. carried out additional experiments to assess a hybrid version of the SPEA that incorporated the same local search components as in their multiobjective genetic local search algorithm. In general, they concluded that the appropriate balance between local search and genetic search depends on two aspects: the algorithm and the available computational time. Recently, Ishibuchi et al. presented an updated version of their previous work where they included a hybrid version of the NSGA-II that also incorporates their local search components [73]. In a related paper, Ishibuchi and Shibata investigated the use of mating restriction in the SPEA and NSGA-II algorithms as a way to limit the crossover between solutions in the flowshop problem [70]. They found that, selecting dissimilar parents improved the search ability of these algorithms in small problems while selecting similar parents was beneficial in larger instances. They also observed that, although mating restriction seems to be beneficial, this depends not only on the problem size but also on the algorithm.

It can be noted that, since the implementation of the multiobjective genetic algorithm proposed in [91], additional strategies have been incorporated to create different versions of the algorithm and improve the results on multiobjective flowshop scheduling problems. It is noted that, the suggested modifications range from the adequate selection of genetic operators to fine-tuning the balance between local search and genetic search. In general, those papers have illustrated the importance of local search for the good performance of these algorithms when tackling multiobjective flowshop scheduling problems.

4.5 A Hybrid Multiobjective Evolutionary Algorithm

A hybrid evolutionary algorithm was proposed for the flowshop scheduling problem with two objectives (minimisation of makespan and total tardiness) by Talbi et al. [109]. The hybrid applied a genetic algorithm to obtain an approximation to the Pareto front and then employed local search to the obtained front. So, once a non-dominated front is obtained using the genetic algorithm, the local search explores neighbours of the solutions in this front and updates the set accordingly until no new non-dominated neighbours are found. The neighbourhood exploration was carried out using the mutation operator of the genetic algorithm. The crossover and mutation operators used were those employed in [91]. An interesting aspect of the study presented by Talbi et al. is that the authors investigated the following selection criteria:

1. The combination of objectives using weights.
2. The parallel selection strategy used in the VEGA algorithm [104].
3. The selection strategy used in the NSGA algorithm [106].
4. A non-dominated sorting selection.
5. A weighted average ranking, where individuals are ranked according to the different objectives separately.
6. An elitist method, where a population of non-dominated individuals is maintained and it participates in the selection for reproduction.

Talbi et al. observed in their experiments that elitist selection was the most beneficial and that the non-Pareto based selection schemes (combination using weights and weighted average ranking) seemed not to be suitable for the problem. They also found that, tuning the elitism pressure was important because high pressure intensifies the exploitation tendency of the good solutions while low elitism pressure favours exploration of new regions in the search space. Another interesting aspect of the study by Talbi et al. is that they compared three ways of fitness sharing: genotypic sharing, phenotypic sharing and a combined approach. In the solution space (genotypic sharing) the distance between two individuals x and y is measured according to the distance between the schedules (represented by a permutation) given by (9).

$$\begin{aligned} dist1(x,y) = |\{(i,j) \in J \times J | i \text{ precedes } j \text{ in the solution } x \\ \text{and } j \text{ precedes } i \text{ in the solution } y\ \}| \ . \end{aligned} \quad (9)$$

In the two-objective space (phenotypic sharing) the distance between two individuals x and y was given by (10).

$$dist2(x,y) = |f_1(x) - f_1(y)| + |f_2(x) - f_2(y)| \ . \quad (10)$$

The third approach combined the distances in both spaces, where $\gamma 1$ and $\gamma 2$ are parameters set to 4.0 and 1.0 respectively:

$$sh(x,y) = 1 - \frac{dist1(x,y)}{\gamma 1} \text{ if } dist1(x,y) < \gamma 1 \ , \ dist2(x,y) \geq \gamma 2 \ . \quad (11)$$

$$sh(x,y) = 1 - \frac{dist2(x,y)}{\gamma 2} \text{ if } dist1(x,y) \geq \gamma 1 \ , \ dist2(x,y) < \gamma 2 \ . \quad (12)$$

$$sh(x,y) = 1 - \frac{dist1(x,y) dist2(x,y)}{\gamma 1 \gamma 2} \text{ if } dist1(x,y) < \gamma 1 \ , \ dist2(x,y) < \gamma 2 \ . \quad (13)$$

Talbi et al. noted that when their algorithm used phenotypic sharing, it produced closer approximations to the Pareto front. But when using genotypic sharing, solutions were found in some areas that the other variant did not cover. They decided to use the combined sharing approach in their final implementation because it appeared to outperform the other two methods by helping to obtain closer approximations to the Pareto front and a better coverage of this front. In their experiments, they also observed that their hybrid evolutionary algorithm performed better as the problem size increased.

4.6 Dynamic Mutation Pareto Genetic Algorithm

Basseur et al. presented a method called dynamic mutation Pareto genetic algorithm and applied it to the flowshop scheduling problem with two objectives: minimisation of total makespan and minimisation of total tardiness [7]. The distinctive feature of their algorithm is that it uses different genetic operators in a simultaneous and adaptive manner during the search. In their approach, several mutation operators are given the same probability at the beginning of the search and then, they are chosen dynamically during the search. The individuals are evaluated before and after the application of the mutation operators. Then, for each mutation operator, an average growth value is calculated and used to adjust the probability assigned to each mutation operator. After applying a mutation operator M, a solution $M(x)$ is generated from a solution x. The progress of a mutation operator M applied to a solution x is 1 if the solution x is dominated by $M(x)$, 0 if x dominates $M(x)$ and 0.5 otherwise. Then, the average $Progress(M(i))$ is calculated by summing all the progresses of the mutation operator M and dividing it by the number of solutions to which the mutation operator was applied. The probability of each mutation operator is adjusted using (14) where η is the number of mutation operators and δ indicates the minimal ratio value permitted for each operator. That is, δ is a parameter that permits to keep each operator even if the progress of the operator is too poor.

$$P_{M(i)} = \frac{Progress(M(i))}{\sum_{j=1}^{\eta} Progress(M(j))} \times (1 - \eta \times \delta) + \delta \ . \tag{14}$$

In their implementation, Basseur et al. used two mutation operators: an exchange (swap) between jobs and the insertion operator, which is the same as the shift change operator used in [91]. They used fitness sharing with a combination of the distance in the solution and decision spaces (see also [109]). Their hybrid consisted of a genetic algorithm followed by a memetic algorithm applied only during a few generations due to its more expensive computational cost. They found improvements over the previous results reported in [109] in both the proximity to the Pareto front and the diversity of the solutions found.

4.7 A Semi-Exact Population Heuristic

Gandibleux et al. proposed the idea of first generating the set of supported (non-dominated solutions produced using weighted vectors) solutions using an exact or heuristic method and then, use these solutions to improve the front by applying a population heuristic [62]. The supported solutions are considered to hold good genetic information. These solutions help to achieve a faster convergence to the Pareto front and also to maintain the diversity of the population. They applied their concept to two bi-criteria combinatorial optimisation problems. One was the single machine scheduling problem (namely permutation scheduling) with two objectives: the minimisation of the total flow time and the minimisation of the maximum tardiness. The other problem was the bi-ojective knapsack problem. The main features of the population heuristic that they used in the second phase of their approach are:

– All solutions ranked one with the non-domination ranking mechanism are copied to the next generation.
– During selection, some good solutions with respect to each objective are copied to the new population as in the VEGA algorithm.
– Among the solutions not selected as above, tournament selection is applied based on dominance with sharing.
– In the initial population, besides the solutions generated randomly, some good solutions with respect to each objective are computed and added to the initial population.
– Local search is applied to all elite individuals except to those that already received local search in the previous generations.

Gandibleux et al. noted that seeding elite solutions permitted the propagation of the superior genetic information to other individuals during the evolution process. Also, when all supported solutions were used to seed the search, the computation time and the number of generations needed was reduced considerably. They suggested that this two-phase method or semi-exact approach can be very useful in problems for which efficient methods exist to solve the single-objective version of the problem or for problems for which efficient greedy algorithms exist.

4.8 Implementations of the Non-Dominated Sorting Genetic Algorithm

Bagchi applied the original NSGA and also an extension of that algorithm to multiobjective flowshop, jobshop and openshop scheduling problems in [4, 5]. The extended approach, called the elitist non-dominated sorting genetic algorithm (ENGA), was an elitist version of the original algorithm in which the selection mechanism was modified to consider the parents and the offspring to form the next generation. Bagchi observed that the non-dominated sorting mechanism augmented with elitism was capable of improving the speed of convergence towards Pareto optimal solutions.

Brizuela et al. also applied the NSGA to the flowshop scheduling problem with three objectives: minimisation of the makespan, minimisation of the mean flow time and minimisation of mean tardiness [13]. They studied the effect of the genetic operators used on the dominance properties of the solutions generated. They compared three mutation operators and observed an influence of the operator used on the quality of the non-dominated solutions generated. They suggested that this effect can be translated into a concept of *non-dominated local search*. Here, the neighbourhood search operators can be adapted during the search according to their influence in the quality of non-dominated solutions produced. A second set of experiments was carried out using three crossover operators. The aim of these experiments was to determine whether or not the distance between parents in the solution space had an influence in the dominance relation between the parents and the offspring after the crossover. They observed that a combination of the genetic operators used in [91] performed the best. They also used two distance measures, one in the solution space and the other in the objective space. For measuring the distance between two solutions x and y in the solution space, a matrix n x n is associated with each permutation of the n jobs representing a schedule. Each element of the matrix $a_{ij} = 1$ if job j is scheduled before job i and $a_{ij} = 0$ otherwise. Then, the normalised domain distance between two individuals x and y is given by (15) where $\oplus$ represents the exclusive-or logical operation and $n(n-1)$ is the maximum number of different elements between two given associated matrices:

$$dn(x,y) = \frac{\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}(x) \oplus a_{ij}(y)}{n(n-1)} \quad . \tag{15}$$

The Euclidean distance was used to measure the difference between individuals in the objective space. The objective function distance (ofd) between two solutions x and y with k objectives is given by (16).

$$ofd(x,y) = \sqrt{\sum_{j=1}^{k}(f_j(x) - f_j(y))^2} \quad . \tag{16}$$

Brizuela et al. applied the selected operators to the NSGA and outperformed the results obtained by the modified version of [4]. They noted that their experiments offered an insight into how non-dominated local search can be performed. This is because different operators produce different results with respect to non-dominance and this could be a first step in an analysis of the landscape in multiobjective combinatorial optimisation problems. A recent related study by Brizuela and Aceves revealed that an order-based crossover operator outperformed the other operators tested when implemented in the NSGA algorithm and applied to flowshop scheduling problems with three criteria: makespan, mean flow time and mean tardiness [12].

5 Timetabling Problems

5.1 Introduction

Timetabling is the activity of scheduling a set of meetings or events in such a way that certain requirements and constraints are satisfied (see [49]). A common feature of many real world timetabling problems is that there are a certain number of constraints (soft and hard). In timetabling, the allocation of resources other than people and locations for the meetings is usually not considered to be a part of the problem. In many timetabling problems, the meetings to be scheduled are already specified and the problem is to schedule them into the available timeslots and locations. However, in some timetabling problems the creation of meetings (relationships between the entities such as teacher-class or exam-invigilator) is also part of the timetabling activity. There has been significant recent research in the area (eg. see [17, 18, 25, 32]). Timetabling problems include: educational timetabling (university and school timetabling), sports timetabling, employee timetabling, transport timetabling and others such as conference timetabling. This Sect. concentrates on educational timetabling, which is a particularly well investigated problem.

5.2 Educational Timetabling Problems

An effective timetabling in academic institutions is crucial for the satisfaction of educational requirements and the efficient utilisation of human and space resources [97]. Educational timetabling problems have many variants including the school timetabling problem (class-teacher timetabling), the university course timetabling problem and the university examination timetabling problem. Many models and formulations have been proposed to describe educational timetabling problems. This Sect. presents a brief description of some educational timetabling problems. For a more detailed analysis refer to [38, 46, 49, 102].

School timetabling. In general terms, this problem usually refers to assigning timeslots and locations so that meetings between teachers and classes can take place (eg. see [6]). The main two features of this type of problem are: 1) the students are grouped in fixed classes and, 2) the meetings and the number of them are predefined, i.e. the curricula of each class is usually known and fixed. Teachers are usually pre-assigned to courses and the number of sessions of each course that the classes have to take is also known. The groups of students are not necessarily disjoint but in general most of them are.

University course timetabling. This activity refers to the assignment of timeslots and locations so that meetings between lecturers and students can take place (eg. see [37, 10]). University students usually have a range of optional courses and therefore, they are not pre-assigned to meetings. The assignment of locations for the lectures may also be considered to be a part of the problem because the size and requirements of each group of students varies more than in the school timetabling problem.

University examination timetabling. This activity refers to the assignment of timeslots and locations so that students can take exams (eg. see [35, 36, 24]). There are some distinct differences between university course timetabling and university examination timetabling. This difference can be illustrated by noting that it is common to assign several exams to one (large) room at the same time. This is clearly nor possible for course timetabling.

5.3 Feasibility and Timetable Quality

The feasibility of solutions in the above timetabling problems varies according to the particular instance. Different institutions have very different ideas about what constitutes a good timetable (eg. see [24]). In general, hard constraints must be satisfied. For example, no person (teacher, lecturer or student) can be present in two meetings at the same time. Soft constraints are those which are desirable but not essential. Examples include spread, compactness and balance of the timetable, free timeslots between meetings, meetings-free days, similarity with previous timetables, timetable flexibility, etc. Blakesley et al. studied the problem of constructing educational timetables from a very interesting perspective: the student's needs [10]. They noted that constructing timetables that satisfy faculty and student preferences may have an unanticipated negative effect on the students needs because the availability of courses is reduced and the course completion time could be enlarged as a consequence. The number and variety of constraints (hard and soft) existing in educational timetabling problems makes it almost impossible to list all of them. For details of soft constraints across all broad classes of educational timetabling problems see [6, 10, 24, 36, 37, 49, 102].

6 Multiobjective Approaches for Educational Timetabling

6.1 Introduction

Although it is generally acknowledged that multiple criteria exist to evaluate solutions in educational timetabling problems, few multiobjective metaheuristics have been applied to this class of problems. It has been pointed out that in the real-world, decision-makers prefer to have a selection of possible timetables from which to choose the most appropriate one [35]. However, the vast majority of approaches use a weighted sum of penalties for evaluating the fitness of solutions and only one timetable (the one with the lowest total penalty) is produced as a result. The goal is usually to attempt to obtain a lower penalty according to criteria defined in the algorithm. But the workability of a timetable depends on how complete and realistic these criteria are. In practical problems, there are three main reasons for the imperfection of timetables: inaccurate prediction of student enrollment, mistakes in the events list or resources availability, and inadequate selection of weights for the soft and hard constraints [97]. Some papers have reported on the application of strategies for producing various alternative solutions. For example, the combination of graph colouring techniques with heuristics was one of the first approaches that were used to produce several (not simultaneously) reasonable timetables [119]. This is overviewed in [22].

6.2 Multi-Phased Approaches

Thompson and Dowsland implemented a multi-phased simulated annealing algorithm for timetabling examinations [111, 112]. The authors modelled the problem as a graph colouring problem and the neighbourhood structure used was the change of colour in a single vertex that corresponds to moving an exam from one timeslot to another. In their approach, the first phase is used to tackle the first objective: the satisfaction of all the hard constraints while the second phase is used to optimise the secondary objective: the minimisation of soft constraints violations. Since the decisions made in previous phases have an influence in the solutions that can be reached in later phases (the solution space may be disconnected), their multi-phased simulated annealing algorithm permits the alteration of decisions made in earlier phases as long as the quality of the solutions with respect to earlier objectives does not deteriorate. In our opinion, the papers by Thompson and Dowsland are among the best reported studies on using simulated annealing for timetabling problems and among the few that approach these problems as multicriteria optimisation problems. A similar study of the application of a multi-phased approach to examination timetabling and practical lab sessions timetabling was reported in [53]. In that investigation, the author pointed out that the decision on which objectives or constraints are to be tackled in each phase depends not only on the importance of the objective but also on the difficulty to achieve it and on its relation with the neighbourhood structure defined (this has also been noted by other researchers [85]).

Ideally, when treating objectives in phases, one objective has to be tackled in each phase in order to eliminate the use of weights. However, this is not always possible in timetabling problems because the number of different objectives can be very large. Therefore, like in the multi-phased approaches described above, the constraints have to be grouped and each group is tackled in each stage of the algorithm. This originates the problem of still having to determine weights to reflect the relative importance of the constraints in the same group. Another drawback of multi-phased methods, is that the solution obtained in an early phase is usually fixed and this may lead to poor solutions in later phases because the solution space may be drastically reduced. A strategy to avoid this can be the implementation of backtracking mechanisms as proposed in [111].

Another multi-phased approach was described in [3] for the course timetabling problem in Spanish universities. The first phase is an interactive process in which students select their courses and the second phase uses a tabu search algorithm for constructing the timetable. In the assignment phase, the following criteria were used to measure the quality of timetables: students course selections must be respected (this is the only hard constraint imposed), section enrollments should be balanced, sections maximum capacities must not be exceeded, clashes in students timetables should be avoided, students timetables should be as good as possible (measured in terms of number of lectures per day, number and length of holes in the timetable and moves between buildings), student language preferences should be respected. The construction of the timetables is divided into two steps. In the first step, the best set of timetables is constructed for each student according to their selection of courses and without taking into consideration the balance of the course sections. In the second step, a global timetable is constructed by combining the timetables of all students to obtain balanced section enrollments and minimise the decrease of the quality of each student timetable.

6.3 Multicriteria Decision-making Techniques

Burke et al. approached the multicriteria examination timetabling problem by grouping nine different constraints into three categories: 1) room capacity, 2) proximity of exams and 3) time and order of exams (see [16] for full details). The nine considered criteria are incommensurable and partially or totally conflicting (at least in their problems). Only one hard constraint was considered in their problems: that conflicting exams must be scheduled in different timeslots. As in the multi-phased approaches described in the previous Sect., the approach by Burke et al. also requires the setting of weights for expressing the relative importance of the different criteria within the same group. They used compromise programming as the basis for their solution method [124]. In compromise programming, the strategy is to find compromise solutions that are close to the ideal point. An ideal point is defined in the criteria space as the vector containing the best possible value for each criterion. Their algorithm uses two phases. In the first one, timetables of high quality are constructed using a graph-colouring heuristic. The second phase attempts to improve the timetables by using a hill-climber and a heavy mutation operator. In each step of the preference space search, multiple applications of the hill-climber are followed by one application of the mutation operator until the distance between the solution and the ideal point has not decreased for a predefined number of iterations. One final solution is chosen from the set of obtained timetables. This is the one with the minimum distance from the ideal point. The authors noted in their experiments that the weights for each criterion, and some parameters of the function to measure the distance from each solution to the ideal point, had a significant influence on the quality of the solutions obtained. This permits the decision-makers to express their preferences before the search. Petrovic and Bykov proposed an approach based on the specification of trajectories in the objective space to tackle multicriteria examination timatabling problems [95]. In their method, the decision makers express their preferences by specifying a point in the k-objective space. Then, a line is drawn between the image of a randomly generated solution and the reference point. A local search is conducted, following the defined trajectory in order to find a solution that is as good as (or better than) the reference solution. Weights are dynamically varied during the search in order to maintain the new solutions close to the defined trajectory. Petrovic and Bykov suggested that their method is more transparent to the decison-makers because it allows them to express their preferences without the need for setting weights.

6.4 Multiobjective Evolutionary Algorithms

One of the few applications reported in the literature of Pareto-based genetic algorithms to timetabling problems is the one by Carrasco and Pato [34]. In that paper, the authors tackled a bi-objective school timetabling problem with a modified version of the NSGA described in [106]. The two conflicting objectives were the minimisation of soft constraint violation from two competitive perspectives: teachers and classes. Penalties were assigned to the violation of constraints and the authors observed that the algorithm was very sensitive to the selection of these penalties. Since the NSGA uses fitness sharing, a measure of distance between two timetables x_i and x_j is required. Carrasco and Pato used (17) for this purpose.

$$d(x_i, x_j) = \frac{\sum_{k=1}^{L} t(k, x_i, x_j)}{L} \ . \tag{17}$$

where L is the total number of lessons and $t(k, x_i, x_j)$ equals 1 if the lesson k occupies the same period in both solutions x_i and x_j and 0 otherwise.

They used a direct representation in which a bi-dimensional matrix represents the timetable. Each row represents one room and each column represents one timeslot. Then, each cell in the matrix contains the lesson that will be taught in the given room at the given period. The creation of meetings (teacher-lesson-class) is carried out before the construction of the timetable. A constructive heuristic that starts scheduling the most difficult lessons first (in terms of lesson duration and the preferences of teachers and classes) was used to initialise the population. An elitist secondary population, composed of some of the non-dominated solutions from the main population was used. Specialised crossover and mutation operators were designed for the chromosome representation described above. The crossover operator was specially designed to create two offspring, one teacher-oriented and the other class-oriented. That is, their specific crossover operator attempts to produce elite timetables with respect to each of the objectives. A repair operator was employed to fix the overlaps that are normally created by the crossover. The mutation operator consisted of removing a number of lessons from the timetable. Then, these lessons are re-scheduled so that the total penalisation is minimised. Although the authors mentioned that several experiments were carried out to assess the effect of the fitness sharing mechanism and the secondary population in the genetic algorithm, a detailed discussion of these effects was not provided. However, they observed that the use of the secondary population was helpful and the algorithm found better timetables than those constructed manually.

Another application of a Pareto-based genetic algorithm was reported by Paquete and Fonseca [94] where the authors implemented a multiobjective evolutionary algorithm [59] for the examination timetabling problem. In that paper, the authors used a direct chromosome encoding and a mutation operator with an independent mutation probability for each gene in the chromosome (no recombination operator was implemented). Each gene in the encoding represents an exam. The mutation probability for each gene is calculated according to the number of timeslots available for each exam and the degree of involvement of that exam in the violation of constraints. Their experiments sought to compare three aspects: Pareto ranking against linear ranking, independent mutation against single-position mutation and different levels of mutation bias. They reported that: the use of Pareto ranking produced better performance in the algorithm, no difference was observed between the two mutation strategies and although a difference was observed between groups of mutation rates, no more details were provided. One interesting aspect in the study by Paquete and Fonseca, is that experiments were carried out considering the execution time as an additional objective and the independent mutation operator produced better performance in these experiments. Another interesting observation made by Paquete and Fonseca, was that each objective handling technique performed better in its own case. That is, Pareto ranking provided better coverage of the objective space while linear aggregation was more effective in minimising the total number of constraint violations across the runs. This may represent an important clue for the implementation of non-dominated local search (see Sect. 4.8) in timetabling problems.

7 Multiobjective Approaches for Personnel Scheduling

Personnel scheduling refers to the construction of shift patterns for employees and it is also known as rostering or employee timetabling [120]. Personnel scheduling problems are multicriteria problems that have certain similarities (but also distinct differences) with educational timetabling problems. They also involve the construction of a schedule that satisfies as much as possible a number of diverse criteria. The criteria are also usually incommensurable and in conflict as they represent the interests of employees and employers and also working regulations. Like in educational timetabling, few multiobjective metaheuristics have been applied to personnel scheduling problems. This Sect. attempts to describe, in a brief manner, some of these approaches.

Jaszkiewicz applied the Pareto simulated annealing algorithm to a multiobjective nurse scheduling problem in Polish hospitals [74]. This algorithm is a population-based extension of simulated annealing proposed for multiobjective combinatorial optimisation problems [48]. The population of solutions explore their neighbourhood similarly to the classical simulated annealing, but weights for each objective are tuned in each iteration in order to assure a tendency to cover the trade-off surface. The weights for each solution are adjusted in order to increase the probability of moving away from its closest neighbourhood in a similar way as in the multiobjective tabu search algorithm of Hansen [66]. In the nurse scheduling problem tackled by Jaszkiewicz in [74], five objectives were identified, four minimisation objectives and one maximisation objective. One initial solution was generated by a constraint-based programming technique and multiple copies of this solution formed the initial population. Three types of neighbourhood structures were defined. In each iteration, one of these structures was selected at random to generate the candidate solutions. Only feasible solutions were explored and if the chosen move violated any constraint another move was tried. The results were reported on a small test problem and the goal of producing better schedules that those generated manually was achieved.

A similar multicriteria approach to the one in [16] also using compromise programming was presented for the nurse scheduling problem in [19]. The main algorithm (based on tabu search) constructs a feasible schedule and iterative improvement of this initial schedule is tried by moving shifts between nurses and never accepting infeasible solutions. The ideal and anti-ideal points are estimated in order to make the mapping from the criteria space onto the preference space. Each personal schedule is considered separately and the sum of distances is used to measure the schedule fitness.

El Moudani et al. described a bi-criterion approach for the airline crew rostering problem [57]. This airline crew rostering problem refers to assigning crew staff to a set of pairings covering all the scheduled flights. A pairing is a sequence of flights that starts and ends at the same airline base while meeting all relevant legal regulations. In this problem, hard constraints include the regulations of Civil Aviation and the airline's internal agreements. Soft constraints include: internal company rules, union agreements, office duties, holidays, assignment preferences and others. The authors tackled the airline crew rostering problem from a bi-criterion perspective in which the first goal was to minimise airline operations cost and the second goal was to maximise the crew staff overall degree of satisfaction. The initial population of solutions was generated using a greedy heuristic specially designed to attempt the maximisation of the overall degree of satisfaction regardless of the operation costs. After this initial population is created, genetic operators are applied to generate new solutions with reduced operations cost at the expense of perhaps reduction on the degree of crew satisfaction. A direct chromosome representation was used, in which each gene represents the pairing and the allele represents the crew member that has been assigned to that pairing. Three specific domain operators were implemented: crossover, mutation and inversion. A local search heuristic was designed to restrict the search space of the mutation and the inversion operators in order to speed-up the discovery of promising solutions. The authors reported that the application of the greedy heuristic to initialise the population required a very short computation time. However, in the subsequent application of the genetic operators, these operators did not show equivalent performance. They noted that the crossover operator was very time consuming (mainly because the set of constraints to be checked was large) and it did not contribute too much to produce new promising solutions. On the other hand, the mutation and inversion operators appeared to be more efficient in the generation of new promising solutions with relatively moderate computing times.

8 Other Relevant Research

8.1 Introduction

This Sect. discusses some of the research recently reported in the literature and that seems to be relevant for future research on multiobjective combinatorial optimisation in general and multiobjective scheduling problems in particular. One aspect that has been recently investigated, is the complexity of the landscape in multiobjective combinatorial optimisation problems. Another aspect is the effect that the evaluation method, used to discriminate between solutions during the search, has on the performance of the algorithm. The adaptation of operators during the search is another interesting issue discussed here.

8.2 Complexity of the Landscape

The paper by Wright and Marett was one of the first attempts to assess the performance of local search algorithms according to the complexity of the landscape in multiobjective problems [122]. To study the shape of the landscape, they measured the correlation between the sum of objectives' improvements and the sum of objectives' detriments when reaching local optima in a steepest descent run. When this correlation is close to $+1$, there is some conflict between the objectives. When the correlation is close to 0, the objectives are dissimilar or not affecting each other. When the correlation is close to -1, the objectives cooperate or reinforce each other. In multiobjective optimisation problems, some objectives may reinforce each other, conflict or be completely uncorrelated. Also, the improvement and detriment of each objective may be different and not constant during the search, not only in their value, but also in the frequency in which they change. These two aspects are related to the properties of the landscape and by studying them, an idea of the complexity of multiobjective combinatorial optimisation problems can be obtained. Another contribution in this direction is the study carried out by Knowles and Corne to analyse the landscape of the multiobjective quadratic assignment problem (mQAP) [80]. They proposed some metrics to measure the correlation between nearby optima in the mQAP. Then, they proposed to use this information to decide which hybrid strategy (incorporating local search) would be more appropriate to approach the Pareto front: 1) to approach the Pareto front and then spread around from there, 2) start the search repeteadly from random solutions or, 3) use a gradual approach towards the Pareto front from all directions in parallel.

8.3 Effect of the Evaluation Method

Another important aspect that has been investigated by some researchers, is the effect of the evaluation method used to discriminate between solutions during the search in Pareto optimisation. The use of subcost guided search was proposed by Wright to deal with compound-objective timetabling problems [121]. In that approach, an improvement of a subcost (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 on in the process. This is because the improvement in one aspect of the solution (a subcost), allow us to conduct a kind of guided diversification towards promising areas of the solution space. Wright carried out experiments with simulated annealing and threshold acceptance and found that the use of subcost guided search improved the performance of both algorithms. Kokolo et al. proposed the concept of α-dominance, which is a relaxed dominance relation [81]. In α-dominance, a small detriment in one or more of the objectives is permitted if an attractive improvement in the other objective(s) is achieved. The hope is that by accepting α-dominating solutions, the search can be widened and the connectedness of the search space can be improved because α-dominating solutions may serve to reach more non-dominated solutions. Burke and Landa Silva applied this concept of relaxed dominance to the multiobjective optimisation of space allocation problems in academic institutions [26, 29]. They compared the performance of an evolutionary annealing algorithm and the PAES approach with respect to the form of dominance used. They found that when using the relaxed dominance, both algorithms obtained better non-dominated fronts. Additional experiments showed that this behaviour was not observed in the algorithms when the hard constraints in the test problems where treated as soft constraints. That is, when the conditions of feasibility were relaxed so that it was easier to visit feasible solutions. Laumanns et al. proposed the concept of ϵ-dominance (ϵ-dominance uses the same concept as the α-dominance proposed earlier by Kokolo et al., i.e a relaxed from of dominance) [82]. They suggested this form of dominance to implement better archiving strategies that overcomes the difficulty of multiobjective evolutionary algorithms have in converging towards the optimal Pareto front and maintain a wide diversity in the population at the same time.

8.4 Use of Adaptive Operators

Applying different operators or heuristics at different stages of the search, or according to the localisation of the solutions with respect to the Pareto optimal front, may also be beneficial. For example, Salman et al. proposed an approach based on a co-operative team of simple heuristics that generate non-dominated solutions for the multiple knapsack problem in a short computation time [101]. The team of heuristics co-operate in such a way that the solutions generated by one heuristic can be improved by another one, or the adequate team of heuristics can be formed to generate solutions for the given problem. Another option is to implement a set of local searchers that attempt to achieve self-improvement and ask for the help of other searchers in the population (perhaps by mating or sharing information) when they cannot achieve further improvement [27]. In this way, the interactions between individuals are minimal and they are carried out in an asynchronous manner. Therefore, the need for niching and fitness sharing strategies to maintain diversity is also reduced. In another example of co-operating heuristics, Burke et al. investigated hyperheuristics (heuristics to select heuristics) to solve two different timetabling problems (course timetabling and nurse rostering [21]. A hyperheuristic as a strategy that is able to choose between a set of so-called low level heuristics [20]. This selection is based solely on performance indicators and not in the knowledge of the problem. Then, the hyperheuristic decides which heuristic to call at each moment during the search. The application of hyperheuristic approaches for Pareto optimisation has been proposed in [28].

9 Final Remarks

It is not within the scope of this paper to present a comprehensive review of multiobjective scheduling problems. It must be stressed that, in particular for machine scheduling problems, there exist in the literature very complete studies on the multiobjective optimisation of these problems (eg. [4, 113]). This paper is focused on the application of modern multiobjective metaheuristics for the optimisation of some types of multiobjective scheduling problems including machine scheduling, educational timetabling and personnel scheduling. The following concluding remarks can be made:

Problem formulation. The conditions of feasibility and the criteria used to measure the quality of solutions in multiobjective scheduling problems vary enormously between the different problem classes (machine scheduling, educational timetabling and personnel scheduling) and between particular instances. Machine scheduling problems (considering the single-objective case too) are among the scheduling problems for which more benchmark theoretical models and test problems exist. Contrary to this, educational timetabling and personnel scheduling problems lack a very large set of widely accepted benchmark models and test problems. In multiobjective machine scheduling problems several criteria have been clearly identified (makespan, tardiness, earliness, lateness, etc.). In educational timetabling and personnel scheduling problems, the criteria that define the multiobjective nature of the problem vary largely between instances.

The application of modern multiobjective metaheuristics. The application of these techniques, and in particular multiobjective evolutionary algorithms, to multicriteria scheduling problems is scarce. This is particularly true for educational timetabling and personnel scheduling. Multiobjective machine scheduling problems (and in particular multiobjective flowshop scheduling) are the problems for which more reports on the application of modern multiobjective metaheuristics exist in the literature. Within educational timetabling problems, the variant that has received more attention from the multiobjective perspective appears to be examination timetabling.

Useful strategies. Several strategies can be identified in the applications considered in this paper. The importance of local search and problem domain knowledge is evident for obtaining good results. Even in recent approaches, local search continues to play an important role, particularly when tackling educational timetabling and personnel scheduling problems. Weighted aggregating functions are still widely used in many approaches and they appear to be adequate when problems are highly constrained. Graph colouring heuristics are widely used in methods for educational timetabling problems and they can be implemented as operators in more elaborated approaches. In educational timetabling and personnel scheduling problems, due to the existence of lots of hard constraints, accepting infeasible solutions during the search has been helpful to improve the connectedness of the solution space and to widen the search in a number of applications. Also, the use of elite solutions according to each of the criteria has been helpful in some recent approaches.

Problem domain knowledge. In the few reported implementations of multiobjective evolutionary algorithms to multicriteria scheduling problems, several components such as representation schemes, initialisation strategies, genetic operators and repairing mechanisms need to be specially designed using problem domain knowledge. Also, in many approaches, only mutation operators are used within evolutionary algorithms for scheduling problems. It is often the case that these operators are in fact, very elaborate heuristics designed to bias the search towards promising regions.

Some promising research directions in the field of multiobjective scheduluing and timetabling are proposed below.

A. Exploit the experiences obtained from research in some of these problems (such as multiobjective flowshop) to produce successful approaches in other multiobjective scheduling problems. For example, the use of weighted vectors to specify search directions towards the Pareto optimal set, the tuning of local search, the selection of adequate genetic operators, the balance between local search and genetic search in hybrid approaches, the use of elitist strategies, the study of the impact of selection mechanisms and diversity measures on the performance of the algorithm, the adaptation of genetic operators probabilities during the search, the use of exact methods or heuristics to quickly approximate the Pareto optimal front followed by heuristics designed to improve the quality of this approximation, etc.

B. Given the importance of local search in this context, it would be interesting to put more effort into studying the landscape of multiobjective scheduling problems in order to design better local search components. Also, some strategies that have been proposed to improve local search for single-objective combinatorial optimisation could be incorporated into multiobjective metaheuristics. For example, the use of various neighbourhood structures or teams of local search heuristics according to the objective being optimised might be useful. Another example is changing the fitness landscape for improving the connectedness of the solution space. Using different evaluation methods to discriminate solutions during the search (aggregating functions and relaxed forms of dominance) can help to obtain better results for multiobjective scheduling problems.

C. Multicriteria educational timetabling and personnel scheduling problems need to be investigated in order to identify the criteria that should be considered when tackling these problems from a multiobjective perspective. It is also important to investigate the conflicting and incommensurable nature of these criteria.

Acknowledgement. The authors thank the anonymous referees for their feedback. Their comments helped us to substantially improve the quality of this paper and to make it more readable. We would also like to thank Vincent T'kindt for his valuable advice in preparing this paper.

References

1. Aarts E., Korts J., Simulated Annealing and Boltzman Machines, Wiley, 1998.
2. Aarts E., Lenstra J.K. (eds.), Local Search in Combinatorial Optimisation, Wiley, 1997.
3. Alvarez-Valdes R., Crespo E., Tamarit J.M., Assigning Students to Course Sections Using Tabu Search, Annals of Operations Research, Vol. 96, pp. 1-16, 2000.
4. Bagchi T.P., Multiobjective Scheduling By Genetic Algorithms, Kluwer Academic Publishers, 1999.
5. Bagchi T.P., Pareto-Optimal Solutions for Multi-objective Production Scheduling Problems, In: [125], pp. 458-471, 2001.
6. Bardadym V.A., Computer-aided School and University Timetabling: The New Wave, In: [32], pp. 22-45, 1996.
7. Basseur M., Seynhaeve F., Talbi E.G., Design of Multi-objective Evolutionary Algorithms to the Flow-shop Scheduling Problem, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, pp. 1151-1156, 2002.
8. Baykasoglu A., Owen S., Gindy N., A Taboo Search Based Approach to Find the Pareto Optimal Set in Multiple Objective Optimisation, Engineering Optimization, Vol. 31, pp. 731-748, 1999.
9. Belton V., Stewart T.J., Multiple Criteria Decision Analysis - An Integrated Approach, Kluwer Academic Publishers, 2002.
10. Blakesley J.F. Murray K.S., Wolf F.H., Murray D., Academic Scheduling, In [17], pp. 223-236, 1998.
11. Blazewicz J., Domschke W.,Pesch E., The Job Shop Scheduling Problem: Conventional and New Solution Techniques, European Journal of Operational Research, Vol. 93, pp. 1-33, 1996.
12. Brizuela C.A., Aceves R., Experimental Genetic Operators Analysis for the Multi-objective Permutation Flowshop, In: [60], pp. 578-592, 2003.
13. Brizuela C., Sannomiya N., Zhao Y., Multi-objective Flow-Shop: Preliminary Results, In: [125], pp. 443-457, 2001.
14. Brucker P., Drexl A., Mohring R., Neumann K., Pesch E., Resource-constrained Project Scheduling: Notation, Classification, Models and, Methods, European Journal of Operational Research, Vol. 112, pp. 3-41, 1999.
15. Brucker P., Knust S., Complexity Results for Scheduling Problems, available on-line at http://www.mathematik.uni-osnabrueck.de/research/OR/class/, 16 July 2003.
16. Burke E., Bykov Y., Petrovic S., A Multicriteria Approach to Examination Timetabling, In: [25], pp. 118-131, 2001.
17. Burke E.K., Carter M.W. (eds.), The Practice and of Automated Timetabling II: Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT 97), Lecture Notes in Computer Science, Vol. 1408, Springer, 1998.

18. Burke E.K., De Causamaecker P. (eds.), 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. 2740, Springer, to appear, 2003.
19. Burke E.K., De Causmaecker P., Petrovic S., Vanden Berghe G., A Multi Criteria Meta-heuristic Approach to Nurse Scheduling, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, pp. 1197-1202, 2002.
20. Burke E.K., Hart E., Kendall G., Newall J., Ross P., Schulemburg S., Hyper-heuristics: an Emerging Direction in Modern Search Technology, In: Glover F.W., Kochenberger G.A. (eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 2003.
21. Burke E.K., Kendall G., Soubeiga E., A Tabu-Search Hyper-Heuristic for Timetabling and Rostering, Accepted for Publication in the Journal of Heuristics, 2003.
22. Burke E.K., Kingston J., De Werra D., Perspectives on Timetabling, to appear in the Handbook of Graph Theory (edited by Jonathan Gross and Jay Yellen), to be published by Chapman Hall/CRC Press, 2003.
23. Burke E.K., Elliman D.G., Weare R., A University Timetabling System Based on Graph Colouring and Constraint Manipulation, Journal of Research on Computing in Education, Vol. 27, No. 1, pp. 1-18, 1994.
24. Burke E.K., Elliman D.G., Ford P.H., Weare R.F., Examination Timetabling in British Universities - A Survey, In: [32], pp. 76-90, 1996.
25. Burke E.K., Erben W. (eds.), 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. 2070, Springer, 2001.
26. Burke E.K., Landa Silva J.D., Improving the Performance of Multiobjective Optimizers by Using Relaxed Dominance, Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), Singapore, 2002.
27. Burke E.K., Landa Silva J.D., On the Performance of Hybrid Population-Based Metaheuristics Based on Cooperative Local Search, Technical Report, Available form the authors, 2003.
28. Burke E.K., Landa Silva J.D., Soubeiga E., Hyperheuristic Approaches for Multiobjective Optimisation, In: Proceedings of the 5th Metaheuristics International Conference (MIC 2003), Kyoto Japan, pp. 11.1-11.6, August 2003.
29. Burke E.K., Landa Silva J.D., The Influence of the Fitness Evaluation Method on the Performance of Multiobjective Optimisers, Technical Report, Available form the authors, 2003.
30. Burke E.K., Newall J.P., Weare R.F., A Memetic Algorithm for University Exam Timetabling, In: [32], pp. 241-250, 1996.
31. Burke E.K., Newall J.P., Weare R.F., Initialisation Strategies and Diversity in Evolutionary Timetabling, Evolutionary Computation, Vol. 6, No. 1, pp. 81-103, 1998.
32. Burke E.K., Ross P. (eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT 1995), Lecture Notes in Computer Science, Vol. 1153, Springer, 1996.
33. Burke E.K., Smith A., Hybrid Evolutionary Techniques for the Maintenance Scheduling Problem, IEEE Transactions on Power Systems, Vol. 15, No. 1, pp. 122-128, 2000.

34. Carrasco M.P., Pato M.V., A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem, In: [25], pp. 3-17, 2001.
35. Carter M.W., A Survey of Practical Applications of Examination Timetabling Algorithms, OR Practice, Vol. 34, No. 2, pp. 193-202, 1986.
36. Carter M.W., Laporte G., Recent Developments in Practical Examination Timetabling, In: [32], pp. 3-21, 1996.
37. Carter M.W., Laporte G., Recent Developments in Practical Course Timetabling, In: [17], pp. 3-19, 1998.
38. Carter M.W., Laporte G., Chinneck J.W., A General Examination Timetabling System, Interfaces, Vol. 24, No. 3, pp. 109-120, 1994.
39. Carter M.W., Laporte G., Lee S.Y., Examination Timetabling: Algorithm Strategies and Applications, Journal of the Operational Research Society, Vol. 47, pp. 373-383, 1996.
40. Chen W.H., Lin C.S., A Hybrid Heuristic to Solve a Task Allocation Problem, Computers and Operations Research, VOl. 27, pp. 287-303, 2000.
41. Coello Coello C.A., Van Veldhuizen D.A., Lamont G.B., Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, 2002.
42. Corne D., Dorigo M., Glover F. (eds.), New Ideas in Optimisation, McGraw Hill, 1999.
43. Corne D., Ogden J., Evolutionary Optimisation of Methodist Preaching Timetables, In: [17], pp. 142-155, 1998.
44. Corne D., Ross P., Peckish Initialisation Strategies for Evolutionary Timetabling, In: [32], pp. 227-240, 1996.
45. Corne D., Ross P., Fang H.L., Fast Practical Evolutionary Timetabling, Selected Papers from the AISB Workshop on Evolutionary Computation, Lecture Notes in Computer Science, Vol. 865, Springer, pp. 220-263, 1994.
46. Costa D., A Tabu Search Algorithm for Computing an Operational Timetable, European Journal of Operational Research, Vol. 76, pp. 98-110, 1994.
47. Cowling P., Kendall G., Soubeiga E., A Hyperheuristic Approach to Scheduling a Sales Summit, In: [25], pp. 176-190, 2001.
48. Czyzak P., Jaszkiewicz A. Pareto Simulated Annealing - a Metaheuristic for Multiple-Objective Combinatorial Optimization, Journal of Multi-Criteria Decision Analysis, Vol. 7, No. 1, pp. 34-47, 1998.
49. de Werra D., An Introduction to Timetabling, European Journal of Operational Research, Vol. 19, pp. 151-162, 1985.
50. Deb K., Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.
51. Deb K., Agrawal S. Pratap A. and Meyarivan T., A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 182-197, 2002.
52. Dorigo M., Maniezzo V., Colorni A., The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man, and Cybernetics - Part B, Vol. 26, No. 1, pp. 1-13, 1996.
53. Dowsland K.A., Simulated Annealing Solutions for Multi-Objective Scheduling and Timetabling, In: Rayward-Smith V.J., Osman I.H., Reeves C.R., Smith G.D. (eds.), Modern Heuristic Search Methods, Wiley, 1996.
54. Dowsland K.A., Off-the-Peg or Made-to-Measure? Timetabling and Scheduling with SA and TS, In: [17], pp. 37-52, 1998.
55. Ehrgott M., Gandibleux X., A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization, OR Spectrum, Vol. 22, No. 4, Springer, pp. 425-460, 2000.

56. Ehrgott M., Klamroth K., Connectedness of Efficient Solutions in Multiple Criteria Combinatorial Optimization, European Journal of Operational Research, Vol. 97, pp. 159-166, 1997.
57. El Moudani W., Nunes Cosenza C.A., de Coligny M., Mora Camino F., A Bi-Criterion Approach for the Airlines Crew Rostering Problem, In: [125], pp. 486-500, 2001.
58. Fonseca C.M., Fleming P.J., An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation, Vol. 3, No. 1, pp. 1-16, 1995.
59. Fonseca C.M., Fleming P.J., Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms - Part 1: A Unified Formulation, IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, No. 1, pp. 26-37, 1998.
60. Fonseca C.M., Fleming P., Zitzler E., Deb K., Thiele L. (eds.), Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003), Lecture Notes in Computer Science, Vol. 2632, Springer, 2003.
61. Gandibleux X., Freville A., Tabu Search Based Procedure for Solving the 0-1 MultiObjective Knapsack Problem: The Two Objectives Case, Journal of Heuristics, Vol. 6, No. 3, pp. 361-383, 2000.
62. Gandibleux X., Morita H., Katoh N., The Supported Solutions Used as a Genetic Information in a Population Heuristics, In: [125], pp. 429-442, 2001.
63. Garey M.R., Johnson D.S., Computers and Intractability - A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.
64. Glover F.W., Kochenberger G.A. (eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, 2003.
65. Glover F., Laguna M., Tabu Search, Kluwer Academic Publishers, 1997.
66. Hansen M.P., Tabu Search for Multiobjective Optimization: MOTS, Technical Report Presented at 13th International Conference on MCDM, Technical University of Denmark, 1997.
67. Hansen P., Mlandenovic N., Variable Neighbourhood Search: Principles and Applications, European Journal of Operational Research, Vol. 130, No. 3, pp. 449-467, 2001.
68. Ishibuchi H., Murata T., A Multi-Objective Genetic Local Search Algorithm and its Application to Flowshop Scheduling, IEEE Transactions on Systems, Man and Cybernetics - Part C: Applications and Reviews, Vol. 28, No. 3, pp. 392-403, 1998.
69. Ishibuchi H., Murata T., Tomioka S., Effectiveness of Genetic Local Search Algorithms, Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 505-512, 1997.
70. Ishibuchi H., Shibata Y., An Empirical Study on the Effect of Mating Restriction on the Search Ability of EMO Algorithms, In: [60], pp. 433-447, 2003.
71. Ishibuchi H., Yoshida T., Murata T., Selection of Initial Solutions for Local Search in Multiobjective Genetic Local Search, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, pp. 950-955, 2002.
72. Ishibuchi H., Yoshida T., Murata T., Balance Between Genetic Search and Local Search in Hybrid Evolutionary Multi-Criterion Optimization Algorithms, Proceedings of the 2002 Genetic and Evolutionary Conference (GECCO 2002), Morgan Kaufmann, pp. 1301-1308, 2002.
73. Ishibuchi H., Yoshida T., Murata T., Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling, IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2, pp. 204-223, 2003.

74. Jaszkiewicz A., A Metaheuristic Approach to Multiple Objective Nurse Scheduling, Foundations of Computing and Decision Sciences, Vol. 22, No. 3, pp. 169-183, 1997.
75. Jaszkiewicz A., Comparison of Local Search-based Metaheuristics on the Multiple Objective Knapsack Problem, Foundations of Computing and Decision Sciences, Vol. 26, No. 1, pp. 99-120, 2001.
76. Jaszkiewicz A., Genetic Local Search for Multi-objective Combinatorial Optimization, European Journal of Operational Research, Vol. 137, No. 1, pp. 50-71, 2002.
77. Jones D.F., Mirrazavi S.K., Tamiz M., Multiobjective Meta-heuristics: An Overview of the Current State-of-the-Art, European Journal of Operational Research, Vol. 137, No. 1, pp. 1-9, 2001.
78. Knowles J., Corne D., Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, Evolutionary Computation, Vol. 8, No. 2, pp. 149-172, 2000.
79. Knowles J., Corne D., On Metrics for Comparing Nondominated Sets, Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), IEEE Press, pp. 711-716, 2002.
80. Knowles J.D., Corne D.W., Towards Landscape Analyses to Inform the Design of a Hybrid Loacl Search for the Multiobjective Quadratic Assignment Problem, In: Abraham A., Ruiz-del-Solar J., Koppen M. (eds.), Soft Computing Systems: Design, Management and Applications, IOS Press, pp. 271-279, 2002.
81. Kokolo I., Hajime K., Shigenobu K., Failure of Pareto-based MOEAs, Does Non-dominated Really Mean Near to Optimal?, Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001), IEEE Press, pp. 957-962, 2001.
82. Laumanns M., Thiele L., Deb K., Zitzler E., Combining Convergence and Diversity in Evolutionary Multiobjective Optimization, Evolutionary Computation, Vol. 10, No. 3, pp. 263-282, 2002.
83. Lee C.Y., Lei L., Pinedo M., Current Trends in Deterministic Scheduling, Annals of Operations Research, Vol. 70, pp. 1-41, 1997.
84. Man K.F., Tang K.S. and Kwong S., Genetic Algorithms: Concepts and Design, Springer, 1999.
85. Marett R., Wright M., A Comparison of Neighbourhood Search Techniques for Multi-Objective Combinatorial Problems, Computers and Operations Research, Vol. 23, No. 5, pp. 465-483, 1996.
86. Michalewicz Z., Fogel D., How to Solve It: Modern Heuristics, Springer, 2000.
87. Miettinen K., Some Methods for Nonlinear Multi-Objective Optimization, In: [125], pp. 1-20, 2001.
88. Murata T., Ishibuchi H., Gen M., Cellular Genetic Local Search for Multi-Objective Optimization, Proceedings of the 2000 Genetic and Evolutionary Computation Conference (GECCO 2000), Morgan Kaufmann, pp. 307-314, 2000.
89. Murata T., Ishibuchi H., Gen M., Specification of Genetic Search Directions in Cellular Multi-objective Genetic Algorithms, In: [125], pp. 82-95, 2001.
90. Murata T., Ishibuchi H., Tanaka H., Genetic Algorithms for Flowshop Scheduling Problems, Computers and Industrial Engineering, Vol. 30, No. 4, pp. 1061-1071, 1996.
91. Murata T., Ishibuchi H., Tanaka H., Multi-Objective Genetic Algorithm and its Applications to Flowshop Scheduling, Computers and Industrial Engineering, Vol. 30, No. 4, pp. 957-968, 1996.
92. Nagar A., Haddock J., Heragu S., Multiple and Bicriteria Scheduling: A Literature Survey, European Journal of Operational Research, Vol 81, pp. 88-104, 1995.

93. Papadimitriou C.H., Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, 1982.
94. Paquete L.F., Fonseca C.M., A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms, Proceedings of the 2001 Metaheuristics International Conference (MIC 2001), pp. 149-153, 2001.
95. Petrovic S., Bykov Y., A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories, to appear In: [18], 2003.
96. Pinedo M., Scheduling, Theory, Algorithms, and Systems, 2nd Edition, Prentice-Hall, 2002.
97. Rankin R.C., Automated Timetabling in Practice, In: [32], pp. 266-279, 1996.
98. Reeves C.R. (ed.), Modern Heuristic Techniques for Combinatorial Problems, McGraw-Hill, 1995.
99. Reeves C., Integrating Local Search into Genetic Algorithms, In: Rayward-Smith V.J., Osman I.H., Reeves C.R., Smith G.D. (eds.), Modern Heuristic Search Methods, Wiley, 1996.
100. Rosenthal R.E., Principles of Multiobjective Optimization, Decision Sciences, Vol. 16, pp. 133-152, 1985.
101. Salman F.S., Kalagnaman J.R., Murthy S., Davenport A., Cooperative Strategies for Solving Bicriteria Sparse Multiple Knapsack Problem, Journal of Heuristics, Vol. 8, pp. 215-239, 2002.
102. Schaerf A., A Survey on Automated Timetabling, Artificial Intelligence Review, Vol. 13, pp. 87-127, 1999.
103. Schaerf A., Local Search Techniques for Large High School Timetabling Problems, IEEE Transactions on Systems, Man and Cybernetics- Part A: Systems and Humans, Vol. 29, No. 4, pp. 368-377, 1999.
104. Schaffer J.D., Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93-100, 1985.
105. Socha K., Knowles J., Samples M., A Max-Min Ant System for the University Course Timetabling Problem, Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002), Lecture Notes in Computer Science, Vol. 2463, Springer, pp. 1-13, 2002.
106. Srivivas N., Deb K., Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation, Vol. 2, No. 3, pp. 221-248, 1995.
107. Steuer Ralph E., Multiple Criteria Optimization: Theory, Computation and Application, Wiley, 1986.
108. Suppapitnarm A., Seffen A., Parks G.T., Clarkson P.J., A Simulated Annealing Algorithm for Multiobjective Optimisation Engineering Optimization, Vol. 33, No. 1, pp. 59-85, 2000.
109. Talbi E.G., Rahoudal M., Mabed M.H., Dhaenens C., A Hybrid Evolutionary Approach for Multicriteria Optimization Problems: Application to the Flow Shop, In: [125], pp. 416-428, 2001.
110. Tan K.C., Lee T.H., Khor E.F., Evolutionary Algorithms for Multi-Objective Optimization: Performance Assessments and Comparisons, Artificial Intelligence Review, Vol. 17, pp. 253-290, 2002.
111. Thompson J.M., Dowsland K.A., General Cooling schedules for a Simulated Annealing Based Timetabling System, In: [32], pp. 345-363, Springer-Verlag, 1996.
112. Thompson J.M., Dowsland K.A., Variants of Simulated Annealing for the Examination Timetabling Problem, Annals of Operations Research, Vol. 63, pp. 105-128, 1996.

113. T'kindt V., Billaut J.C., Multicriteria Scheduling: Theory, Models and Algorithms, Springer, 2002.
114. Ulungu E.L., Teghem J., Multi-objective Combinatorial Optimization Problems: a Survey, Journal of Multi-Criteria Decision Analysis, Vol. 3, pp. 83-104, 1994.
115. Ulungu E.L., Teghem J. Fortemps P.H., Tuyttens D., MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems, Journal of Multi-criteria Decision Analysis, Vol. 8, pp. 221-236, 1999.
116. Vaessens R.J.M., Aarts E.H.L. and Lenstra J.K., Job Shop Scheduling by Local Search, INFORMS Journal on Computing, Vol 8, No. 3, pp. 302-317, 1996.
117. Varela R., Vela C.R., Puente J., Gomez A., Vidal A. M., Solving Job-shop Scheduling Problems by Means of Genetic Algorithms, In: Chambers Lance (ed.) The Practical Handbook of Genetic Algorithms Applications, Chapman:Hall/CRC, 2001.
118. Voss S., Martello S., Osman I.H. and Rucairol C. (eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, 1999.
119. Welsh D.J.A., Powell M.B., An Upper Bound for the Chromatic Number of a Graph and its Applications to Timetabling Problems, The Computer Journal, Vol. 10, pp. 85-86, 1967.
120. Wren A., Scheduling, Timetabling and Rostering, a Special Relationship?, In: [32], pp. 46-75, 1996.
121. Wright Mike, Subcost-Guided Search - Experiments with Timetabling Problems, Journal of Heuristics, Vol. 7, pp. 251-260, 2001.
122. Wright Mike B., Marett Richard C., A Preliminary Investigation into the Performance of Heuristic Search Methods Applied to Compound Combinatorial Problems, In: Osman I.H., Kelly J.P. (eds.), Meta-Heuristics: Theory and Applications, Kluwer Academic Publishers, pp. 299-317, 1996.
123. Yannakakis M., Computational Complexity, In: Aarts E. and Lenstra J.K. (eds.), Local Search in Combinatorial Optimization, Wiley, 1997.
124. Zeleny M., Compromise Programming, In: Cochrane J.L., Zeleny M. (eds.): Multiple Criteria Decision Making, University of South Carolina Press, Columbia, pp. 262-301, 1973.
125. Zitzler E., Deb K., Thiele L., Coello Coello C.A., Corne D. (eds.), Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001), Lecture Notes in Computer Science, Vol. 1993, Springer, 2001.
126. Zitzler E., Thiele L., Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp. 257-271, 1999.
127. Zitzler E., Thiele L., Laumanns M., Fonseca C.M., da Fonseca V.G., Performance Assessment of Multiobjective Optimizers: An Analysis and Review, IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2, pp. 117-132, 2003.