Chapter 5

A GENTLE INTRODUCTION TO MEMETIC ALGORITHMS

Pablo Moscato
Grupo de Engenharia de Computação em Sistemas Complexos,
Departamento de Engenharia de Computação e Automação Industrial,
Faculdade de Engenharia Eletrónica e de Computação, Universidade Estadual de Campinas,
C.P. 6101, Campinas, SP, CEP 13083-970, Brazil
E-mail: moscato@densis.fee.unicamp.br

Carlos Cotta
Departamento de Lenguajes y Ciencias de la Computación,
Escuela Técnica Superior de Ingeniería Informática, Universidad de Málaga,
Complejo Tecnológico (3.2.49), Campus de Teatinos,
29071-Málaga, Spain
E-mail: ccottap@lcc.uma.es

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 $\mathcal{NP}$ 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 Dawkins [52], the word *'meme'* denotes an analogous to the gene in the context of cultural evolution [154]. 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. 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 [169]. 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 [88] initially stated and Wolpert and Macready [224] 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* [42] is commonly used to denote the process of incorporating problem knowledge, it is not surprising that MAs are sometimes called 'Hybrid Evolutionary Algorithms' [51] as well. One of the first algorithms to which the MA label was assigned dates from 1988 [169], and was regarded by many as a hybrid of *traditional* Genetic Algorithms (GAs) and *SimulatedAnnealing* (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* [96] used to study "*the evolution of cooperation*" [6] (see also [164,171] for more recent theoretical results in this field). 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 "*clonation*", which has a strong similarity with the more recent "*go with the winners*" algorithms [4, 182]. 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 [74,157] and other authors proposing "*island models*" for GAs. Spacialization is now being recognized as the "catalyzer" responsible of a variety of phenomena [163,164]. This is an important research issue, currently only understood in a rather heuristic way. However, some proper undecidability results of have been obtained for related problems [80] 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 [73,158] (see other references and the discussion in [154]). Particularly coming from the GA field, several authors were introducing *problem-domain knowledge* in a variety of ways. In [154] 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

$\mathcal{NP}$-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, 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 may 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 $V = \{v_0, \ldots, v_{n-1}\}$, a vector of capacities $C = \{c_0, \ldots, c_{m-1}\}$, and a matrix of capacity constraints $M = \{m_{ij}\colon 0 \le i < m, 0 \le j < n\}$. 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 $\mathbb{N}_n = \{0, 1, \ldots, n-1\}$, the answer set $ans_P(x)$ for an instance x is simply the power set of $\mathbb{N}_n$, i.e., 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 $M \cdot B \le C'$. 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, being the goal 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* $g : \mathcal{S}_P(x) \rightarrow ans_P(x)$. 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 $g(s) = g(b_0b_1 \cdots b_{n-1}) = \{i | b_i = 1\}$. 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 $\mathbb{N}_n$ [77]. In this case, the growth function may consist of applying a greedy construction algorithm [45], 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" on while the search algorithm will act, and of course, indirectly moving in the image set $ans_P(x)$. 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 $s \in \mathcal{S}$ a set $\mathcal{N}(s) \subseteq \mathcal{S}$ 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* configuration $s' \in S$. In many situations, this property is self-evident and no explicit demonstration is required. It is important since even if having a valid representation (recall the definition above), it is necessary to guarantee that *a priori* 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 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 clauses 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 is in this case the objective function of the associated $\mathcal{NP}$ 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* [106]. 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 difference of the guiding function of the 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 interpreting 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 Figure 5.1.

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

```
Procedure Local-Search-Engine (current)
begin
    repeat
        new ← GenerateNeighbor(current);
        if ($F_g(new) \prec_{\mathcal{F}} F_g(current)$) then
            current ← new;
        endif
    until TerminationCriterion();
    return current;
end
```

Figure 5.1. A local search algorithm.

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 certain binomial distribution [189]. 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.

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*, $(k \geq 2)$ 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 Figure 5.1. 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 $\mathcal{NP}$–hard problems (e.g., see [62,199]). 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. [104]. 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 $\mathcal{NP}$–hard problem is contained in class PLS, since that would imply that $\mathcal{NP} = \text{co-}\mathcal{NP}$ [226], 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 [189]. The first property, *respect,* represents the exploitative 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 obliged 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 behaviour 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 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 to the best parents). A typical example of a blind recombination operator is the classical *Uniform crossover* [209]. 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 behaviour 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 behaviour (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 heuristic recombination operator focused in the first aspect, *Dynastically Optimal Recombination* (DOR) [43] must be mentioned. This operator explores the dynastic potential set (i.e., 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 A* algorithm in order to minimize the cost of traversing the dynastic potential. The goal of this operator is thus finding 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 [188]. These operators are based on generating an incomplete child using a non-heuristic procedure (e.g., the $\mathbf{RAR}_\omega$ operator [187]), 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) [162]. 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 [155].

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, being the term *crossover* a reasonable name conveying the low complexity (yet not always used in this context) of these. 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 parents (the larger the size of these pieces of information, the lower the number of them needed to complete the child) [46]. 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 Designing a Memetic Algorithm

In light of the above considerations, it is possible to provide a general template for a memetic algorithm. As mentioned in Section 2.3, this template is very similar to that of a local search procedure acting on a set of $|pop| \geq 2$ configurations. This is shown in Figure 5.2.

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 [130,208]. Another possibility, the Local-Search-Engine presented in Section 2.3 could be used as shown in Figure 5.3.

As to 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 the core of the memetic algorithm. Essentially, this procedure can be seen as a pipelined process comprising n_{op} stages. Each of these stages consists of taking $arity_{in}^{j}$ configurations from the previous stage, generating $arity_{out}^{j}$ new configurations by applying an operator op^{j}. This pipeline is restricted to have $arity_{in}^{1} = popsize$. The whole process is sketched in Figure 5.4.

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.

```
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
```

Figure 5.2. A population-based search algorithm.

```
Procedure GenerateInitialPopulation
begin
    Initialize $pop$ using EmptyPopulation();
    parfor $j \leftarrow 1$ to $popsize$ do
        $i \leftarrow$ GenerateRandomConfiguration();
        $i \leftarrow$ Local-Search-Engine ($i$);
        InsertInPopulation individual $i$ to $pop$;
    endparfor
    return $pop$
end
```

Figure 5.3. Injecting high-quality solutions in the initial population.

```
Procedure GenerateNewPopulation ($pop$)
begin
    $buffer^0 \leftarrow pop$;
    parfor $j \leftarrow 1$ to $n_{op}$ do
        Initialize $buffer^j$ using EmptyPopulation();
    endparfor
    parfor $j \leftarrow 1$ 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
    endparfor;
    return $buffer^{n_{op}}$
end
```

Figure 5.4. The pipelined GenerateNewPopulation procedure.

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, applying the latter after recombination is not as problematic as in GAs.

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 [194,202] 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 $|newpop|/posize$ 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 (e.g., [9,78]). As a general guideline, the comma strategy is usually regarded as less prone to stagnation, being the ratio $|newpop|/posize \simeq 6$ a common choice [8]. 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 [222]. 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 Figure 5.2, 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 [50]. If this measure falls below a predefined threshold, the population is considered at 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 [99] 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 keeping 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 Figure 5.5.

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

Figure 5.5. The RestartPopulation procedure.

The procedure shown in Figure 5.5 is also known as the *random-immigrant* strategy [38]. A different possibility is activating 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 the restarting). 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.

3 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 be illustrative of the practical impact of these optimization techniques.

3.1 Traditional $\mathcal{NP}$ Optimization Problems

Traditional $\mathcal{NP}$ Optimization problems constitute one of the most typical battlefields of MAs. A remarkable history of successes has been reported with respect to the application of MAs to $\mathcal{NP}$–hard problems such as the following: GRAPH PARTITIONING [18,19,139,142,143], MIN NUMBER PARTITIONING [14], MAX INDEPENDENT SET [2, 90,200], BIN-PACKING [195], MIN GRAPH COLORING [39,41,60,64], SET COVERING [11], SINGLE MACHINE SCHEDULING WITH SETUP-TIMES AND DUE-DATES [65,119,146], PARALLEL MACHINE SCHEDULING [33,35,135,148], MIN GENERALISED ASSIGNMENT [36], MULTIDIMENSIONAL KNAPSACK [12,45,77], NONLINEAR INTEGER PROGRAMMING [210], QUADRATIC ASSIGNMENT [17,29,137,141,142], SET PARTITIONING [120], and particularly on the MIN TRAVELLING SALESMAN PROBLEM [66,67,73,75,76,97,110,138, 142,156,189].

Regarding the theory of $\mathcal{NP}$-Completeness, most of them can be cited as *"classical"* as they appeared in Karp's notorious paper [108] on the reducibility of combinatorial problems. Remarkably, in most of them the authors claim that they have developed the best heuristic for the problem at hand. This is important since these problems have been addressed with several with different approaches from the combinatorial optimization toolbox and almost all general-purpose algorithmic techniques have been tested on them.

3.2 Other Combinatorial Optimization Problems

The MA paradigm is not limited to the above mentioned classical problems. There exist additional "non-classical" combinatorial optimization problems of similar or higher complexity in whose resolution MAs have revealed themselves as outstanding techniques. As an example of these problems, one can cite *partial shape matching* [176], *Kauffman NK Landscapes* [140], *spacecraft trajectory design* [48], *frequency allocation* [109], *multiperiod network design* [69], *degree-constrained minimum spanning tree problem* [190], *uncapacitated hub location* [1], *placement problems* [98,115,201], *vehicle routing* [101,102], *transportation problems* [71,170], *task allocation* [86], *maintenance scheduling* [25]–[27], *open shop scheduling* [34,63,124], *flowshop scheduling* [30,159,160], *project scheduling* [168,177,191], *warehouse scheduling* [216], *production planning* [57,149], *timetabling* [20–24,128,151,152,179,180,192], *rostering* [53,153], and *sport games scheduling* [40].

Obviously, this list is by no means complete since its purpose is simply to document the wide applicability of the approach for combinatorial optimization.

3.3 Machine Learning and Robotics

Machine learning and robotics are two closely related fields since the different tasks involved in the control of robots are commonly approached using artificial neural networks and/or classifier systems. MAs, generally cited as "genetic hybrids" have been used in both fields, i.e., in general optimization problems related to machine learning (e.g., the training of artificial neural networks), and in robotic applications. With respect to the former, MAs have been applied to *neural network training* [100,155,212,227], *pattern recognition* [3], *pattern classification* [113,145], and *analysis of time series* [173].

As to the application of MAs to robotics, work has been done in *reactive rulebase learning in mobile agents* [47], *path planning* [172,183,225], *manipulator motion planning* [197], *time optimal control* [31], etc.

3.4 Electronics and Engineering

Electronics and engineering are also two fields in which these methods have been actively used. For example, with regard to engineering problems, work has been done in the following areas: *structure optimization* [228], *system modeling* [215], *aeronautic design* [16,186], *trim loss minimization* [174], *traffic control* [205], and *power planning* [213]. As to practical applications in the field of electronics, the following list can illustrate the numerous areas in which these techniques have been utilized: *semiconductor manufacturing* [111], *circuit design* [83,87,220], *computer aided design* [13], *multilayered periodic strip grating* [7], *analogue network synthesis* [81], and *service restoration* [5].

3.5 Molecular Optimization Problems

We have selected this particular class of computational problems, involving nonlinear optimization issues, to help the reader to identify a common trend in the literature. Unfortunately, the authors continue referring to their technique as 'genetic', although they are closer in spirit to MAs [94].

The Caltech report that gave its name to the, at that time incipient, field of MAs [154] discussed a metaheuristic which can be viewed as a hybrid of GAs and SA developed with M.G. Norman in 1988. In recent years, several papers applied hybrids of GAS with SA or other methods to a variety of molecular optimization problems [10,49,54,59,68, 82,105,107,117,123,129,131,136,147,178,203,204,211,221,230,231]. Hybrid population approaches like this can hardly be catalogued as being 'genetic', but this denomination has appeared in previous work by Deaven and Ho [55] and then cited by J. Maddox in *Nature* [132]. Other fields of application include *cluster physics* [167]. Additional work has been done in [56,92,93,184,185,221]. Other evolutionary approaches to a variety of molecular problems can be found in: [59,89,91,134,144,193,214]. Their use for design problems is particularly appealing [37,107,223]. They have also been applied in protein design [58,118], probably due to energy landscape structure associated with proteins (see the discussion in [155] and the literature review in [94]).

3.6 Other Applications

In addition to the application areas described above, MAs have been also utilized in other fields such as, for example, *medicine* [84,85,217], *economics* [122,175], *oceanography* [161], *mathematics* [196,218,219], *imaging science and speech processing* [28,114,133,198,229], etc.

For further information about MA applications we suggest querying bibliographical databases or web browsers for the keywords *'memetic algorithms'* and *'hybrid genetic algorithm'*. We have tried to be illustrative rather than exhaustive, pointing out some selected references for well-known application areas. This means that, with high probability, many important contributions may have been inadvertently left out.

4 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 the other area. Our current work on the *MemePool Project* goes in that direction. Fourth, an important and pioneering group of MAs, that of *Scatter Search,* is challenging the rôle 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.

4.1 Lessons Learned

In 1998, Applegate, Bixby, Cook, and Chvatal established new breakthrough results for the MIN TSP. Interestingly enough, this work supports our view that MAs will have a central rôle as a problem solving methodology. For instance, this research team 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.[1] The approach, according to Bixby: "...*involves ideas from polyhedral combinatorics and combinatorial optimization, integer and linear programming, computer science data structures and*

[1] See: http://www.crpc.rice.edu/CRPC/newsArchive/tsp.html

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 of MAs, that of a synergy of different approaches. However, due to a comment by Johnson and McGeoch ([103], p. 301), we previously suggested that the connection with MAs might be stronger since there is an analogy with *multi-parent* recombination. In a recent unpublished manuscript, *"Finding Tours in the TSP"*, by the same authors (Bixby et al.), available from their web site, this suspicion has been now confirmed. They present results on running an optimal algorithm for solving the MIN WEIGHTED HAMILTONIAN CYCLE PROBLEM in a subgraph formed by the union of 25 Chained Lin-Kernighan tours. The approach consistently finds the optimal solution to the original MIN TSP instances with up to 4461 cities. They also attempted to apply this idea to an instance with 85,900 cities (the largest instance in TSPLIB) and from that experience they convinced themselves that it also works well for such large instances.

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, following by the dynastical-optimal recombination method the authors named *tour merging* gave a non-optimal tour of only 0.0002% excess 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.

4.2 Computational Complexity of Recombination Problems

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. We note that the dynastical optimal recombination, as it was the tour merging procedure just discussed, can lead to solving another $\mathcal{NP}$-hard problem of a smaller size. To try to avoid intractability of the new subproblem generated, we may wish to find interesting cases with worst-case polynomial time complexity for the generated subproblem, yet helpful towards the primary goal of solving the original optimization problem. Having these ideas in mind we will discuss two recently introduced computational complexity classes.

We will define the classes PMA and uPMA by referring to three analogous algorithms to the ones that define the class of *Polynomial Local Search* problems (PLS). These definitions (specially for PMA) are particularly dependent on an algorithm called *k-merger,* that will help to formalize the notion of *recombination of a set of k given solutions,* as generally used by most MAS (as well as other population approaches). The input of a *k*-merger algorithm is a set S_{par} of $k \geq 2$ *feasible* solutions, so

$S_{par} \subseteq sol_P(x)$ They can be informally called "parent" solutions and, if successful, the k-merger delivers as output *at least one* feasible solution. For the uPMA class the construction of the new solution is less restricted than for PMA. In general, recombination processes can be very complex with many side restrictions involving the detection, the preservation or avoidance, and the feasible combination of *features* already present in the parent solutions.

Definition 5.1 (Definition (uPMA)). *Let x be an instance of an optimization problem P. With $\mathcal{M}_P(S_{par}, x) \subseteq sol_P(x)$ we denote the set of all possible outputs that the* k-merger *algorithm can give if it receives as input the pair (S_{par}, x) for problem P. (Note that the set $\mathcal{M}_P(S_{par}, x)$ generalizes the idea of* dynastic potential *for more than two parents* [43]*).*

A recombination problem $P/\mathcal{M}$ belongs to uPMA if there exist three polynomial-time algorithms *p-starter, p'-evaluator*, and *k-merger* (where p, p' and k are integer numbers such that $p' \geq p \geq k \geq 2$) that satisfy the following properties:

- Given an input x (formally a string $\in \{0, 1\}^*$), the *p-starter* determines whether $x \in I_P$ and in this case produces a set of p different feasible solutions $\{y_1, y_2, \ldots, y_p\} \subseteq sol_P(x)$.
- Given an instance $x \in I_P$ and an input (formally a string $\in \{0, 1\}^*$), the *p'-evaluator* determines whether this input represents a set of feasible solutions, i.e. $\{y_1, y_2, \ldots, y_{p'}\} \subset sol_P(x)$ and in that case it computes the value of the objective function associated to each one of them, i.e. $m_P(y_j, x), \forall j = 1, \ldots, p'$.
- Given an instance $x \in I_P$ and a set of k feasible solutions $S_{par} \subseteq sol_P(x)$, the *k-merger* determines whether the set S_{par} is a *k-merger optimum,* and, if it is not, it outputs at least one feasible solution $y' \in \mathcal{M}_P(S_{par}, x)$ with strictly better value of m_P.

Analogously, the PMA class is more restricted since it embodies a particular type of recombination. For uPMA the type of recombination is implicit in the way the group neighborhood $\mathcal{M}$ is defined. However, the definition for PMA is still general enough to encompasses most of the recombination procedures used in practical population-based approaches.

Definition 5.2 (PMA). *A recombination problem $P/\mathcal{M}$ belongs to PMA if there exist three polynomial-time algorithms* p-starter, p'-evaluator, *and* k-merger *(where p, p' and k are integer numbers such that $p' \geq p \geq k \geq 2$), such that the* p-starter *and* p'-evaluator *satisfy the same properties required by the uPMA class but the* k-merger *is constrained to be of a particular type, i.e.:*

- Given an instance $x \in I_P$ and a set of k feasible solutions $S_{par} \subseteq sol_P(x)$, the *k-merger* determines whether the set S_{par} is a *k-merger optimum,* and, if it is not, it does the following:
  —For each $y \in S_{par}$, it solves n_1 polynomial-time decision problems $\{\Pi_1(y), \ldots, \Pi_{n_1}(y)\}$. Let D be a matrix of $k \times n_1$ Boolean coefficients formed by the output of all these decision problems, i.e. $D_{i,j} = \Pi_j(y_i)$.

—It creates a set of n_2 *constraints* C, such that C can be partitioned in two subsets, i.e. $C = C_{in} \cup C_{out}$. Each constraint $c \in C$ is represented by a predicate π_c such that its associated decision problem $\Pi_c(y)$ can be solved in polynomial-time for all $y \in sol_P(x)$. Any predicate π_c is a polynomial-time computable function that has as input the Boolean matrix D and the instance x. It is required that *at least one* predicate π_c^* to be a non-constant function of *at least two different elements of* S_{par}.

—It outputs at least one *offspring,* i.e., another feasible solution $y' \in \mathcal{M}_P(S_{par}, x)$ with strictly better value of m_P, (i.e. $m_P(y', x) < \max\{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a minimization problem, and $m_P(y', x) > \min\{m_P(y_1, x), m_P(y_2, x), \ldots, m_P(y_k, x)\}$ for a maximization problem) subject to

$$\max_{y' \in sol_P(x)} \left[\left(\sum_{(c \in C_{in}) \wedge \Pi_c(y')} w_c \right) - \left(\sum_{(c \in C_{out}) \wedge \Pi_c(y')} w_c \right) \right] \quad (1)$$

where w_c is an integer weight associated to constraint c.

Current research is being conducted to identify problems, and their associated recombination procedures, such that membership, in either PMA or uPMA, can be proved. It is also hoped that after some initial attempts on challenging problems completeness and reductions for the classes can be properly defined.

4.3 Exploiting Fixed-parameter Tractability Results

An interesting new avenue of research can be established by appropriately linking results from the theory of fixed-parameter tractability and the development of recombination algorithms. This is an avenue that goes both ways. On one direction, efficient, (i.e., polynomial-time), fixed-parameter algorithms can be used as *"out of the box"* tools to create efficient recombination procedures. They can be used as dynastical optimal recombination procedures or *k*-merger algorithms. On the opposite direction, since MAs are typically designed to deal with large instances and scale pretty well with their size, using both techniques together can produce *complete MAs* allowing to extend the benefits of fixed-parameter tractability. From a software engineering perspective, the combination is perfect both from code and algorithmic reuse.

A parameterized problem can be generally viewed as a problem that has as input two 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 $k \ll |x|$) that constitutes a *parameter.* For a maximization (resp. minimization) problem P, the induced language $L_P(param)$ is the parameterized language consisting of all pairs $\langle x, k \rangle$ where $x \in I_P$ and $OPT(x) \geq k$ (resp. $OPT(x) \leq k$). If there exists an algorithm solving $L_P(param)$ in time $O(f(k)|x|^\alpha)$, where $|x|$ is a measure of the instancesize, $f(k)$ 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 language recognition problem belongs to the computational complexity class FPT.

To illustrate on this topic, we can discuss one of the most emblematic FPT problems:

VERTEX COVER (fixED PARAMETER, DECISION VERSION)
Instance: A graph $G(V, E)$.
Parameter: A positive integer k.
Question: Is there a set $V' \subseteq V$, such that for every edge $(u, v) \in E$, at least u or v is a member of V' and $|V'| \leq k$?

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, Kanj, and Yia [32]. This method can solve the parameterized version of vertex cover in time $O(1.271^k k^2 + kn)$. However, using this method together with the speed-up method proposed by Neidermeier and Rossmanith [166], the problem can be solved in $O(1.271^k + n)$, i.e., linear in n.

The definition of the FPT class allows $f(k)$ to grow arbitrarily fast. For the parameterized version of PLANAR DOMINATING SET the best known algorithm has $f(k) = 11^k$, but for other problems, like VERTEX COVER this situation is not that dramatic. In addition, the new general method to improve FPT algorithms that run on time $O(q(k)\beta^k + p(n))$ (where p and q are polynomials and β is a small constant) has been developed by Niedermeier and Rossmanith [166]. Their method interleaves the reduction to a problem kernel and bounded search methods. If it is the case that β^k is the size of the bounded search tree and $q(k)$ is the size of the problem kernel, the new technique is able to get rid of the $q(k)$ factor allowing the algorithm to be $O(\beta^k + p(n))$. Another problem that belongs to the FPT class is:

HITTING SET FOR SIZE THREE SETS (fixED PARAMETER, DECISION VERSION)
Instance: A collection C of subsets of size three of a finite set S.
Parameter: A positive integer k.
Question: Is there a subset $S' \subseteq S$, with $|S'| \leq k$ which allows S' contain at least one element from each subset in C?

It can be seen as a generalization of the VERTEX COVER problem for hypergraphs, where there is an hyperedge between three vertices (instead of only two as in VERTEX COVER), and the task is to find a minimal subset of vertices covering all edges. Recently, Niedermeier and Rossmanith have presented an algorithm that solves this problem in time $O(2.311^k + n)$ [165]. We leave as an exercise to the reader the task of finding recombination operators that take advantage of this algorithm. We also leave as an exercise the following, "*is it possible to use the* $O(2.311^k + n)$ *to construct a k-merger algorithm for the optimization problem* MIN 3-HITTING SET *that satisfies the criteria required by the PMA and uPMA definitions?*"

4.4 Addition of negative knowledge

During the past five years, the addition of problem-dependent knowledge to enhance population-based approaches received a renewed interest among MAs researchers [2, 12,19,70,82,189,204]. Earlier papers had emphasized the relevance [79,125,154,207] of this issue, as well as Lawrence Davis from his numerous writings, also supported this idea when there was still not such a consensus, and many researchers were skeptical about their central role.

However, as many other things, the addition of problem-dependent, and instance-dependent knowledge can be addressed in different ways. Today, it is reasonable to continue finding new ways to incorporate *negative knowledge* in MAs. It is also challenging to find ways of extracting (or we may better say induce), "negative knowledge" from the current population of solutions. We quote M. Minsky [150]:

> We tend to think of knowledge in positive terms – and of experts as people who know what to do. But a 'negative' way to seem competent is, simply, never to make mistakes.

This is what heuristics do best, reducing the computational effort by radically changing the search space, so certainly there is no paradox here. Most heuristics, however, have a "positive bias". For instance, in [181] we can read:

> The strategy developed by Lin [126] for the TSP is to obtain several local optima and then identify edges that are common to all of them. These are then fixed, thus reducing the time to find more local optima. This idea is developed further in [72,127].

Curiously enough, although this strategy has been around for more than three decades it is hardly accepted by some researchers regardless all empirical evidence of the benefits of MAs that exploit, by using recombination, the correlation of highly evolved solutions [15,112,142,154,155,157].

Already recognized by R. Hofmann in 1993, there is a certain curious asymmetry in this strategy. We tend to think of common edges as pieces of "positive" information to be passed on between generations. However, after a few generations of an MA, a highly evolved (yet still diverse) population, might consistently avoid certain edges. *What is the population of agents "suggesting" about these edges ?* For instance, given an edge, we can define two tours as being equivalent if they both contain the edge, but also they can be defined equivalent *if both do not.* Hofmann recognized this duality and came with the denomination of *negative edge formae* for the latter induced equivalence class, that describes the set of tours that do not share this particular edge (see [95], p. 35).

At this point it is important to distinguish knowledge from *belief.* Formally, knowledge is generally defined as a *true belief,* for instance "agent a knows ϕ if a believes ϕ and ϕ is true". So a way of adding knowledge to the agents is to actually to attempt to *prove* things the agents believe.

For instance, if we are trying to solve an Euclidean 2-dimensional instance of the MIN TSP, it is always the case that the optimal tour must not have two crossing edges. This is a true fact that can be proven with a theorem. Since 2-opt moves can efficiently remove crossing edges with the aid of associate data structures and range queries, all agents in an MAs generally have it incorporated in their local search methods. However, it is also the case that other predicates, involving more than two edges, can be proved in all circumstances for the optimal tour (i.e., a predicate that is always true for all instances under consideration), yet not efficient associated local search methods have been found. This said, we envision that given an optimization with only one objective function, we might use methods from multi-objective MAs, using as auxiliary objective functions the number of violations of over a set of properties that the optimal solution should have. Research in this direction is also under way.

4.5 Guided Memetic Algorithms

In [181] we can read:

> As is common with heuristics, one can also argue for exactly the opposite idea. Once such common features are detected, they are *forbidden* rather than fixed. This justification is the following: If we fear that the global optimum is escaping us, this could be because our heuristic is "fooled" by this tempting features. Forbidding them could finally put us on the right track towards the global optimum. This is called *denial* in [206].

We also envision that future generations of MAs will work in at least two levels and two time scales. During the short-time scale, a set of agents would be searching in the search space associated to the problem. This is what most MAs do today. However, a long-time scale would *adapt the algorithms* associated with the agents. Here we encompass all of them, the individual search strategies, the recombination operators, etc. For instance, an MA that uses *guided local search* for an adaptive strategy has been proposed in [97]. Recombination operators with different(*rebel*, *obsequent*, and *conciliator behaviors*) were introduced in [14]. We think that these new promising directions need to be investigated, providing adaptive recombination mechanisms that take information of the progress of the current population. Moreover, it is intriguing the idea of using a combination of deterministic search procedures with a stochastic MA. The combination of both would help to systematically explore sub-regions of the search space and *guarantee* a coverage that currently is only left to some random procedure that aim to diversify the search (like the so-called *heavy mutation* procedures).

4.6 Modal Logics for MAs and Belief Search

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 coordinated as a formal paradigm. However, their achievements are well-recognized. From [121]:

> 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 of applied research is that it allows the introduction of increasingly more complex heuristics, unfortunately most of the time parameterized by *ad-hoc* numbers. 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* [61], as it has been recognized in the past yet it is hardly being mentioned in the current metaheuristics literature.

To illustrate how Belief Search can work in an MA setting, we will use an illustrative example. Let's assume that the formula $\mathbf{B}_a^i \phi$ has the following meaning *"agent i believes with strength (at least) a that ϕ is true"*, such that the strength values a are restricted to be rational numbers in [0,1]. Let's also suppose we accept as an axiom that from ψ being true we can deduce $\mathbf{B}_1^i\psi$ for all *i*. Now let's suppose that our agents are trying to solve a MIN TSP and that the particular instance being considered is Euclidean and two-dimensional. Let ϕ_k represent the proposition *"edge e_k is present in the optimum tour"* and let $\chi_{k,l}$ be true if edges e_k and e_l cross each other, and false otherwise. It can be proved that for such type of instances *"if edges e_k and e_l cross each other, then e_k and e_l can not both be present in the optimal tour"*. Then we can assume that this is known by all agents, and by the previous axiom we can deduce that agent 2 now believes $\mathbf{B}_1^2(\chi_{k,l} \rightarrow \neg(\phi_k \wedge \phi_l))$. Now let's suppose that agent 1 believes, with strength 0.4, that *"either edge e_k or e_l, but not both, is present in the optimal tour"*. We will represent this as $\mathbf{B}_{0.4}^1\phi_{k,l}$. We will not enter into the discussion of how that agent reached that belief and we take it as a fact. Now let us suppose that another agent believes, at a level 0.7 that $\chi_{k,l} \rightarrow \phi_{k,l}$, then we write $\mathbf{B}_{0.7}^3(\chi_{k,l} \rightarrow \phi_{k,l})$. Note that this kind of assumption confuses our common sense, since in general we do not see any relationship between the fact that two edges cross and that we can deduce that as a consequence one of them should be present in the optimum tour. However, note that agent 3 believes in this relationship (at a 0.7 level) for *a particular* pair of edges e_k and e_l. Now, what can we say about the *distributed belief* of this group of three agents ? How can we recombine this information ?

According to $\mathbf{PL}_n^{\otimes}$, a multi-agent epistemic logic recently introduced by Boldrin and Saffiotti, the opinions shared by different a set of *n* agents can be *recombined* in a distributed belief. Using $\mathbf{PL}_n^{\otimes}$ we can deduce $\mathbf{D}_{0.82}\phi_{k,l}$. The distributed belief about proposition $\phi_{k,l}$ is stronger than any individual belief about it, and is even stronger than what you would get if any agent would believe the three facts.

4.7 The MemePool Project: Our Meeting Place for Algorithm and Knowledge Reuse

As we have said before, we believe that these different research directions can blend together and make an outstanding contribution when used in combination.

One interesting example of these new MAs, is a recent work by Lamma et al. [116] applied to the field of diagnosing digital circuits. In their approach, they differentiate between genes and "*memes*". The latter group codes for the agents 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 reach 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 clear this point 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. They might be able to use it, but that is anecdotal at the moment. What it is important to remark is that algorithms like this one might be used *out-of-the-box* if a proper, world-wide based, algorithmic framework were created.

On the other hand, we remarked how results of logic programming and belief revision may help the current status of metaheuristics. The current situation where everybody comes with new names for the same basic techniques, and most contributions are just the addition of new parameters to guide the search, is a futile research direction. It may be possible, that belief search guided MAs can be a valid tool to help systematize the scenario. 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. Towards that end, we hope to convince researchers in logic programming to start to address these issues, challenging them with the task of guiding MAs for large-scale combinatorial optimization. The MemePool Project2 seems to be a good scenario for such an interdisciplinary and international enterprise. Whether it will work or not relies more on our ability to work cooperatively more than anything else.

BIBLIOGRAPHY

[1] H.S. Abdinnour (1998) A hybrid heuristic for the uncapacitated hub location problem. *European Journal of Operational Research,* **106**(2–3), 489–99.

[2] C.C. Aggarwal, J.B. Orlin and R.P. Tai (1997) Optimized crossover for the independent set problem. *Operations Research,* **45**(2), 226–234.

[3] J. Aguilar and A. Colmenares (1998) Resolution of pattern recognition problems using a hybrid genetic/random neural network learning algorithm. *Pattern Analysis and Applications,* **1**(1), 52–61.

[4] D. Aldous and U. Vazirani (1994) “Go with the winners” algorithms. In: *Proceedings of the 35th Annual Symposium on Foundations of Computer Science.* IEEE, Los Alamitos, CA, pp. 492–501.

[5] A. Augugliaro, L. Dusonchet and E. Riva-Sanseverino (1998) Service restoration in compensated distribution networks using a hybrid genetic algorithm. *Electric Power Systems Research,* **46**(1), 59–66.

[6] R. Axelrod and W.D. Hamilton (1981) The evolution of cooperation. *Science,* **211**(4489), 1390–1396.

[7] K. Aygun, D.S. Weile and E. Michielssen (1997) Design of multilayered periodic strip gratings by genetic algorithms. *Microwave and Optical Technology Letters,* **14**(2), 81–85.

[8] T. Bäck and F. Hoffmeister (1991) Adaptive search by evolutionary algorithms. In: W. Ebeling, M. Peschel and W. Weidlich (eds.), *Models of Selforganization in Complex Systems,* number 64 in Mathematical Research, Akademie-Verlag, pp. 17–21.

[9] Th. Bäck (1996) *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York.

[2] http://www.densis.fee.unicamp.br/~moscato/memepool.html

[10] M.J. Bayley, G. Jones, P. Willett and M.P. Williamson (1998) Genfold: A genetic algorithm for folding protein structures using NMR restraints. *Protein Science,* **7**(2), 491–499.

[11] J. Beasley and P.C. Chu (1996) A genetic algorithm for the set covering problem. *European Journal of Operational Research,* **94**(2), 393–404.

[12] J. Beasley and P.C. Chu (1998) A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics,* **4**, 63–86.

[13] B. Becker and R. Drechsler (1994) Ofdd based minimization of fixed polarity Reed-Muller expressions using hybrid genetic algorithms. In: *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processor.* IEEE, Los Alamitos, CA, pp. 106–110.

[14] R. Berretta and P. Moscato (1999) The number partitioning problem: An open challenge for evolutionary computation? In: D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization.* McGraw-Hill, pp. 261–278.

[15] K.D. Boese (1995) Cost versus Distance in the Traveling Salesman Problem. Technical Report TR-950018, UCLA CS Department.

[16] A.H.W. Bos (1998) Aircraft conceptual design by genetic/gradient-guided optimization. *Engineering Applications of Artificial Intelligence,* **11**(3), 377–382.

[17] D. Brown, C. Huntley and A. Spillane (1989) A parallel genetic heuristic for the quadratic assignment problem. In: J. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, pp. 406–415.

[18] T.N. Bui and B.R. Moon (1996) Genetic algorithm and graph partitioning. *IEEE Transactions on Computers,* **45**(7), 841–855.

[19] T.N. Bui and B.R. Moon (1998) GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **17**(3), 193–204.

[20] E.K. Burke, D.G. Elliman and R.F. Weare (1995) A hybrid genetic algorithm for highly constrained timetabling problems. In: *Proceedings of the Sixth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Francisco, CA, pp. 605–610.

[21] E.K. Burke, K.S. Jackson, J.H. Kingston and R.F. Weare (1997) Automated timetabling: The state of the art. *The Computer Journal*, **40**(9), 565–571.

[22] E.K. Burke and J.P. Newall (1997) A phased evolutionary approach for the timetable problem: An initial study. In: *Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference.* Springer-Verlag, Dunedin, New Zealand, pp. 1038–1041.

[23] E.K. Burke, J.P. Newall and R.F. Weare (1996) A memetic algorithm for university exam timetabling. In: E.K. Burke and P. Ross (eds.), *The Practice and Theory of Automated Timetabling,* volume 1153 of *Lecture Notes in Computer Science.* Springer-Verlag, pp. 241–250.

[24] E.K. Burke, J.P. Newall and R.F. Weare (1998) Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation,* **6**(1), 81–103.

[25] E.K. Burke and A.J. Smith (1997) A memetic algorithm for the maintenance scheduling problem. In: *Proceedings of the ICONIP/ANZIIS/ANNES '97 Conference.* Springer-Verlag, Dunedin, New Zealand, pp. 469–472.

[26] E.K. Burke and A.J. Smith (1999) A memetic algorithm to schedule grid maintenance. In: *Proceedings of the International Conference on Computational Intelligence for Modelling Control and Automation, Vienna: Evolutionary Computation and Fuzzy Logic for Intelligent Control, Knowledge Acquisition and Information Retrieval.* IOS Press 1999, pp. 122–127.

[27] E.K. Burke and A.J. Smith (1999) A multi-stage approach for the thermal generator maintenance scheduling problem. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE, Washington D.C., pp. 1085–1092.

[28] S. Cadieux, N. Tanizaki and T. Okamura (1997) Time efficient and robust 3-D brain image centering and realignment using hybrid genetic algorithm. In: *Proceedings of the 36th SICE Annual Conference.* IEEE, pp. 1279–1284.

[29] J. Carrizo, F.G. Tinetti and P. Moscato (1992) A computational ecology for the quadratic assignment problem. In: *Proceedings of the 21st Meeting on Informatics and Operations Research.* Buenos Aires, SADIO.

[30] S. Cavalieri and P. Gaiardelli (1998) Hybrid genetic algorithms for a multiple-objective scheduling problem. *Journal of Intelligent Manufacturing,* **9**(4), 361–367.

[31] N. Chaiyaratana and A.M.S. Zalzala (1999) Hybridisation of neural networks and genetic algorithms for time-optimal control. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE, Washington D.C., pp. 389–396.

[32] Jianer Chen, lyad A. Kanj and Weijia Jia (1999) Vertex cover: further observations and further improvements. In: *Proceedings of the 25th International Worksh. Graph-Theoretic Concepts in Computer Science,* number 1665 in *Lecture Notes in Computer Science.* Springer-Verlag, pp. 313–324.

[33] R. Cheng and M. Gen (1996) Parallel machine scheduling problems using memetic algorithms. In: *1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems,* Vol. 4. IEEE, New York, NY, pp. 2665–2670.

[34] R. Cheng, M. Gen and Y. Tsujimura (1999) A tutorial survey of job-shop scheduling problems using genetic algorithms, ii. hybrid genetic search strategies. *Computers & Industrial Engineering,* **37**(1–2), 51–55.

[35] R.W. Cheng and M. Gen (1997) Parallel machine scheduling problems using memetic algorithms. *Computers & Industrial Engineering,* **33**(3–4), 761–764.

[36] P.C. Chu and J. Beasley (1997) A genetic algorithm for the generalised assignment problem. *Computers & Operations Research,* **24**, 17–23.

[37] D.E. Clark and D.R. Westhead (1996) Evolutionary algorithms in computer-aided molecular design. *Journal of Computer-aided Molecular Design,* **10**(4), 337–358.

[38] H.G. Cobb and J.J. Grefenstette (1993) Genetic algorithms for tracking changing environments. In: S. Forrest (ed.), *Proceedings of the Fifth International*

Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 529–530.

[39] P.E. Coll, G.A. Durán and P. Moscato (1999) On worst-case and comparative analysis as design principles for efficient recombination operators: A graph coloring case study. In: D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization.* McGraw-Hill, pp. 279–294.

[40] D. Costa (1995) An evolutionary tabu search algorithm and the NHL scheduling problem. *INFOR,* **33**(3), 161–178.

[41] D. Costa, N. Dubuis and A. Hertz (1995) Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics,* **1**(1), 105–128.

[42] C. Cotta (1998) A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Communications,* **11**(3–4), 223–224.

[43] C. Cotta, E. Alba and J.M. Troya (1998) Utilising dynastically optimal form a recombination in hybrid genetic algorithms. In: A.E. Eiben, Th. Bäck, M. Schoenauer and H.-P. Schwefel (eds.), *Parallel Problem Solving From Nature V,* volume 1498 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, pp. 305–314.

[44] C. Cotta, E. Alba and J.M. Troya (1999) Stochastic reverse hillclimbing and iterated local search. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE, Washington D.C., pp. 1558–1565.

[45] C. Cotta and J.M. Troya (1998) A hybrid genetic algorithm for the 0–1 multiple knapsack problem. In: G.D. Smith, N.C. Steele and R.F. Albrecht (eds.), *Artificial Neural Nets and Genetic Algorithms 3.* Springer-Verlag, Wien New York, pp. 251–255.

[46] C. Cotta and J.M. Troya (2000) On the influence of the representation granularity in heuristic forma recombination. In: J. Carroll, E. Damiani, H. Haddad and D. Oppenheim (eds.), *ACM Symposium on Applied Computing 2000.* ACM Press, pp. 433–439.

[47] C. Cotta and J.M. Troya (2000) Using a hybrid evolutionary A* approach for learning reactive behaviors. In: S. Cagnoni et al. (eds.), *Real-World Applications of Evolutionary Computation,* volume 1803 *of Lecture Notes in Computer Science.* Springer-Verlag, Edinburgh, pp. 347–356.

[48] T. Crain, R. Bishop, W. Fowler and K. Rock (1999) Optimal interplanetary trajectory design via hybrid genetic algorithm/recursive quadratic program search. In: *Ninth AAS/AIAA Space Flight Mechanics Meeting.* Breckenridge CO, pp. 99–133.

[49] T. Dandekar and P. Argos (1996) Identifying the tertiary fold of small proteins with different topologies from sequence and secondary structure using the genetic algorithm and extended criteria specific for strand regions. *Journal of Molecular Biology,* **256**(3), 645–660.

[50] Y. Davidor and O. Ben-Kiki (1992) The interplay among the genetic algorithm operators: Information theory tools used in a holistic way. In: R. Männer and B. Manderick (eds.), *Parallel Problem Solving From Nature II.* Elsevier Science Publishers B.V., Amsterdam, pp. 75–84.

[51] L. Davis (1991) *Handbook of Genetic Algorithms.* Van Nostrand Reinhold Computer Library, New York.

[52] R. Dawkins (1976) *The Selfish Gene.* Clarendon Press, Oxford.

[53] P. de Causmaecker, G. van den Berghe and E.K. Burke (1999) Using tabu search as a local heuristic in a memetic algorithm for the nurse rostering problem. In: *Proceedings of the Thirteenth Conference on Quantitative Methods for Decision Making,* pages abstract only, poster presentation. Brussels, Belgium.

[54] P.A. de Souza, R. Garg and V.K. Garg (1998) Automation of the analysis of Mossbauer spectra. *Hyperfine Interactions,* **112**(1–4), 275–278.

[55] D.M. Deaven and K.O. Ho (1995) Molecular-geometry optimization with a genetic algorithm. *Physical Review Letters,* **75**(2), 288–291.

[56] D.M. Deaven, N. Tit, J.R. Morris and K.M. Ho (1996) Structural optimization of Lennard-Jones clusters by a genetic algorithm. *Chemical Physics Letters,* **256**(1–2), 195–200.

[57] N. Dellaert and J. Jeunet (2000) Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm. *International Journal of Production Research,* **38**(5), 1083–1099.

[58] J.R. Desjarlais and T.M. Handel (1995) New strategies in protein design. *Current Opinion in Biotechnology,* **6**(4), 460–466.

[59] R. Doll and M.A. VanHove (1996) Global optimization in LEED structure determination using genetic algorithms. *Surface Science,* **355**(1–3), L393–L398.

[60] R. Dorne and J.K. Hao (1998) A new genetic local search algorithm for graph coloring. In: A.E. Eiben, Th. Bäck, M. Schoenauer and H.-P. Schwefel (eds.), *Parallel Problem Solving From Nature V,* volume 1498 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, pp. 745–754.

[61] R. Englemore and T. Morgan (eds.) (1988) *Blackboard Systems.* Addison-Wesley.

[62] C. Ersoy and S.S. Panwar (1993) Topological design of interconnected LAN/MAN networks. *IEEE Journal on Selected Areas in Communications,* **11**(8), 1172–1182.

[63] J. Fang and Y. Xi (1997) A rolling horizon job shop rescheduling strategy in the dynamic environment. *International Journal of Advanced Manufacturing Technology,* **13**(3), 227–232.

[64] C. Fleurent and J.A. Ferland (1997) Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research,* **63,** 437–461.

[65] P.M. França, A.S. Mendes and P. Moscato (1999) Memetic algorithms to minimize tardiness on a single machine with sequence-dependent setup times. In: *Proceedings of the 5th International Conference of the Decision Sciences Institute, Athens, Greece,* pp. 1708–1710.

[66] B. Freisleben and P. Merz (1996) 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.* IEEE Press, pp. 616–621.

[67] B. Freisleben and P. Merz (1996) New genetic local search operators for the traveling salesman problem. In: H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel (eds.), *Parallel Problem Solving from Nature IV,* volume 1141 *of Lecture Notes in Computer Science.* Springer, pp. 890-900.

[68] R.T. Fu, K. Esfarjani, Y. Hashi, J. Wu, X. Sun and Y. Kawazoe (1997) Surface reconstruction of Si (001) by genetic algorithm and simulated annealing method. *Science Reports of The Research Institutes Tohoku University Series A-Physics Chemistry and Metallurgy,* **44**(1), 77–81.

[69] B.L. Garcia, P. Mahey and L.J. LeBlanc (1998) Iterative improvement methods for a multiperiod network design problem. *European Journal of Operational Research,* **110**(1), 150–165.

[70] M. Gen and R. Cheng (1997) *Genetic Algorithms and Engineering Design.* Wiley Series in Engineering Design and Automation. John Wiley & Sons (Sd).

[71] M. Gen, K. Ida and L. Yinzhen (1998) Bicriteria transportation problem by hybrid genetic algorithm. *Computers & Industrial Engineering,* **35**(1–2), 363–366.

[72] A.J. Goldstein and A.B. Lesk (1975) Common feature techniques for discrete optimization. *Comp. Sci. Tech. Report 27,* Bell. Tel. Labs, March.

[73] M. Gorges-Schleuter (1989) ASPARAGOS: An asynchronous parallel genetic optimization strategy. In: J. David Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers, pp. 422–427.

[74] M. Gorges-Schleuter (1991) Explicit parallelism of genetic algorithms through population structures. In: H.-P. Schwefel and R. Manner (eds.), *Parallel Problem Solving from Nature.* Springer-Verlag, pp. 150–159.

[75] M. Gorges-Schleuter (1991) *Genetic Algorithms and Population Structures—A Massively Parallel Algorithm.* PhD thesis, University of Dortmund, Germany.

[76] M. Gorges-Schleuter (1997) Asparagos96 and the Traveling Salesman Problem. In: T. Baeck, Z. Michalewicz and X. Yao (eds.), *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation. Indianapolis, USA.* IEEE Press, pp. 171–174.

[77] J. Gottlieb (2000) Permutation-based evolutionary algorithms for multidimensional knapsack problems. In: J. Carroll, E. Damiani, H. Haddad and D. Oppenheim (eds.), *ACM Symposium on Applied Computing 2000.* ACM Press, pp. 408–114.

[78] J. Gottlieb and T. Kruse (2000) Selection in evolutionary algorithms for the traveling salesman problem. In: J. Carroll, E. Damiani, H. Haddad and D. Oppenheim (eds.), *ACM Symposium on Applied Computing 2000.* ACM Press, pp. 415–421.

[79] J. J. Grefenstette (1987) Incorporating Problem Specific Knowledge into Genetic Algorithms. In: L. Davis (ed.), *Genetic Algorithms and Simulated Annealing, Research Notes in Artificial Intelligence.* Morgan Kaufmann Publishers, pp. 42–60.

[80] P. Grim (1997) The undecidability of the spatialized prisoner's dilemma. *Theory and Decision,* **42**(1), 53–80.

[81] J.B. Grimbleby (1999) Hybrid genetic algorithms for analogue network synthesis. In: *Proceedings of the 1999 Congress on Evolutionary Computation,* IEEE, Washington D.C., pp. 1781–1787.

[82] J.R. Gunn (1997) Sampling protein conformations using segment libraries and a genetic algorithm. *Journal of Chemical Physics,* **106**(10), 4270–4281.

[83] M. Guotian and L. Changhong (1999) Optimal design of the broadband stepped impedance transformer based on the hybrid genetic algorithm. *Journal of Xidian University,* **26**(1), 8–12.

[84] O.C.L. Haas, K.J. Burnham and J.A. Mills (1998) Optimization of beam orientation in radiotherapy using planar geometry. *Physics in Medicine and Biology,* **43**(8), 2179–2193.

[85] O.C.L. Haas, K.J. Burnham, J.A. Mills, C.R. Reeves and M.H. Fisher (1996) Hybrid genetic algorithms applied to beam orientation in radiotherapy. In: *Fourth European Congress on Intelligent Techniques and Soft Computing Proceedings,* Vol. 3. Verlag Mainz, Aachen, Germany, pp. 2050–2055.

[86] A.B. Hadj-Alouane, J.C. Bean and K.G. Murty (1999) A hybrid genetic/optimization algorithm for a task allocation problem. *Journal of Scheduling,* **2**(4).

[87] S.P. Harris and E.G. Ifeachor (1998) Automatic design of frequency sampling filters by hybrid genetic algorithm techniques. *IEEE Transactions on Signal Processing,* **46**(12), 3304–3314.

[88] W.E. Hart and R.K. Belew (1991) Optimizing an arbitrary function is hard for the genetic algorithm. In: R.K. Belew and L.B. Booker (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo CA, pp. 190–195.

[89] B. Hartke (1993) Global geometry optimization of clusters using genetic algorithms. *Journal of Physical Chemistry,* **97**(39), 9973–9976.

[90] M. Hifi (1997) A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems. *Journal of the Operational Research Society,* **48**(6), 612–622.

[91] R. Hirsch and C.C. Mullergoymann (1995) Fitting of diffusion-coefficients in a 3-compartment sustained-release drug formulation using a genetic algorithm. *International Journal of Pharmaceutics,* **120**(2), 229–234.

[92] K.M. Ho, A.A. Shvartsburg, B.C. Pan, Z.Y. Lu, C.Z. Wang, J.G. Wacker, J.L. Fye and M.F. Jarrold (1998) Structures of medium-sized silicon clusters. *Nature,* **392**(6676), 582–585.

[93] S. Hobday and R. Smith (1997) Optimisation of carbon cluster geometry using a genetic algorithm. *Journal of The Chemical Society-Faraday Transactions,* **93**(22), 3919–3926.

[94] R.J.W. Hodgson (2000) Memetic algorithms and the molecular geometry optimization problem. In: *Proceedings of the 2000 Congress on Evolutionary Computation.* IEEE Service Center, Piscataway, NJ, pp. 625–632.

[95] Reimar Hofmann (1993) Examinations on the algebra of genetic algorithms. Master's thesis, Technische Universität München, Institut fü Informatik.

[96] D.R. Hofstadter (1983) Computer tournaments of the prisoners-dilemma suggest how cooperation evolves. *Scientific American,* **248**(5), 16–23.

[97] D. Holstein and P. Muscat (1999) Memetic algorithms using guided local search: A case study. In: D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization.* McGraw-Hill, pp. 235–244.

[98] E. Hopper and B. Turton (1999) A genetic algorithm for a 2d industrial packing problem. *Computers & Industrial Engineering,* **37**(1–2), 375–378.

[99] M. Hulin (1997) An optimal stop criterion for genetic algorithms: A bayesian approach. In: Th. Bäck (ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA, pp. 135–143.

[100] T. Ichimura and Y. Kuriyama (1998) Learning of neural networks with parallel hybrid ga using a royal road function. In: *1998 IEEE International Joint Conference on Neural Networks,* Vol. 2, IEEE, New York, NY, pp. 1131–1136.

[101] J. Berger, M. Salois and R. Begin (1998) A hybrid genetic algorithm for the vehicle routing problem with time windows. In: R.E. Mercer and E. Neufeld (eds.), *Advances in Artificial Intelligence. 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence.* Springer-Verlag, Berlin, pp. 114–127.

[102] W.R. Jih and Y.J. Hsu (1999) Dynamic vehicle routing using hybrid genetic algorithms. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE, Washington D.C., pp. 453–458.

[103] D.S. Johnson and L.A. McGeoch (1997) The traveling salesman problem: A case study. In: E.H.L. Aarts and J.K. Lenstra (eds,), *Local Search in Combinatorial Optimization.* Wiley, Chichester, pp. 215–310.

[104] D.S. Johnson, C.H. Papadimitriou and M. Yannakakis (1988) How easy is local search? *Journal of Computers and System Sciences*, **37**, 79–100.

[105] G. Jones, P. Willett, R.C. Glen, A.R. Leach and R. Taylor (1997) Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology,* **267**(3), 727–748.

[106] T.C. Jones (1995) *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico.

[107] B.M. Kariuki, H. Serrano-Gonzalez, R.L. Johnston and K.D.M. Harris (1997) The application of a genetic algorithm for solving crystal structures from powder diffraction data. *Chemical Physics Letters*, **280**(3–4), 189–195.

[108] R.M. Karp (1972) Reducibility among combinatorial problems. In: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations.* Plenum, New York NY, pp. 85–103.

[109] I.E. Kassotakis, M.E. Markaki and A.V. Vasilakos (2000) A hybrid genetic approach for channel reuse in multiple access telecommunication networks. *IEEE Journal on Selected Areas in Communications*, **18**(2), 234–243.

[110] K. Katayama, H. Hirabayashi and H. Narihisa (1998) Performance analysis for crossover operators of genetic algorithm. *Transactions of the Institute of Electronics, Information and Communication Engineers,* **J81D-I**(6), 639–650.

[111] T.S. Kim and G.S. May (1999) Intelligent control of via formation by photosensitive BCB for MCM-L/D applications. *IEEE Transactions on Semiconductor Manufacturing,* **12**, 503–515.

[112] S. Kirkpatrick and G. Toulouse (1985) Configuration space analysis of traveling salesman problems. *J. Physique,* **46**, 1277–1292.

[113] K. Krishna and M. Narasimha-Murty (1999) Genetic k-means algorithm. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, **29**(3), 433–439.

[114] K. Krishna, K.R. Ramakrishnan and M.A.L. Thathachar (1997) Vector quantization using genetic k-means algorithm for image compression. In: *1997 International Conference on Information, Communications and Signal Processing,* Vol. 3, IEEE, New York, NY, pp. 1585–1587.

[115] R.M. Krzanowski and J. Raper (1999) Hybrid genetic algorithm for transmitter location in wireless networks. *Computers, Environment and Urban Systems,* **23**(5), 359–382.

[116] E. Lamma, L.M. Pereira and F. Riguzzi (2000) Multi-agent logic aided lamarckian learning. Technical Report DEIS-LIA-00-004, Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna (Italy), LIA Series no. 44 (submitted for publication).

[117] E. Landree, C. Collazo-Davila and L.D. Marks (1997) Multi-solution genetic algorithm approach to surface structure determination using direct methods. *Acta Crystallographica Section B—Structural Science*, **53**, 916–922.

[118] G.A. Lazar, J.R. Desjarlais and T.M. Handel (1997) De novo design of the hydrophobic core of ubiquitin. *Protein Science*, **6**(6), 1167–1178.

[119] C.Y. Lee (1994) Genetic algorithms for single machine job scheduling with common due date and symmetric penalties. *Journal of the Operations Research Society of Japan*, **37**(2), 83–95.

[120] D. Levine (1996) A parallel genetic algorithm for the set partitioning problem. In: I.H. Osman and J.P. Kelly (eds.), *Meta-Heuristics: Theory & Applications.* Kluwer Academic Publishers, pp. 23–35.

[121] H.R. Lewis and C.H. Papadimitriou (1998) *Elements of the Theory of Computation.* Prentice-Hall, Inc., Upper Saddle River, New Jersey.

[122] F. Li, R. Morgan and D. Williams (1996) Economic environmental dispatch made easy with hybrid genetic algorithms. In: *Proceedings of the International Conference on Electrical Engineering,* Vol. 2. Int. Acad. Publishers, Beijing, China, pp. 965–969.

[123] L.P. Li, T.A. Darden, S.J. Freedman, B.C. Furie, B. Furie, J.D. Baleja, H. Smith, R.G. Hiskey and L.G. Pedersen (1997) Refinement of the NMR solution structure of the gamma-carboxyglutamic acid domain of coagulation factor IX using molecular dynamics simulation with initial Ca2+ positions determined by a genetic algorithm. *Biochemistry,* **36**(8), 2132–2138.

[124] C.F. Liaw (2000) A hybrid genetic algorithm for the open shop scheduling problem. *European Journal of Oprational Research,* **124**(1), 28–42.

[125] G.E. Liepins and M.R. Hilliard (1987) Greedy Genetics. In: J.J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms and their Applications.* Lawrence Erlbaum Associates, Cambridge, MA, pp. 90–99.

[126] S. Lin (1965) Computer solutions of the traveling salesman problem. *Bell System Technical Journal,* **10,** 2245–2269.

[127] S. Lin and B. Kernighan (1973) An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research,* **21**, 498–516.

[128] S.E. Ling (1992) Integrating genetic algorithms with a prolog assignment program as a hybrid solution for a polytechnic timetable problem. In: *Parallel Problem Solvingfrom Nature II.* Elsevier Science Publisher B.V., pp. 321–329.

[129] D.M. Lorber and B.K. Shoichet (1998) Flexible ligand docking using conformational ensembles. *Protein Science,* **7**(4), 938–950.

[130] S.J, Louis, X. Yin and Z.Y. Yuan (1999) Multiple vehicle routing with time windows using genetic algorithms. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* Washington D.C., pp. 1804–1808. IEEE Neural Network Council—Evolutionary Programming Society—Institution of Electrical Engineers.

[131] A.L. MacKay (1995) Generalized crystallography. *THEOCHEM-Journal of Molecular Structure,* **336**(2–3), 293–303.

[132] J. Maddox (1995) Genetics helping molecular-dynamics. *Nature,* **376**(6537), 209–209.

[133] K.E. Mathias and L.D. Whitley (1994) Noisy function evaluation and the delta coding algorithm. In: *Proceedings of the SPIE—The International Society for Optical Engineering,* pp. 53–64.

[134] A.C.W. May and M.S. Johnson (1994) Protein-structure comparisons using a combination of a genetic algorithm, dynamic-programming and least-squares minimization. *Protein Engineering,* **7**(4), 475–485.

[135] A.S. Mendes, F.M. Muller, P.M. França and P. Moscato (1999) Comparing metaheuristic approaches for parallel machine scheduling problems with sequence-dependent setup times. In: *Proceedings of the 15th International Conference on CAD/CAM Robotics & Factories of the Future,* Aguas de Lindoia, Brazil.

[136] L.D. Merkle, G.B. Lament, G.H. Jr. Gates and R. Pachter (1996) Hybrid genetic algorithms for minimization of a polypeptide specific energy model. In: *Proceedings of 1996 IEEE International Conference on Evolutionary Computation.* IEEE, New York, NY, pp. 396–400.

[137] P. Merz and B. Freisleben (1997) A Genetic Local Search Approach to the Quadratic Assignment Problem. In: T. Bäck (ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA, pp. 465–172.

[138] P. Merz and B. Freisleben (1997) Genetic local search for the TSP: new results. In: *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation.* IEEE Press, pp. 159–164.

[139] P. Merz and B. Freisleben (1998) Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. In: A.E. Eiben, T. Back, M. Schoenauer and H.-P. Schwefel (eds.), *Parallel Problem Solvingfrom Nature V,* volume 1498 *of Lecture Notes in Computer Science.* Springer-Verlag, pp. 765–774.

[140] P. Merz and B. Freisleben (1998) On the effectiveness of evolutionary search in high-dimensional *NK*-landscapes. In: *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation.* IEEE Press, pp. 741–745.

[141] P. Merz and B. Freisleben (1999) A comparion of Memetic Algorithms, Tabu Search, and ant colonies for the quadratic assignment problem. In: *Proceedings of the 1999 Congress on Evolutionary Computation. Washington D.C.* IEEE Service Center, Piscataway, NJ, pp. 2063–2070.

[142] P. Merz and B. Freisleben (1999) Fitness landscapes and memetic algorithm design. In: D. Corne, M. Dorigo and F. Glover (eds.), *New Ideas in Optimization.* McGraw-Hill, pp. 245–260.

[143] P. Merz and B. Freisleben (2000) Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation,* **8**(1), 61–91.

[144] J.C. Meza, R.S. Judson, T.R. Faulkner and A.M. Treasurywala (1996) A comparison of a direct search method and a genetic algorithm for conformational searching. *Journal of Computational Chemistry,* **17**(9), 1142–1151.

[145] M. Mignotte, C. Collet, P. Pérez and P. Bouthemy (2000) Hybrid genetic optimization and statistical model based approach for the classification of shadow shapes in sonar imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* **22**(2), 129–141.

[146] D.M. Miller, H.C. Chen, J. Matson and Q. Liu (1999) A hybrid genetic algorithm for the single machine scheduling problem. *Journal of Heuristics*, **5**(4), 437–454.

[147] S.T. Miller, J.M. Hogle and D.J. Filman (1996) A genetic algorithm for the *ab initio* phasing of icosahedral viruses. *Acta Crystallographica Section D—Biological Crystallography,* **52**, 235–251.

[148] L. Min and W. Cheng (1998) Identical parallel machine scheduling problem for minimizing the makespan using genetic algorithm combined with simulated annealing. *Chinese Journal of Electronics,* **7**(4), 317–321.

[149] X.G. Ming and K.L. Mak (2000) A hybrid hopfield network–genetic algorithm approach to optimal process plan selection. *International Journal of Production Research,* **38**(8), 1823–1839.

[150] M. Minsky (1994) Negative expertise. *International Journal of Expert Systems,* **7**(1), 13–19.

[151] A. Monfroglio (1996) Hybrid genetic algorithms for timetabling. *International Journal of Intelligent Systems,* **11**(8), 477–523.

[152] A. Monfroglio (1996) Timetabling through constrained heuristic search and genetic algorithms. *Software—Practice and Experience,* **26**(3), 251–279.

[153] A. Montfroglio (1996) Hybrid genetic algorithms for a rostering problem. *Software—Practice and Experience,* **26**(7), 851–862.

[154] P. Moscato (1989) 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.

[155] P. Moscato (1993) 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.

[156] P. Moscato and M.G. Norman (1992) A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. In: M. Valero, E. Onate, M. Jane, J.L. Larriba and B. Suarez (eds.), *Parallel Computing and Transputer Applications.* IOS Press, Amsterdam, pp. 177–186.

[157] H. Mühlenbein (1991) Evolution in time and space—the parallel genetic algorithm. In: Gregory J.E. Rawlins (ed.), *Foundations of Genetic Algorithms.* Morgan Kaufmann Publishers, pp. 316–337.

[158] H. Mühlenbein M. Gorges-Schleuter and O. Krämer (1988) Evolution Algorithms in Combinatorial Optimization. *Parallel Computing,* **7**, 65–88.

[159] T. Murata and H. Ishibuchi (1994) Performance evaluation of genetic algorithms for flowshop scheduling problems. In: *Proceedings of the First IEEE Conference on Evolutionary Computation,* Vol. 2. IEEE, New York, NY, pp. 812–817.

[160] T. Murata, H. Ishibuchi and H. Tanaka (1996) Genetic algorithms for flowshop scheduling problems. *Computers & Industrial Engineering,* 30(4), 1061-1071.

[161] M. Musil, M.J. Wilmut and N.R. Chapman (1999) A hybrid simplex genetic algorithm for estimating geoacoustic parameters using matched-field inversion. *IEEE Journal of Oceanic Engineering,* **24**(3), 358–369.

[162] Y. Nagata and Sh. Kobayashi (1997) Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. In: Th. Bäck (ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms, East Lansing, EUA.* Morgan Kaufmann, San Mateo, CA, pp. 450–457.

[163] M. Nakamaru, H. Matsuda and Y. Iwasa (1998) The evolution of social interaction in lattice models. *Sociological Theory and Methods,* **12**(2), 149–162.

[164] M. Nakamaru, H. Nogami and Y. Iwasa (1998) Score-dependent fertility model for the evolution of cooperation in a lattice. *Journal of Theoretical Biology,* **194**(1), 101–124.

[165] R. Niedermeier and P. Rossmanith (2000) 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.

[166] R. Niedermeier and P. Rossmanith (2000) A general method to speed up fixed-parameter-tractable algorithms. *Information Processing Letters,* **73**, 125–129.

[167] J.A. Niesse and H.R. Mayne (1996) Global geometry optimization of atomic clusters using a modified genetic algorithm in space-fixed coordinates. *Journal of Chemical Physics,* **105**(11), 4700–1706.

[168] A.L. Nordstrom and S. Tufekci (1994) A genetic algorithm for the talent scheduling problem. *Computers & Operations-Research,* **21**(8), 927–940.

[169] M.G. Norman and P. Moscato (1991) A competitive and cooperative approach to complex combinatorial search. Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA, 1989. expanded version published at the *Proceedings of the 20th Informatics and Operations Research Meeting,* Buenos Aires (20th JAIIO), August pp. 3.15–3.29.

[170] A.G.N. Novaes, J.E.S. De-Cursi and O.D. Graciolli (2000) A continuous approach to the design of physical distribution systems. *Computers & Operations Research,* **27**(9), 877–893.

[171] M.A. Nowak and K. Sigmund (1998) Evolution of indirect reciprocity by image scoring. *Nature,* **393**(6685), 573–577.

[172] P. Osmera (1995) Hybrid and distributed genetic algorithms for motion control. In: V. Chundy and E. Kurekova (eds.), *Proceedings of the Fourth International Symposium on Measurement and Control in Robotics,* pp. 297–300.

[173] R. Ostermark (1999) A neuro-genetic algorithm for heteroskedastic time-series processes: empirical tests on global asset returns. *Soft Computing,* **3**(4), 206–220.

[174] R. Ostermark (1999) Solving a nonlinear non-convex trim loss problem with a genetic hybrid algorithm. *Computers & Operations Research,* **26**(6), 623–635.

[175] R. Ostermark (1999) Solving irregular econometric and mathematical optimization problems with a genetic hybrid algorithm. *Computational Economics,* **13**(2), 103–115.

[176] E. Ozcan and C.K. Mohan (1998) Steady state memetic algorithm for partial shape matching. In: V.W. Porto, N. Saravanan and D. Waagen (eds.), *Evolutionary Programming VII,* volume 1447 of *Lecture Notes in Computer Science.* Springer, Berlin, pp. 527–236.

[177] L. Ozdamar (1999) A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews),* **29**(1), 44–59.

[178] M.N. Pacey, X.Z. Wang, S.J. Haake and E.A. Patterson (1999) The application of evolutionary and maximum entropy algorithms to photoelastic spectral analysis. *Experimental Mechanics,* **39**(4), 265–273.

[179] B. Paechter, A. Cumming, M.G. Norman and H. Luchian Extensions to a Memetic timetabling system. In: E.K. Burke and P. Ross (eds.), *The Practice and Theory of Automated Timetabling,* volume 1153 of *Lecture Notes in Computer Science.* Springer Verlag, pp. 251–265.

[180] B. Paechter, R.C. Rankin and A. Cumming (1998) Improving a lecture timetabling system for university wide use. In: E.K. Burke and M. Carter (eds.),

The Practice and Theory of Automated Timetabling II, volume 1408 of *Lecture Notes in Computer Science.* Springer Verlag, pp. 156–165.

[181] C.H. Papadimitriou and K. Steiglitz (1982) *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[182] M. Peinado and T. Lengauer (1997) Parallel “go with the winners algorithms” in the LogP Model. In: IEEE Computer Society Press (ed.), *Proceedings of the 11th International Parallel Processing Symposium.* Los Alamitos, California, pp. 656–664.

[183] O.K. Pratihar, K. Deb and A. Ghosh (1999) Fuzzy-genetic algorithms and mobile robot navigation among static obstacles. In: *Proceedings of the 1999 Congress on Evolutionary Computation.* IEEE, Washington D.C., pp. 327–334.

[184] N. Pucello, M. Rosati, G. D’Agostino, F. Pisacane, V. Rosato and M. Celino (1997) Search of molecular ground state via genetic algorithm: Implementation on a hybrid SIMD-MIMD platform. *International Journal of Modern Physics C.* **8**(2), 239–252.

[185] W.J. Pullan (1997) Structure prediction of benzene clusters using a genetic algorithm. *Journal of Chemical Information and Computer Sciences,* **37**(6), 1189–1193.

[186] D. Quagliarella and A. Vicini (1998) Hybrid genetic algorithms as tools for complex optimisation problems. In: P. Blonda, M. Castellano and A. Petrosino (eds.), *New Trends in Fuzzy Logic II. Proceedings of the Second Italian Workshop on Fuzzy Logic.* World Scientific, Singapore, pp. 300–307.

[187] N.J. Radcliffe (1994) The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence,* **10**, 339–384.

[188] N.J. Radcliffe and P.D. Surry (1994) Fitness Variance of Formae and Performance Prediction. In: L.D. Whitley and M.D. Vose (eds.), *Proceedings of the Third Workshop on Foundations of Genetic Algorithms.* Morgan Kaufmann, San Francisco, pp. 51–72.

[189] N.J. Radcliffe and P.D. Surry (1994) Formal Memetic Algorithms. In: T. Fogarty (ed.), *Evolutionary Computing: AISB Workshop,* volume 865 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, pp. 1–16.

[190] G.R. Raidl and B.A. Julstron (2000) A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In: J. Carroll, E. Damiani, H. Haddad and D. Oppenheim (eds.), *ACM Symposium on Applied Computing 2000.* ACM Press, pp. 440^t45.

[191] E. Ramat, G. Venturini, C. Lente and M. Slimane (1997) Solving the multiple resource constrained project scheduling problem with a hybrid genetic algorithm. In: Th. Bäck (ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms.* Morgan Kaufmann, San Francisco CA, pp. 489–496.

[192] R.C. Rankin (1996) Automatic timetabling in practice. In: *Practice and Theory of Automated Timetabling. First International Conference. Selected Papers.* Springer-Verlag, Berlin, pp. 266–279.

[193] M.L. Raymer, P.C. Sanschagrin, W.F. Punch, S. Venkataraman, E.D. Goodman and L.A. Kuhn (1997) Predicting conserved water-mediated and polar ligand

interactions in proteins using a k-nearest-neighbors genetic algorithm. *Journal of Molecular Biology,* **265**(4), 445–464.

[194] I. Rechenberg (1973) *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog Verlag, Stuttgart.

[195] C. Reeves (1996) Hybrid genetic algorithms for bin-packing and related problems. *Annals of Operations Research,* **63**, 371–396.

[196] C. Reich (2000) Simulation if imprecise ordinary differential equations using evolutionary algorithms. In: J. Carroll, E. Damiani, H. Haddad and D. Oppenheim (eds.), *ACM Symposium on Applied Computing 2000.* ACM Press, pp. 428–432.

[197] M.A. Ridao, J. Riquelme, E.F. Camacho and M. Toro (1998) An evolutionary and local search algorithm for planning two manipulators motion. In: A.P. Del Pobil, J. Mira and M. Ali (eds.), *Tasks and Methods in Applied Artificial Intelligence,* volume 1416 of *Lecture Notes in Computer Science,* Springer-Verlag, Berlin Heidelberg, pp. 105–114.

[198] C.F. Ruff, S.W. Hughes and D.J. Hawkes (1999) Volume estimation from sparse planar images using deformable models. *Image and Vision Computing,* **17**(8), 559–565.

[199] S.M. Sait and H. Youssef (2000) *VLSI Design Automation: Theory and Practice.* McGraw-Hill Book Co. (copublished by IEEE), Europe.

[200] A. Sakamoto, X.Z. Liu and T. Shimamoto (1997) A genetic approach for maximum independent set problems. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences,* **E80A**(3), 551–556.

[201] V. Schnecke and O. Vornberger (1997) Hybrid genetic algorithms for constrained placement problems. *IEEE Transactions on Evolutionary Computation,* **1**(4), 266–277.

[202] H.-P. Schwefel (1984) Evolution strategies: a family of non-linear optimization techniques based on imitating some principles of natural evolution. *Annals of Operations Research,* **1**, 165–167.

[203] K. Shankland, W.I.F. David and T. Csoka (1997) Crystal structure determination from powder diffraction data by the application of a genetic algorithm. *Zeitschrift Fur Kristallographie,* **212**(8), 550–552.

[204] K. Shankland, W.I.F. David T. Csoka and L. McBride (1998) Structure solution of ibuprofen from powder diffraction data by the application of a genetic algorithm combined with prior conformational analysis. *International Journal of Pharmaceutics,* **165**(1), 117–126.

[205] D. Srinivasan, R.L. Cheu, Y.P. Poh and A.K.C. Ng (2000) Development of an intelligent technique for traffic network incident detection. *Engineering Applications of Artificial Intelligence,* **13**(3), 311–322.

[206] K. Steiglitz and P. Weiner (1968) Some improved algorithms for computer solution of the traveling salesman problem. In: *Proceedings of the Sixth Allerton Conference on Circuit and System Theory.* Urbana, Illinois, pp. 814–821.

[207] J.Y. Suh and Dirk Van Gucht (1987) Incorporating heuristic information into genetic search. In: J.J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms and their Applications,* Lawrence Erlbaum Associates. Cambridge, MA, pp. 100–107.

[208] P.D. Surry and N.J. Radcliffe (1996) Inoculation to initialise evolutionary search. In: T.C. Fogarty (ed.), *Evolutionary Computing: AISB Workshop,* number 1143 in *Lecture Notes in Computer Science.* Springer-Verlag, pp. 269–285.

[209] G. Syswerda (1989) Uniform crossover in genetic algorithms. In: J.D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms.* Morgan Kaufmann, San Mateo, CA, pp. 2–9.

[210] T. Taguchi, T. Yokota and M. Gen (1998) Reliability optimal design problem with interval coefficients using hybrid genetic algorithms. *Computers & Industrial Engineering,* **35**(1–2), 373–376.

[211] K.Y. Tam and R.G. Compton (1995) GAMATCH—a genetic algorithm-based program for indexing crystal faces. *Journal of Applied Crystallography,* **28**, 640–645.

[212] A.P. Topchy, O.A. Lebedko and V.V. Miagkikh (1996) Fast learning in multilayered networks by means of hybrid evolutionary and gradient algorithms. In: *Proceedings of International Conference on Evolutionary Computation and its Applications,* pp. 390–398.

[213] A.J. Urdaneta, J.F. Gómez, E. Sorrentino, L. Flores and R. Díaz (1999) A hybrid genetic algorithm for optimal reactive power planning based upon successive linear programming. *IEEE Transactions on Power Systems,* **14**(4), 1292–1298.

[214] A.H.C. vanKampen, C.S. Strom and L.M.C Buydens (1996) Legalization, penalty and repair functions for constraint handling in the genetic algorithm methodology. *Chemometrics and Intelligent Laboratory Systems,* **34**(1), 55–68.

[215] L. Wang and J. Yen (1999) Extracting fuzzy rules for system modeling using a hybrid of genetic algorithms and kalman filter. *Fuzzy Sets and Systems,* **101**(3), 353–362.

[216] J.P. Watson, S. Rana, L.D. Whitley and A.E. Howe (1999) The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling. *Journal of Scheduling,* **2**(2), 79–98.

[217] R. Wehrens, C. Lucasius, L. Buydens and G. Kateman (1993) HIPS, A hybrid self-adapting expert system for nuclear magnetic resonance spectrum interpretation using genetic algorithms. *Analytica Chimica ACTA,* **277**(2), 313–324.

[218] P. Wei and L.X. Cheng (1999) A hybrid genetic algorithm for function optimization. *Journal of Software,* **10**(8), 819–823.

[219] X. Wei and F. Kangling (2000) A hybrid genetic algorithm for global solution of nondifferentiable nonlinear function. *Control Theory & Applications,* **17**(2), 180–183.

[220] D.S. Weile and E. Michielssen (1999) Design of doubly periodic filter and polarizer structures using a hybridized genetic algorithm. *Radio Science,* **34**(1), 51–63.

[221] R.P. White, J.A. Niesse and H.R. Mayne (1998) A study of genetic algorithm approaches to global geometry optimization of aromatic hydrocarbon microclusters. *Journal of Chemical Physics,* **108**(5), 2208–2218.

[222] D. Whitley (1987) Using reproductive evaluation to improve genetic search and heuristic discovery. In: J.J. Grefenstette (ed.), *Proceedings of the Second International Conference on Genetic Algorithms and their Applications.* Lawrence Erlbaum Associates, Cambridge, MA, pp. 108–115.

[223] P. Willett (1995) Genetic algorithms in molecular recognition and design. *Trends in Biotechnology,* **13**(12), 516–521.

[224] D.H. Wolpert and W.G. Macready (1997) No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,* **1**(1), 67–82.

[225] J. Xiao and L. Zhang (1997) Adaptive evolutionary planner/navigator for mobile robots. *IEEE Transactions on Evolutionary Computation,* **1**(1), 18–28.

[226] M. Yannakakis (1997) Computational complexity. In: E.H.L. Aarts and J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization.* Wiley, Chichester, pp. 19–55.

[227] X. Yao (1993) Evolutionary artificial neural networks. *International Journal of Neural Systems,* **4**(3), 203–222.

[228] I.C. Yeh (1999) Hybrid genetic algorithms for optimization of truss structures. *Computer Aided Civil and Infrastructure Engineering,* **14**(3), 199–206.

[229] M. Yoneyama, H. Komori and S. Nakamura (1999) Estimation of impulse response of vocal tract using hybrid genetic algorithm—a case of only glottal source. *Journal of the Acoustical Society of Japan,* **55**(12), 821–830.

[230] C.R. Zacharias, M.R. Lemes and A.D. Pino (1998) Combining genetic algorithm and simulated annealing: a molecular geometry optimization study. *THEOCHEM—Journal of Molecular Structure,* **430**(29–39).

[231] M. Zwick, B. Lovell and J. Marsh (1996) Global optimization studies on the 1–d phase problem. *International Journal of General Systems,* **25**(1), 47–59.