A Modern Introduction to Memetic Algorithms

Pablo Moscato and Carlos Cotta

1 Introduction and historical notes

The generic denomination of *'Memetic Algorithms'* (MAs) is used to encompass a broad class of metaheuristics (i.e. general purpose methods aimed to guide an underlying heuristic). The method is based on a population of agents and proved to be of practical success in a variety of problem domains and in particular for the approximate solution of **NP**-hard optimization problems.

Unlike traditional evolutionary computation (EC) methods, MAs are intrinsically concerned with exploiting *all available knowledge* about the problem under study. The incorporation of problem domain knowledge is not an optional mechanism, but a fundamental feature that characterizes MAs. This functioning philosophy is perfectly illustrated by the term "memetic". Coined by R. Dawkins [62], the word *'meme'* denotes an analogous to the gene in the context of cultural evolution [177]. In Dawkins' words:

> *"Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation."*

This characterization of a meme suggest that in cultural evolution processes, information is not simply transmitted unaltered between individuals.

Pablo Moscato
Centre for Bioinformatics, Biomarker Discovery and Information-based Medicine, The University of Newcastle, University Drive, Callaghan NSW 2308, Australia, e-mail: Pablo.Moscato@newcastle.edu.au

Carlos Cotta
Departamento de Lenguajes y Ciencias de la Computación, Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga, Campus de Teatinos, 29071 - Málaga, Spain, e-mail: ccottap@lcc.uma.es

In contrast, it is processed and enhanced by the communicating parts. This enhancement is accomplished in MAs by incorporating heuristics, approximation algorithms, local search techniques, specialized recombination operators, truncated exact methods, etc. In essence, most MAs can be interpreted as a search strategy in which a population of optimizing agents cooperate and compete [202]. The success of MAs can probably be explained as being a direct consequence of the *synergy* of the different search approaches they incorporate.

The most crucial and distinctive feature of MAs, the inclusion of problem knowledge mentioned above, is also supported by strong theoretical results. As Hart and Belew [108] initially stated and Wolpert and Macready [276] later popularized in the so-called *No-Free-Lunch Theorem*, a search algorithm strictly performs in accordance with the amount and quality of the problem knowledge they incorporate. This fact clearly underpins the exploitation of problem knowledge intrinsic to MAs. Given that the term *hybridization* is often used to denote the process of incorporating problem knowledge [39], it is not surprising that MAs are sometimes called 'Hybrid Evolutionary Algorithms' [61] (hybrid EAs) as well. One of the first algorithms to which the MA label was assigned dates from 1988 [202], and was regarded by many as a hybrid of *traditional* Genetic Algorithms (GAs) and *Simulated Annealing* (SA). Part of the initial motivation was to find a way out of the limitations of both techniques on a well-studied combinatorial optimization problem the MIN EUCLIDEAN TRAVELING SALESMAN problem (MIN ETSP). According to the authors, the original inspiration came from *computer game tournaments* [111] used to study *"the evolution of cooperation"* [8, 190]. That approach had several features which anticipated many current algorithms in practice today. The competitive phase of the algorithm was based on the new allocation of search points in configuration phase, a process involving a *"battle"* for survival followed by the so-called *"cloning"*, which has a strong similarity with *'go with the winners'* algorithms [4, 213]. The cooperative phase followed by local search may be better named *"go-with-the-local-winners"* since the optimizing *agents* were arranged with a topology of a two dimensional toroidal lattice. After initial computer experiments, an insight was derived on the particular relevance that the *"spatial"* organization, when coupled with an appropriate set of rules, had for the overall performance of population search processes. A few months later, Moscato and Norman discovered that they shared similar views with other researchers [100, 185] and other authors proposing *"island models"* for GAs. Spacialization is now being recognized as the "catalyzer" responsible of a variety of phenomena [189, 190]. This is an important research issue, currently only understood in a rather heuristic way. However, some proper undecidability results have been obtained for related problems [102] giving some hope to a more formal treatment.

Less than a year later, in 1989, Moscato and Norman identified several authors who were also pioneering the introduction of heuristics to improve the solutions before recombining them [99, 186] (see other references and the discussion in [177]). Particularly coming from the GA field, several authors were introducing *problem-domain knowledge* in a variety of ways. In [177] the denomination of *'memetic algorithms'* was introduced for the first time. It was also suggested that *cultural evolution* can be a better working metaphor for these metaheuristics to avoid *"biologically constrained"* thinking that was restricting progress at that time.

Ten years later, albeit unfortunately under different names, MAs have become an important optimization approach, with several successes in a variety of classical **NP**-hard optimization problems. We aim to provide an updated and self-contained introduction to MAs, focusing on their technical innards and formal features, but without loosing the perspective of their practical application and open research issues.

2 Memetic Algorithms

Before proceeding to the description of MAs, it is necessary to provide some basic concepts and definitions. Several notions introduced in the first subsection are strongly related to the field of computational complexity. Nevertheless, they may be presented in a slightly different way and pace for the sake of the subsequent development. These basic concepts will give rise to the notions of local search and population-based search, upon which MAs are founded. This latter class of search settles the scenario for *recombination*, a crucial mechanism in the functioning of MAs that will be studied to some depth. Finally, a basic algorithmic template and some guidelines for designing MAs will be presented.

2.1 Basic Concepts

An *algorithm* is a detailed step-by-step procedure for solving a *computational problem.* A computational problem P denotes a class of algoritmically-doable tasks, and it has an input domain set of *instances* denoted I_P. For each instance $x \in I_P$, there is an associated set $sol_P(x)$ which denotes the *feasible* solutions for problem P given instance x. The set $sol_P(x)$ is also known as the set of *acceptable* or *valid* solutions.

We are expected to deliver an algorithm that solves problem P; this means that our algorithm, given instance $x \in I_P$, must return at least one element y from a set of *answers* $ans_P(x)$ (also called *given solutions*) that satisfies the requirements of the problem. This is the first design issue to face. To be precise, depending on the kind of answers expected, computational problems can be classified into different categories; for instance:

- finding *all* solutions in $sol_P(x)$, i.e., *enumeration* problems.

- counting *how many* solutions exist in $sol_P(x)$, i.e. *counting* problems.
- determining whether the set $sol_P(x)$ *is empty or not*, i.e., *decision* problems.
- finding a solution in $sol_P(x)$ maximizing or minimizing a given function, i.e., *optimization* problems.

In this chapter, we will focus on the last possibility, that is, a problem will be considered *solved* by finding a certain feasible solution, i.e. either finding an *optimal* $y \in sol_P(x)$ or giving an indication that no such feasible solution exists. It is thus convenient in many situations to define a Boolean *feasibility* function $feasible_P(x, y)$ in order to identify whether a given solution $y \in ans_P(x)$ is acceptable for an instance $x \in I_P$ of a computational problem P, i.e., checking if $y \in sol_P(x)$.

An algorithm is said to *solve* problem P if it can fulfill this condition for any given instance $x \in I_P$. This definition is certainly too broad, so a more restrictive characterization for our problems of interest is necessary. This characterization is provided by restricting ourselves to the so-called *combinatorial optimization* problems . These constitute a special subclass of computational problems in which for each instance $x \in I_P$:

- the cardinality of $sol_P(x)$ is finite.
- each solution $y \in sol_P(x)$ has a *goodness integer value* $m_P(y, x)$, obtained by means of an associated *objective function* m_P.
- a partial order $\prec_P$ is defined over the set of goodness values returned by the objective function, allowing determining which of two goodness values is preferable.

An instance $x \in I_P$ of a combinatorial optimization problem P is solved by finding the best solution $y^* \in sol_P(x)$, i.e., finding a solution y^* such that no other solution $y \prec_P y^*$ exists if $sol_P(x)$ is not empty. It is very common to have $\prec_P$ defining a total order. In this case, the best solution is the one that maximizes (or minimizes) the objective function.

As an example of a combinatorial optimization problem consider the 0-1 Multiple Knapsack Problem (0-1 MKP). Each instance x of this problem is defined by a vector of profits $V = \{v_0, \cdots, v_{n-1}\}$, a vector of capacities $C = \{c_0, \cdots, c_{m-1}\}$, and a matrix of capacity constraints $M = \{m_{ij} \ : 0 \leqslant i < m,\ 0 \leqslant j < n\}$. Intuitively, the problem consists in selecting a set of objects so as to maximize the profit of this set without violating the capacity constraints. If the objects are indexed with the elements of the set $\mathbb{N}_n = \{0, 1, \cdots, n-1\}$, the answer set $ans_P(x)$ for an instance x is simply the power set of $\mathbb{N}_n$, that is, each subset of $\mathbb{N}_n$ is a possible answer. Furthermore, the set of feasible answers $sol_P(x)$ is composed of those subsets whose incidence vector B verifies $M \cdot B \leqslant C$. Finally, the objective function is defined as $m_P(y, x) = \sum_{i \in y} v_i$, i.e., the sum of profits for all selected objects, the goal being to maximize this value.

Notice that, associated with a combinatorial optimization problem, we can define its *decisional* version. To formulate the decision problem, an integer goodness value K is considered, and instead of trying to find the best solution of instance x, we ask whether x has a solution whose goodness is equal or better than K. In the above example, we could ask whether a feasible solution y exists such that its associated profit is equal or better than K.

2.2 Search Landscapes

As mentioned above, having defined the concept of combinatorial optimization problem the goal is finding at least one of the optimal solutions for a given instance. For this purpose, a search algorithm must be used. Before discussing search algorithms, three entities must be discussed. These are the *search space*, the *neighborhood relation*, and the *guiding function*. It is important to consider that, for any given computational problem, these three entities can be instantiated in several ways, giving rise to different optimization tasks.

Let us start by defining the concept of search space for a combinatorial problem P. To do so, we consider a set $\mathcal{S}_P(x)$, whose elements have the following properties:

- Each element $s \in \mathcal{S}_P(x)$ represents at least one answer in $ans_P(x)$.
- For decision problems: at least one element of $sol_P(x)$ that stands for a 'Yes' answer must be represented by one element in $\mathcal{S}_P(x)$.
- For optimization problems: at least one *optimal* element y^* of $sol_P(x)$ is represented by one element in $\mathcal{S}_P(x)$.

Each element of $\mathcal{S}_P(x)$ will be termed a *configuration*, being related to an answer in $ans_P(x)$ by a *growth function* $g : \mathcal{S}_P(x) \to ans_P(x)$. Note that the first requirement refers to $ans_P(x)$ and not to $sol_P(x)$, i.e., some configurations in the search space may correspond to infeasible solutions. Thus, the search algorithm may need being prepared to deal with this fact. If these requirements have been achieved, we say that we have a *valid representation* or *valid formulation* of the problem. For simplicity, we will just write $\mathcal{S}$ to refer to $\mathcal{S}_P(x)$ when x and P are clear from the context. People using biologically-inspired metaphors like to call $\mathcal{S}_P(x)$ the *genotype space* and $ans_P(x)$ denotes the *phenotype space*, so we appropriately refer to g as the *growth function*.

To illustrate this notion of search space, consider again the case of the 0-1 MKP. Since solutions in $ans_P(x)$ are subsets of $\mathbb{N}_n$, we can define the search space as the set of n-dimensional binary vectors. Each vector will represent the incidence vector of a certain subset, i.e., the growth function g is defined as $g(s) = g(b_0 b_1 \cdots b_{n-1}) = \{i \mid b_i = 1\}$. As mentioned above, many binary vectors may correspond to infeasible sets of objects. Another possibility is defining the search space as the set of permutations of elements in $\mathbb{N}_n$ [101]. In this case, the growth function may consist of applying a greedy construction algorithm, considering objects in the order provided by the permutation. Unlike the binary search space previously mentioned, all configurations represent feasible solutions in this case.

The role of the search space is to provide a "ground" where the search algorithm will act. Important properties of the search space that affect the dynamics of the search algorithm are related with the accessibility relationships between the configurations. These relationships are dependent of a *neighborhood function* $\mathcal{N} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$. This function assigns to each element $s \in S$ a set $\mathcal{N}(s) \subseteq S$ of neighboring configurations of s. The set $\mathcal{N}(s)$ is called the *neighborhood* of s and each member $s' \in \mathcal{N}(s)$ is called a *neighbor* of s.

It must be noted that the neighborhood depends on the instance, so the notation $\mathcal{N}(s)$ is a simplified form of $\mathcal{N}_P(s, x)$ since it is clear from the context. The elements of $\mathcal{N}(s)$ need not be listed explicitly. In fact, it is very usual to define them *implicitly* by referring to a set of possible *moves*, which define *transitions* between configurations. Moves are usually defined as *"local"* modifications of some part of s, where "locality" refers to the fact that the move is done on a single solution to obtain another single solution. This "locality", is one of the key ingredients of *local search*, and actually it has also given the name to the whole search paradigm.

As examples of concrete neighborhood definitions, consider the two representations of solutions for the 0-1 MKP presented above. In the first case (binary representation), moves can be defined as changing the values of a number of bits. If just one bit is modified at a time, the resulting neighborhood structure is the n-dimensional binary hypercube. In the second case (permutation representation), moves can be defined as the interchange of two positions in the permutation. Thus, two configurations are neighboring if, and only if, they differ in exactly two positions.

This definition of locality presented above is not necessarily related to "closeness" under some kind of distance relationship between configurations (except in the tautological situation in which the distance between two configurations s and s' is defined as the number of moves needed to reach s' from s). As a matter of fact, it is possible to give common examples of very complex neighborhood definitions unrelated to intuitive distance measures.

An important feature that must be considered when selecting the class of moves to be used in the search algorithm is its *"ergodicity"*, that is the ability, given any $s \in S$ to find a sequence of moves that can reach *all other* configurations $s' \in S$. In many situations this property is self-evident and no explicit demonstration is required. It is important since even if we have a valid representation (recall the definition above), it is necessary to guarantee *a priori* that at least one optimal solution is reachable from any given initial solution. Again, consider the binary representation of solutions for a 0-1 MKP instance. If moves are defined as single bit-flips, it is easily seen that any configuration s' can be reached from another configuration s in exactly h moves, where h is the Hamming distance between these configurations. This is not always the case though.

The last entity that must be defined is the *guiding function*. To do so, we require a set $\mathcal{F}$ whose elements are termed *fitness* values (typically $\mathcal{F} \equiv \mathbb{R}$), and a partial order $\prec_{\mathcal{F}}$ on $\mathcal{F}$ (typically, but not always, $\prec_{\mathcal{F}} \equiv <$). The guiding function is defined as a function $F_g : \mathcal{S} \to \mathcal{F}$ that associates to each configuration $s \in \mathcal{S}$ a value $F_g(s)$ that assesses the quality of the solution. The behavior of the search algorithm will be "controlled" by these fitness values.

Notice that for optimization problems there is an obvious direct connection between the guiding function F_g and the objective function m_P (and hence between partial orders $\prec_P$ and $\prec_{\mathcal{F}}$). As a matter of fact, it is very common to enforce this relationship to the point that both terms are usually considered equivalent. However, this equivalence is not necessary and, in many situations, not even desirable. For decision problems, since a solution is a 'Yes' or 'No' answer, associated guiding functions usually take the form of *distance to satisfiability*.

A typical example is the Boolean Satisfiability Problem, i.e., determining whether a Boolean expression in conjunctive normal form is satisfiable. In this case, solutions are assignments of Boolean values to variables, and the objective function m_P is a binary function returning 1 if the solution satisfies the Boolean expression, and returning 0 otherwise. This objective function could be used as guiding function. However, a much more typical choice is to use the number of satisfied clauses in the current configuration as guiding function, i.e., $F_g(s) = \sum_i f_i(s)$, the sum over clause indexes i of $f_i(s)$, defined as $f_i(s) = 0$ for a yet unsatisfied clause i, and $f_i(s) = 1$ if the clause i is satisfied. Hence, the goal is to maximize this number. Notice that the guiding function in this case is the objective function of the associated **NP**-hard optimization problem called Max SAT.

The above differentiation between objective function and guiding function is also very important in the context of constrained optimization problems, i.e., problems for which, in general, $sol_P(x)$ is chosen to be a proper subset of $ans_P(x)$. Since the growth function establishes a mapping from $\mathcal{S}$ to $ans_P(x)$, the search algorithm might need processing both feasible solutions (whose goodness values are well-defined) and infeasible solutions (whose goodness values are ill-defined in general). In many implementations of MAs for these problems, a guiding function is defined as a weighted sum of the value of the objective function and the distance to feasibility (which accounts for the constraints). Typically, a higher weight is assigned to the constraints, so as to give preference to feasibility over optimality. Several other remedies to this problem abound, including resorting to multi-objective techniques.

The combination of a certain problem instance and the three entities defined above induces a so-called *fitness landscape* [127]. Essentially, a fitness landscape can be defined as a weighted digraph, in which the vertices are configurations of the search space $\mathcal{S}$, and the arcs connect neighboring configurations. The weights are the differences between the guiding function values of the two endpoint configurations. The search can thus be seen as the process of "navigating" the fitness landscape using the information provided by the guiding function. This is a very powerful metaphor; it allows interpretations in terms of well-known topographical objects such as *peaks*, *valleys*, *mesas*, etc, of great utility to visualize the search progress, and to grasp factors affecting the performance of the process. In particular, the important notion of *local* optimum is associated to this definition of fitness landscape. To be precise, a local optimum is a vertex of the fitness landscape whose guiding function value is better than the values of all its neighbors. Notice that different moves define different neighborhoods and hence different fitness landscapes, even when the same problem instance is considered. For this reason, the notion of local optimum is not intrinsic to a problem instance as it is, sometimes, erroneously considered.

2.3 Local vs. Population-Based Search

The definitions presented in the previous subsection naturally lead to the notion of *local search algorithm*. A local search algorithm starts from a configuration $s_0 \in \mathcal{S}$, generated at random or constructed by some other algorithm. Subsequently, it iterates using at each step a transition based on the neighborhood of the current configuration. Transitions leading to preferable (according to the partial order $\prec_{\mathcal{F}}$) configurations are accepted, i.e., the newly generated configuration turns to be the current configuration in the next step. Otherwise, the current configuration is kept. This process is repeated until a certain termination criterion is met. Typical criteria are the realization of a pre-specified number of iterations, not having found any improvement in the last m iterations, or even more complex mechanisms based on estimating the probability of being at a local optimum [44]. Due to these characteristics, the approach is metaphorically called *"hill climbing"*. The whole process is sketched in Algorithm 1.

The selection of the particular type of moves (also known as *mutation* in the context of GAs) to use does certainly depend on the specific characteristics of the problem and the representation chosen. There is no general advice for this, since it is a matter of the available computer time for the whole process as well as other algorithmic decisions that include ease of coding, etc. In some cases some moves are conspicuous, for example it can be the change of the value of one single variable or the swap of the values of two different variables. Sometimes the "step" may also be composed of a chain of transitions. For instance, in relation with MAs, Radcliffe and Surry introduced the concept of *Binomial Minimal Mutation*, where the number of mutations to perform is selected according to a certain binomial distribution [229]. In the context of fitness landscapes, this is equivalent to a redefinition of the neighborhood relation, considering two configurations as neighbors when there exists a chain of transitions connecting them.

Algorithm 1: A Local Search Algorithm

```
Procedure Local-Search-Engine (current);
begin
    repeat
        new ← GenerateNeighbor(current);
        if F_g(new) ≺_F F_g(current) then
            current ← new;
        endif
    until TerminationCriterion() ;
    return current;
end
```

Local search algorithms are thus characterized by keeping a single configuration at a time. The immediate generalization of this behavior is the simultaneous maintenance of k, $(k \geqslant 2)$ configurations. The term *population-based* search algorithms has been coined to denote search techniques behaving this way.

The availability of several configurations at a time allows the use of new powerful mechanisms for traversing the fitness landscape in addition to the standard mutation operator. The most popular of these mechanisms, the recombination operator, will be studied in more depth in the next section. In any case, notice that the general functioning of population-based search techniques is very similar to the pseudocode depicted in Algorithm 1. As a matter of fact, a population-based algorithm can be imagined as a procedure in which we sequentially visit vertices of a hypergraph. Each vertex of the hypergraph represents a set of configurations in $\mathcal{S}_P(x)$, i.e., a population. The next vertex to be visited, i.e., the new population, can be established according to the composition of the neighborhoods of the different transition mechanisms used in the population algorithm. Despite the analogy with local search, it is widely accepted in the scientific literature to apply the denomination 'local' just to one-configuration-at-a-time search algorithms. For this reason, the term 'local' will be used with this interpretation in the remainder of the article.

2.4 Recombination

As mentioned in the previous section, local search is based on the application of a mutation operator to a single configuration. Despite the apparent simplicity of this mechanism, "mutation-based" local search has revealed itself a very powerful mechanism for obtaining good quality solutions for **NP**−hard problems. For this reason, some researchers have tried to provide a more theoretically-solid background to this class of search. In this line, it is worth mentioning the definition of the *Polynomial Local Search* class (PLS) by Johnson et al. [126]. Basically, this complexity class comprises a problem and an associated search landscape such that we can decide in polynomial time if we can find a better solution in the neighborhood. Unfortunately, it is very likely that no **NP**−hard problem is contained in class PLS, since that would imply that **NP**=co-**NP** [279], a conjecture usually assumed to be false. This fact has justified the quest for additional search mechanisms to be used as stand-alone operators or as complements to standard mutation.

In this line, recall that population-based search allowed the definition of generalized move operators termed *recombination* operators. In essence, recombination can be defined as a process in which a set S_{par} of n configurations (informally referred to as "parents") is manipulated to create a set $S_{desc} \subseteq sol_P(x)$ of m new configurations (informally termed "descendants"). The creation of these descendants involves the identification and combination of features extracted from the parents.

At this point, it is possible to consider properties of interest that can be exhibited by recombination operators [229]. The first property, *respect*, represents the exploitation side of recombination. A recombination operator is said to be *respectful*, regarding a particular type of features of the configurations, if, and only if, it generates descendants carrying all basic features common to all parents. Notice that, if all parent configurations are identical, a respectful recombination operator is forced to return the same configuration as a descendant. This property is termed *purity*, and can be achieved even when the recombination operator is not generally respectful.

On the other hand, *assortment* represents the exploratory side of recombination. A recombination operator is said to be *properly assorting* if, and only if, it can generate descendants carrying any combination of compatible features taken from the parents. The assortment is said to be *weak* if it is necessary to perform several recombinations within the offspring to achieve this effect.

Finally, *transmission* is a very important property that captures the intuitive rôle of recombination. An operator is said to be transmitting if every feature exhibited by the offspring is present in at least one of the parents. Thus, a transmitting recombination operator combines the information present in the parents but does not introduce new information. This latter task is usually left to the mutation operator. For this reason, a non-transmitting recombination operator is said to introduce *implicit mutation*.

The three properties above suffice to describe the abstract input/output behavior of a recombination operator regarding some particular features. It provides a characterization of the possible descendants that can be produced by the operator. Nevertheless, there exist other aspects of the functioning of recombination that must be studied. In particular, it is interesting to consider how the construction of $\mathcal{S}_{desc}$ is approached.

First of all, a recombination operator is said to be *blind* if it has no other input than $\mathcal{S}_{par}$, i.e., it does not use any information from the problem instance. This definition is certainly very restrictive, and hence is sometimes relaxed as to allow the recombination operator to use information regarding the problem constraints (so as to construct feasible descendants), and possibly the fitness values of configurations $y \in \mathcal{S}_{par}$ (so as to bias the generation of descendants toward the best parents). A typical example of a blind recombination operator is the classical *Uniform crossover* [253]. This operator is defined on search spaces $\mathcal{S} \equiv \Sigma^n$, i.e., strings of n symbols taken from an alphabet Σ. The construction of the descendant is done by randomly selecting at each position one of the symbols appearing in that position in any of the parents. This random selection can be totally uniform or can be biased according to the fitness values of the parents as mentioned before. Furthermore, the selection can be done so as to enforce feasibility (e.g., consider the binary representation of solutions in the 0-1 MKP). Notice that, in this case, the resulting operator is neither respectful nor transmitting in general.

The use of blind recombination operators has been usually justified on the grounds of not introducing excessive bias in the search algorithm, thus preventing extremely fast convergence to suboptimal solutions. This is questionable though. First, notice that the behavior of the algorithm is in fact biased by the choice of representation and the mechanics of the particular operators. Second, there exist widely known mechanisms (e.g., spatial isolation) to hinder these problems. Finally, it can be better to quickly obtain a suboptimal solution and restart the algorithm than using blind operators for a long time in pursuit of an asymptotically optimal behavior (not even guaranteed in most cases).

Recombination operators that use problem knowledge are commonly termed *heuristic* or *hybrid*. In these operators, problem information is utilized to guide the process of constructing the descendants. This can be done in a plethora of ways for each problem, so it is difficult to provide a taxonomy of heuristic recombination operators. Nevertheless, there exist two main 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. A heuristic recombination operator can focus in one of these aspects, or in both of them simultaneously.

As an example of a heuristic recombination operator focusing on the first aspect, Dynastically Optimal Recombination (DOR) [53] can be mentioned. This operator explores the *dynastic potential* (i.e., the set of possible children) of the configurations being recombined, so as to find the best member of this set (notice that, since configurations in the dynastic potential are entirely composed of features taken from any of the parents, this is a transmitting operator). This exploration is done using a subordinate complete algorithm, and its goal is thus to find the best combination of parental features giving rise to a feasible child. Hence, this operator is monotonic in the sense that any child generated is at least as good as the best parent.

Examples of heuristic recombination operators concentrating on the selection of non-parental features, one can cite the *patching-by-forma-completion* operators proposed by Radcliffe and Surry [228]. These operators are based on generating an incomplete child using a non-heuristic procedure (e.g., the RAR_ω operator [227]), and then completing the child either using a local hill climbing procedure restricted to non-specified features (*locally optimal forma completion*) or a global search procedure that finds the globally best solution carrying the specified features (*globally optimal forma completion*). Notice the similarity of this latter approach with DOR.

Finally, there exist some operators trying to exploit knowledge in both of the above aspects. A distinguished example is the *Edge Assembly Crossover* (EAX) [188]. EAX is a specialized operator for the TSP (both for symmetric and asymmetric instances) in which the construction of the child comprises two-phases: the first one involves the generation of an incomplete child via the so-called E-sets (subtours composed of alternating edges from each parent); subsequently, these subtours are merged into a single feasible subtours using a greedy repair algorithm. The authors of this operator reported impressive results in terms of accuracy and speed. It has some similarities with the recombination operator proposed in [178].

A final comment must be made in relation to the computational complexity of recombination. It is clear that combining the features of several solutions is in general computationally more expensive than modifying a single solution (i.e., a mutation). Furthermore, the recombination operation will be usually invoked a large number of times. For this reason, it is convenient (and in many situations mandatory) to keep it at a low computational cost. A reasonable guideline is to consider an $O(N \log N)$ upper bound for its complexity, where N is the size of the input (the set S_{par} *and* the problem instance x). Such limit is easily affordable for blind recombination operators, which are called *crossover*, a reasonable name to convey their low complexity (yet not always used in this context). However, this limit can be relatively astringent in the case of heuristic recombination, mainly when epistasis (non-additive inter-feature influence on the fitness value) is involved. This admits several solutions depending upon the particular heuristic used. For example, DOR has exponential worst case behavior, but it can be made affordable by picking larger pieces of information from each parent (the larger the size of these pieces of information, the lower the number of them needed to complete the child) [52]. In any case, consider that heuristic recombination operators provide better solutions than blind recombination operators, and hence they need not be invoked the same number of times.

2.5 A Memetic Algorithm Template

In light of the above considerations, it is possible to provide a general template for a memetic algorithm. As mentioned in Subsection 2.3, this template is

Algorithm 2: A Population-Based Search Algorithm

```
Procedure Population-Based-Search-Engine;
begin
    Initialize pop using GenerateInitialPopulation();
    repeat
        newpop ← GenerateNewPopulation(pop);
        pop ← UpdatePopulation (pop, newpop);
        if pop has converged then
            pop ← RestartPopulation(pop);
        endif
    until TerminationCriterion() ;
end
```

Algorithm 3: Injecting high-quality solutions in the initial population.

```
Procedure GenerateInitialPopulation;
begin
    Initialize pop using EmptyPopulation();
    for j ← 1 to popsize do
        i ← GenerateRandomConfiguration();
        i ← Local-Search-Engine (i);
        InsertInPopulation individual i to pop;
    endfor
    return pop;
end
```

very similar to that of a local search procedure acting on a set of $|pop| \geqslant 2$ configurations. This is shown in Algorithm 2.

This template requires some explanation. First of all, the GenerateInitialPopulation procedure is responsible for creating the initial set of $|pop|$ configurations. This can be done by simply generating $|pop|$ random configurations or by using a more sophisticated seeding mechanism (for instance, some constructive heuristic), by means of which high-quality configurations are injected in the initial population [252]. Another possibility is to use the Local-Search-Engine presented in Subsection 2.3 as shown in Algorithm 3.

As for the TerminationCriterion function, it can be defined very similarly to the case of Local Search, i.e., setting a limit on the total number of iterations, reaching a maximum number of iterations without improvement, or having performed a certain number of population restarts, etc.

The GenerateNewPopulation procedure is at the core of memetic algorithms. Essentially, this procedure can be seen as a pipelined process comprising n_{op} stages. Each of these stages consists of taking $arity^j_{in}$ configurations from the previous stage, generating $arity^j_{out}$ new configurations by applying an operator op^j. This pipeline is restricted to have $arity^1_{in} = popsize$. The whole process is sketched in Algorithm 4.

Algorithm 4: The pipelined GenerateNewPopulation procedure.

Procedure GenerateNewPopulation (*pop*);
begin
  $buffer^0 \leftarrow pop$;
  for $j \leftarrow 1$ **to** n_{op} **do**
    Initialize $buffer^j$ **using** EmptyPopulation();
  endfor
  for $j \leftarrow 1$ **to** n_{op} **do**
    $S^j_{par} \leftarrow$ ExtractFromBuffer ($buffer^{j-1}$, $arity^j_{in}$);
    $S^j_{desc} \leftarrow$ ApplyOperator (op^j, S^j_{par});
    for $z \leftarrow 1$ **to** $arity^j_{out}$ **do**
      InsertInPopulation individual $S^j_{desc}[z]$ **to** $buffer^j$;
    endfor
  endfor
  return $\text{buffer}^{n_{op}}$;
end

This template for the GenerateNewPopulation procedure is usually instantiated in GAs by letting $n_{op} = 3$, using a selection, a recombination, and a mutation operator. Traditionally, mutation is applied after recombination, i.e., on each child generated by the recombination operator. However, if a heuristic recombination operator is being used, it may be more convenient to apply mutation *before* recombination. Since the purpose of mutation is simply to introduce new features in the configuration pool, using it in advance is also possible. Furthermore, the *smart* feature combination performed by the heuristic operator would not be disturbed this way.

This situation is slightly different in MAs. In this case, it is very common to let $n_{op} = 5$, inserting a Local-Search-Engine right after applying op^2 and op^4 (respectively recombination and mutation). Due to the local optimization performed after mutation, their combined effect (i.e., mutation + local search) cannot be regarded as a simple disruption of a computationally-demanding recombination. Note also that the interplay between mutation and local search requires the former to be different than the neighborhood structure used in the latter; otherwise mutations can be readily reverted by local search, and their usefulness would be negligible.

The UpdatePopulation procedure is used to reconstruct the current population using the old population *pop* and the newly generated population *newpop*. Borrowing the terminology from the evolution strategy [230, 238] community, there exist two main possibilities to carry on this reconstruction: the *plus* strategy and the *comma* strategy. In the former, the current population is constructed taken the best *popsize* configurations from $pop \cup newpop$. As to the latter, the best *popsize* configurations are taken just from *newpop*. In this case, it is required to have $|newpop| > popsize$, so as to put some selective pressure on the process (the bigger the $|newpop|/popsize$ ratio, the stronger the pressure). Otherwise, the search would reduce to a random wandering through $\mathcal{S}$.

There are a number of studies regarding appropriate choices for the UpdatePopulation procedure (see e.g., [9]). As a general guideline, the comma strategy is usually regarded as less prone to stagnation, with the ratio $|newpop|/popsize \simeq 6$ being a common choice [10]. Nevertheless, this option can be somewhat computationally expensive if the guiding function is complex and time-consuming. Another common alternative is using a plus strategy with a low value of $|newpop|$, analogous to the so-called *steady-state* replacement strategy in GAs [274]. This option usually provides a faster convergence to high-quality solutions. However, care has to be taken with premature convergence to suboptimal regions of the search space, i.e., all configurations in the population being very similar to each other, hence hindering the exploration of other regions of $\mathcal{S}$.

The above consideration about premature convergence leads to the last component of the template shown in Algorithm 2, the restarting procedure. First of all, it must be decided whether the population has degraded or has not. To do so, it is possible to use some measure of information diversity in the population such as Shannon's entropy [60]. If this measure falls below a predefined threshold, the population is considered to be in a degenerate state. This threshold depends upon the representation (number of values per variable, constraints, etc.) and hence must be determined in an *ad-hoc* fashion. A different possibility is using a probabilistic approach to determine with a desired confidence that the population has converged. For example, in [119] a Bayesian approach is presented for this purpose.

Once the population is considered to be at a degenerate state, the restart procedure is invoked. Again, this can be implemented in a number of ways. A very typical strategy is to keep a fraction of the current population (this fraction can be as small as one solution, the current best), and substituting the remaining configurations with newly generated (from scratch) solutions, as shown in Algorithm 5.

The procedure shown in Algorithm 5 is also known as the *random-immigrant* strategy [33]. Another possibility is to activate a *strong* or *heavy* mutation operator in order to drive the population away from its current location in the search space. Both options have their advantages and disadvantages. For example, when using the random-immigrant strategy, one has to take some caution to prevent the preserved configurations to take over the population (this can be achieved by putting a low selective pressure, at least in the first iterations after a restart). As to the heavy mutation strategy, one has to achieve a tradeoff between an excessively strong mutation that would destroy any information contained in the current population, and a not so strong mutation that would cause the population to converge again in a few iterations.

Algorithm 5: The RestartPopulation procedure.

```
Procedure RestartPopulation (pop);
begin
    Initialize newpop using EmptyPopulation();
    #preserved ← popsize · %preserve;
    for j ← 1 to #preserved do
        i ← ExtractBestFromPopulation(pop);
        InsertInPopulation individual i to newpop;
    endfor
    for j ← #preserved + 1 to popsize do
        i ← GenerateRandomConfiguration();
        i ← Local-Search-Engine (i);
        InsertInPopulation individual i to newpop;
    endfor
    return newpop;
end
```

2.6 Designing an Effective Memetic Algorithm

The general template of MAs depicted in the previous section must be instantiated with precise components in order to be used for solving a specific problem. This instantiation has to be done carefully so as to obtain an effective optimization tool. We will address some design issues in this section.

A first obvious remark is that there exist no general approach for the design of effective MAs. This observation is based on different proofs depending on the precise definition of *effective* in the previous statement. Such proofs may involve classical complexity results and conjectures if 'effective' is understood as 'polynomial-time', the NFL Theorem if we consider a more general set of performance measures, and even Computability Theory if we relax the definition to arbitrary decision problems. For these reasons, we can only define several *design heuristics* that will likely result in good-performing MAs, but without explicit guarantees for this.

This said, MAs are commonly implemented as evolutionary algorithms endowed with a local search component (recall previous section), and as such can benefit from the theoretical corpus available for EAs. This is particularly applicable to some basic aspects such as the representation of solutions in terms of meaningful information units [59, 228]. Focusing now on more specific aspects of MAs, the first consideration that must be clearly taken into account is the interplay among the local search component and the remaining operators, mostly with respect to the characteristics of the search landscape. A good example of this issue can be found in the work of Merz and Freisleben on the TSP [85]. They consider the use of a highly intensive local search procedure –the Lin-Kernighan heuristic [157]– and note that the average distance between local optima is similar to the average distance between a local optimum and the global optimum. For this reason, they introduce a distance-preserving crossover (DPX) operator that generate offspring whose distance from the parents is the same as the distance between the parents themselves. Such an operator is likely to be less effective if a not-so-powerful local improvement method, e.g., 2-opt, was used, inducing a different distribution of local optima.

In addition to the particular choice (or choices) of local search operator, there remains the issue of determining an adequate parameterization for the procedure, namely, how much effort must be spent on each local search, how often the local search must be applied, and –were it not applied to every new solution generated– how to select the solutions that will undergo local improvement. Regarding the first two items, there exists theoretical evidence [143, 251] that an inadequate parameter setting can turn the algorithmic solution from easily solvable to non-polynomially solvable. Besides, there are obvious practical limitations in situations where the local search and/or the fitness function is computationally expensive. This fact admits different solutions. On the one hand, the use of surrogates (i.e., fast approximate models of the true function) to accelerate evolution is an increasingly popular option in such highly demanding problems [104, 155, 272, 273, 283]. On the other hand, partial lamarckism [42, 112, 212], where not every individual is subject to local search, is commonly used as well. The precise value for the local search application probability (or multiple values when more than one local search procedure is available) largely depends on the problem under consideration [123], and its determination is in many cases an art. For this reason, adaptive and self-adaptive mechanisms have been defined in order to let the algorithm learn what the most appropriate setting is (see Section 3.2).

As to the selection of individuals that will undergo local search, most common options are random-selection, and fitness-based selection, where only the best individuals are subject to local improvement. Nguyen *et al.* [197] also consider a 'stratified' approach, in which the population is sorted and divided into n levels (n being the number of local search applications), and one individual per level is randomly selected. Their experimentation on some continuous functions indicates that this strategy and improve-the-best (i.e., applying local search to the best n individuals) provide better results than random selection. Such strategies can be readily deployed on a structured MA as defined by Moscato *et al.* [15, 21, 83, 172, 169], where good solutions flow upwards within a tree-structured population, and layers are explicitly available. Other population management strategies are nevertheless possible, see [19, 218, 219, 249].

3 Algorithmic Extensions of Memetic Algorithms

The algorithmic template and design guidelines described in the previous section can characterize most basic incarnations of MAs, namely population-based algorithms endowed with static local search for single-objective optimization. However, more sophisticated approaches can be conceived, and certainly required, in certain applications. This section is aimed at providing an overview of more advanced algorithmic extensions used in the MA realm.

3.1 Multiobjective Memetic Algorithms

Multiobjective problems are frequent in real-world applications. Rather than having a single objective to be optimized, the solver is faced with multiple, partially conflicting objectives. As a result, there is no *a priori* single optimal solution ,but rather a collection of optimal solutions, providing different trade-offs among the objectives considered. In this scenario, the notion of Pareto-dominance is essential: given two solutions $s, s' \in sol_P(x)$, s is said to dominate s' if it is better than s' in at least one of the objectives, and it is no worse in the remaining ones. This clearly induces a partial order $\prec_P$, since given two solutions it may be the case that none of them dominates the other. This collection of optimal solutions is termed the optimal Pareto front, or the optimal non-dominated front.

Population-based search techniques, in particular evolutionary algorithms (EAs), are naturally fit to deal with multiobjective problems, due to the availability of a population of solutions which can approach the optimal Pareto front from different directions. There is extensive literature on the deployment of EAs in multiobjective settings, and the reader is referred to [36, 35, 63, 287], among others, for more information on this topic. MAs can obviously benefit from this corpus of knowledge. However, MAs typically incorporate a local search mechanism, and it has to be adapted to the multiobjective setting as well. This can be done in different ways [132], which can be roughly classified into two major classes: scalarizing approaches, and Pareto-based approaches. The scalarizing approaches are based on the use of some aggregation mechanism to combine the multiple objectives into a single scalar value. This is usually done using a linear combination of the objective values, with weights that are either fixed (at random or otherwise) for the whole execution of the local search procedure [266], or adapted as the local search progresses [106]. As to Pareto-based approaches, they consider the notion of Pareto-dominance for deciding transitions among neighboring solutions, typically coupled with the use of some measure of crowding to spread the search, e.g, [133].

A full-fledged multiobjective MA (MOMA) is obtained by appropriately combining population-based and local search-based components for multiobjective optimization. Again, the strategy used in the local search mechanism can be used to classify most MOMAs. Thus, two proposals due to Ishibuchi and Murata [121, 122] and to Jaszkiewicz [124, 125] are based on the use of random scalarization each time a local search is to be used. Alternatively, a single-objective local search could be used to optimize individual objectives [120]. Ad hoc mating strategies based on the particular weights chosen at each local search invocation (whereby the solutions to be recombined are picked according to these weights) are used as well. A related approach –including the on-line adjustment of scalarizing weights– is followed by Guo *et al.* [105, 106, 107]. On the other hand, a MA based on PAES (Pareto Archived Evolution Strategy) was defined by Knowles and Corne [134, 135]. More recently, a MOMA based on particle swarm optimization (PSO) has been defined by Liu *et al.* [152, 162]. In this algorithm, an archive of non-dominated solutions is maintained and randomly sampled to obtain reference points for particles. A different approach is used by Schuetze *et al.* [237] for numerical-optimization problems. The continuous nature of solution variables allows using their values for computing search directions. This fact is exploited in their local search procedure (HCS for Hill Climber with Sidestep) for directing the search toward specific regions (e.g., along the Pareto front) when required.

3.2 Adaptive Memetic Algorithms

When some design guidelines were given in Section 2.6, the fact that these were heuristics that ultimately relied on the problem-knowledge available was stressed. This is not a particular feature of MAs, but affects the field of metaheuristics as a whole. Indeed, one of the keystones in practical metaheuristic problem-solving is the necessity of customizing the solver for the problem at hand [51]. Therefore, it is not surprising that attempts to transfer a part of this tuning effort to the metaheuristic technique itself have been common. Such attempts can take place at different levels, or can affect different components of the algorithm. The first –and more intuitive one– is the parametric level involving the numerical values of parameters, such as the operator application rates. Examples of this can be found in early EAs, see for example [61]. A good up-to-date overview of these approaches (actually broader in scope, covering more advanced topics than parameter adaptation) can be found in [247]. Focusing specifically on MAs, this kind of adaptation has been applied in [11, 164, 175, 176].

A slightly more general approach –termed 'meta-lamarckian learning' [204] by Ong and Keane– takes place at the algorithmic level. They consider a setting in which the MA has a collection of local search operators available, and how the selection of the particular operator(s) to be applied to a specific solution can be done on the basis of past performance of the operator, or on the basis of the similarity of the solution to previous successful cases of operator application. Some analogies can also be drawn here with hyper-heuristics [54], a high-level heuristic that controls the application of a set of low-level heuristics to solutions, using strategies ranging from pure random to performance-based rules. See [28] for a recent comprehensive overview of hyperheuristics.

In general terms, the approaches mentioned before are based on static, hard-wired mechanisms that the MA uses to react to the environment. Hence, they can be regarded as adaptive, but not as self-adaptive [205]. In the latter case, the actual definition of the search mechanisms can evolve during the search. This is a goal that has been pursued for long in MAs. Back in the early days of the field, it was already envisioned that future generations of MAs would work in at least two levels and two time scales [179]. During the short-time scale, a set of agents would be searching in the search space associated to the problem. The long-time scale would *adapt the algorithms* associated with the agents. Here we encompass individual search strategies, recombination operators, etc. A simple example of this kind of self-adaptation can be found in the so-called multi-memetic algorithms, in which each solution carries a gene that indicates which local search has to be applied on it. This can be a simple pointer to an existing local search operator, or even the parametrization of a general local search template, with items such as the neighborhood to use, acceptance criterion, etc. [141]. Going beyond, a grammar can be defined to specify a more complex local search operator [140, 142]. At an even higher level, this evolution of local search operators can be made fully symbiotic, rather than merely endosymbiotic. For this purpose, two co-evolving populations can be considered: a population of solutions, and a population of local search operators. These two populations co-operate by means of an appropriate pairing mechanism, that associates solutions with operators. The latter receive fitness in response on their ability to improve solutions, thus providing a fully self-adaptive strategy for exploring the search landscape [244, 245, 246].

3.3 Complete Memetic Algorithms

The combination of exact techniques with metaheuristics is an increasingly popular approach. Focusing on local search techniques, Dumitrescu and Stützle [73] have provided a classification of methods in which exact algorithms are used to strengthen local search, i.e., to explore large neighborhoods, to solve exactly some subproblems, to provide bounds and problem relaxations to guide the search, etc. Some of these combinations can be also found in the literature on population-based methods. For example, exact techniques –such as branch-and-bound (BnB) [53] or dynamic programming [90] among others– have been used to perform recombination (recall Section 2.4), and approaches in which exact techniques solved some subproblems provided by EAs date back to 1995 [45]. See also [76] for a large list of references regarding local search/exact hybrids.

Puchinger and Raidl [220] have provided a classification of this kind of hybrid techniques in which algorithmic combinations are either collaborative (sequential or intertwined execution of the combined algorithms) or integrative (one technique works inside the other one, as a subordinate). Some of the exact/metaheuristic hybrid approaches defined before are clearly integrative –i.e., using an exact technique to explore neighborhoods. Further examples are the use of BnB in the decoding process [221] of a genetic algorithm (i.e., exact method within a metaheuristic technique), or the use of evolutionary techniques for the strategic guidance of BnB [139] (metaheuristic approach within an exact method).

As to collaborative combinations, a sequential approach in which the execution of a MA is followed by a branch-and-cut method can be found in [131]. Intertwined approaches are also popular. For example, Denzinger and Offerman [66] combine genetic algorithms and BnB within a parallel multi-agent system. These two algorithms also cooperate in [45, 88], the exact technique providing partial promising solutions, and the metaheuristic returning improved bound. A related approach involving beam search and full-fledged MAs can be found in [89, 92, 93].

It must be noted that most hybrid algorithms defined so far that involve exact techniques and metaheuristics are not complete, in the sense that they do not guarantee an optimal solution (an exception is the proposal of French *et al.* [86], combining an integer-programming BnB approach with GAs for MAX-SAT). Thus, the term 'complete MA' may be not fully appropriate. Nevertheless, many of these hybrids can be readily adapted for completeness purposes, although obviously time and/or space requirements will grow faster-than-polynomial in general.

4 Applications of Memetic Algorithms

This section will provide an overview of the numerous applications of MAs. This overview is far from exhaustive since new applications are being developed continuously. However, it is intended to illustrate the practical impact of these optimization techniques. We have focused on recent applications, namely in the last five years (that is, from 2004 onwards). Readers interested in earlier applications (which are also manifold) can refer to [109, 180, 181, 182]. We have organized references in five major areas: machine learning and knowledge discovery (Table 1), traditional combinatorial optimization (Table 2), planning, scheduling and timetabling (Table 3), bioinformatics (Table 4), and electronics, engineering, and telecommunications (Table 4). As mentioned before, we have tried to be illustrative rather than exhaustive, pointing out some selected references from these well-known application areas.

Table 1 Applications in machine learning and knowledge discovery

Data Mining and Knowledge Discovery	Image analysis	[37, 67, 68, 77, 211]
	Fuzzy clustering	[70]
	Feature selection	[243, 286]
	Pattern recognition	[94]
Machine Learning	Decision trees	[144]
	Inductive learning	[69]
	Neural networks	[64, 65, 103, 110, 159, 168, 195, 262]

Table 2 Applications in combinatorial optimization

Binary & Set Problems	Binary quadratic programming	[173]
	Knapsack problem	[87, 88, 105, 107, 222]
	Low autocorrelation sequences	[91]
	Max-SAT	[18, 223]
	Set covering	[125]
Graph-based Problems	Crossdock optimization	[2, 154]
	Graph coloring	[38]
	Graph matching	[12]
	Hamiltonian cycle	[32]
	Maximum cut	[270]
	Quadratic assignment	[72, 255]
	Routing problems	[19, 20, 56, 57, 74]
		[80, 145, 146, 147]
		[218, 259, 263]
	Spanning tree	[79, 231]
	Steiner tree	[131]
	TSP	[21, 161, 163, 196, 271]
Constrained Optimization	Golomb ruler	[46, 48]
	Social golfer	[47]
	Maximum density still life	[89, 90]

Table 3 Applications in planning, scheduling, timetabling, and manufacturing. Check also [49].

Manufacturing	Assembly line	[226, 257, 265]
	Flexible manufacturing	[5, 31, 187, 258]
	Lot sizing	[16]
	Multi-tool milling	[13]
	Supply chain network	[280]
Planning	Temporal planning	[235]
Scheduling	Flowshop scheduling	[82, 84, 152, 158, 160, 184, 209, 240, 241]
	Job-shop	[27, 96, 97, 98, 224, 267, 268, 278]
	Parallel machine scheduling	[184, 277]
	Project scheduling	[29]
	Single machine scheduling	[166, 184]
Timetabling	Driver scheduling	[153]
	Examination timetabling	[216]
	Rostering	[3, 22, 206]
	Sport league	[236]
	Train timetabling	[239]
	University course	[151, 215, 233]

Table 4 Applications in bioinformatics

PHYLOGENY	Phylogenetic inference	[43, 93, 275]
	Consensus tree	[217]
MICROARRAYS	Biclustering	[208]
	Feature Selection	[55, 284, 285]
	Gene ordering	[169, 183]
SEQUENCE ANALYSIS	Shortest common supersequence	[42, 92]
	DNA sequencing	[71]
PROTEIN SCIENCE	Sequence assignment	[269]
	Structure comparison	[140]
	Structure prediction	[14, 40, 203, 234, 281]
SYSTEMS BIOLOGY	Gene regulatory networks	[200, 250]
	Cell models	[232]
BIOMEDICINE	Drug therapy design	[194, 264]

Table 5 Applications in electronics, telecommunications and engineering

ELECTRONICS	Analog circuit design	[58, 170]
	Circuit partitioning	[34]
	Electromagnetism	[23, 104, 210]
	Filter design	[254]
	VLSI design	[7, 171, 256]
ENGINEERING	Chemical kinetics	[136, 137]
	Crystallography	[212]
	Drive design	[24, 25]
	Power systems	[26]
	Structural optimization	[129]
	System modelling	[1, 260]
COMPUTER SCIENCE	Code optimization	[207]
	Information forensics	[242]
	Information theory	[41]
	Software engineering	[6]
TELECOMMUNICATIONS	Antenna design	[114, 115, 116, 117]
	Mobile networks	[128, 225]
	P2P networks	[174, 191, 192]
	Wavelength Assignment	[78]
	Wireless networks	[113, 118, 130, 138]

Although these fields encompass the vast majority of applications of MAs, it must be noted that success stories are not restricted to these major fields. To cite an example, there are several applications of MAs in economics, e.g., in portfolio optimization [165], risk analysis [167], and labour-market delineation [81]. For further information about MA applications we suggest querying bibliographical databases or web browsers for the keywords *'memetic algorithms'* and *'hybrid genetic algorithms'*.

5 Challenges and Future Directions

The future seems promising for MAs. This is the combination of several factors. First, MAs (less frequently disguised under different names) are showing a remarkable record of efficient implementations, providing very good results in practical problems. Second, there are reasons to believe that some new attempts to do theoretical analysis can be conducted. This includes the worst-case and average-case computational complexity of recombination procedures. Third, the ubiquitous nature of distributed systems, like networks of workstations for example, plus the inherent asynchronous parallelism of MAs and the existence of web-conscious languages like Java, all together are an excellent combination to develop highly portable and extendable object-oriented frameworks allowing algorithmic reuse. These frameworks might allow the users to solve subproblems using commercial codes or well-tested software from other users who might be specialists in another area. Fourth, an important and pioneering group of MAs, that of *Scatter Search* [95, 148], is challenging the role of randomization in recombination. We expect that, as a healthy reaction, we will soon see new types of powerful MAs that blend in a more appropriate way both exhaustive (either truncated or not) and systematic search methods.

5.1 Learning from Experience

In 1998, Applegate, Bixby, Cook, and Chvatal established new breakthrough results for the Min TSP. They solved to optimality an instance of the TSP of 13,509 cities corresponding to all U.S. cities with populations of more than 500 people. The approach, according to Bixby: *"...involves ideas from polyhedral combinatorics and combinatorial optimization, integer and linear programming, computer science data structures and algorithms, parallel computing, software engineering, numerical analysis, graph theory, and more"*. The solution of this instance demanded the use of three Digital AlphaServer 4100s (with a total of 12 processors) and a cluster of 32 Pentium-II PCs. The complete calculation took approximately three months of computer time. The code has certainly more than 1,000 pages and is based on state-of-the-art techniques from a wide variety of scientific fields.

The philosophy is the same in the case of MAs, that of a synergy of different approaches. Actually, their approach can possibly be classified as the most complex MA ever built for a given combinatorial optimization problem. One of the current challenges is to develop simpler algorithms that achieve these impressive results. The approach of running a local search algorithm (Chained Lin Kernighan) to produce a collection of tours, followed by the dynastically optimal recombination method called *tour merging* , produced a non-optimal tour only 0.0002 % above the proved optimal tour for the

13,509 cities instance. We take this as a clear proof of the benefits of the MA approach and that more work is needed in developing good strategies for *complete memetic algorithms*, i.e. those that systematically and synergistically use randomized and deterministic methods and can prove optimality.

An open line for the design of this kind of algorithms may be the exploitation of FPT (fixed-parameter tractability) results, see Subsection 5.2. Related to this, it must be noted that we still lack a formal framework for recombination, similar for instance to the one for Local Search [126, 279]. In this sense, an interesting new direction for theoretical research arose after the introduction of two computational complexity classes, the PMA class (for Polynomial Merger Algorithms problems) and its unconstrained analogue, the uPMA class (see [180]). These classes are defined analogously to the class of Polynomial Local Search (PLS). Conducting research to identify problems, and their associated recombination procedures, such that membership, in either PMA or uPMA, can be proved is a definitely important task. It is also hoped that after some initial attempts on challenging problems, completeness and reductions for these classes can be properly defined [50].

5.2 Exploiting FPT results

An interesting new avenue of research can be established by appropriately linking results from the theory of fixed-parameter tractability (FPT) and the development of recombination algorithms. A parameterized problem can be generally viewed as a problem with two input components, i.e. a pair $\langle x, k\rangle$. The former is generally an instance (i.e. $x \in I_P$) of some other decision problem P and the latter is some numerical aspect of the former (generally a positive integer assumed $k \ll |x|$, where $|x|$ is the size of instance x) that constitutes a *parameter*, for example, the maximum node degree in a certain graph-based problem, the maximum number of elements in the solution of a subset-selection problem, etc. If there exists an algorithm solving the problem in time $O(f(k)|x|^{\alpha})$, where $f(k)$ is an *arbitrary* function depending on k only, and α a constant independent of k or n, the parameterized problem is said to be *fixed-parameter tractable* and the decision problem belongs to the computational complexity class FPT. Note that by following this parameterized approach, the complexity analysis becomes multidimensional, in contrast to the classical one-dimensional approach, in which only the instance size is considered (thus failing to distinguish structural properties that may make a particular problem instance hard or easy).

To illustrate this topic, consider one of the most emblematic FPT problems, namely VERTEX COVER: given a graph $G(V, E)$, find a subset $S \subseteq V$ of k vertices, such that for every edge $(u, v) \in E$, at least u or v is a member of S. Here, the number k of vertices in S is taken as a parameter and factored out from the problem input. In general, efficient FPT algorithms are based on the techniques of *reduction to a problem kernel* and *bounded search trees*. To understand the techniques, the reader may check a method by Chen *et al.* [30]. This method can solve the parameterized version of vertex cover in time $O(1.271^k k^2 + kn)$. Furthermore, using this method together with the speed-up method proposed by Neidermeier and Rossmanith [199], the problem can be solved in $O(1.271^k + n)$, i.e. linear in n for fixed k. The relevance of this result is more evident by noting that VERTEX COVER is an **NP**-hard problem. Thus, FPT results provide an efficient way for provably solving **NP**-hard problems for fixed parameter values.

The combination of FPT results and recombination operators is an avenue that goes both ways. In one direction, efficient, (i.e. polynomial-time), fixed-parameter algorithms can be used as *"out of the box"* tools to create efficient recombination procedures, i.e., recall some of the procedures mentioned in Subsection 3.3. Conversely, since MAs are typically designed to deal with large instances and scale pretty well with problem size, using both techniques together can produce *complete MAs*, thus extending the benefits of fixed-parameter tractability. From a software engineering perspective, the combination is perfect both from code and algorithmic reuse.

5.3 Belief Search in Memetic Algorithms

As a logical consequence of the possible directions that MAs can take, it is reasonable to affirm that more complex schemes evolving solutions, agents, as well as representations, will soon be implemented. Some theoretical computer science researchers dismiss heuristics and metaheuristics since they are not scholarly structured as a formal paradigm. However, their achievements are well-recognized. From [150]:

> *"Explaining and predicting the impressive empirical success of some of these algorithms is one of the most challenging frontiers of the theory of computation today."*

This comment is even more relevant for MAs since they generally present even better results than single-agent methods. Though metaheuristics are extremely powerful in practice, we agree that one problem with the current trend in applied research is that it allows the introduction of increasingly more complex heuristics, unfortunately most of the time parameterized by *ad-hoc* values. Moreover, some metaheuristics, like some *ant-systems* implementations, can basically be viewed as particular types of MAs. This is the case if you allow the "ants" to use *branch-and-bound* or *local search* methods. In addition, these methods for distributed recombination of information (or beliefs) have some points in common with *blackboard systems* [75], as it has been recognized in the past, yet it is hardly being mentioned in the current metaheuristics literature [180].

To illustrate how Belief Search can work in an MA setting, consider for example $\mathbf{PL}_n^{\otimes}$, a multi-agent epistemic logic introduced by Boldrin and Saffiotti [17]. According to this formalism, the opinions shared by a set of n agents can be *recombined* in a distributed belief. Using it, we can deduce the distributed belief about properties of solutions, and this can be stronger than any individual belief about it (see [50] for detailed examples with numerical values).

One interesting application of these new MAs is due to Lamma *et al.* [149] for diagnosing digital circuits. In their approach, they differentiate between genes and *"memes"*. The latter group codes for the agent beliefs and assumptions. Using a logic-based technique, they modify the memes according on how the present beliefs are contradicted by integrity constraints that express observations and laws. Each agent keeps a population of chromosomes and finds a solution to the belief revision problem by means of a genetic algorithm. A *Lamarckian* operator is used to modify a chromosome using belief revision directed mutations, oriented by tracing logical derivations. As a consequence, a chromosome will satisfy a larger number of constraints. The evolution provided by the Darwinian operators, allow agents to improve the chromosomes by gaining on the experience of other agents. Central to this approach is the Lamarckian operator appropriately called *Learn*. It takes a chromosome and produces a revised chromosome as output. To achieve that, it eliminates some derivation paths that lead to contradictions.

Surprisingly enough (and here we remark the first possibility of using the theory of *fixed-parameter tractability*), the learning is achieved by finding a *hitting set* which is not necessarily minimal. The authors make this point clear by saying that: *"a hitting set generated from these support sets is not necessarily a contradiction removal set and therefore is not a solution to the belief revision problem."* The authors might not be aware of the $O(2.311^k+n)$ exact algorithm for MIN 3-HITTING SET [198]. They might be able to use it, but that is anecdotal at the moment. What is important is that algorithms like this one might be used *out-of-the-box* if a proper, world-wide based, algorithmic framework was created.

On the other hand, we noted how results of logic programming and belief revision might help improving the current status of metaheuristics. The current situation where everybody comes with new names for the same basic techniques, and where most contributions are just the addition of new parameters to guide the search, is a futile research direction. It is possible that belief-search-guided MAs will prove to be a valid tool to help systematize the construction of these guided metaheuristics. In particular, if the discussion is based on which multi-agent logic performs better, rather than which parameters work better for specific problems or instances. To this end, we hope to convince researchers in logic programming to address these issues and to face the difficult task of guiding MAs for large-scale combinatorial optimization.

6 Conclusions

We believe that the future looks good for MAs. This belief is based on the following. First of all, MAs are showing a great record of efficient implementations, providing very good results in practical problems, as the reader may have noted in Section 4. We also have reasons to believe that we are close to some major leaps forward in our theoretical understanding of these techniques, including for example the worst-case and average-case computational complexity of recombination procedures. On the other hand, the ubiquitous nature of distributed systems is likely to boost the deployment of MAs on large-scale, computationally demanding optimization problems.

We also see as a healthy sign the systematic development of other particular optimization strategies. If any of the simpler metaheuristics (SA, TS, VNS, GRASP, etc.) performs the same as a more complex method (GAs, MAs, Ant Colonies, etc.), an "elegance design" principle should prevail and we must either resort to the simpler method, or to the one that has less free parameters, or to the one that is easier to implement. Such a fact should defy us to adapt the complex methodology to beat a simpler heuristic, or to check if that is possible at all. An unhealthy sign of current research, however, are the attempts to encapsulate metaheuristics on stretched confinements. Fortunately, such attempts are becoming increasingly less frequent. Indeed, combinations of MAs with other metaheuristics such as differential evolution [193, 201, 261], particle swarm optimization [152, 158, 159, 161, 162, 209, 214, 248, 282], or ant-colony optimization [156] are not unusual nowadays. As stated before, the future looks promising for MAs.

Acknowledgements

This chapter is an updated second edition of [180], refurbished with new references and the inclusion of sections on timely topics which were not fully addressed in the first edition. Carlos Cotta acknowledges the support of Spanish Ministry of Science and Innovation, under project TIN2008-05941.

References

1. R. Ahmad, H. Jamaluddin, and M.A. Hussain. Application of memetic algorithm in modelling discrete-time multivariable dynamics systems. *Mechanical Systems and Signal Processing*, 22(7):1595–1609, 2008.
2. U. Aickelin and A. Adewunmi. Simulation optimization of the crossdock door assignment problem. In *UK Operational Research Society Simulation Workshop 2006 (SW 2006)*, Leamington Spa, UK, March 11 2006.

3. U. Aickelin and P. White. Building better nurse scheduling algorithms. *Annals of Operations Research*, 128:159–177, 2004.
4. D. Aldous and U. Vazirani. "Go with the winners" algorithms. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 492–501, Los Alamitos, CA, 1994. IEEE Press.
5. J.E. Amaya, C. Cotta, and Fernández A.J. A memetic algorithm for the tool switching problem. In M.J. Blesa et al., editors, *Hybrid Metaheuristics 2008*, volume 5296 of *Lecture Notes in Computer Science*, pages 190–202, Berlin Heidelberg, 2008. Springer-Verlag.
6. A. Arcuri and X. Yao. A memetic algorithm for test data generation of object-oriented software. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2048–2055, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
7. S. Areibi and Z. Yang. Effective Memetic Algorithms for VLSI design = genetic algorithms plus local search plus multi-level clustering. *Evolutionary Computation*, 12(3):327–353, 2004.
8. R. Axelrod and W.D. Hamilton. The evolution of cooperation. *Science*, 211(4489):1390–1396, 1981.
9. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
10. T. Bäck and F. Hoffmeister. Adaptive search by evolutionary algorithms. In W. Ebeling, M. Peschel, and W. Weidlich, editors, *Models of Self-organization in Complex Systems*, number 64 in Mathematical Research, pages 17–21. Akademie-Verlag, 1991.
11. N.K. Bambha, S.S. Bhattacharyya, J. Teich, and E. Zitzler. Systematic integration of parameterized local search into evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):137–155, 2004.
12. T. Bärecke and M. Detyniecki. Memetic algorithms for inexact graph matching. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 4238–4245, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
13. N. Baskar, P. Asokan, R. Saravanan, and G. Prabhaharan. Selection of optimal machining parameters for multi-tool milling operations using a memetic algorithm. *Journal of Materials Processing Technology*, 174(1-3):239–249, 2006.
14. A. Bazzoli and A.G.B. Tettamanzi. A memetic algorithm for protein structure prediction in a 3D-Lattice HP model. In G.R. Raidl et al., editors, *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 1–10, Berlin, 2004. Springer Verlag.
15. R. Berretta, C. Cotta, and P. Moscato. Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem. In M. Resende and J. Pinho de Sousa, editors, *Metaheuristics: Computer-Decision Making*, pages 65–90. Kluwer Academic Publishers, Boston MA, 2003.
16. R. Berretta and L.F. Rodrigues. A memetic algorithm for a multistage capacitated lot-sizing problem. *International Journal of Production Economics*, 87(1):67–81, 2004.
17. L. Boldrin and A. Saffiotti. A modal logic for merging partial belief of multiple reasoners. *Journal of Logic and Computation*, 9(1):81–103, 1999.
18. M. Borschbach and A. Exeler. A tabu history driven crossover operator design for memetic algorithm applied to max-2SAT-problems. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 605–606, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
19. M. Boudia, C. Prins, and M. Reghioui. An effective memetic algorithm with population management for the split delivery vehicle routing problem. In T. Bartz-Beielstein et al., editors, *Hybrid Metaheuristics 2007*, volume 4771 of *Lecture Notes in Computer Science*, pages 16–30. Springer-Verlag, 2007.

20. H. Bouly, D.-C. Dang, and A. Moukrim. A memetic algorithm for the team orienteering problem. In M. Giacobini et al., editors, *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 649–658. Springer-Verlag, 2008.
21. L. Buriol, P.M. França, and P. Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, 2004.
22. E. K. Burke, P. De Causmaecker, and G. van den Berghe. Novel metaheuristic approaches to nurse rostering problems in belgian hospitals. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 44, pages 44.1–44.18. Chapman Hall/CRC Press, 2004.
23. S. Caorsi, A. Massa, M. Pastorino, and A. Randazzo. Detection of PEC elliptic cylinders by a memetic algorithm using real data. *Microwave and Optical Technology Letters*, 43(4):271–273, 2004.
24. A. Caponio, G. Leonardo Cascella, F. Neri, N. Salvatore, and M. Sumner. A fast adaptive memetic algorithm for online and offline control design of pmsm drives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):28–41, 2007.
25. A. Caponio, F. Neri, G.L. Cascella, and N. Salvatore. Application of memetic differential evolution frameworks to PMSM drive design. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 2113–2120, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
26. E.G. Carrano, B.B. Souza, and O.M. Neto. An immune inspired memetic algorithm for power distribution system design under load evolution uncertainties. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 3251–3257, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
27. A. Caumond, P. Lacomme, and N. Tchernev. A memetic algorithm for the job-shop with time-lags. *Computers & OR*, 35(7):2331–2356, 2008.
28. K. Chakhlevitch and P. Cowling. Hyperheuristics: Recent developments. In C. Cotta, M. Sevaux, and K. Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 3–29. Springer-Verlag, Berlin Heidelberg, 2008.
29. A.H.L. Chen and C.-C. Chyu. A memetic algorithm for maximizing net present value in resource-constrained project scheduling problem. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 2401–2408, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
30. J. Chen, I.A. Kanj, and W. Jia. Vertex cover: further observations and further improvements. In *Proc. 25th Int. Worksh. Graph-Theoretic Concepts in Computer Science*, number 1665 in Lecture Notes in Computer Science, pages 313–324. Springer-Verlag, 1999.
31. J.-H. Chen and J.-H. Chen. Multi-objective memetic approach for flexible process sequencing problems. In M. Ebner et al., editors, *GECCO-2008 Late-Breaking Papers*, pages 2123–2128, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
32. X. S. Chen, M. H. Lim, and D. C. Wunsch II. A memetic algorithm configured via a problem solving environment for the hamiltonian cycle problems. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2766–2773, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
33. H.G. Cobb and J.J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 529–530, San Mateo, CA, 1993. Morgan Kaufmann.
34. S. Coe, S. Areibi, and M. Moussa. A hardware memetic accelerator for VLSI circuit partitioning. *Computers & Electrical Engineering*, 33(4):233–248, 2007.
35. C.A. Coello Coello and G.B. Lamont. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, New York, 2004.

36. C.A. Coello Coello, D.A. Van Veldhuizen, and G.B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5 of *Genetic Algorithms and Evolutionary Computation.* Kluwer Academic Publishers, 2002.
37. O. Cordón, S. Damas, and J. Santamaria. A scatter search algorithm for the 3D image registration problem. In X. Yao et al., editors, *Parallel Problem Solving From Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 471–480, Berlin, 2004. Springer-Verlag.
38. D. Cosmin, J.-K. Hao, P. Kuntz. Diversity control and multi-parent recombination for evolutionary graph coloring. In C. Cotta and P. Cowling, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 5482 of *Lecture Notes in Computer Science*, pages 121–132, Tübingen, 2009. Springer-Verlag.
39. C. Cotta. A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications*, 11(3-4):223–224, 1998.
40. C. Cotta. Hybrid evolutionary algorithms for protein structure prediction in the HPNX model. In B. Reusch, editor, *Computational intelligence, Theory and Applications*, Advances in Soft Computing, pages 525–534, Berlin Heidelberg, 2004. Springer-Verlag.
41. C. Cotta. Scatter search and memetic approaches to the error correcting code problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pages 51–60, Berlin, 2004. Springer Verlag.
42. C. Cotta. Memetic algorithms with partial lamarckism for the shortest common supersequence problem. In J. Mira and J.R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 84–91, Berlin Heidelberg, 2005. Springer-Verlag.
43. C. Cotta. Scatter search with path relinking for phylogenetic inference. *European Journal of Operational Research*, 169(2):520–532, 2005.
44. C. Cotta, E. Alba, and J.M. Troya. Stochastic reverse hillclimbing and iterated local search. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1558–1565, Washington D.C., 1999. IEEE.
45. C. Cotta, J.F. Aldana, A.J. Nebro, and J.M. Troya. Hybridizing genetic algorithms with branch and bound techniques for the resolution of the TSP. In D.W. Pearson, N.C. Steele, and R.F. Albrecht, editors, *Artificial Neural Nets and Genetic Algorithms 2*, pages 277–280, Wien New York, 1995. Springer-Verlag.
46. C. Cotta, I. Dotú, A.J. Fernández, and P. Van Hentenryck. A memetic approach to Golomb rulers. In T.P. Runarsson et al., editors, *Parallel Problem Solving from Nature IX*, volume 4193 of *Lecture Notes inComputer Science*, pages 252–261. Springer-Verlag, Berlin Heidelberg, 2006.
47. C. Cotta, I. Dotú, A.J. Fernández, and P. Van Hentenryck. Scheduling social golfers with memetic evolutionary programming. In *Hybrid Metaheuristic 2006*, volume 4030 of *Lecture Notes in Computer Science*, pages 150–161. Springer-Verlag, 2006.
48. C. Cotta and A. Fernández. A hybrid GRASP - evolutionary algorithm approach to golomb ruler search. In X. Yao et al., editors, *Parallel Problem Solving From Nature VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 481–490, Berlin, 2004. Springer-Verlag.
49. C. Cotta and A.J. Fernández. Memetic algorithms in planning, scheduling, and timetabling. In K.P. Dahal, K.C. Tan, and P.I. Cowling, editors, *Evolutionary Scheduling*, volume 49 of *Studies in Computational Intelligence*, pages 1–30. Springer-Verlag, 2007.
50. C. Cotta and P. Moscato. Evolutionary computation: Challenges and duties. In A. Menon, editor, *Frontiers of Evolutionary Computation*, pages 53–72. Kluwer Academic Publishers, Boston MA, 2004.

51. C. Cotta, M. Sevaux, and K. Sörensen. *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*. Springer-Verlag, Berlin Heidelberg, 2008.
52. C. Cotta and J.M. Troya. On the influence of the representation granularity in heuristic forma recombination. In J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, editors, *ACM Symposium on Applied Computing 2000*, pages 433–439. ACM Press, 2000.
53. C. Cotta and J.M. Troya. Embedding branch and bound within evolutionary algorithms. *Applied Intelligence*, 18(2):137–153, 2003.
54. P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to schedule a sales submit. In E. Burke and W. Erben, editors, *PATAT 2000*, volume 2079 of *Lecture Notes in Computer Science*, pages 176–190, Berlin Heidelberg, 2008. Springer-Verlag.
55. M. Cox, N. Bowden, P. Moscato, R. Berretta, R. I. Scott, and J. S. Lechner-Scott. Memetic algorithms as a new method to interpret gene expression profiles in multiple sclerosis. *Multiple Sclerosis*, 13(Suppl. 2):S205, 2007.
56. J.-C. Créput and A. Koukam. The memetic self-organizing map approach to the vehicle routing problem. *Soft Computing*, 12(11):1125–1141, 2008.
57. M.A. Cruz-Chavez, O. Díaz-Parra, D. Juárez-Romero, and M. G. Martínez-Rangel. Memetic algorithm based on a constraint satisfaction technique for VRPTW. In L. Rutkowski et al., editors, *9th Artificial Intelligence and Soft Computing Conference*, volume 5097 of *Lecture Notes in Computer Science*, pages 376–387. Springer-Verlag, 2008.
58. M.J. Dantas, L. da C. Brito, and P.H. de Carvalho. Multi-objective Memetic Algorithm applied to the automated synthesis of analog circuits. In *Advances in Artificial Intelligence*, volume 4140 of *Lecture Notes in computer Science*, pages 258–267. Springer-Verlag, 2006.
59. Y. Davidor. Epistasis Variance: Suitability of a Representation to Genetic Algorithms. *Complex Systems*, 4(4):369–383, 1990.
60. Y. Davidor and O. Ben-Kiki. The interplay among the genetic algorithm operators: Information theory tools used in a holistic way. In R. Männer and B. Manderick, editors, *Parallel Problem Solving From Nature II*, pages 75–84, Amsterdam, 1992. Elsevier Science Publishers B.V.
61. L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold Computer Library, New York, 1991.
62. R. Dawkins. *The Selfish Gene.* Clarendon Press, Oxford, 1976.
63. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Chichester, UK, 2001.
64. M. Delgado, M.P. Cuellar, and M.C. Pegalajar. Multiobjective hybrid optimization and training of recurrent neural networks. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 38(2):381–403, 2008.
65. M. Delgado, M.C. Pegalajar, and M.P. Cuellar. Memetic evolutionary training for recurrent neural networks: an application to time-series prediction. *Expert Systems*, 23(2):99–115, 2006.
66. J. Denzinger and T. Offermann. On cooperation between evolutionary algorithms and other search paradigms. In *6th International Conference on Evolutionaey Computation*, pages 2317–2324. IEEE Press, 1999.
67. V. di Gesù, G. Lo Bosco, F. Millonzi, and C. Valenti. Discrete tomography reconstruction through a new memetic algorithm. In M. Giacobini et al., editors, *Applications of Evolutionary Computing*, volume 4974 of *Lecture Notes in Computer Science*, pages 347–352. Springer-Verlag, 2008.
68. V. di Gesù, G. Lo Bosco, F. Millonzi, and C. Valenti. A memetic algorithm for binary image reconstruction. In *Combinatorial Image Analysis*, pages 384–395, 2008.
69. F. Divina. *Hybrid Genetic Relational Search for Inductive Learning.* PhD thesis, Department of Computer Science, Vrije Universiteit, Amsterdam, the Netherlands, 2004.

70. A.-D. Do and S.Y. Cho. Memetic algorithm based fuzzy clustering. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2398–2404, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
71. B. Dorronsoro, E. Alba, G. Luque, and P. Bouvry. A self-adaptive cellular memetic algorithm for the DNA fragment assembly problem. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 2656–2663, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
72. Z. Drezner. Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers & Operations Research*, 35(3):717–736, MAR 2008.
73. I. Dumitrescu and T. Stützle. Combinations of local search and exact algorithms. In Günther R. Raidl et al., editors, *Applications of Evolutionary Computing: EvoWorkshops 2003*, volume 2611 of *LNCS*, pages 212–224. Springer, 2003.
74. A. El-Fallahi, C. Prins, and R. Wolfler Calvo. A memetic algorithm and a tabu search for the multi-compartment vehicle routing problem. *Computers & OR*, 35(5):1725–1741, 2008.
75. R. Englemore and T. Morgan (eds.). *Blackboard Systems.* Addison-Wesley, 1988.
76. S. Fernandes and H. Lourenço. Hybrids combining local search heurisitcs with exact algorithms. In F. Almeida et al., editors, *V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, pages 269–274, Las Palmas, Spain, 2007.
77. E. Fernández, M. Graña, and J. Ruiz-Cabello. An instantaneous memetic algorithm for illumination correction. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pages 1105–1110, Portland, Oregon, 20-23 June 2004. IEEE Press.
78. T. Fischer, K. Bauer, and P. Merz. Distributed memetic algorithm for the routing and wavelength problem. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature X*, volume 5199 of *Lecture Notes in Computer Science*, pages 879–888, Berlin Heidelberg, 2008. Springer-Verlag.
79. T. Fischer and P. Merz. A memetic algorithm for the optimum communication spanning tree problem. In T. Bartz-Beielstein et al., editors, *Hybrid Metaheuristics 2007*, volume 4771 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2007.
80. G. Fleury, P. Lacomme, and C. Prins. Evolutionary algorithms for stochastic arc routing problems. In G.R. Raidl et al., editors, *Applications of Evolutionary Computing*, volume 3005 of *Lecture Notes in Computer Science*, pages 501–512, Berlin, 2004. Springer Verlag.
81. F. Flórez-Revuelta, J.M. Casado-Díaz, L. Martínez-Bernabeu, and R. Gómez-Hernández. A memetic algorithm for the delineation of local labour markets. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature X*, volume 5199 of *Lecture Notes in Computer Science*, pages 1011–1020, Berlin Heidelberg, 2008. Springer-Verlag.
82. P. M. França, J. N. D. Gupta, A. S. Mendes, P. Moscato, and K. J. Veltnik. Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. *Computers and Industrial Engineering*, 48:491–506, 2005.
83. P. M. França, A. S. Mendes, and P. Moscato. A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operational Research*, 132:224–242, 2001.
84. P.M. França, G. Tin, and L.S. Buriol. Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions. *International Journal of Production Research*, 44(5):939–957, 2006.
85. B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, Nagoya, Japan*, pages 616–621. IEEE Press, 1996.

86. A.P. French, A.C. Robinson, and J.M. Wilson. Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems. *Journal of Heuristics*, 7(6):551–564, 2001.
87. J. E. Gallardo, C. Cotta, and A. J. Fernández. A hybrid model of evolutionary algorithms and branch-and-bound for combinatorial optimization problems. In *2005 Congress on Evolutionary Computation*, pages 2248–2254, Edinburgh, UK, 2005. IEEE Press.
88. J.E. Gallardo, C. Cotta, and A.J. Fernández. Solving the multidimensional knapsack problem using an evolutionary algorithm hybridized with branch and bound. In J. Mira and J.R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 21–30, Berlin Heidelberg, 2005. Springer-Verlag.
89. J.E. Gallardo, C. Cotta, and A.J. Fernández. A multi-level memetic/exact hybrid algorithm for the still life problem. In T.P. Runarsson et al., editors, *Parallel Problem Solving from Nature IX*, volume 4193 of *Lecture Notes inComputer Science*, pages 212–221. Springer-Verlag, Berlin Heidelberg, 2006.
90. J.E. Gallardo, C. Cotta, and A.J. Fernández. A memetic algorithm with bucket elimination for the still life problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 73–84, Budapest, 10-12 April 2006. Springer-Verlag.
91. J.E. Gallardo, C. Cotta, and A.J. Fernández. A memetic algorithm for the low autocorrelation binary sequence problem. In H. Lipson, editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation Conference*, pages 1226–1233. ACM Press, 2007.
92. J.E. Gallardo, C. Cotta, and A.J. Fernández. On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):77–83, 2007.
93. J.E. Gallardo, C. Cotta, and A.J. Fernández. Reconstructing phylogenies with memetic algorithms and branch-and-bound. In S. Bandyopadhyay, U. Maulik, and J. Tsong-Li Wang, editors, *Analysis of Biological Data: A Soft Computing Approach*, pages 59–84. World Scientific, 2007.
94. S. García, J. R. Cano, and F. Herrera. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition*, 41(8):2693–2709, August 2008.
95. F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–684, 2000.
96. M.A. González, C.R. Vela, M.R. Sierra, I. González Rodríguez, and R. Varela. Comparing schedule generation schemes in memetic algorithms for the job shop scheduling problem with sequence dependent setup times. In A.F. Gelbukh and C.A. Reyes García, editors, *5th Mexican International Conference on Artificial Intelligence*, volume 4293 of *Lecture Notes in Computer Science*, pages 472–482. Springer-Verlag, 2006.
97. M.A. González, C.R. Vela, and R. Varela. Scheduling with memetic algorithms over the spaces of semi-active and active schedules. In *Artificial Intelligence and Soft Computing*, volume 4029 of *Lecture Notes in computer Science*, pages 370–379. Springer-Verlag, Berlin Heidelberg, 2006.
98. I. González-Rodríguez, C.R. Vela, and J. Puente. A memetic approach to fuzzy job shop based on expectation model. In *2007 IEEE International Conference on Fuzzy Systems*, pages 1–6, 2007.
99. M. Gorges-Schleuter. ASPARAGOS: An asynchronous parallel genetic optimization strategy. In J. David Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 422–427. Morgan Kaufmann Publishers, 1989.
100. M. Gorges-Schleuter. Explicit Parallelism of Genetic Algorithms through Population Structures. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 150–159. Springer-Verlag, 1991.

101. J. Gottlieb. Permutation-based evolutionary algorithms for multidimensional knapsack problems. In J. Carroll, E. Damiani, H. Haddad, and D. Oppenheim, editors, *ACM Symposium on Applied Computing 2000*, pages 408–414. ACM Press, 2000.
102. P. Grim. The undecidability of the spatialized prisoner's dilemma. *Theory and Decision*, 42(1):53–80, 1997.
103. A. Guillén, H. Pomares, J. González, I. Rojas, L.J. Herrera, and A. Prieto. Parallel multi-objective memetic RBFNNs design and feature selection for function approximation problems. In F. Sandoval, A. Prieto, J. Cabestany, and M. Graña, editors, *9th International Work-Conference on Artificial Neural Networks*, volume 4507 of *Lecture Notes in Computer Science*, pages 341–350. Springer-Verlag, 2007.
104. F.G. Guimarães, F. Campelo, H. Igarashi, D.A. Lowther, and J.A. Ramírez. Optimization of cost functions using evolutionary algorithms with local learning and local search. *IEEE Transactions on Magnetics*, 43(4):1641–1644, 2007.
105. X.P. Guo, Z.M. Wu, and G.K. Yang. A hybrid adaptive multi-objective memetic algorithm for 0/1 knapsack problem. In *AI 2005: Advances in Artificial Intelligence*, volume 3809 of *Lecture Notes in Artificial Intelligence*, pages 176–185. Springer-Verlag, Berlin Heidelberg, 2005.
106. X.P. Guo, G.K. Yang, and Z.M. Wu. A hybrid self-adjusted memetic algorithm for multi-objective optimization. In *4th Mexican International Conference on Artificial Intelligence*, volume 3789 of *Lecture Notes in Computer Science*, pages 663–672, Berlin Heidelberg, 2005. Springer-Verlag.
107. X.P. Guo, G.K. Yang, Z.M. Wu, and Z.H. Huang. A hybrid fine-timed multi-objective memetic algorithm. *IEICE Transactions on Fundamentals of Electronics Communication and Computer Sciences*, E89A(3):790–797, 2006.
108. W.E. Hart and R.K. Belew. Optimizing an arbitrary function is hard for the genetic algorithm. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 190–195, San Mateo CA, 1991. Morgan Kaufmann.
109. W.E. Hart, N. Krasnogor, and J.E. Smith. *Recent advances in memetic algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, 2005.
110. C. Hervás and M. Silva. Memetic algorithms-based artificial multiplicative neural models selection for resolving multi-component mixtures based on dynamic responses. *Chemometrics and Intelligent Laboratory Systems*, 85(2):232–242, 2007.
111. D.R. Hofstadter. Computer tournaments of the prisoners-dilemma suggest how cooperation evolves. *Scientific American*, 248(5):16–23, 1983.
112. C. Houck, J.A. Joines, M.G. Kay, and J.R. Wilson. Empirical investigation of the benefits of partial lamarckianism. *Evolutionary Computation*, 5(1):31–60, 1997.
113. C.-H. Hsu. Uplink MIMO-SDMA optimisation of smart antennas by phase-amplitude perturbations based on memetic algorithms for wireless and mobile communication systems. *IET Communications*, 1(3):520–525, 2007.
114. C.-H. Hsu, P.-H. Chou, W.-J. Shyr, and Y.-N. Chung. Optimal radiation pattern design of adaptive linear array antenna by phase and amplitude perturbations using memetic algorithms. *International Journal of Innovative Computing, Information and Control*, 3(5):1273–1287, 2007.
115. C.-H. Hsu and W.-J. Shyr. Memetic algorithms for optimizing adaptive linear array patterns by phase-position perturbations. *Circuits Systems and Signal Processing*, 24(4):327–341, 2005.
116. C.-H. Hsu and W.-J. Shyr. Optimizing linear adaptive broadside array antenna by amplitude-position perturbations using memetic algorithms. In R. Khosla, R.J. Howlett, and L.C. Jain, editors, *9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems*, volume 3681 of *Lecture Notes in Computer Science*, pages 568–574. Springer-Verlag, 2005.
117. C.-H. Hsu, W.-J. Shyr, and C.-H. Chen. Adaptive pattern nulling design of linear array antenna by phase-only perturbations using memetic algorithms. In *First In-*

ternational Conference on Innovative Computing, Information and Control, pages 308–311, Beijing, China, 2006. IEEE Computer Society.

118. D. Huang, C. Leung, and C. Miao. Memetic algorithm for dynamic resource allocation in multiuser OFDM based cognitive radio systems. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 3861–3866, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
119. M. Hulin. An optimal stop criterion for genetic algorithms: A bayesian approach. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 135–143, San Mateo, CA, 1997. Morgan Kaufmann.
120. H. Ishibuchi, Y. Hitotsuyanagi, N. Tsukamoto, and Y. Nojima. Use of heuristic local search for single-objective optimization in multiobjective memetic algorithms. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature X*, volume 5199 of *Lecture Notes in Computer Science*, pages 743–752, Berlin Heidelberg, 2008. Springer-Verlag.
121. H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In T. Fukuda and T. Furuhashi, editors, *1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE Press.
122. H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(3):392–403, 1998.
123. H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, 2003.
124. A. Jaszkiewicz. Genetic local search for multiple objective combinatorial optimization. *European Journal of Operational Research*, 137(1):50–71, 2002.
125. A. Jaszkiewicz. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. *Annals of Operations Research*, 131(1-4):135–158, 2004.
126. D.S. Johnson, C.H. Papadimitriou, and M. Yannakakis. How easy is local search ? *Journal of Computers and System Sciences*, 37:79–100, 1988.
127. T.C. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, 1995.
128. B. Karaoğlu, H. Topçuoğlu, and F. Gürgen. Evolutionary algorithms for location area management. In F. Rothlauf et al., editors, *Applications of Evolutionary Computing*, volume 3449 of *LNCS*, pages 175–184, Lausanne, Switzerland, 30 March-1 April 2005. Springer Verlag.
129. A. Kaveh and M. Shahrouzi. Graph theoretical implementation of memetic algorithms in structural optimization of frame bracing layouts. *Engineering Computatins*, 25(1-2):55–85, 2008.
130. S.-S. Kim, A.E. Smith, and J.-H. Lee. A memetic algorithm for channel assignment in wireless FDMA systems. *Computers & OR*, 34(6):1842–1856, 2007.
131. G.W. Klau, I. Ljubić, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G.R. Raidl, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. *GECCO 04: Genetic and Evolutionary Computation Conference*, 3102(Part 1):1304–1315, 2004.
132. J. Knowles and D. Corne. Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects. In W.E. Hart, N. Krasnogor, and J. E. Smith, editors, *Recent Advances in Memetic Algorithms*, volume 166 of *Studies in Fuzziness and Soft Computing*, pages 313–352. Springer-Verlag, 2005.
133. J. Knowles and D.W. Corne. Approximating the non-dominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
134. J.D. Knowles and D. W. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC00)*, pages 325–332, Piscataway, NJ, 2000. IEEE Press.

135. J.D. Knowles and D.W. Corne. A Comparison of Diverse Aproaches to Memetic Multiobjective Combinatorial Optimization. In Annie S. Wu, editor, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pages 103–108, 2000.
136. A.V. Kononova, K.J. Hughes, M. Pourkashanian, and D.B. Ingham. Fitness diversity based adaptive memetic algorithm for solving inverse problems of chemical kinetics. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2366–2373, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
137. A.V. Kononova, D.B. Ingham, and M. Pourkashanian. Simple scheduled memetic algorithm for inverse problems in higher dimensions: Application to chemical kinetics. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 3906–3913, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
138. A. Konstantinidis, K. Yang, H.-H. Chen, and Q. Zhang. Energy-aware topology control for wireless sensor networks using memetic algorithms. *Computer Communications*, 30(14-15):2753–2764, 2007.
139. K. Kostikas and C. Fragakis. Genetic programming applied to mixed integer programming. In M. Keijzer et al., editors, *7th European Conference on Genetic Programming*, volume 3003 of *Lecture Notes in Computer Science*, pages 113–124, Berlin Heidelberg, 2004. Springer-Verlag.
140. N. Krasnogor. Self generating metaheuristics in bioinformatics: The proteins structure comparison case. *Genetic Programming and Evolvable Machines*, 5(2):181–201, June 2004.
141. N. Krasnogor, B.P. Blackburne, E.K. Burke, and J.D. Hirst. Multimeme algorithms for protein structure prediction. In J.J. Merelo et al., editors, *Parallel Problem Solving From Nature VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 769–778. Springer-Verlag, Berlin, 2002.
142. N. Krasnogor and S.M. Gustafson. A study on the use of "self-generation" in memetic algorithms. *Natural Computing*, 3(1):53–76, 2004.
143. N. Krasnogor and J. Smith. Memetic algorithms: The polynomial local search complexity theory perspective. *Journal of Mathematical Modelling and Algorithms*, 7(1):3–24, 2008.
144. M. Kretowski. A memetic algorithm for global induction of decision trees. In V. Geffert et al., editors, *34th Conference on Current Trends in Theory and Practice of Computer Science*, volume 4910 of *Lecture Notes in Computer Science*, pages 531–540. Springer-Verlag, 2008.
145. M. Kubiak and P. Wesolek. Accelerating local search in a memetic algorithm for the capacitated vehicle routing problem. In C. Cotta and J.I. van Hemert, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 4446 of *Lecture Notes in Computer Science*, pages 96–107. Springer-Verlag, 2007.
146. P. Lacomme, C. Prins, and W. Ramdane-Cherif. Competitive memetic algorithms for arc routing problems. *Annals of Operations Research*, 131(1-4):159–185, 2004.
147. P. Lacomme, C. Prins, and W. Ramdane-Cherif. Evolutionary algorithms for periodic arc routing problems. *European Journal of Operational Research*, 165(2):535–553, 2005.
148. M. Laguna and R. Martí. *Scatter Search. Methodology and Implementations in C.* Kluwer Academic Publishers, Boston MA, 2003.
149. E. Lamma, L. M. Pereira, and F. Riguzzi. Multi-agent logic aided lamarckian learning. Technical Report DEIS-LIA-00-004, Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna (Italy), 2000.
150. H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1998.
151. R. Lewis and B. Paechter. Finding feasible timetables using group-based operators. *IEEE Transactions on Evolutionary Computation*, 11(3):397–413, 2007.

152. B.-B. Li, L. Wang, and B. Liu. An effective PSO-based hybrid algorithm for multiobjective permutation flow shop scheduling. *IEEE Transactions on Systems, Man and Cybernetics Part B*, 38(4):818–831, 2008.
153. J. Li and R. S. K. Kwan. A self adjusting algorithm for driver scheduling. *Journal of Heuristics*, 11(4):351–367, 2005.
154. A. Lim, B. Rodrigues, and Y. Zhu. Airport gate scheduling with time windows. *Artificial Intelligence Review*, 24(1):5–31, 2005.
155. D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In D. Thierens et al., editors, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1288–1295, London, 7-11 July 2007. ACM Press.
156. K.K. Lim, Y.-S. Ong, M.H. Lim, X. Chen, and A. Agarwal. Hybrid ant colony algorithms for path planning in sparse graphs. *soft Computing*, 12(10):981–994, 2008.
157. S. Lin and B. Kernighan. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research*, 21:498–516, 1973.
158. B. Liu, L. Wang, and Y. Jin. An effective PSO-based memetic algorithm for flow shop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):18–27, 2007.
159. B. Liu, L. Wang, Y. Jin, and D. Huang. Designing neural networks using PSO-based memetic algorithm. In D. Liu, S. Fei, Z.-G. Hou, H. Zhang, and C. Sun, editors, *4th International Symposium on Neural Networks*, volume 4493 of *Lecture Notes in Computer Science*, pages 219–224. Springer-Verlag, 2007.
160. B. Liu, L. Wang, and Y.-H. Jin. An effective hybrid particle swarm optimization for no-wait flow shop scheduling. *International Journal of Advances Manufacturing Technology*, 31(9-10):1001–1011, 2007.
161. B. Liu, L. Wang, Y.-H. Jin, and D.-X. Huang. An effective PSO-based memetic algorithm for TSP. In *Intelligent Computing in Signal Processing and Pattern Recognition*, volume 345 of *Lecture Notes in Control and Information Sciences*, pages 1151–1156. Springer-Verlag, 2006.
162. D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho. A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):42–50, 2007.
163. Y.-H. Liu. A memetic algorithm for the probabilistic traveling salesman problem. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 146–152, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
164. M. Lozano, F. Herrera, N. Krasnogor, and D. Molina. Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation*, 12(3):273–302, 2004.
165. E. Lumanpauw, M. Pasquier, and C. Quek. MNFS-FPM: A novel memetic neuro-fuzzy system based financial portfolio management. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2554–2561, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
166. R. Maheswaran, S.G. Ponnambalam, and C. Aravindan. A meta-heuristic approach to single machine scheduling problems. *International Journal of Advanced Manufacturing Technology*, 25(7-8):772–776, 2005.
167. D.G. Maringer. Finding the relevant risk factors for asset pricing. *Computational Statistics & Data Analysis*, 47(2):339–352, 2004.
168. F. J. Martínez-Estudillo, C. Hervás-Martínez, A.C. Martínez-Estudillo, and D. Ortiz-Boyer. Memetic algorithms to product-unit neural networks for regression. In J. Cabestany, A.Prieto, and F. Sandoval Hernández, editors, *8th International Work-Conference on Artificial Neural Networks*, volume 3512 of *Lecture Notes in Computer Science*, pages 83–90. Springer-Verlag, 2005.
169. A. Mendes, C. Cotta, V. Garcia, P.M. França, and P. Moscato. Gene ordering in microarray data using parallel memetic algorithms. In T. Skie and C.-S. Yang, editors,

Proceedings of the 2005 International Conference on Parallel Processing Workshops, pages 604–611, Oslo, Norway, 2005. IEEE Press.
170. A. Mendes, P.M. França, C. Lyra, C. Pissarra, and C. Cavellucci. Capacitor placement in large-sized radial distribution networks. *IEE Proceedings*, 152(4):496–502, 2005.
171. A. Mendes and A. Linhares. A multiple-population evolutionary approach to gate matrix layout. *International Journal of Systems Science*, 35(1):13–23, 2004.
172. A. S. Mendes, P. M. França, and P. Moscato. Fitness landscapes for the total tardiness single machine scheduling problem. *Neural Network World*, 2(2):165–180, 2002.
173. P. Merz and K. Katayama. Memetic algorithms for the unconstrained binary quadratic programming problem. *Biosystems*, 78(1-3):99–118, 2004.
174. P. Merz and S. Wolf. Evolutionary local search for designing peer-to-peer overlay topologies based on minimum routing cost spanning trees. In T.P. Runarsson et al., editors, *Parallel Problem Solving from Nature IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 272–281. Springer-Verlag, Berlin Heidelberg, 2006.
175. D. Molina, F. Herrera, and M. Lozano. Adaptive local search parameters for real-coded memetic algorithms. In D. Corne et al., editors, *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 888–895, Edinburgh, Scotland, UK, 2-5 September 2005. IEEE Press.
176. D. Molina, M. Lozano, and F. Herrera. Memetic algorithms for intense continuous local search methods. In M.J. Blesa et al., editors, *Hybrid Metaheuristics 2008*, volume 5296 of *Lecture Notes in Computer Science*, pages 58–71, Berlin Heidelberg, 2008. Springer-Verlag.
177. P. Moscato. 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.
178. P. Moscato. An Introduction to Population Approaches for Optimization and Hierarchical Objective Functions: The Role of Tabu Search. *Annals of Operations Research*, 41(1-4):85–121, 1993.
179. P. Moscato. Memetic algorithms: A short introduction. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 219–234. McGraw-Hill, 1999.
180. P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Publishers, Boston MA, 2003.
181. P. Moscato and C. Cotta. Memetic algorithms. In T. González, editor, *Handbook of Approximation Algorithms and Metaheuristics*, chapter 22. Taylor & Francis, 2006.
182. P. Moscato, C. Cotta, and A. Mendes. Memetic algorithms. In G.C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 53–85. Springer-Verlag, Berlin Heidelberg, 2004.
183. P. Moscato, A. Mendes, and R. Berretta. Benchmarking a memetic algorithm for ordering microarray data. *Biosystems*, 88(1-2):56–75, 2007.
184. P. Moscato, A. Mendes, and C. Cotta. Scheduling and production & control. In G. C. Onwubolu and B. V. Babu, editors, *New Optimization Techniques in Engineering*, pages 655–680. Springer-Verlag, Berlin Heidelberg, 2004.
185. H. Mühlenbein. Evolution in Time and Space – The Parallel Genetic Algorithm. In Gregory J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 316–337. Morgan Kaufmann Publishers, 1991.
186. H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7:65–88, 1988.
187. A. Muruganandam, G. Prabhaharan, P. Asokan, and V. Baskaran. A memetic algorithm approach to the cell formation problem. *International Journal of Advanced Manufacturing Technology*, 25(9-10):988–997, 2005.
188. Y. Nagata and Sh. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In T. Bäck, editor, *Proceedings of the*

Seventh International Conference on Genetic Algorithms, pages 450–457, San Mateo, CA, 1997. Morgan Kaufmann.
189. M. Nakamaru, H. Matsuda, and Y. Iwasa. The evolution of social interaction in lattice models. *Sociological Theory and Methods*, 12(2):149–162, 1998.
190. M. Nakamaru, H. Nogami, and Y. Iwasa. Score-dependent fertility model for the evolution of cooperation in a lattice. *Journal of Theoretical Biology*, 194(1):101–124, 1998.
191. F. Neri, N. Kotilainen, and M. Vapa. An adaptive global-local memetic algorithm to discover resources in P2P networks. In M. Giacobini et al., editors, *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 61–70. Springer-Verlag, 2007.
192. F. Neri, N. Kotilainen, and M. Vapa. A memetic-neural approach to discover resources in P2P networks. In C. Cotta and J. van Hemert, editors, *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*, pages 113–129. Springer-Verlag, Berlin Heidelberg, 2008.
193. F. Neri and V. Tirronen. On memetic differential evolution frameworks: A study of advantages and limitations in hybridization. In J. Wang, editor, *2008 IEEE World Congress on Computational Intelligence*, pages 2135–2142, Hong Kong, 1-6 June 2008. IEEE Computational Intelligence Society, IEEE Press.
194. F. Neri, J. Toivanen, G. L. Cascella, and Y.-S. Ong. An adaptive multimeme algorithm for designing HIV multidrug therapies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(2):264–278, April 2007.
195. R. Neruda and S. Slusny. Variants of memetic and hybrid learning of perceptron networks. In *18th International Workshop on Database and Expert Systems Applications*, pages 158–162. IEEE Computer Society, 2007.
196. H.D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga. Implementation of an effective hybrid GA for large-scale traveling salesman problems. *IEEE Transactions on systems, Man and Cybernetics Part B*, 37(1):92–99, 2007.
197. Q. H. Nguyen, Y.-S. Ong, and N. Krasnogor. A study on the design issues of memetic algorithm. In D. Srinivasan and L. Wang, editors, *2007 IEEE Congress on Evolutionary Computation*, pages 2390–2397, Singapore, 25-28 September 2007. IEEE Computational Intelligence Society, IEEE Press.
198. R. Niedermeier and P. Rossmanith. An efficient fixed parameter algorithm for 3-hitting set. Technical Report WSI-99-18, Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, 1999. Technical Report, Revised version accepted in *Journal of Discrete Algorithms*, August 2000.
199. R. Niedermeier and P. Rossmanith. A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters*, 73:125–129, 2000.
200. N. Noman and H. Iba. Inferring gene regulatory networks using differential evolution with local search heuristics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(4):634–647, October 2007.
201. N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search. *IEEE Transactions on Evolutionary Computation*, 12(1):107–125, 2008.
202. M.G. Norman and P. Moscato. A competitive and cooperative approach to complex combinatorial search. In *Proceedings of the 20th Informatics and Operations Research Meeting*, pages 3.15–3.29, Buenos Aires, 1989.
203. M.T. Oakley, D. Barthel, Y. Bykov, J.M. Garibaldi, E.K. Burke, N. Krasnogor, and J.D. Hirst. Search strategies in structural bioinformatics. *Current Protein & Peptide Science*, 9(3):260–274, 2008.
204. Y.-S. Ong and A.J. Keane. Meta-lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, 2004.
205. Y.-S. Ong, M.-H. Lim, N. Zhu, and K.W. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(1):141–152, 2006.

206. E. Özcan. Memetic algorithms for nurse rostering. In Pinar Yolum et al., editors, *Computer and Information Sciences - ISCIS 2005, 20^{th} International Symposium (ISCIS)*, volume 3733 of *Lecture Notes in Computer Science*, pages 482–492, Berlin Heidelberg, October 2005. Springer-Verlag.
207. E. Özcan and E. Onbasioglu. Memetic algorithms for parallel code optimization. *International Journal of Parallel Programming*, 35(1):33–61, 2007.
208. P. Palacios, D. Pelta, and A. Blanco. Obtaining biclusters in microarrays with population-based heuristics. In F. Rothlauf et al., editors, *Applications of Evolutionary Computing*, volume 3907 of *Lecture Notes in Computer Science*, pages 115–126. Springer-Verlag, Berlin Heidelberg, 2006.
209. Q-K Pan, L. Wang, and B. Qian. A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems. *Journal of Engineering Manufacture*, 222(4):519–539, 2008.
210. M. Pastorino. Stochastic optimization methods applied to microwave imaging: A review. *IEEE Transactions on Antennas and Propagation*, 55(3, Part 1):538–548, 2007.
211. M. Pastorino, S. Caorsi, A. Massa, and A. Randazzo. Reconstruction algorithms for electromagnetic imaging. *IEEE Transactions of Instrumentation and Measurement*, 53(3):692–699, 2004.
212. W Paszkowicz. Properties of a genetic algorithm extended by a random self-learning operator and asymmetric mutations: A convergence study for a task of powder-pattern indexing. *Analytica Chimica Acta*, 566(1):81–98, 2006.
213. M. Peinado and T. Lengauer. Parallel "go with the winners algorithms" in the LogP Model. In *Proceedings of the 11th International Parallel Processing Symposium*, pages 656–664, Los Alamitos, California, 1997. IEEE Computer Society Press.
214. Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis. Memetic particle swarm optimization. *Annals of Operations Research*, 156(1):99–127, 2007.
215. S. Petrovic and E. K. Burke. University timetabling. In J. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 45. Chapman Hall/CRC Press, 2004.
216. S. Petrovic, V. Patel, and Y. Yang. Examination timetabling with fuzzy constraints. In *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 313–333. Springer-Verlag, Berlin Heidelberg, 2005.
217. S. Pirkwieser and G.R. Raidl. Finding consensus trees by evolutionary, variable neighborhood search, and hybrid algorithms. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 323–330, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
218. C. Prins, C. Prodhon, and R.W. Calvo. A memetic algorithm with population management (MA | PM) for the capacitated location-routing problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 183–194. Springer-Verlag, Budapest, 10-12 April 2006.
219. C. Prodhom and C. Prins. A memetic algorithm with population management (MA|PM) for the periodic location-routing problem. In M.J. Blesa et al., editors, *Hybrid Metaheuristics 2008*, volume 5296 of *Lecture Notes in Computer Science*, pages 43–57, Berlin Heidelberg, 2008. Springer-Verlag.
220. J. Puchinger and G.R. Raidl. Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In J. Mira and J.R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 41–53. Springer-Verlag, 2005.
221. J. Puchinger, G.R. Raidl, and G. Koller. Solving a real-world glass cutting problem. In J. Gottlieb and G.R. Raidl, editors, *4th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 3004 of *Lecture Notes in Computer Science*, pages 165–176, Berlin Heidelberg, 2004. Springer-Verlag.

222. J. Puchinger, G.R. Raidl, and U. Pferschy. The core concept for the Multidimensional Knapsack Problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 195–208. Springer-Verlag, Budapest, 10-12 April 2006.
223. M. Qasem and A. Prugel-Bennett. Complexity of Max-SAT using stochastic algorithms. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 615–616, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
224. B. Qian, L. Wang, D.-X. Huang, and X. Wang. Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. *International Journal of Advanced Manufacturing Technology*, 35(9–10):1014–1027, January 2008.
225. A. Quintero and S. Pierre. On the design of large-scale cellular mobile networks using multi-population memetic algorithms. In A. Abraham et al., editors, *Engineering Evolutionary Intelligent Systems*, volume 82 of *Studies in Computational Intelligence*, pages 353–377. Springer-Verlag, 2008.
226. M. Rabbani, A. Rahimi-Vahed, and S.A. Torabi. Real options approach for a mixed-model assembly line sequencing problem. *International Journal of Advanced Manufacturing Technology*, 37(11-12):1209–1219, 2008.
227. N.J. Radcliffe. The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10:339–384, 1994.
228. N.J. Radcliffe and P.D. Surry. Fitness Variance of Formae and Performance Prediction. In L.D. Whitley and M.D. Vose, editors, *Proceedings of the 3rd Workshop on Foundations of Genetic Algorithms*, pages 51–72, San Francisco, 1994. Morgan Kaufmann.
229. N.J. Radcliffe and P.D. Surry. Formal Memetic Algorithms. In T. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, volume 865 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1994.
230. I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart, 1973.
231. D.A.M. Rocha, E.F.G. Goldbarg, and M.C. Goldbarg. A memetic algorithm for the biobjective minimum spanning tree problem. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 222–233. Springer-Verlag, 2006.
232. F.J. Romero-Campero, H. Cao, M. Camara, and N. Krasnogor. Structure and parameter estimation for cell systems biology models. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 331–338, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
233. O. Rossi-Doria and B. Paechter. A memetic algorithm for university course timetabling. In *Combinatorial Optimisation 2004 Book of Abstracts*, page 56, Lancaster, UK, 2004. Lancaster University.
234. E.E. Santos and E. Santos, Jr. Effective computational reuse for energy evaluations in protein folding. *International Journal of Artificial Intelligence Tools*, 15(5):725–739, 2006.
235. M Schoenauer, P Saveant, and V Vidal. Divide-and-evolve: A new memetic scheme for domain-independent temporal planning. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3906 of *Lecture Notes in Computer Science*, pages 247–260. Springer-Verlag, Budapest, 10-12 April 2006.
236. J. Schönberger, D. C. Mattfeld, and H. Kopfer. Memetic algorithm timetabling for non-commercial sport leagues. *European Journal of Operational Research*, 153:102–116, 2004.
237. O. Schuetze, G. Sanchez, and C.A. Coello Coello. A new memetic strategy for the numerical treatment of multi-objective optimization problems. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and*

evolutionary computation, pages 705–712, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
238. H.-P. Schwefel. Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of natural evolution. *Annals of Operations Research*, 1:165–167, 1984.
239. Y. Semet and M. Schoenauer. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In *Proceedings of the 2005 Congress on Evolutionary Computation*, pages 2752–2759, Edinburgh, UK, 2005. IEEE Press.
240. M. Sevaux, A. Jouglet, and C. Oğuz. Combining constraint programming and memetic algorithm for the hybrid flowshop scheduling problem. In *ORBEL 19th annual conference of the SOGESCI-BVWB*, Louvain-la-Neuve, Belgium, 2005.
241. M. Sevaux, A. Jouglet, and C. Oğuz. 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.
242. W. Sheng, G. Howells, M. Fairhurst, and F. Deravi. A memetic fingerprint matching algorithm. *IEEE Transactions on Information Forensics and Security*, 2(3, Part 1):402–412, 2007.
243. W. Sheng, X. Liu, and M. Fairhurst. A niching memetic algorithm for simultaneous clustering and feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 20(7):868–879, 2008.
244. J. E. Smith. Credit assignment in adaptive memetic algorithms. In H. Lipson, editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation Conference*, pages 1412–1419. ACM Press, 2007.
245. J.E. Smith. Co-evolution of memetic algorithms: Initial investigations. In J.J. Merelo et al., editors, *Parallel Problem Solving From Nature VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 537–548. Springer-Verlag, Berlin, 2002.
246. J.E. Smith. Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):6–17, 2007.
247. J.E. Smith. Self-adaptation in evolutionary algorithms for combinatorial optimization. In C. Cotta, M. Sevaux, and K. Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 31–57. Springer-Verlag, Berlin Heidelberg, 2008.
248. S.-M. Soak, S.-W. Lee, N.P. Mahalik, and B.-H. Ahn. A new memetic algorithm using particle swarm optimization and genetic algorithm. In *Knowledge-based Intelligent Information and Engineering Systems*, volume 4251 of *Lecture Notes in Artificial Intelligence*, pages 122–129. Springer-Verlag, 2006.
249. K. Sörensen and M. Sevaux. MA | PM: memetic algorithms with population management. *Computers & OR*, 33:1214–1225, 2006.
250. C. Spieth, F. Streichert, J. Supper, N. Speer, and A. Zell. Feedback memetic algorithms for modeling gene regulatory networks. In *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2005)*, pages 61–67, La Jolla, CA, 2005. IEEE Press.
251. D. Sudholt. Memetic algorithms with variable-depth search to overcome local optima. In M. Keijzer et al., editors, *GECCO '08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 787–794, Atlanta, GA, USA, 12-16 July 2008. ACM Press.
252. P.D. Surry and N.J. Radcliffe. Inoculation to initialise evolutionary search. In T.C. Fogarty, editor, *Evolutionary Computing: AISB Workshop*, number 1143 in Lecture Notes in Computer Science, pages 269–285. Springer-Verlag, 1996.
253. G. Syswerda. Uniform crossover in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1989. Morgan Kaufmann.
254. K. Tagawa and M. Matsuoka. Optimum design of surface acoustic wave filters based on the Taguchi's quality engineering with a memetic algorithm. In T.P. Runarsson

et al., editors, *Parallel Problem Solving from Nature IX*, volume 4193 of *Lecture Notes inComputer Science*, pages 292–301. Springer-Verlag, Berlin Heidelberg, 2006.
255. J. Tang, M. H. Lim, Y.-S. Ong, and M.J. Er. Parallel memetic algorithm with selective local search for large scale quadratic assignment problems. *International Journal of Innovative Computing, Information and Control*, 2(6):1399–1416, 2006.
256. M. Tang and X. Yao. A memetic algorithm for VLSI floorplanning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):62–69, 2007.
257. R. Tavakkoli-Moghaddam and A. R. Rahimi-Vahed. A Memetic Algorithm for Multi-Criteria Sequencing Problem for a Mixed-Model Assembly Line in a JIT Production System. In *2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, pages 10350–10355, Vancouver, BC, Canada, July 2006. IEEE.
258. R. Tavakkoli-Moghaddam, N. Safaei, and M. Babakhani. Solving a dynamic cell formation problem with machine cost and alternative process plan by memetic algorithms. In *International Symposium on Stochastic Algorithms: Foundations and Applications, LNCS*, volume 3, 2005.
259. R. Tavakkoli-Moghaddam, A. R. Saremi, and M. S. Ziaee. A memetic algorithm for a vehicle routing problem with backhauls. *Applied mathematics and Computation*, 181(2):1049–1060, 2006.
260. Y. Tenne and S.W. Armfield. A memetic algorithm using a trust-region derivative-free optimization with quadratic modelling for optimization of expensive and noisy black-box functions. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 389–415. Springer-Verlag, 2007.
261. V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi. A memetic differential evolution in filter design for defect detection in paper production. In M. Giacobini et al., editors, *Applications of Evolutionary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 320–329. Springer-Verlag, 2007.
262. J. Togelius, T. Schaul, J. Schmidhuber, and F. Gómez. Countering poisonous inputs with memetic neuroevolution. In G. Rudolph et al., editors, *Parallel Problem Solving from Nature X*, volume 5199 of *Lecture Notes in Computer Science*, pages 610–619, Berlin Heidelberg, 2008. Springer-Verlag.
263. F. Tricoire. Vehicle and personnel routing optimization in the service sector: application to water distribution and treatment. *4OR-A Quarterly Journal of Operations Research*, 5(2):165–168, 2007.
264. S.-M. Tse, Y. Liang, K.-S. Leung, K.-H. Lee, and T.S.K. Mok. A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):84–91, 2007.
265. H.E. Tseng, W.P. Wang, and H.Y. Shih. Using memetic algorithms with guided local search to solve assembly sequence planning. *Expert Systems With Applications*, 33(2):451–467, 2007.
266. E.L. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens. MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.
267. R. Varela, J. Puente, and C. R. Vela. Some issues in chromosome codification for scheduling with genetic algorithms. In L. Castillo, D. Borrajo, M. A. Salido, and A. Oddi, editors, *Planning, Scheduling and Constraint Satisfaction: From Theory to Practice*, volume 117 of *Frontiers in Artificial Intelligence and Applications*, pages 1–10. IOS Press, 2005.
268. R. Varela, D. Serrano, and M. Sierra. New codification schemas for scheduling with genetic algorithms. In J. Mira and J. R. Álvarez, editors, *Artificial Intelligence and Knowledge Engineering Applications: a Bioinspired Approach*, volume 3562 of *Lecture Notes in Computer Science*, pages 11–20, Berlin Heidelberg, 2005. Springer-Verlag.

269. J. Volk, T. Herrmann, and K. Wuethrich. Automated sequence-specific protein NMR assignment using the memetic algorithm match. *Journal of Biomolecular NMR*, 41(3):127–138, 2008.
270. J. Wang. A memetic algorithm with genetic particle swarm optimization and neural network for maximum cut problems. In K. Li, M. Fei, G.W. Irwin, and S. Ma, editors, *International Conference on Life System Modeling and Simulation*, volume 4688 of *Lecture Notes in Computer Science*, pages 297–306. Springer-Verlag, 2007.
271. Y. Wang and J. Qin. A memetic-clustering-based evolution strategy for traveling salesman problems. In J. Yao et al., editors, *2nd International Conference on Rough Sets and Knowledge Technology*, volume 4481 of *Lecture Notes in Computer Science*, pages 260–266. Springer-Verlag, 2007.
272. E.F. Wanner, F.G. Guimarães, R.H.C. Takahashi, and P.J. Fleming. Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria. *Evolutionary Computation*, 16(2):185–224, 2008.
273. E.F. Wanner, F.G. Guimarães, R.H.C. Takahashi, D.A. Lowther, and J.A. Ramírez. Multiobjective memetic algorithms with quadratic approximation-based local search for expensive optimization in electromagnetics. *IEEE Transactions on Magnetics*, 44(6):1126–1129, 2008.
274. D. Whitley. Using reproductive evaluation to improve genetic search and heuristic discovery. In J.J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, pages 108–115, Cambridge, MA, July 1987. Lawrence Erlbaum Associates.
275. T.L. Williams and M.L. Smith. The role of diverse populations in phylogenetic analysis. In M. Keijzer et al., editors, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 287–294, Seattle, Washington, USA, 8-12 July 2006. ACM Press.
276. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
277. F. Xhafa and B. Duran. Parallel memetic algorithms for independent job scheduling in computational grids. In C. Cotta and J. van Hemert, editors, *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, volume 153 of *Studies in Computational Intelligence*, pages 219–239. Springer-Verlag, Berlin Heidelberg, 2008.
278. J.-H. Yang, L. Sun, H.P. Lee, Y. Qian, and Y.-C. Liang. Clonal selection based memetic algorithm for job shop scheduling problems. *Journal of Bionic Engineering*, 5(2):111–119, 2008.
279. M. Yannakakis. Computational complexity. In E.H.L. Aarts and J.K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 19–55. Wiley, Chichester, 1997.
280. W.-C. Yeh. An efficient memetic algorithm for the multi-stage supply chain network problem. *International Journal of Advanced Manufacturing Technology*, 29(7-8):803–813, 2006.
281. X. Zhao. Advances on protein folding simulations based on the lattice HP models with natural computing. *Applied Soft Computing*, 8(2):1029–1040, 2008.
282. Z. Zhen, Z. Wang, Z. Gu, and Y. Liu. A novel memetic algorithm for global optimization based on PSO and SFLA. In L. Kang, Y. Liu, and S. Y. Zeng, editors, *2nd International Symposium on Advances in Computation and Intelligence*, volume 4683 of *Lecture Notes in Computer Science*, pages 127–136. Springer-Verlag, 2007.
283. Z. Zhou, Y.-S. Ong, M.-H. Lim, and B.-S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing*, 11(10):957–971, 2007.
284. Z. Zhu and Y.-S. Ong. Memetic algorithms for feature selection on microarray data. In D. Liu et al., editors, *4th International Symposium on Neural Networks*, volume 4491 of *Lecture Notes in Computer Science*, pages 1327–1335. Springer-Verlag, 2007.
285. Z. Zhu, Y.-S. Ong, and M. Dash. Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition*, 40(11):3236–3248, 2007.

286. Z. Zhu, Y.-S. Ong, and M. Dash. Wrapper-filter feature selection algorithm using a memetic framework. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):70–76, 2007.
287. E. Zitzler, M. Laumanns, and S. Bleuler. A Tutorial on Evolutionary Multiobjective Optimization. In X. Gandibleux et al., editors, *Metaheuristics for Multiobjective Optimisation*, volume 535 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag, 2004.