Memetic Algorithms in Planning, Scheduling, and Timetabling

Carlos Cotta and Antonio J. Fernández

Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain.

{ccottap,afdez}@lcc.uma.es

Abstract. Memetic algorithms (MAs) constitute a metaheuristic optimization paradigm based on the systematic exploitation of knowledge about the problem being solved, and the synergistic combination of ideas taken from other population-based and trajectory-based metaheuristics. They have been successfully deployed on a plethora of hard combinatorial optimization problems, amongst which scheduling, planning and timetabling are distinguished examples due to their practical interest. This work examines the application of MAs to problems in these domains. We describe the basic architecture of a MA, and present some guidelines to the design of a successful MA for these applications. An overview of the existing literature on the topic is also provided. We conclude with some reflections on the lessons learned, and the future directions that research could take in this area.

1 Introduction

Back in the late 1960s and early 1970s, it began to be evident that there existed many practical problems for which neither the exact resolution nor approximate approaches with realistic performance guarantees were acceptable in practice. Motivated by this fact, several researchers laid the foundations of what we now know as *evolutionary algorithms* [1–4] (EAs). Despite some voices claiming that such approaches constituted "an admission of defeat", these techniques have steadily grown in usage and understanding to become what they nowadays represent: the cutting-edge approach to real-world optimization. Certainly, this has also been the case for other related techniques, such as *simulated annealing* [5] (SA), *tabu search* [6] (TS), etc. The term *metaheuristics* has been coined to denote them.

The development of metaheuristics in general, and EAs in particular, reached a critical point in the mid 1990s, when the need of exploiting problem knowledge was clearly exposed. The formulation of the *No Free Lunch Theorem* (NFL) by Wolpert and Macready [7] made it definitely clear that a search algorithm performs in strict accordance with the amount and quality of the problem knowledge they incorporate. Quite interestingly, this line of thinking had already been advocated by several researchers in the late 1980s and early 1990s, e.g., Hart and

Belew [8], Davis [9], and Moscato [10]. It was precisely in the work of Moscato where the paradigm of *memetic algorithms* [11–13] (MAs) started.

MAs are a family of metaheuristics that try to blend several concepts from tightly separated –in their origins– families such as EAs and SA. The adjective 'memetic' comes from the term 'meme', coined by R. Dawkins [14] to denote an entity that is analogous to the *gene* in the context of cultural evolution. The purpose of the analogy (sometimes over-stressed in the literature) is to emphasize the departure from biologically-inspired mechanisms of evolution, to more general processes where actual information is manipulated, learned, and transmitted. Due to the way in which this can be implemented, it is often the case that MAs are used under a different name (e.g., 'hybrid EAs', 'Lamarckian EAs', etc.), and sometimes with a very restrictive meaning. At any rate, we can say that a MA is a search strategy in which a population of optimizing agents synergistically cooperate and compete [10]. These agents are explicitly concerned with using knowledge from the problem being solved, as suggested by both theory and practice [15]. A more detailed description of the algorithmic pattern of MAs is given in Sect. 2.

As mentioned above, the *raison d'être* of metaheuristics is the fact that many problems are very difficult to solve using classical approaches. This is precisely the case for many problems in the area of industrial planning, scheduling, timetabling, etc. All these problems have something in common: a set of entities have to be arranged in a time-like fashion, subject to some particular constraints (based on, e.g., precedence, resource consumption, etc.), and usually with some cost figure associated (to be optimized). Not surprisingly, MAs have been extensively used to solve this kind of problems. In this work, we shall analyze the deployment of MAs on this domain. To this end, we shall provide some general design guidelines in Sect. 3, and an overview of the relevant applications of MAs in Sect. 4, trying to highlight the successful strategies. This chapter will end with a summary of lessons learned, and some current and emerging research trends in MAs for scheduling, planning, and timetabling.

2 Memetic Algorithms

MAs are population-based metaheuristics. This means that the algorithm maintain a *population* of candidate solutions for the problem at hand, i.e., a pool comprising several tentative solutions. In the EA terminology, each of these candidate solutions is called *individual*. However, the nature of MAs suggests *agent* as a more appropriate term here. In essence, this is due to the fact that 'individual' denotes a *passive* entity, subject to some evolutionary rules, whereas 'agent' implies the existence of an *active* behavior, purposefully directed to solving the optimization problem at hand. This active behavior is reflected in several typical components of the algorithm such as (but not exclusively as) local search add-ons. We shall return to this point later in this section.

A general sketch of a MA is shown in Fig. 1. As in EAs, the population of agents is subject to processes of competition and mutual cooperation. Competi-

Memetic Algorithm

```
     INPUT:  An instance I of problem P.
     OUTPUT: A solution sol.

     // Generate Initial Population
 1:  for j ← 1:popsize do
 2:    let ind ← GenerateHeuristicSolution(I)
 3:    let pop[j] ← LocalImprover (ind, I)
 4:  endfor
 5:  repeat // Basic Generational Loop
       // Selection
 6:    let breeders ← SelectFromPopulation(pop)
       // Pipelined reproduction
 7:    let auxpop[0] ← pop
 8:    for j ← 1:#op do
 9:      let auxpop[j] ← ApplyOperator (op[j], auxpop[j − 1], I)
10:    endfor
11:    let newpop ← auxpop[#op]
       // Replacement
12:    let pop ← UpdatePopulation (pop, newpop)
       // Check Population Convergence
13:    if Converged(pop) then
14:      let pop ← RestartPopulation(pop, I)
15:    endif
16:  until MA-TerminationCriterion(pop, I)
17:  return Best (pop, I)
```

Fig. 1. The general template of a memetic algorithm

tion is achieved via the standard procedures of selection (line 6) and replacement (line 12): using the information provided by an *ad hoc* guiding function (*fitness* function in the EA terminology), the goodness of agents in *pop* is evaluated; subsequently, a sample of them is selected for reproduction according to this goodness measure. This information is later used to determine which agents will be removed from the population in order to make room for the newly created ones. In both cases –selection and replacement– any of the well-known strategies used in EAs can be utilized, e.g., tournament, ranking, or fitness-proportionate selection, plus or comma replacement, etc. In addition to these, there are other strategies that could be used here, and that have been proved successful in several scheduling domains (see Sect. 4).

As to cooperation, it is accomplished through reproduction. At this stage, new agents are created by using the existing ones. This is done by utilizing a number of reproductive *operators*. Many different such operators can be used in a MA, as illustrated in the general pseudocode shown in Fig. 1, lines 7–11. As it can be seen, an array *op* of operators can be in general assumed. These operators

(whose number is denoted by $\#op$) are sequentially applied to the population in a pipeline fashion, thus resulting in several intermediate populations $auxpop[i]$, $0 \leqslant i \leqslant \#op$, where $auxpop[0]$ is initialized to pop, and $auxpop[\#op]$ is the final offspring. In practice, the most typical situation involves utilizing just three operators: recombination, mutation, and local improvement. Notice that in line 9 of the pseudocode, these operators receive not only the solutions they must act on, but also the problem instance I. This illustrates the fact that in MAs these operators are problem-aware, and base their functioning on their knowledge about the problem being solved. This is an important difference with respect to classical EAs.

Recombination is the process that best encapsulates mutual cooperation among several agents (typically two of them, but a higher number is possible [16]). This is done by constructing new solutions using the *relevant* information contained in a number of selected parents. By "relevant", it is implied that the information pieces manipulated bear some important information in order to determine the goodness (or badness) of the solutions. This is an interesting notion that departs from the classical manipulation of symbols for a certain syntax of candidate solutions, typical of plain EAs. We shall return to this in next section.

The other classical operator –mutation– plays the role of *keeping the pot boiling*, continuously injecting new material in the population, but at a low rate (otherwise the search would degrade to a random walk in the solution space). This interpretation of mutation reflects the classical view, dominant in the genetic algorithm arena [17]. Certainly, evolutionary programming practitioners [1] would disagree with this characterization, claiming the crucial role for mutation. According to this latter view, recombination is generally regarded as a *macro-mutation* process. While this is something that could be accepted to some extent in many EA applications in which recombination is a mere random-shuffler of information, the situation is quite different for MAs. Indeed, recombination is usually performed in a *smart* way as anticipated before, and hence it provides a central contribution to the search.

Lastly, one of the most distinctive components of MAs is the use of local improvers. To understand their philosophy, let us consider the following abstract formulation: first of all, assume a graph whose vertices are solutions, and whose edges connect pairs of vertices such that the corresponding solutions differ in some (typically small) amount of relevant information. Now, a local improver is a process that starts at a certain vertex, and moves to an adjacent vertex, provided that the neighboring solution is better than the current solution. Thus, the local improver tries to find an "uphill" (in terms of improving the value provided by the guiding function) path in the graph whose definition was sketched before (formally termed *fitness landscape* [18]). Notice that this description is very simplistic, and that many variations may exist in the way in which the neighbor is selected, the precise criterion under which it is accepted or rejected, and the termination condition for the local improvement.

As anticipated before, the utilization of local improvers (notice that several different ones could be used in different points of the algorithm) is one of the

most characteristic features of MAs. It is mainly because of the use of this mechanism for improving solutions on a local (and even autonomous) basis that the term 'agent' is deserved. Thus, the MA can be viewed as a collection of agents performing an autonomous exploration of the search space, cooperating some times via recombination, and competing for computational resources due to the use of selection/replacement mechanisms.

There is another interesting element in the pseudocode shown in Fig. 1, namely the RestartPopulation process (lines 13–15). This process is very important in order to make an appropriate use of the computational resources: should the population reach a state in which all agents were very similar to each other, the generation of new improved solutions would be very unlikely. This phenomenon is known as *convergence*, and can be identified using measures such as Shannon's entropy [19]. If this measure falls below a predefined threshold, the population is considered at a degenerate state. This threshold depends upon the representation of the problem being used, and must therefore be determined in an *ad hoc* fashion.

It must be noted that while most MA applications can be dissected using the general template presented here, it is obviously possible to conceive algorithms somehow departing from it, that could nevertheless be catalogued as MAs. This fact notwithstanding, the general principles depicted in this section should still be applicable to these MAs.

3 Design Principles for Effective MAs

In order to solve a particular optimization problem in the area of planning and scheduling, the general template of MAs depicted in Sect. 2 must be instantiated with precise problem-aware components. No general approach for the design of effective MAs exists in a well-defined sense, and hence this design phase must be addressed from a heuristic point of view as well. Let us consider in the following the main decisions that have to be taken.

3.1 Representation

The first element that one has to decide is the *representation* of solutions. This notion must be understood not as the way solutions are encoded (something that is subject to considerations of memory consumption, manipulation complexity, etc.), but as the abstract formulation of solutions, as regarded by the reproductive operators [20]. In this sense, recall the notion of "relevant" information introduced in Sect. 2: given a certain representation of solutions, these are expressed via some *information units*; if the operators used for manipulating solutions are problem-aware, these information units they identify must be important to determine whether a solution is good or not. The evolutionary dynamics of the system would then drive the population to retain those positive information units, making negative units disappear. This is better illustrated with an example: consider a problem whose solution space is composed of all

permutations of n elements; there are several types of information units in such solutions [21], e.g.,

- *positional*, i.e., element e appears in position j.
- *precedence*, i.e., element e appears before/after element e'.
- *adjacency*, i.e., element e appears next to element e'.

The relevance of each type of information unit will obviously depend on the problem being solved. For example, adjacency information is important for the Travelling Salesman Problem (TSP), but positional information is less so. On the other hand, it seems that positional information is relevant when minimizing makespan in a permutation flowshop problem [22], and adjacency information is more irrelevant in this case. Therefore, an edge-manipulation operator such as edge-recombination [23] (ER) will perform better than position-based operators such as partially-mapped crossover [24] (PMX) or uniform cycle crossover [22] (UCX) on the TSP, but the latter will be better on permutation flowshop problems.

There have been several attempts for quantifying how good a certain set of information units is for representing solutions for a specific problems. Among these we can cite *epistasis* (non-additive influence on the guiding function of combining several information units) [25, 26], *fitness variance of formae* (variance of the values returned by the guiding function, measured across a representative subset of solutions carrying this information unit) [27], and *fitness correlation* (correlation in the values of the guiding function for parents and offspring) [28, 29]. Notice that in addition to using a quality metric of representations to predict the performance of a certain pre-existing operator (i.e., *inverse analysis*), new *ad hoc* operators can be defined to manipulate the best representation (*direct analysis*) [13]. This is for example done in [22] for permutation flowshop scheduling, once the positional representation is revealed as the most promising.

It is also important to note that whatever the metric used to quantify the goodness of a particular representation is, there are other considerations that can play a central role, namely, the presence of constraints. Typically, these are handled in three ways: (1) by using penalty functions that guide the search to the feasible region, (2) by using repairing mechanisms that take infeasible solutions back to the feasible region, and (3) by defining reproductive operators that always remain in the feasible region. In the first two cases, the complexity of the representation and the operators can be kept at a lower level[1]. In the latter case, responsibility has to be taken either by representation or by operators to ensure feasibility, and this comes at the cost of an increased complexity. Focusing on representations, the use of *decoders* is a common option to ensure feasibility. The basic idea is to use a complex genotype-to-phenotype mapping that not only produces feasible solutions, but can also provide additional problem knowledge and hence solutions of better quality. For example, a greedy permutational decoder is used in [30] for a nurse scheduling problem. A related approach is used in [31–33] in the context of job shop scheduling.

[1] At least from the functional point of view; from the practical point of view, more sophisticated strategies can obviously result in improved performance.

3.2 Reproductive Operators

The generation of new solutions during the reproductive stage is done by manipulating the relevant pieces of information identified. To do so, the user could resort to any of the generic templates defined for that purpose, e.g., *random respectful recombination*, *random assorting recombination*, and *random transmitting recombination* among others [34]. These are generic templates, in the sense that they blindly process abstract information units. However, in order to ensure top performance, reproductive operators must not only manipulate the relevant information, but must do so in a sensible way, that is, using problem knowledge.

There are many ways to achieve this inclusion of problem knowledge. From a rather general stand-point, there are two major aspects into which problem knowledge can be injected: the selection of the parental features that will be transmitted to the descendant, and the selection of non-parental features that will be added to it. Regarding the former issue, there exists evidence that transmission of common features is beneficial for some problems (e.g., [23, 35]). After this initial transmission, the offspring can be completed in several ways. For example, Radcliffe and Surry [27] have proposed the use of local improvers or implicit enumeration schemes. These implicit enumeration schemes can also be used to find the best combination of the information units present in the parents [36] (in this case, the resulting solution would not necessarily respect common features, unless forced to do so). This operation is monotonic in the sense that any child generated is at least as good as the best parent. Ibaraki [37] uses dynamic programming for this purpose, in a single-machine scheduling problem.

To some extent, the above discussion is also applicable to mutation operators, although these exhibit a clearly different role: they must introduce new information as indicated in Sect. 2. The typical procedure is removing some information units from a single solution, and either complete it at random, or use any of the completion procedures described before. Several considerations must be made here though. The first one is the lower relevance that mutation may have in some memetic contexts. Indeed, mutation is sometimes not used in MAs, and instead it is embedded into the local search component, e.g., see [38, 39] in job shop scheduling, and [40] in single machine scheduling, among others. The reason is the widespread usage in MAs of re-starting procedures for refreshing the population when a stagnation point is reached (see Sect. 3.4). In some applications, it may be better to achieve faster convergence and then re-start, than diversifying continuously the search using mutation in pursuit of steady (yet slower) progress.

In other applications, re-start strategies are not used and mutation is thus more important. In these cases, it is not unusual to use several mutation operators, either by considering different basic neighborhoods (e.g., [41] in open shop scheduling, and [42] in single machine scheduling), or by defining *light* and *heavy* mutations that introduce different amounts of new information (e.g., [43, 44] in timetabling, and [45] in flowshop scheduling). Note that according to the operator-based view of representations presented in Sect. 3.1, the use of multi-

ple operators may imply the consideration of different solution representations at different stages of the reproductive phase. This feature of MAs is also exhibited by other metaheuristics such as *variable neighborhood search* [46] (VNS).

Problem-knowledge can also be attained by using of constructive heuristics. These can be used for instance to create the initial population, as depicted in Fig. 1, line 2. For example, Yeh [47] uses a greedy heuristic for this purpose in a flowshop scheduling problem. Other examples of heuristic initialization can be found in [48, 31, 49] for job shop scheduling, and in [43, 50, 51] for timetabling.

3.3 Local Search

The presence of a local search (LS) component is usually regarded as the distinctive feature of MAs with respect to plain evolutionary algorithms. To some extent, it is true that most MAs incorporate LS; this said, the naïve equation MA = EA + LS is an oversimplification that should be avoided [11–13]. Indeed, there are metaheuristic approaches whose philosophy is strongly connected to that of MAs, but that cannot be called "evolutionary" unless a very broad meaning of the term (i.e., practically encompassing every population-based metaheuristic) were assumed. The scatter search metaheuristic [52] is a good example of this situation. On the other hand, there are MA that rely heavily on the use of knowledge-augmented recombination operators, rather than on LS operators, e.g., [36, 53]. Be that as it may, this is not an obstacle to state that LS is commonly used in MAs (i.e., EA + LS $\subset$ MA), and usually has a determining influence on the final performance.

As sketched in Sect. 2, LS can be typically modelled as a trajectory in the search space, that is, an ordered sequence of solutions such that neighboring solutions in this sequence differ in some *small* amount of information. Of course, some implementations can depart from this idealized description. As an example, we can consider MA applications in which the LS component is implemented via TS (tabu search, Sect. 1), such as [54, 55] in flowshop scheduling, [41] in openshop scheduling [56–60] in timetabling, or [61, 62] in maintenance scheduling, among others. Some implementations of TS are endowed with intensification strategies that resume the search from previous elite solutions (hence, rather than a linear sequence, the path traversed by TS can be regarded as a branching trajectory). Additionally, a feature of the utilization of TS –which is shared with MAs that use other LS components as SA (simulated annealing, Sect. 1), e.g., [39, 63] in flowshop scheduling, and also [44, 61] in maintenance scheduling– is the fact that the quality of solutions in the trajectory is not monotonically increasing, but can eventually decrease in order to escape from a local optimum. Obviously, at the end of the process the best solution found (rather than the last one in the trajectory) is kept.

Several issues must be considered when implementing the LS component. One of them is the termination criterion for the search. In classical hill climbing (HC) techniques, it makes sense to stop the search whenever a local optimum is found. However, this is not directly applicable to LS techniques with global optimization capabilities such as TS or SA. In these cases (and indeed in the

case of plain HC) it is customary to define a maximum computational effort to be devoted to each LS invocation. On one hand, this means that the final solution need not be a local optimum, as some incorrect characterizations of MAs as EAs in the space of local optima would suggest. On the other hand, it is necessary to define an appropriate balance between the effort of LS and that of the underlying population-based search. This issue has been also acknowledged in other domains, e.g., [64], and the use of *partial Lamarckism* has been suggested, i.e., not using LS on every new solution computed, but only on some of them, selected at random with some probability or on the basis of quality, or only in every $k-$th generation; see [65] for an analysis of these strategies in a multi-objective permutation flowshop problem.

Similarly to the remaining reproductive operators, the selection of a particular LS scheme can be done in light of quality metrics computed on the resulting fitness landscape. *Fitness distance correlation* [66, 67] (FDC) has been proposed as a hardness measure. In essence, FDC is the correlation between the quality of local optima and their closeness to the global optimum. If this FDC coefficient is high (that is, quality tends to be larger for increasing closeness to the optimum), the natural dynamics of the MA (i.e., get closer to local optima by virtue of the LS component) would also lead it close to the global optimum. This *a priori* analysis can be helpful to estimate the potential effectiveness of a particular LS scheme. Its usefulness as a tool for dynamically acquiring information on the fitness landscape is much more questionable, since some general theoretical results indicate that information is conserved during optimization (i.e., information that is apparently gained during the run is in fact a result of *a priori* knowledge) [68].

Another important issue that must be considered during landscape analysis is its global topology, and more precisely, whether it is regular or not, and in the latter case whether the irregularity is in some sense connected to quality or not. This issue has been analyzed by Bierwirth *et al.* [69] for a job-shop scheduling problem. They found that high quality solutions are also highly connected, and hence there is a beneficial drift force that makes that random walks tend to land closer to high quality solutions than to low quality solutions. Of course, the contrary might be true for a certain problem, and this could hinder good performance of the LS algorithm. At any rate, the same consideration regarding the use of multiple mutation operators done in Sect. 3.2 is applicable here, and multiple LS schemes can be used within a MA, see e.g., [65, 51].

3.4 Managing Diversity

As mentioned in Sect. 3.2, there are different views on how to manage the diversity of information in the population. In essence, we can make a distinction between methods for *preserving* diversity, and methods for *restoring* diversity. Clearly, the use of mutation operators falls within the first class. This does not exhaust the possibilities though. For example, the strategy of injecting in the population "random immigrants" [70] –i.e., completely new solutions– could be used. This has been done in [71] for task allocation on multiprocessors. A much more widespread option is the utilization of structured populations [72]: rather

than maintaining a panmictic pool of agents in which any two of them can mate, or in which a new solution can potentially replace any existing solution, the population is endowed with a precise topology; both mating and replacement are confined to *neighboring* agents. This causes a slowdown in the propagation of information across the population, and hence hinders the apparition of *super-agents* that might quickly take the population over and destroy diversity.

Different population topologies are reported in the literature, i.e., unidirectional/bidirectional rings, grids, hypercubes, etc. In the context of MAs and scheduling, the ternary tree topology has been particularly successful, see e.g., [73–80, 45]. MAs endowed with this topology usually combine it with the use of a quality-based mechanism for placing solutions. More precisely, each internal node of the tree is forced to have a better solution than that of its immediate descendants in the tree. If at a certain point of the run an agent bears a better solution than that of its parent's, they exchange their solutions. This way, there is a continuous upwards flow of better solutions that also guarantees that when recombination is attempted, the intervening solutions are of similar quality.

The alternative (or better, the complement) to these diversity preservation mechanisms is the use of diversity restoration procedures. These are triggered whenever it is detected that the population has stagnated, or is close to a such a dead-end state. This situation can be detected by monitoring the population composition as mentioned in Sect. 2, or by analyzing the population dynamics [81]. In either case, a re-starting procedure is activated. These procedures can be implemented in different ways. A possibility is the utilization of triggered hypermutation [82] (cf. heavy mutation, see Sect. 3.2), as it is done in [45] for a flowshop scheduling problem with sequence dependent family setups. Alternatively, the population can be refreshed by using the previously mentioned random-immigrant strategy, i.e., keeping a fraction of the existing agents (typically some percentage of the best ones), and renewing the rest of the population with random (or heuristically constructed solutions).

We would like to close this section by emphasizing again the heuristic nature of the design principles described in this and previous sections. There is still much room for research in methodological aspects of MAs (e.g., see [83]), and the open-minded philosophy of MAs make them suitable for incorporating mechanisms from other optimization techniques.

4 Applications in Planning, Scheduling, and Timetabling

Scheduling problems can take many forms, and adopt many variants, so we have opted for considering four major subclasses, namely machine scheduling, timetabling, manpower scheduling, and industrial planning. Note that this classification aims to provide a general view of MA applications in this field, rather than a conclusive taxonomy.

4.1 Machine Scheduling

In a broad sense, machine scheduling amounts to organizing in a time-like fashion a set of jobs that have to be processed in a collection of machines. This general definition admits numerous variants in terms of (1) the number of machines onto which the schedule must be arranged (e.g., one or many), (2) the precise constraints involved in the arrangement (e.g., precedence constraints, setup times, etc.), and (3) the quality measure being optimized (e.g., makespan, total tardiness, number of tardy jobs, etc.). The reader may be convinced of the competence of MAs by noting that almost every conceivable instantiation of this generic problem family has been tackled with MAs in the literature.

One of the most well-studied problems in this area is *single machine scheduling* (SMS), i.e., the scheduling of n jobs on a single processor, subject to different constraints and/or cost functions. Many different SMS problems have been solved with MAs. For example, França *et al.* [74, 78] and Mendes *et al.* [79] tackle the SMS problem with sequence-dependent setup times and due dates, aiming to minimizing the total tardiness of the schedule (i.e., the sum of the tardiness of each job). This is done with a structured MA (with ternary tree topology, as described in Sect. 3.4) using two different schemes for both local search (insertion and swaps) and mutation (light and heavy, cf. Sect. 3.2). Sevaux and Dauzère-Pérès [42] also tackle the SMS problem with MAs, considering the weighted number of late jobs as quality measure. They use a low-cost local search scheme whose complexity is $O(n^2)$, n being the number of jobs, and compare different decoding functions for computing a feasible schedule from a plain permutation of the jobs. The same SMS problem with the added requirement of *robustness* (that is, high quality solutions remaining good when some changes take place in the input data) is tackled by Sevaux and Sörensen in [84]. Maheswaran *et al.* [40] consider a related objective function, namely the minimization of the total weighted tardiness. They use a simple local search scheme based on swaps, which is terminated as soon as the first fitness improvement is achieved.

Parallel machine scheduling (PMS) is the natural generalization of the SMS to multiple processors. Cheng and Gen's work [85] is one of the first memetic approaches to PMS. They consider a MA with a sophisticated decoding mechanism based on heuristics for the SMS. França *et al.* [73], Mendes *et al.* [75, 77], and Moscato *et al.* [80] tackle successfully the PMS using the same structured MA described before for the SMS. This approach is also used by these authors in flowshop scheduling [76, 45], see below. Also, Bonfim and Yamakami [86] present an interesting proposal in which a simple MA (using a plain hill-climber for local search) is endowed with a neural network in order to evaluate solutions.

Flowshop scheduling (FSS) is another conspicuous problem variant, in which the jobs have to be processed on m machines in the same order. Yamada and Reeves [87, 88, 54] consider a MA that uses with *path relinking* [89] (PR) for recombination. PR is a method that explores a sequence of solutions defined by two endpoints: given initial solution s and a final solution s', relevant attributes of the former are successively dropped, and substituted by attributes from the latter. Along this path, a local search procedure can be eventually triggered. Yeh

[47] tackles the problem with a MA that incorporates a greedy method to inject a single high-quality solution in the population, and a local search scheme based in two neighborhoods (swaps and insertions).

Several authors have also considered *hybrid flowshop scheduling* (HFSS) problems, in which jobs have to be sequenced through k stages, being a number of identical machines available for each stage (this number being in general different for each stage). Sevaux *et al.* [90, 91] have tackled this problem combining MAs with *constraint programming* (CP) (see Sect. 5.1). In this case, the CP method (actually a branch-and-bound algorithm) is used as local search mechanism. Ždánský and Poživil [55] also deal with the HFSS. The main feature of their MA is the use of TS as an embedded method for performing local search.

FSS problems can be generalized to *job-shop scheduling* (JSS) problems, in which each job follows its own technological processing sequence on the set of machines, and further to *open-shop scheduling* (OSS) problems, in which no processing sequence is imposed for jobs (i.e., a job requires being processed in some subset of machines, and this can be done in any order). The JSS problem has been dealt by Yamada and Nakano [38, 92, 93] using the PR approach mentioned before for recombination. Also for the JSS, Wang and Zheng [94, 39] consider a MA that incorporates simulated annealing as local search mechanism. Quite interestingly, they name their approach GASA or "modified GA" rather than MA. As to the OSS problem, MAs have been used by Liaw [41]. In this case, the MA features tabu search as local search mechanism, and uses two *ad hoc* heuristics for initializing the population.

4.2 Timetabling

Timetabling consists basically of allocating a number of events to a finite number of time periods (also called *slots*) in such a way that a certain set of constraints is satisfied. Two types of constraints are usually considered, the so called *hard constrains*, that is, those constraints that have to be fulfilled under all circumstances, and *soft constraints*, that is, those constraints that should be fulfilled if possible. In some cases, it is not possible to fully satisfy all the constraints, and the aim turns to be finding good solutions subject to certain quality criteria (e.g., minimizing the number of violated constraints, or alternatively maximizing the number of satisfied hard constraints, whilst the number of violated soft constraints is minimized).

Timetabling arises in many different forms that differ mainly in the kind of event (e.g., exams, lectures, courses, etc.) to be scheduled. By the mid 1990s, it was already suggested that incorporating some amount of local search within evolutionary algorithms might enhance the quality of final solutions [95, 96], and experiments with directed and targeted mutation were addressed [97]. From then on, MAs have been shown to be specially useful to tackle timetabling in each of its modalities as shown in the following.

The *university exam timetabling* (i.e., scheduling a number of exams in a given set of sessions avoiding clashes, e.g., no student has two exams at the

same time) is one of the instances that have attracted more interest for evolutionary techniques. For example, Burke *et al.* [43] propose a MA that uses a combination of mutation and local search. The local search component consists of a hill climber applied after the mutation process (two operators are proposed). In general, this algorithm does not perform well in highly constrained problems, and the reason seems to be that the local search operator is less effective. A similar idea taking into account recombination operators instead of mutation operators is investigated by Burke and Newall [98] with unproductive results. They also describe in [99] a multi-stage memetic algorithm that decomposes larger problems into smaller components, and then applies to each of the subproblems a similar MA to that in [43]. The basic idea is to decompose the set of events in k subsets phases ($k = 3$ in the paper) and then schedule the original set of events in k phases. To avoid non-schedulable subsets, an idea from heuristic sequencing methods is applied, choosing subsets according to a smart ordering. This proposals follows the maxima of *divide and conquer* with the aim of reducing the complexity of the problem. Indeed, this idea improves both the time to find a solution, and the quality of solutions with respect to the original MA applied over the whole original problem. Batenburg and Palenstijn [60] describe an alternative multi-stage algorithm constructed from the replacement of the MA used in [99] by TS.

Several authors have also suggested memetic solutions to tackle the problem of producing a periodical (usually weekly) timetable to allocate a set of lecturers and/or students in a certain number of time slots and rooms, with varying facilities and student capacities (i.e., the *university course timetabling problem*). For instance, Alkan and Özcan [100] use a set of violation directed mutation operators, and a violation direct hill climbing operator in their MAs. Rossi-Doria and Paechter [51] describe a different proposal where local search consists of a stochastic process that transforms infeasible timetables into feasible ones. Local search is applied initially on each solution in the initial population, and from then on, on each child generated between generations. Wilke *et al.* [101] consider a variant of the problem in the context of high schools. Their MA incorporates a mechanism for self-adapting the parameters that control the application of local search. Additional information on memetic approaches to course and exam timetabling problems can be found in [102, 50, 103, 104].

Public transportation scheduling is another timetabling problem that is attracting increasing interest, specially in the railway area. For instance, Greistorfer [57, 58] is concerned with the problem of finding a schedule of train arrivals in a railway station with a number of different lines passing through it. To obtain such a schedule, a MA incorporating tabu search is used. Semet and Schoenauer [105] deal with the problem of minimizing the resulting delays in train timetables in case of small perturbations of the traffic. To do so, they consider an indirect approach based on permutations representing train ordering, and combine it with ILOG CPLEX, a mathematical programming tool. The cooperation is performed in an autonomous way: initially the EA computes a good solution that is then provided as input to CPLEX.

The *automatic timetabling of sport leagues* is another variant that has also been solved successfully by MAs. Costa [56] describes an evolutionary tabu search (ETS) algorithm that combines the mechanisms of GAs and tabu search. The basic idea is replacing the mutation step in the GA by a search in the space of feasible solutions commanded by the tabu search. This choice of TS for performing LS is dictated by the reported superiority of this approach over other LS techniques such as SA in the domain of graph coloring [106, 107]. The ETS is applied to construct schedules on the National Hockey League of North America. Schönberger *et al.* [108] propose a MA to schedule a regional table-tennis league in Germany. Here, constraint programming (see Sect. 5.1) is used to experiment with the order in which the decision variables are instantiated in the heuristic.

4.3 Manpower Scheduling

Manpower scheduling (also called *rostering* or *human scheduling*) is concerned with the arrangement of employee timetables in an institution. To solve the problem, a set of constraints or preferences (involving the employees, the employers, and even the customers) must be considered. The goal is to find the best shifts and resource assignments that satisfy these constraints.

One of the most popular instances is *nurse scheduling*, i.e., allocating the shifts (day and night shifts, holidays, etc.) for nurses under various constraints. Traditionally, this problem has been tackled via an integer programming formulation, although it has also attracted the attention of the evolutionary community and many proposals of MAs have been done. For instance, Aickelin [109] analyzes the effect of a family of GAs applied to nurse rostering. He concludes that basic GAs cannot solve the problem, and that the results can be improved via specialized operators and local searches. To do so, De Causmaecker and van den Berghe [110] propose the use of tabu search as a local heuristic in a MA.

Burke *et al.* [59] present a number of MAs which use a steepest descent improvement heuristic. These MAs only consider locally optimal solutions. The MAs are compared to previously published TS results and, later, hybridized with this TS algorithm (either as local improvement or as an improved method to be applied over the best solution found by the MA; in both cases this combination produces the best overall results in terms of quality of the solutions). Gröbner and Wilke [111] describe a MA that incorporates reparing operators, applied at an adaptive rate (cf. [101] for timetabling). Burke *et al.* [112] discuss a set of MAs that apply local search on every solution in the population. The range of these MAs varies from those already presented in [59] to new ones that consider random selection in different stages of the algorithm (e.g., in the local search step, in the parent selection, etc.). Özcan [113] has also tackled the nurse rostering problem via a memetic approach based on the same setting proposed in [100] for timetabling. Basically, Özcan proposes a self-adaptive violation-directed hierarchical hill climbing (VDHC) method as a part of the MA; VDHC provides a framework for the cooperation of a set of hill climbers targeted to specific constraint types. The idea is very similar to the VNS approach.

A different problem –*driver scheduling*– is tackled by Li and Kwan [114]. They present a GA with fuzzy evaluation (GAFE) that can be catalogued as a MA. The basic idea is similar to that of the GRASP metaheuristic [115] in the sense that GAFE also applies a greedy heuristic to obtain feasible solutions and performs searches based on multiple solutions to improve the local optimum. In GAFE, fuzzy set theory is applied in the evaluation of potential shifts based on fuzzified criteria represented by fuzzy membership functions. These functions are weighted and the GA is precisely employed to tune these weights. This same approach is extended in [116] by a self-adjusting approach that can be viewed as a particular hybrid of population-based search and local search.

4.4 Industrial Planning

Industrial planning comprises those activities directed to the development, optimization, and maintenance of industrial processes. Roughly speaking, this amounts to producing a list of activities (a plan) to achieve some pre-defined goal. In some contexts, planning can be considered a prior stage to scheduling, the latter being responsible for arranging in time those planned activities. However, this distinction is not always clear. An example of this can be found in *maintenance scheduling*, that is, organizing the activities required to keep a certain set of facilities at a functioning level. Typically, this involves fulfilling several constraints related to the external demands the system has to serve (e.g., an electricity transmission system must keep supplying the demanded energy even if some station is down due to maintenance reasons). Additionally, maintenance costs must be kept low, thus introducing an optimality criterion.

Several maintenance scheduling problems have been attacked with MAs. Burke and Smith [117, 118] consider the maintenance of thermal generators. A rolling maintenance plan is sought, such that capacity and output constraints are not violated, and such that the total combined cost of production and maintenance is minimized. They compared MAs incorporating either HC, SA, or TS. It was shown that the MA with TS performed slightly better than the HC-based and SA-based variants. The influence of local search was determinant in the performance, to the point that heuristic initialization of the populations seems to exert a negligible effect. Digalakis and Margaritis [61] further studied this problem from the point of view of parallel multi-population MAs. In their experiments, a MA with multiple populations using different local search techniques produces better results than homogeneous multi-population MAs, using just one kind of local improver.

Burke and Smith [44, 62] also address the maintenance problem of a British regional electricity grid. As before, MAs with HC, SA, and TS are compared. In order to alleviate the cost of local search, it is limited to a small number of iterations after the first local optimum is found. In this case, the HC-based and the TS-based MAs perform similarly, providing the best results and outperforming the SA-based MA. This result was consistent with previous work by the authors [119] indicating that TS was better than SA in terms of solution quality. TS was also slightly better than HC, but at the cost of a higher running time.

There have been other attempts to deploy MAs on industrial planning problems. Not related with maintenance scheduling, but sharing several important features, Evans and Fletcher [120] have considered the *boiler scheduling* problem. The goal is scheduling the operation of boilers in a power plant, so as to optimize the production of pressurized steam. The problem exhibits some production constraints that are considered by an *ad hoc* heuristic in order to produce an initial population of feasible solutions. Local search is implemented as a simple hill-climbing step, i.e., a solution is modified, and the change is kept only if it results in a quality improvement. Notice that this problem is also strongly related to the area of power scheduling, one of whose most conspicuous members is the *unit commitment* problem. Although MAs have been applied here as well, e.g., [121], an overview of these applications is beyond the scope of this work.

5 Directions for Future Developments

Unlike other optimization techniques, MAs were explicitly conceived as a eclectic paradigm, open to the integration of other techniques (metaheuristic or not). Ultimately, this ability to synergistically combine with diverse methods is one of the major reasons of their success. The availability of numerous alternative (and certainly complementary) optimization trends, tools, and methods thus offers a huge potential for future developments. We shall provide an overview of these possibilities in this section.

5.1 Hybridization with Constraint Programming

Most scheduling problems can be naturally formulated as constraint satisfaction problems (CSPs) involving a high number of constraints. In evolutionary approaches, constraint handling represents a difficult task that can be managed in different ways as mentioned in Sect. 3.1, e.g., using a suitable encoding in the space of feasible solutions, or integrating constraints in the evaluation process in form of penalty functions, among other approaches. In any case, dealing with constraints is essential for solving scheduling problems.

A natural way to manage constraints and CSPs is *constraint programming* [122–126] (CP). CP is a sound programming paradigm based on strong theoretical foundations [127] that represents a heterogeneous field of research ranging from theoretical topics in mathematical logic to practical applications in industry. As a consequence, CP is attracting also widespread commercial interest since it is suitable for modelling a wide variety of optimizations problems, particularly, problems involving heterogeneous constraints and combinatorial search.

Optimization in CP is usually based on a form of branch and bound (although other alternative models are also proposed, e.g. [128]), that is, as soon as a solution is found, a further constraint is added, so that from that point on, the value of the optimizing criterion must be better than the value just found. This causes the system to backtrack until a better solution is found. When no further solutions can be found the optimum value is known. CP techniques are complete

methods, and thus always guarantee in optimization problems that (1) if there exist at least a solution, the solution found is optimal, and (2) if a solution is not found, it is guaranteed that no solution exist. CP techniques have already been applied to a wide range of scheduling and resource allocation problems (e.g., [129–132]), and there exist many successful applications (e.g., [133–135]).

Compared to evolutionary algorithms, CP systems present some advantages. For instance, one could argue that these systems do not require excessive tuning to fit the problem, and thus are usually easier to modify and maintain; they can also handle certain classes of constraints better than MAs, e.g., preferences [136, 137] since, these can be directly modelled as soft constraints, and one has the possibility of controlling which of them are relaxed (whereas, in general, in evolutionary techniques constraints are simply part of the evaluation function, or are present in the representation of solutions). However, the nature of complete-search techniques of CP is also its main drawback since the time needed to find the optimal solution can be prohibitive for large problems. Hence, stochastic techniques (e.g., MAs) may be better when the search space is huge.

In fact, we can say that CP and MAs are two complementary worlds that clearly can profit one from the other. The hybridization of both approaches opens very interesting lines of research. In this sense, some appealing hybrid proposals to scheduling problems have recently appeared. We have already mentioned some of these, i.e., [108, 90, 91]. Further in this line, Backer *et al.* [138] describe a method for using local search techniques within a CP framework, and apply this technique to vehicle routing problems. To avoid the search getting trapped in local minima, they investigate the use of several meta-heuristics (from a simple TS method to guided local search). Also, Yun and Gen [139] use CP techniques for dealing with the preemptive and non-preemptive case in a single machine job-shop scheduling problem. They consider constraints of different types (e.g., temporal constraints, resource constraints, etc.) and use them for generating the initial population. Merlot *et al.* [140] propose a three-phase hybrid algorithm to deal with timetabling problems. In the first phase, CP is applied with the aim of obtaining a feasible solution (if any): a specialized constraint propagation algorithm is firstly applied to reduce the domain of the constrained variables and then, when no further reduction is possible, enumeration strategies are applied to reactivate the propagation. Moreover, with the aim of improving quality, the solution obtained by the CP method is used as starting point of a simulated annealing-based phase and, in a third phase a hill climber is also used.

In general, we argue that MAs can help CP to tackle larger problems and CP can help MAs to reduce drastically the search space by removing infeasible regions what would allow to focus the evolution in the promising regions. Some initial steps have already been done in this exciting line of researching [141].

5.2 Emergent Technologies

It is clear that our world is getting increasingly complex at an accelerated rate, at least from a technological point of view (famous Moore's Law being just an example of this trend). In order to cope with the optimization problems to come,

in particular those from the areas of planning and scheduling, optimization tools have to adapt to this complexity. This means that traditional, one-dimensional, sequential approaches must move aside to make room for the next generation of optimization techniques. Focusing in MAs, some of the topics that will become increasingly important in the next years are multi-objective optimization, self-adaptation, and autonomous functioning.

Starting with *multi-objective optimization* (MOO), it is clear that the existence of many different cost functions for a single problem (e.g., machine scheduling, cf. Sect. 4.1) is an indication of (1) the richness of these problems, and (2) the inappropriateness of single-objective optimization to grasp many of their practical implications. Although MOO is hardly an emerging paradigm (in the sense of having been extensively studied in the last decades), the development of multi-objective MAs for scheduling and planning is still a developing field. Several proposals have been made, e.g., [63, 65, 142], but clearly, there is still a long way to go in exploring new strategies for adapting MAs to MOO.

Another crucial feature of MAs that deserves further exploration is *self-adaptation*. As anticipated in [11], future MAs will work in at least two levels and two time scales: in the short-time scale, a set of agents would explore the search space associated to the problem; in the long-time scale the MA would adapt the heuristics associated with the agents. This idea is at the core of the *memeplexes* suggested by Krasnogor and Smith [143]. Some work has already been done in this area [116]. Very related ideas are also currently being developed in *hyperheuristics*, see e.g. [144, 145]. A hyperheuristic is a high-level heuristic which adaptively controls the combination of several low-level heuristics. Hyperheuristics have been successfully applied to scheduling problems [146–150], and offer interesting prospects for their combination with MAs.

Finally, *autonomous functioning* is another feature that has to be boosted in near-future MAs. Recall the use of the term "agent" in the description of the functional pattern of MAs (see Sect. 3). Indeed, the original conception of MAs envisioned the search as a rather decoupled process, that is, with inter-agent communication being less frequent than individual improvement. This fits very well with the behavior of multi-agent systems, which have been also applied to planning and scheduling with satisfactory results [151–153]. Enhancing the autonomous component of MA agents would redound in new possibilities for their efficient parallelization in distributed systems, as well as open a plethora of research lines such as, e.g., the use of epistemic logic systems for modelling the distributed belief of the agents on the optimal solution [154].

5.3 Other Interesting Scheduling Problems

There exist several scheduling problems that have not been treated –to the best of our knowledge– by MAs. These do not just provide challenging optimization tasks, but can also open new scenarios for further research on MAs for scheduling.

The *scheduling of social tournaments* (SST) has attracted significant attention in recent years since they arise in many practical applications, and induce

highly combinatorial problems. SST problems may be considered either as instances of timetabling problems (e.g., timetabling of sport leagues) or rostering problems (e.g., judge assignments). One of the most popular SST instances is that known as the *social golfer problem* (problem #10 in the CSPLib[2]): it consists of trying to schedule $g \times s$ golfers into g groups of s players over w weeks, such that no golfer plays in the same group with any other golfer more than once. The problem can be regarded as an optimization problem if for two given numbers g and s we ask for the maximum number of weeks the golfers can play together. An instance to the problem is characterized by a triple $w - g - s$. The initial question consisted of scheduling 32 golfers in a local golf club (i.e., $g = 8$ and $s = 4$). The optimal solution for this instance is not yet known, and the current best known solution is a 9 week schedule (i.e., $w = 9$). There also exist interesting instances and variants for this problem as the *Kirkman's schoolgirl problem* [155], the *debating tournament problem* and the *judge assignment* [156].

Pattern sequencing problems have also important applications, especially in the field of production planning (for instance, in *talent scheduling* [157, 158]). Those problems generally consist of finding a permutation of predetermined production patterns (groupings of some elementary order types) with respect to different objectives. These objectives may represent, e.g., handling costs or stock capacity restrictions, which usually leads to NP-hard problems. In these problems, the use of heuristics to construct near-optimal pattern sequences is generally assumed to be appropriate [159].

6 Concluding Remarks

One of the main conclusions that can be drawn from the extensive literature on MAs for planning, scheduling, and timetabling is they constitute a versatile and effective optimization paradigm. Indeed, MAs are one of the primary weapons in our arsenal for dealing with problems in this area. They provide an appropriate framework to seamlessly integrate successful heuristics into a single search engine. In this sense, MAs should not be regarded as competitors, but as integrators. Whenever non-hybrid metaheuristics start to reach their limits, MAs are the next natural step.

There is an important empirical component in the design of MAs. However, this does not imply that MAs are just a *plug-and-play* approach. The user can benefit from the methodological corpus available for both population-based and trajectory-based search techniques. Design by analogy is another powerful strategy in this area: although very diverse at first sight, scheduling problems have strong underlying connections; hence, knowledge transfer from one subarea to another one is not just feasible, but also likely to be useful. The selection of reproductive operators and/or local-search strategies is at any rate an open problem in methodological terms. Some guidelines for LS design in specific applications are available as shown in Sect. 4, but these are very specific, and hard to

[2] `http://www.csplib.org`

generalize. For this reason, computational considerations, such as the affordable running time, remain one of the governing factors in taking decisions with this regard. For example, more sophisticated LS techniques can provide better results regarding solution quality than plain HC, but the improvement is likely to take place after longer run times. This must be taken into account when dealing with complex scheduling problems in which evaluating local moves is computationally expensive, or in which the size of neighborhoods is huge.

New computational challenges will rise in the years to come. Scheduling problems will not just become a matter of large-scale optimization, but will also become richer and more complex. Consider for example the situation in machine scheduling, where technological developments in manufacturing processes and production strategies will result in new (multiple) objectives to optimize, additional constraints to be considered, etc. New methods will start to play an essential role, e.g., safe kernelization techniques, commonly used in the realm of parameterized complexity [160]. Metaheuristics will have to adapt to this new scenario, and eclecticism appears to be essential for this. The future looks promising for MAs.

Acknowledgments

This work was partially supported by Spanish MCyT under contracts TIN2004-7943-C04-01 and TIN2005-08818-C04-01. Thanks are also due to the reviewers for their useful comments.

References

1. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. John Wiley & Sons, New York (1966)
2. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
3. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
4. Schwefel, H.P.: Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diplomarbeit, Technische Universität Berlin, Hermann Föttinger–Institut für Strömungstechnik (1965)
5. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220** (1983) 671–680
6. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston, MA (1997)
7. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1(1)** (1997) 67–82
8. Hart, W.E., Belew, R.K.: Optimizing an arbitrary function is hard for the genetic algorithm. In Belew, R.K., Booker, L.B., eds.: Proceedings of the 4^{th} International Conference on Genetic Algorithms, San Mateo CA, Morgan Kaufmann (1991) 190–195
9. Davis, L.D.: Handbook of Genetic Algorithms. Van Nostrand Reinhold Computer Library, New York (1991)

10. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989)
11. Moscato, P.: Memetic algorithms: A short introduction. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, Maidenhead, Berkshire, England, UK (1999) 219–234
12. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Boston MA (2003) 105–144
13. Moscato, P., Cotta, C., Mendes, A.S.: Memetic algorithms. In Onwubolu, G.C., Babu, B.V., eds.: New Optimization Techniques in Engineering. Springer-Verlag, Berlin Heidelberg (2004) 53–85
14. Dawkins, R.: The Selfish Gene. Clarendon Press, Oxford (1976)
15. Culberson, J.: On the futility of blind search: An algorithmic view of "No Free Lunch". Evolutionary Computation **6** (1998) 109–127
16. Eiben, A.E., Raue, P.E., Ruttkay, Z.: Genetic algorithms with multi-parent recombination. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving From Nature III. Volume 866 of Lecture Notes in Computer Science. Springer-Verlag (1994) 78–87
17. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA (1989)
18. Jones, T.C.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico (1995)
19. Davidor, Y., Ben-Kiki, O.: The interplay among the genetic algorithm operators: Information theory tools used in a holistic way. In Männer, R., Manderick, B., eds.: Parallel Problem Solving From Nature II, Amsterdam, Elsevier Science Publishers B.V. (1992) 75–84
20. Radcliffe, N.J.: Non-linear genetic representations. In Männer, R., Manderick, B., eds.: Parallel Problem Solving From Nature II, Amsterdam, Elsevier Science Publishers B.V. (1992) 259–268
21. Fox, B.R., McMahon, M.B.: Genetic operators for sequencing problems. In Rawlins, G.J.E., ed.: Foundations of Genetic Algorithms I, San Mateo, CA, Morgan Kaufmann (1991) 284–300
22. Cotta, C., Troya, J.M.: Genetic forma recombination in permutation flowshop problems. Evolutionary Computation **6** (1998) 25–44
23. Mathias, K., Whitley, L.D.: Genetic operators, the fitness landscape and the traveling salesman problem. In Männer, R., Manderick, B., eds.: Parallel Problem Solving From Nature II, Amsterdam, Elsevier Science Publishers B.V. (1992) 221–230
24. Goldberg, D.E., Lingle Jr., R.: Alleles, loci and the traveling salesman problem. In Grefenstette, J.J., ed.: Proceedings of the 1^{st} International Conference on Genetic Algorithms, Hillsdale NJ, Lawrence Erlbaum Associates (1985) 154–159
25. Davidor, Y.: Epistasis Variance: Suitability of a Representation to Genetic Algorithms. Complex Systems **4** (1990) 369–383
26. Davidor, Y.: Epistasis variance: A viewpoint on GA-hardness. In Rawlins, G.J.E., ed.: Foundations of Genetic Algorithms I, San Mateo, CA, Morgan Kaufmann (1991) 23–35
27. Radcliffe, N.J., Surry, P.D.: Fitness Variance of Formae and Performance Prediction. In Whitley, L.D., Vose, M.D., eds.: Foundations of Genetic Algorithms III, San Francisco, CA, Morgan Kaufmann (1994) 51–72

28. Manderick, B., de Weger, M., Spiessens, P.: The Genetic Algorithm and the Structure of the Fitness Landscape. In Belew, R.K., Booker, L.B., eds.: Proceedings of the 4^{th} International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufmann (1991) 143–150
29. Dzubera, J., Whitley, L.D.: Advanced Correlation Analysis of Operators for the Traveling Salesman Problem. In Schwefel, H.P., Männer, R., eds.: Parallel Problem Solving from Nature III. Volume 866 of Lecture Notes in Computer Science., Dortmund, Germany, Springer-Verlag, Berlin, Germany (1994) 68–77
30. Aickelin, U., Dowsland, K.: Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. Journal of Scheduling **3** (2000) 139–153
31. Puente, J., Vela, C.R., Prieto, C., Varela, R.: Hybridizing a genetic algorithm with local search and heuristic seeding. In Mira, J., Álvarez, J.R., eds.: Artificial Neural Nets Problem Solving Methods. Volume 2687 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2003) 329–336
32. Varela, R., Serrano, D., Sierra, M.: New codification schemas for scheduling with genetic algorithms. In Mira, J., Álvarez, J.R., eds.: Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach. Volume 3562 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2005) 11–20
33. Varela, R., Puente, J., Vela, C.R.: Some issues in chromosome codification for scheduling with genetic algorithms. In Castillo, L., Borrajo, D., Salido, M.A., Oddi, A., eds.: Planning, Scheduling and Constraint Satisfaction: From Theory to Practice. Volume 117 of Frontiers in Artificial Intelligence and Applications. IOS Press (2005) 1–10
34. Radcliffe, N.J.: The algebra of genetic algorithms. Annals of Mathematics and Artificial Intelligence **10** (1994) 339–384
35. Oğuz, C., Ercan, M.F.: A genetic algorithm for hybrid flow-shop scheduling with multiprocessor tasks. Journal of Scheduling **8** (2005) 323–351
36. Cotta, C., Troya, J.M.: Embedding branch and bound within evolutionary algorithms. Applied Intelligence **18** (2003) 137–153
37. Ibaraki, T.: Combination with dynamic programming. In Bäck, T., Fogel, D., Michalewicz, Z., eds.: Handbook of Evolutionary Computation. Oxford University Press, New York NY (1997) D3.4:1–2
38. Yamada, T., Nakano, R.: A genetic algorithm with multi-step crossover for job-shop scheduling problems. In: Proceedings of the 1^{st} International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, Sheffield, UK, Institution of Electrical Engineers (1995) 146–151
39. Wang, L., Zheng, D.Z.: A modified genetic algorithm for job-shop scheduling. International Journal of Advanced Manufacturing Technology **20** (2002) 72–76
40. Maheswaran, R., Ponnambalam, S.G., Aranvidan, C.: A meta-heuristic approach to single machine scheduling problems. International Journal of Advanced Manufacturing Technology **25** (2005) 772–776
41. Liaw, C.F.: A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research **124** (2000) 28–42
42. Sevaux, M., Dauzère-Pérès, S.: Genetic algorithms to minimize the weighted number of late jobs on a single machine. European Journal of Operational Research **151** (2003) 296–306
43. Burke, E.K., Newall, J., Weare, R.: A memetic algorithm for university exam timetabling. In Burke, E.K., Ross, P., eds.: Practice and Theory of Automated Timetabling. Volume 1153 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1996)

44. Burke, E.K., Smith, A.J.: A memetic algorithm to schedule planned grid maintenance. In Mohammadian, M., ed.: Computational Intelligence for Modelling, Control and Automation, IOS Press (1999) 12–127
45. França, P.M., Gupta, J.N.D., Mendes, A.S., Moscato, P., Veltnik, K.J.: Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. Computers and Industrial Engineering **48** (2005) 491–506
46. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research **130** (2001) 449–467
47. Yeh, W.C.: A memetic algorithm fo the $n/2$/Flowshop/αF+ βC$_{max}$ scheduling problem. International Journal of Advanced Manufacturing Technology **20** (2002) 464–473
48. Varela, R., Gómez, A., Vela, C.R., Puente, J., Alonso, C.: Heuristic generation of the initial population in solving job shop problems by evolutionary strategies. In Mira, J., Sánchez-Andrés, J.V., eds.: Foundations and Tools for Neural Modeling. Volume 1606 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1999) 690–699
49. Varela, R., Puente, J., Vela, C.R., Gómez, A.: A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. European Journal of Operational Research **145** (2003) 57–71
50. Burke, E.K., Petrovic, S.: Recent research directions in automated timetabling. European Journal of Operational Research **140** (2002) 266–280
51. Rossi-Doria, O., Paechter, B.: A memetic algorithm for university course timetabling. In: Combinatorial Optimisation 2004 Book of Abstracts, Lancaster, UK, Lancaster University (2004) 56
52. Laguna, M., Martí, R.: Scatter Search. Methodology and Implementations in C. Kluwer Academic Publishers, Boston MA (2003)
53. Nagata, Y., Kobayashi, S.: Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In Bäck, T., ed.: Proceedings of the Seventh International Conference on Genetic Algorithms, East Lansing, EE.UU., San Mateo, CA, Morgan Kaufmann (1997) 450–457
54. Yamada, T., Reeves, C.R.: Solving the C_{sum} permutation flowshop scheduling problem by genetic local search. In: 1998 IEEE International Conference on Evolutionary Computation, Piscataway, NJ, IEEE Press (1998) 230–234
55. Žďánský, M., Poživil, J.: Combination genetic/tabu search algorithm for hybrid flowshops optimization. In: Proceedings of ALGORITMY 2002 – Conference on Scientific Computing, Vysoke Tatry, Podbanske, Slovakia (2002) 230–236
56. Costa, D.: An evolutionary tabu search algorithm and the NHL scheduling problem. INFOR **33** (1995) 161–178
57. Greistorfer, P.: Hybrid genetic tabu search for a cyclic scheduling problem. In Voß, S., Martello, S., Osman, I.H., Roucairol, C., eds.: Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, Boston, MA (1998) 213–229
58. Greistorfer, P.: Genetic tabu search extensions for the solving of the cyclic regular max-min scheduling problem. In: International Conference on Operations Research (OR98), Zürich, Schwitzerland (1998)
59. Burke, E.K., Cowling, P.I., De Causmaecker, P., van den Berghe, G.: A memetic approach to the nurse rostering problem. Applied Intelligence **15** (2001) 199–214
60. Batenburg, K.J., Palenstijn, W.J.: A new exam timetabling algorithm. In Heskes, T., Lucas, P., Vuurpijl, L., Wiegerinck, W., eds.: Proceedings of the 15^{th}

Belgian-Dutch Conference on Artificial Intelligence BNAIC'03, Nijmegen, The Netherlands (2003) 19–26
61. Digalakis, J., Margaritis, K.: A multipopulation memetic model for the maintenance scheduling problem. In: Proceedings of the 5^{th} Hellenic European Conference on Computer Mathematics and its Applications, Athens, Greece, LEA Press (2001) 318–323
62. Burke, E.K., Smith, A.J.: A memetic algorithm to schedule planned maintenance for the national grid. Journal of Experimental Algorithmics **4** (1999) 1–13
63. Ponnambalam, S.G., Mohan Reddy, M.: A GA-SA multiobjective hybrid search algorithm for integrating lot sizing and sequencing in flow-line scheduling. International Journal of Advanced Manufacturing Technology **21** (2003) 126–137
64. Houck, C., Joines, J.A., Kay, M.G., Wilson, J.R.: Empirical investigation of the benefits of partial lamarckianism. Evolutionary Computation **5** (1997) 31–60
65. 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 **7** (2003) 204–223
66. Jones, T.C., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L.J., ed.: Proceedings of the 6^{th} International Conference on Genetic Algorithms, Morgan Kaufmann (1995) 184–192
67. Merz, P., Freisleben, B.: Fitness landscapes and memetic algorithm design. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, Maidenhead, Berkshire, England, UK (1999) 245–260
68. English, T.M.: Evaluation of evolutionary and genetic optimizers: No free lunch. In Fogel, L.J., Angeline, P.J., Bäck, T., eds.: Evolutionary Programming V, Cambridge, MA, MIT Press (1996) 163–169
69. Bierwirth, C., Mattfeld, D.C., Watson, J.P.: Landscape regularity and random walks for the job shop scheduling problem. In Gottlieb, J., Raidl, G.R., eds.: Evolutionary Computation in Combinatorial Optimization. Volume 3004 of Lecture Notes in Computer Science., Berlin, Springer-Verlag (2004) 21–30
70. Grefenstette, J.J.: Genetic algorithms for changing environments. In Männer, R., Manderick, B., eds.: Parallel Problem Solving from Nature II, Amsterdam, North-Holland Elsevier (1992) 137–144
71. Hadj-Alouane, A.B., Bean, J.C., Murty, K.G.: A hybrid genetic/optimization algorithm for a task allocation problem. Journal of Scheduling **2** (1999) 181–201
72. Tomassini, M.: Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time. Springer-Verlag (2005)
73. França, P.M., Mendes, A.S., Müller, F., Moscato, P.: Memetic algorithms applied to the single machine and parallel machine scheduling problems. In: Anais da Primeira Oficina de Planejamento e Controle da Produção em Sistemas de Manufatura, Campinas, SP, Brazil (1999)
74. França, P.M., Mendes, A.S., Moscato, P.: Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times. In Despotis, D.K., Zopounidis, C., eds.: Proceedings of the 5^{th} International Conference of the Decision Sciences Institute, Athens, Greece (1999) 1708–1710
75. Mendes, A.S., Müller, F., França, P.M., Moscato, P.: Comparing meta-heuristic approaches for parallel machine scheduling problems with sequence-dependent setup times. In: Procedings of the 15^{th} International Conference on CAD/CAM Robotics and Factories of the Future, Águas de Lindóia, SP, Brazil (1999) 1–6

76. França, P.M., Gupta, J.N.D., Mendes, A.S., Moscato, P., Veltnik, K.J.: Metaheuristic approaches for the pure flowshop manufacturing cell problem. In: Proceedings of the 7^{th} International Workshop on Project Management and Scheduling, Osnabrück, Germany (2000) 128–130
77. Mendes, A.S., França, P.M., Moscato, P.: Fuzzy-evolutionary algorithms applied to scheduling problems. In: Proceedings of the 1^{st} World Conference on Production and Operations Management, Seville, Spain (2000) 1–10
78. França, P.M., Mendes, A.S., Moscato, P.: A memetic algorithm for the total tardiness single machine scheduling problem. European Journal of Operational Research **132** (2001) 224–242
79. Mendes, A.S., França, P.M., Moscato, P.: Fitness landscapes for the total tardiness single machine scheduling problem. Neural Network World **2** (2002) 165–180
80. Moscato, P., Mendes, A., Cotta, C.: Scheduling and production & control. In Onwubolu, G.C., Babu, B.V., eds.: New Optimization Techniques in Engineering. Springer-Verlag, Berlin Heidelberg (2004) 655–680
81. Cotta, C., Alba, E., Troya, J.M.: Stochastic reverse hillclimbing and iterated local search. In: Proceedings of the 1999 Congress on Evolutionary Computation, Washington D.C., IEEE Neural Network Council - Evolutionary Programming Society - Institution of Electrical Engineers (1999) 1558–1565
82. Cobb, H.G.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington DC (1990)
83. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms. PhD thesis, Faculty of Engineering, Computer Science and Mathematics. University of the West of England. Bristol, United Kingdom (2002)
84. Sevaux, M., Sörensen, K.: A genetic algorithm for robust schedules. In: Proceedings of the 8^{th} International Workshop on Project Management and Scheduling. (2002) 330–333
85. Cheng, R., Gen, M.: Parallel machine scheduling problems using memetic algorithms. Computers and Industrial Engineering **33** (1997) 761–764
86. Bonfim, T.R., Yamakami, A.: Neural network applied to the coevolution of the memetic algorithm for solving the makespan minimization problem in parallel machine scheduling. In Ludermir, T.B., de Souto, M.C.P., eds.: Proceedings of the 7^{th} Brazilian Symposium on Neural Networks (SBRN 2002), Recife, Brazil, IEEE Computer Society (2002) 197–199
87. Yamada, T., Reeves, C.R.: Permutation flowshop scheduling by genetic local search. In: Proceedings of the 2^{nd} International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, London, UK, Institution of Electrical Engineers (1997) 232–238
88. Reeves, C.R., Yamada, T.: Genetic algorithms, path relinking and the flowshop sequencing problem. Evolutionary Computation **6** (1998) 230–234
89. Glover, F.: Scatter search and path relinking. In Corne, D., Dorigo, M., Glover, F., eds.: New Methods in Optimization. McGraw-Hill, London (1999) 291–316
90. Sevaux, M., Jouglet, A., Oğuz, C.: MLS+CP for the hybrid flowshop scheduling problem. In: Workshop on the Combination of metaheuristic and local search with Constraint Programming techniques, Nantes, France (2005)
91. Sevaux, M., Jouglet, A., Oğuz, C.: Combining constraint programming and memetic algorithm for the hybrid flowshop scheduling problem. In: ORBEL 19^{th} annual conference of the SOGESCI-BVWB, Louvain-la-Neuve, Belgium (2005)

92. Yamada, T., Nakano, R.: A fusion of crossover and local search. In: IEEE International Conference on Industrial Technology ICIT'96, Shangai, China, IEEE Press (1996) 426–430
93. Yamada, T., Nakano, R.: Scheduling by genetic local search with multi-step crossover. In Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.P., eds.: Parallel Problem Solving From Nature IV. Volume 1141 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1996) 960–969
94. Wang, L., Zheng, D.Z.: An effective hybrid optimization strategy for job-shop scheduling problems. Computers & Operations Research **28** (2001) 585–596
95. Paechter, B., Cumming, A., Luchian, H.: The use of local search suggestion lists for improving the solution of timetable problems with evolutionary algorithms. In Fogarty, T.C., ed.: AISB Workshop on evolutionary computing. Volume 993 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1995) 86–93
96. Rankin, B.: Memetic timetabling in practice. In Burke, E.K., Ross, P., eds.: Proceedings of the 1^st International Conference on the Practice and Theory of Automated Timetabling. Volume 1153 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1996) 266–279
97. Paechter, B., Cumming, A., Norman, M.G., Luchian, H.: Extensions to a memetic timetabling system. In Burke, E.K., Ross, P., eds.: Proceedings of the 1^{st} International Conference on the Practice and Theory of Automated Timetabling. Volume 1153 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (1996) 251–265
98. Burke, E.K., Newall, J.P.: Investigating the benefits of utilising problem specific heuristics within a memetic timetabling algorithm. Workin Paper NOTTCS-TR-97-6, dept. of Computer Science, University of Nottingham, UK (1997)
99. Burke, E.K., Newall, J.: A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation **3** (1999) 63–74
100. Alkan, A., Özcan, E.: Memetic algorithms for timetabling. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Canberra, Australia, IEEE Press (2003) 1796–1802
101. Wilke, P., Gröbner, M., Oster, N.: A hybrid genetic algorithm for school timetabling. In McKay, B., Slaney, J., eds.: AI 2002: Advances in Artificial Intelligence, 15^{th} Australian Joint Conference on Artificial Intelligence. Volume 2557 of Lecture Notes in Computer Science., Canberra, Australia, Springer (2002) 455–464
102. Burke, E.K., Jackson, K., Kingston, J.H., Weare, R.F.: Automated university timetabling: The state of the art. The Computer Journal **40** (1997) 565–571
103. Burke, E.K., Landa Silva, J.D.: The design of memetic algorithms for scheduling and timetabling problems. In Krasnogor, N., Hart, W., Smith, J., eds.: Recent Advances in Memetic Algorithms. Volume 166 of Studies in Fuzziness and Soft Computing. Springer-Verlag (2004) 289–312
104. Petrovic, S., Burke, E.K.: University timetabling. In Leung, J., ed.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapman Hall/CRC Press (2004)
105. Semet, Y., Schoenauer, M.: An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In: Proceedings of the 2005 Congress on Evolutionary Computation, Edinburgh, UK, IEEE Press (2005) 2752–2759
106. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39** (1987) 345–351

107. Costa, D.: On the use of some known methods for T-colorings of graphs. Annals of Operations Research **41** (1993) 343–358
108. Schönberger, J., Mattfeld, D.C., Kopfer, H.: Memetic algorithm timetabling for non-commercial sport leagues. European Journal of Operational Research **153** (2004) 102–116
109. Aickelin, U.: Nurse rostering with genetic algorithms. In: Proceedings of young operational research conference 1998, Guildford, UK (1998)
110. De Causmaecker, P., van den Berghe, G.: Using tabu search as a local heuristic in a memetic algorithm for the nurse rostering problem. In: Proceedings of the 13^{th} Conference on Quantitative Methods for Decision Making (ORBEL XIII), Brussels (1999)
111. Gröbner, M., Wilke, P.: Optimizing employee schedules by a hybrid genetic algorithm. In Boers, E.J.W., et al., eds.: Applications of Evolutionary Computing 2001. Volume 2037 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2001) 463–472
112. Burke, E.K., De Causmaecker, P., van den Berghe, G.: Novel metaheuristic approaches to nurse rostering problems in belgian hospitals. In Leung, J., ed.: Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapman Hall/CRC Press (2004) 44.1–44.18
113. Özcan, E.: Memetic algorithms for nurse rostering. In Yolum, P., Güngör, T., Gürgen, F.S., Özturan, C.C., eds.: Computer and Information Sciences - ISCIS 2005, 20^{th} International Symposium (ISCIS). Volume 3733 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2005) 482–492
114. Li, J., Kwan, R.S.K.: A fuzzy genetic algorithm for driver scheduling. European Journal of Operational Research **147** (2003) 334–344
115. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
116. Li, J., Kwan, R.S.K.: A self adjusting algorithm for driver scheduling. Journal of Heuristics **11** (2005) 351–367
117. Burke, E.K., Smith, A.J.: Hybrid evolutionary techniques for the maintenance scheduling problem. IEEE Transactions on Power Systems **15** (2000) 122–128
118. Burke, E.K., Smith, A.J.: A memetic algorithm for the maintenance scheduling problem. In: Proceedings of the 4^{th} International Conference on Neural Information Processing ICONIP'97, Dunedin, New Zealand, Springer-Verlag (1997) 469–474
119. Burke, E., Clark, J., Smith, J.: Four methods for maintenance scheduling. In Smith, G., Steele, N., Albrecht, R., eds.: Artificial Neural Nets and Genetic Algorithms 3, Wien New York, Springer-Verlag (1998) 264–269
120. Evans, S., Fletcher, I.: A variation on a memetic algorithm for boiler scheduling. In Hotz, L., Krebs, T., eds.: Proceedings Workshop Planen und Konfigurieren (PuK-2003), Hamburg, Germany (2003)
121. Valenzuela, J., Smith, A.: A seeded memetic algorithm for large unit commitment problems. Journal of Heuristics **8** (2002) 173–195
122. Marriot, K., Stuckey, P.J.: Programming with constraints. The MIT Press, Cambridge, Massachusetts (1998)
123. Smith, B.M.: A tutorial on constraint programming. Research Report 95.14, University of Leeds, School of Computer Studies, England (1995)
124. Dechter, R.: Constraint processing. Morgan Kaufmann (2003)
125. Apt, K.R.: Principles of constraint programming. Cambridge University Press (2003)

126. Frühwirth, T., Abdennadher, S.: Essentials of constraint programming. Cognitive Technologies Series. Springer-Verlag (2003)
127. Tsang, E.: Foundations of constraint satisfaction. Academic Press, London and San Diego (1993)
128. Larrosa, J., Morancho, E., Niso, D.: On the practical use of variable elimination in constraint optimization problems: 'still life' as a case study. Journal of Artificial Intelligence Research **23** (2005) 421–440
129. Baptiste, P., Le Pape, C.: Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. Constraints **5** (2000) 119–139
130. Barták, R.: Constraint satisfaction for planning and scheduling. In Vlahavas, I., Vrakas, D., eds.: Intelligent Techniques for Planning. Idea Group, Hershey, PA, USA (2005)
131. Le Pape, C.: Constraint-based scheduling: A tutorial. Proceedings of the 1^{st} International Summer School on Constraint Programming (2005)
132. Kilby, P., Prosser, P., Shaw, P.: A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. Constraints **5** (2000) 389–414
133. Oliveira, E., Smith, B.M.: A combined constraint-based search method for single-track railway scheduling problem. In Brazdil, P., Jorge, A., eds.: Proceedings of the 10^{th} Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving. Volume 2258 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2001) 371–378
134. Wallace, R.J., Freuder, E.C.: Supporting dispatchability in schedules with consumable resources. Journal of Scheduling **8** (2005) 7–23
135. Khemmoudj, M.O.I., Porcheron, M., Bennaceur, H.: Using constraint programming and local search for scheduling of electricité de france nuclear power plant outages. In: Proceedings of the Workshop on the Combination of Metaheuristic and Local Search with Constraint Programming techniques, Nantes, France (2005)
136. Bistarelli, S.: Semirings for Soft Constraint Solving and Programming. Volume 2962 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg (2004)
137. Rossi, F.: Preference reasoning. In van Beek, P., ed.: Proceedings of the 11^{th} International Conference on Principles and Practice of Constraint Programming. Volume 3709 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2005) 9–12
138. Backer, B.D., Furnon, V., Shaw, P., Kilby, P., Prosser, P.: Solving vehicle routing problems using constraint programming and metaheuristics. Journal of Heuristics **6** (2000) 501–523
139. Yun, Y.S., Gen, M.: Advanced scheduling problem using constraint programming techniques in SCM environment. Computers & Industrial Engineering **43** (2002) 213–229
140. Merlot, L.T.G., Boland, N., Hughes, B.D., Stuckey, P.J.: A hybrid algorithm for the examination timetabling problem. In Burke, E.K., Causmaecker, P.D., eds.: Proceedings of the 4^{th} International Conference on the Practice and Theory of Automated Timetabling. Volume 2740 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2003) 207–231

141. Terashima, H.: Combinations of GAs and CSP strategies for solving examination timetabling problems. PhD thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey (1998)
142. Landa Silva, J.D., Burke, E.K., Petrovic, S.: An introduction to multiobjective metaheuristics for scheduling and timetabling. In X., G., M., S., Sörensen K. and, T.V., eds.: Metaheuristic for Multiobjective Optimisation. Volume 535 of Lecture Notes in Economics and Mathematical Systems. Springer (2004) 91–129
143. Krasnogor, N., Smith, J.E.: Emergence of profitable search strategies based on a simple inheritance mechanism. In Spector, L., et al., eds.: Proceedings of the 2001 Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2001) 432–439
144. Kendall, G., Soubeiga, E., Cowling, P.I.: Choice function and random hyperheuristics. In: Proceedings of the 4^{th} Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02). (2002) 667–671
145. Burke, E.K., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyperheuristics: an emerging direction in modern search technology. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Boston MA (2003) 457–474
146. Cowling, P.I., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In Burke, E., Erben, W., eds.: Selected Papers of the 3^{rd} PATAT conference. Volume 2079 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2000) 176–190
147. Cowling, P.I., Kendall, G., Soubeiga, E.: Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In Cagnoni, S., et al., eds.: Applications of Evolutionary Computing. Volume 2279 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2002) 1–10
148. Cowling, P.I., Kendall, G., Soubeiga, E.: Hyperheuristics: A robust optimisation method applied to nurse scheduling. In Merelo, J.J., et al., eds.: Parallel Problem Solving from Nature VII. Volume 2439 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2002) 851–860
149. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics **9** (2003) 451–470
150. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A graph-based hyper-heuristic for educational timetabling problems. European Journal of Operational Research (2006) In press.
151. Cowling, P.I., Ouelhadj, D., Petrovic, S.: Dynamic scheduling of steel casting and milling using multi-agents. Production Planning and Control **15** (2002) 1–11
152. Cowling, P.I., Ouelhadj, D., Petrovic, S.: A multi-agent architecture for dynamic scheduling of steel hot rolling. Journal of Intelligent Manufacturing **14** (2002) 457–470
153. Ouelhadj, D., Petrovic, S., Cowling, P.I., Meisels, A.: Inter-agent cooperation and communication for agent-based robust dynamic scheduling in steel production. Advanced Engineering and Informatics **18** (2005) 161–172
154. Cotta, C., Moscato, P.: Evolutionary computation: Challenges and duties. In Menon, A., ed.: Frontiers of Evolutionary Computation. Kluwer Academic Publishers, Boston MA (2004) 53–72
155. Barnier, N., Brisset, P.: Solving Kirkman's schoolgirl problem in a few seconds. Constraints **10** (2005) 7–21
156. Dotú, I., del Val, A., van Hentenryck, P.: Scheduling social tournaments. In van Beek, P., ed.: Proceedings of the 11^{th} International Conference on Principles and

Practice of Constraint Programming. Volume 3709 of Lecture Notes in Computer Science., Berlin Heidelberg, Springer-Verlag (2005) 845
157. Cheng, T.C.E., Diamond, J.: Optimal scheduling in film production to minimize talent hold cost. Journal of Optimization Theory and Applications **79** (1993) 197–206
158. Smith, B.M.: Constraint programming in practice: scheduling a rehearsal. Research Report APES-67-2003, APES group (2003)
159. Fink, A., Voß, S.: Applications of modern heuristic search methods to pattern sequencing problems. Computers & Operations Research **26** (1999) 17–34
160. Downey, R., Fellows, M.: Parameterized Complexity. Springer-Verlag (1998)