A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues

Natalio Krasnogor and Jim Smith

***Abstract*—The combination of evolutionary algorithms with local search was named "memetic algorithms" (MAs) (Moscato, 1989). These methods are inspired by models of natural systems that combine the evolutionary adaptation of a population with individual learning within the lifetimes of its members. Additionally, MAs are inspired by Richard Dawkin's concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement (Dawkins, 1976). In the case of MA's, "memes" refer to the strategies (e.g., local refinement, perturbation, or constructive methods, etc.) that are employed to improve individuals. In this paper, we review some works on the application of MAs to well-known combinatorial optimization problems, and place them in a framework defined by a general syntactic model. This model provides us with a classification scheme based on a computable index D, which facilitates algorithmic comparisons and suggests areas for future research. Also, by having an abstract model for this class of metaheuristics, it is possible to explore their design space and better understand their behavior from a theoretical standpoint. We illustrate the theoretical and practical relevance of this model and taxonomy for MAs in the context of a discussion of important design issues that must be addressed to produce effective and efficient MAs.**

***Index Terms*—Design issues, evolutionary global–local search hybrids, memetic algorithms (MAs), model, taxonomy.**

I. Introduction

EVOLUTIONARY ALGORITHMS (EAs) are a class of search and optimization techniques that work on a principle inspired by nature: *Darwinian Evolution*. The concept of natural selection is captured in EAs. Specifically, solutions to a given problem are codified in so-called chromosomes. The evolution of chromosomes due to the action of crossover, mutation, and natural selection are simulated through computer code.

It is now well established that *pure* EAs are not well suited to fine tuning search in complex combinatorial spaces and that hybridization with other techniques can greatly improve the efficiency of search [3]–[6]. The combination of EAs with local search (LS) was named "memetic algorithms" (MAs) in [1]. MAs are extensions of EAs that apply separate processes to refine individuals, for example, improving their fitness by hill-climbing.

Manuscript received July 15, 2003; revised January 27, 2005.

N. Krasnogor is with School of Computer Science and Information Technology, University of Nottingham, Nottingham NG7 2RD, U.K. (e-mail: natalio.krasnogor@nottingham.ac.uk).

J. Smith is with the Faculty of Computing, Engineering and Mathematical Sciences, University of the West of England, Bristol, BS16 1QY England, U.K. (e-mail: james.smith@uwe.ac.uk).

Digital Object Identifier 10.1109/TEVC.2005.850260

These methods are inspired by models of adaptation in natural systems that combine the evolutionary adaptation of a population with individual learning within the lifetimes of its members. The choice of name is inspired by Richard Dawkins' concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement [2]. In the context of heuristic optimization, a meme is taken to represent a learning or development strategy. Thus, a memetic model of adaptation exhibits the plasticity of individuals that a strictly genetic model fails to capture.

In the literature, MAs have also been named hybrid genetic algorithms (GAs) (e.g., [7]–[9]), genetic local searchers (e.g., [10]), Lamarckian GAs (e.g., [11]), and Baldwinian GAs (e.g., [12]), etc. As noted above, they typically combine local search heuristics with the EAs' operators, but combinations with constructive heuristics or exact methods may also be considered within this class of algorithms. We adopt the name of MAs for this metaheuristic, because we think it encompasses all the major concepts involved by the other ones, and for better or worse has become the *de facto* standard, e.g., [13]–[15].

EAs and MAs have been applied in a number of different areas, for example, operational research and optimization, automatic programming, and machine and robot learning. They have also been used to study and optimize of models of economies, immune systems, ecologies, population genetics, and social systems, and the interaction between evolution and learning, to name but a few applications.

From an optimization point of view, MAs have been shown to be both more efficient (i.e., requiring orders of magnitude fewer evaluations to find optima) and more effective (i.e., identifying higher quality solutions) than traditional EAs for some problem domains. As a result, MAs are gaining wide acceptance, in particular, in well-known combinatorial optimization problems where large instances have been solved to optimality and where other metaheuristics have failed to produce comparable results (see for example [16] for a comparison of MAs against other approaches for the quadratic assignment problem).

II. Goals, Aims, and Methods

Despite the impressive results achieved by some MA practitioners, the process of designing effective and efficient MAs currently remains fairly ad hoc and is frequently hidden behind problem-specific details. This paper aims to begin the process of placing MA design on a sounder footing. In order to do this, we begin by providing some examples of MAs successfully applied to well-known combinatorial optimization problems, and draw out those differences which specifically arise from the hybridization of the underlying EA, as opposed to being design choices within the EA itself. These studies are exemplars, in the

sense that they represent a wide range of applications and algorithmic options for a MA.

The first goal is to define a syntactic model which enables a better understanding of the interplay between the different component parts of an MA. A syntactic model is devoid of the semantic intricacies of each application domain and hence exposes the bare bones of this metaheuristic to scrutiny. This model should be able to represent the many different parts that compose an MA, determine their roles and interrelations.

With such a model, we can construct a taxonomy of MAs, the second goal of this paper. This taxonomy is of practical and theoretical relevance. It will allow for more sensible and fair comparisons of approaches and experimental designs. At the same time, it will provide a conceptual framework to deal with more difficult questions about the general behavior of MAs. Moreover, it will suggest directions of innovation in the design and development of MAs.

Finally, by having a syntactic model and a taxonomy, the process of more clearly identifying which of the many components (and interactions) of these complex algorithms relate to which of these design issues should be facilitated.

The rest of this paper is organized as follows. In Section III, we motivate our definition of the class of metaheuristics under consideration, and give examples of the type of design issues that have motivated this study. This is followed in Section IV by a review of some applications of MAs to well-known problems in combinatorial optimization and bioinformatics. Section V presents a syntax-only model for MAs and a taxonomy of possible architectures for these metaheuristics is given in Section VI. In Section VII, we return to the discussion of design issues, showing how some of these can be aided by the insights given by our model. Finally, we conclude with a discussion and conclusions in Section VIII.

III. Background

A. Defining the Subject of Study

In order to be able to define a syntactic model and taxonomy, we must first clarify what we mean by an MA. It has been argued that the success of MAs is due to the tradeoff between the exploration abilities of the EA, and the exploitation abilities of the local search used. The well-known results of MAs over multistart local search (MSLS) [17] and greedy randomized adaptive search procedure (GRASP) [8] suggest that, by transferring information between different runs of the local search (by means of genetic operators) the MA is capable of performing a much more efficient search. In this light, MAs have been frequently described as *genetic local search* which might be thought as the following process [18]:

> In each generation of GA, apply the LS operator to all solutions in the offspring population, before applying the selection operator.

Although many MAs indeed use this formula this is a somewhat restrictive view of MAs, and we will show in the following sections that many other ways have been used to hybridize EAs with LS with impressive results.

In [19], the authors present an algebraic formalization of MAs. In their approach, an MA is a very special case of GA where just one period of local search is performed. As we will show in the following sections, MAs are used in a *plethora* of alternative ways and not just in the way the formalism introduced in [19] suggests.

It has recently been argued by Moscato that the class of MAs should be extended to contain not only "EA-based MAs," but effectively include any population-based approach based on a "k-merger" operator to combine information from solutions [13], creating a class of algorithms called the *polynomial merger algorithm* (PMA). However, PMA ignores mutation and selection as important components of the evolutionary metaheuristic. Rather, it focuses exclusively on recombination, or it is more general form, the "k-merger" operator. Therefore, we do not use this definition here, as we feel that it is both restrictive (in that it precludes the possibility of EAs which do not use recombination), and also so broad that it encompasses such a wide range of algorithms as to make analysis difficult.

As the limits of "what is" and "what is not" an MA are stretched, it becomes more and more difficult to assess the benefit of each particular component of the metaheuristic in search or optimization. *A priori* formalizations such as [13] and [19] inevitably leave out many demonstrably successful MAs and can seriously limit analysis and generalization of the (already complex) behavior of MAs. Our intention is to provide an *a posteriori* model of MAs, using *algorithms* as data; that is, applications of MAs that have been proven successful. It will be designed in such a way to encompass those algorithms. Thus, we use a commonly accepted definition, which may be summarized as [20]:

> An MA is an EA that includes one or more local search phases within its evolutionary cycle.

While this definition clearly limits the scope of our study, it does not curtail the range of algorithms that can fit this scope. As with any formal model and taxonomy, ours will have its own "outsiders," but hopefully they will be less numerous than those left aside by [13] and [19]. The extension of our model to other population-based metaheuristics is being considered in a separate paper.

Finally, we should note that we have restricted the survey part of this paper to MAs approaches for single-objective combinatorial optimization problems (as opposed to multiobjective or numerical optimization problems). This is not because MAs are unsuited to these domains—they have been very successfully applied to the fields of multiobjective optimization (see, e.g., [21]–[24], an extensive bibliography can be found in [25]), and numerical optimization (see, e.g., [26]–[32]). Rather, the reason for this omission is partly practical, to do with the space this large field would demand. It is also partly because we wish to introduce our ideas in the context of the simple algorithm `Standard_Local_Search(...)`, where it is straightforward to define a neighborhood, improvement, and the concept of local optimality. When we consider multiobjective problems, the whole concept of optimality becomes clouded by the tradeoffs between objectives, and dominance relations are usually preferred. Similarly, in the case of numerical optimization, the concept of local optimality is clouded by the difficulty, in the absence of derivative information, of knowing when a solution is truly locally optimal, as opposed to say, a point, a very small

distance away. Nevertheless, it is worth stressing that the issues cloud the exposition, rather than invalidate the concept of "schedulers" which leads to our syntactic model and taxonomy, and the subsequent design guidelines which can equally well be applied in these more complex domains.

B. Design Issues for MAs

Having provided a fairly broad-brush definition of the class of metaheuristics that we are concerned with, it is still vital to note that the design of "competent" [33] MAs raises a number of important issues which must be addressed by the practitioner.

Perhaps the foremost of these issues may be stated as:

"What is the best tradeoff between local search and the global search provided by evolution?"

This leads naturally to questions such as the following.

- Where and when should local search be applied within the evolutionary cycle?
- Which individuals in the population should be improved by local search, and how should they be chosen?
- How much computational effort should be allocated to each local search?
- How can the genetic operators best be integrated with local search in order to achieve a synergistic effect?

As we will see in the following sections, there are a host of possible answers to these questions, and it is important to use both empirical experience and theoretical reasoning in the search for answers. The aim of our syntactic model is to provide a sound basis for understanding and comparing the effects of different schemes. The use of a formal model aids in this by making some of the design choices more explicit, and by providing a means of comparing the existing MA literature with the (far broader) body of research into EAs.

Similarly, while theoretical understanding of the interplay between local and global search is much less developed than that of "pure" EAs, it is possible to look in that literature for tools and concepts that may aid in the design of competent MAs, for example:

- Is a Baldwinian or Lamarckian model of improvement to be preferred?
- What fitness landscape(s) does the population of the MA operate on?
- What local optima are the MAs operating with?
- How can we engineer MAs that efficiently traverse large neutral plateaus and avoid deep local optima?

IV. Some Example Applications of MAs in Optimization and Search

In this section, we will briefly comment on the use of MAs on different combinatorial optimization problems and adaptive landscapes. Applications to *traveling salesman problem* (TSP), *quadratic assignment problem* (QAP), *binary quadratic programming* (BQP), *minimum graph coloring* (MGC), and *protein structure prediction problem* (PSP) will be reviewed.

This section does not pretend to be an exhaustive bibliography survey, but rather a gallery of well-known applications of MAs from which some architectural and design conclusions might be drawn. In [34], a comprehensive bibliography can be found.

For the definition of the problems, the notation in [35] will be used. The reader interested in the complexity and approximability results of those problems is referred to the previous reference. The pseudocode used to illustrate the different algorithms is shown as used by the respective authors, with only some minor changes made for the sake of clarity.

In [36], a "standard" local search algorithm is defined in terms of a local search problem. Because this standard algorithm is implicit in many MAs, we repeat it here.

Standard_Local_Search(x)
Begin
  produce a starting solution s
  to problem instance x;
  Repeat Until (locally optimal) **Do**
    using s and x generate the next
    neighbor $n_{x,s}$;
    If ($n_{x,s}$ is better than s) **Then**
      $s := n_{x,s}$;
    Fi
  Od
End.

Algorithm Standard_Local_Search(...) captures the intuitive notion of searching a neighborhood as a means of identifying a better solution. It does not specify tie-breaking policies, neighborhood structure, etc.

This algorithm uses a "greedy" rather than a "steepest" policy, i.e., it accepts the first better neighbor that it finds. In general, a given solution might have several better neighbors, and the rule that assigns one of the (potentially many) better neighbors to a solution is called a *pivot rule*. The selection of the pivot rule or rules to use in a given instantiation of the standard local search algorithm has tremendous impact on the complexity of the search and potentially in the quality of the solutions explored.

Note also that the algorithm above implies that local search continues until a local optima is found. This may take a long time, and in the continuous domain proof of local optimality may be decidedly nontrivial. Many of the local search procedures embedded within the MAs in the literature are not standard in this sense, that is, they usually perform a shorter "truncated" local search.

A. MAs for the TSP

The TSP is one of the most studied combinatorial optimization problems. It is defined by the following.

Traveling Salesman Problem
Instance: A set C of m cities, and for each pair of cities $c_i, c_j \in C$ a distance $d(c_i, c_j) \in \mathbf{N}$.
Solution: A tour of C, i.e., a permutation $\pi : [1 \dots m] \mapsto [1 \dots m]$.
Measure: The length of the tour, i.e., $d(\pi) = d(\{c_{\pi(m)}, c_{\pi(1)}\}) + \sum_{i=1}^{i=m-1} d(\{c_{\pi(i)}, c_{\pi(i+1)}\})$.
Aim: minimum length tour $\pi^* : \forall \pi \neq \pi^* \; d(\pi) > d(\pi^*)$.

In [37], a short review on early MAs for the TSP is presented, where an MA was defined by the following skeleton code.

```
Genetic_Local_Search(P ∈ S^l)
Begin
  /* λ, μ, m ≥ 1 */
  For i := 1 To μ Do
    Iterative_Improvement (s_i);
  Od
  stop_criterion:= false
  While (¬ stop_criterion) Do
    P' := ∅;
    For i := 1 To λ Do
      /* Mate */
      M_i ∈ P^m;
      /* Recombine */
      s_i ∈ H_m(M_i);
      Iterative_Improvement (s_i);
      P' := P' ∪ {s_i};
    Od
    /* Select */
    P :∈ (P ∪ P')^μ;
    evaluate stop_criterion
  Od
End.
```

Here, we can regard Iterative_Improvement(...) as a particular instantiation of Standard_Local_Search(...), and appropriate code should be used to initialize the population, mate solutions, and select the next generation. Note that the mutation stage was replaced by the local search procedure. Also, a $(\mu + \lambda)$ selection strategy was applied. The use of local search and the absence of mutation is a clear difference between Genetic_Local_Search(...) and standard EAs.

In [37], early works on the application of MAs to the TSP were commented on. Those works used different instantiations of the above skeleton to produce near-optimal solution for small instances of the problem. Although the results were not definitive, they were very encouraging, and many of the following applications of MAs to the TSP (and also to other NPO problems) were inspired by those early works.

In [38], the MA GLS_Based_Memetic_Algorithm(...) is used which has several nonstandard features. For details, the reader is referred to [13] and [38]. We are interested here in remarking the two important differences with the MA Genetic_Local_Search(...) shown previously. In this MA, the local search procedure is used after the application of each of the genetic operators and not only once in every iteration of the EA. These two metaheuristics differ also in that in the last case a clear distinction is made between mutations and local search. In [38], the local search used is based on the powerful guided local search (GLS) metaheuristic [39]. This algorithm was compared against MSLS, GLS, and a second MA, where the local search engine was the same basic move used by GLS without the guiding strategy. In this paper, results were presented from experiments using instances taken from TSPLIB [40] and fractal instances [41]. In no case was the MSLS able to achieve an optimal tour unlike the other three approaches. Out of 31 instances tested, the GLS_Based_Memetic_Algorithm(...) solved 24 to optimality, MSLS 0, MA with simple local search 10, and GLS 16. It is interesting to note that the paper was not intended as a "better than" paper but rather as a pedagogical paper, where the MAs were exposed as a new metaheuristic in optimization.

```
GLS_Based_Memetic_Algorithm
Begin
  Initialize population;
  For i := 1 To sizeOf(population) Do
    individual := population_i;
    individual := Local − Search − Engine(individual);
    Evaluate(individual);
  Od
  Repeat Until (termination_condition)
Do
    For j := 1 To #recombinations Do.
      selectToMerge a set S_par ⊆ population;
      offspring = Recombine(S_par, x);
      offspring = Local − Search− Engine(offspring);
      Evaluate(offspring);
      Add offspring to population;
    Od
    For j := 1 To #mutations Do
      selectToMutate an individual in population;
      Mutate(individual);
      individual = Local − Search−; Engine(individual)
      Evaluate(individual);
      Add individual to population;
    Od
    population=SelectPop(population);
    If (population has converged) Then
      population=RestartPop(population);
    Fi
  Od
End.
```

Merz and Freisleben in [42]–[44] show many different combinations of local search and genetic search for the TSP [in both its symmetric (STSP) and asymmetric (ATSP) versions], while defining purpose-specific crossover and mutation operators. In [42], the following code was used to conduct the simulations.

```
STSP-GA
Begin
  Initialize pop P with
Nearest − Neighbor(...);
  For i := 1 To popsize(P) Do
    Lin − Kernighan − Opt(individual_i), i ∈ P;
```

```
  Od
  Repeat Until (converged) Do
    For i := 0 To #crossover Do
      Select two parents i_a, i_b ∈ P
      randomly;
      i_c = DPX − STSP(i_a, i_b);
      Lin-Kernighan-Opt (i_c);
      With probability m_p do
      Mutation-STSP (i_c);
      Replace an individual of P by i_c;
    Od
  Od
End.
```

In this pseudocode, the authors employ specialized crossover and mutation operators for the TSP (and a similar algorithm for the ATSP). As in previous examples, the initial population is a set of local optima, in this case, with respect to $\texttt{Lin}-\texttt{Kernighan}-\texttt{Opt}(\ldots)$. In this case, the LK heuristic is also applied to the results of crossover and mutation. The authors motivate this, saying:

> ...and let a GA operate on the set of local optima to determine the global optimum.

However, they also note that this can lead to a disastrous loss of diversity, which prompts their use of a selection strategy which is neither a $(\mu+\lambda)$ nor a (μ,λ) but a hybrid between the two, whereby the new offspring replaces the most similar member of the population, (subject to elitism). As the authors remark, the *large step Markov chains* and *iterated-Lin-Kernighan* techniques are special cases of their algorithm.

In [44], the authors change their optimization scheme to one similar to GLS_Based_Memetic_Algorithm(...), which has a more traditional mutation and selection scheme and in [43] they use the same scheme as $\texttt{STSP}-\texttt{GA}(\ldots)$ but after finalization of the GA run, postprocessing by means of local search is performed.

It is important to notice that Merz and Freisleben's MAs are perhaps the most successful metaheuristics for TSP and ATSP, and a predecessor of the schemes described was the winning algorithm of the *First International Contest on Evolutionary Optimization*.

In [45], Nagata and Kobayashi described a powerful MA with an intelligent crossover, in which the local searcher is embedded in the genetic operator. The authors of [46] describe a detailed study of Nagata and Kobayashi's work, and relate it to the local searcher used by Merz and Freisleben.

B. MAs for the QAP

The QAP is found in the core of many practical problems such as facility location, architectural design, VLSI optimization, etc. Also, the TSP and GP can be recast as special cases of QAP. The problem is formally defined as the following.

Quadratic Assignment Problem
Instance: A,B matrices of $n \times n$.
Solution: A permutation $\pi : [1 \ldots m] \mapsto [1 \ldots m]$.
Measure: The cost of the permutation, i.e., $C(\pi) = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} a_{i,j} \cdot b_{\pi(i),\pi(j)}$
Aim: Minimun cost permutation $\pi^* : \forall \pi \neq \pi^* C(\pi) > C(\pi^*)$.

Because of the nature of QAP, it is difficult to treat with exact methods, and many heuristics and metaheuristics have been used to solve it. In this section, we will briefly comment on the application of MAs to the QAP.

In [9], the following MA described as a "hybrid GA metaheuristic" is proposed.

```
Genetic_Hybrid_Algorithm(H_1, H_2)
Begin
  P := ∅;
  For i := 1 To m Do
    generate a random permutation p;
    Add H_1(p) to P;
  Od
  Sort P;
  For i := 1 To number_of_generations Do
    For j := 1 To num_offspring_per_
    generation Do
      select two parents p_1, p_2 from P;
      child := crossover(p_1, p_2 2);
      Add H_2(child) to P;
    Od
    Sort P;
    Cull(P, num_of fspring_per_generation);
  Od
  Return the best p ∈ P;
End.
```

In the code shown above, H1(...) and H2(...) are initialization and improvement heuristics, respectively. In particular, the authors reports on experiments where H2(...) is a tabu search (TS) heuristic. At the time that paper was written, their MA was one of the best heuristics available (in terms of solution quality for standard test instances).

It is interesting to remark that as in Genetic_Local_Search(...)and GLS_Based_Memetic_Algorithm(...), the GA is seeded with a high-quality initial population, which is the output of an initial local search strategy, H_1 (TS in this case). Again, we find that the selection strategy, represented by Cull(...), is a $(\mu+\lambda)$ strategy as in the previous MAs. The authors further increase the selection pressure by using a mating selection strategy. As in Genetic_Local_Search(...), no explicit mutation strategy is used: Fleurent and Ferland regard H1(...) and H2(...) as mutations that are applied with a probability 1. As in Genetic_Local_Search(...), the optimization step is applied only to the newly generated individual, that is, the output of the crossover stage.

In [16], results were reported which are improvements to those in the paper previously commented, and for other metaheuristics for *QAP*. The sketch of the algorithm used is the following.

```
QAP_MA
Begin
  Initialize population P;
  For i := 1 To sizeof(P) Do
    individual := P_i;
    individual:=Local_Search(individual);
  Od
  Repeat Until (terminate=True) Do
    For i := 1 To #recombinations Do
      Select two parents i_a, i_b ∈ P
      randomly;
      i_c := Recombine(i_a, i_b);
      i_c := Local_Search(i_c);
      Add individual i_c to P;
    Od
    P := Select(P);
    If (Pconverged) Then
      For i := 1 To sizeof(P), i ≠ index(Best)
Do
        individual := P_i;
        individual:=Local_Search(Mutate
        (individual));
      Od
    Fi
  Od
End.
```

Regardless of the new representation and crossover on which the MA relies to perform its search, it should be particularly noted that Mutation(...) is applied only when a diversity crisis arises, and immediately after mutating a solution of the population a new local search improvement is performed. Because the selection strategy is again a $(\mu + \lambda)$ strategy, it may be the case that an old individual, i.e., one that survived many generations, goes through local search many times unlike in Genetic_Hybrid_Algorithm(...).

In this case, the initial population is obtained by the use of the local search engine. As a marginal comment, we can mention that the local search procedure employed was a variant of $2-Opt(\ldots)$ also known as the *pairwise interchange heuristic*.

C. MAs for the BQP

Binary quadratic programming is defined by the following.

Binary Quadratic Programming Problem
Instance: Symmetric rational $n \times n$ matrix $Q = (q_{i,j})$.
Solution: Binary vector of x of length n.
Measure: The benefit of x, i.e,
$f(x) = x^t Q x = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} q_{i,j} \cdot x_i \cdot x_j$
Aim: Maximum benefit solution $x^* : \forall x \neq x^* \; f(x) < f(x^*)$

As well as being a well-known *NP-Hard* problem, BQP has many applications, i.e., financial analysis, CAD problems, machine scheduling, etc. In [47], the authors used an MA with the same architecture as in QAP_MA(...) but tailored for BQP, and they were able to improve over previous approaches based on TS and simulated annealing (SA). They also were able to find new best solutions for instances in the ORLIB [48].

D. MAs for the MGC

The MGC is one of the most studied problems in graph theory, with many applications in the area of scheduling and timetabling. Its definition is the following.

Graph Coloring
Instance: Graph $G = (V, E)$.
Solution: S, a coloring of G, i.e.,
a parition of V into disjoint sets $v_1, \ldots, v_k$
such that each v_i is an independent set for G
Measure: Cardinality k of the coloring
Aim: Minimum k coloring: $S^* : \forall S \neq S^* \; k(S) \geq k(S^*)$.

In [49], an MA was presented for this problem which used an embedded kind of Standard_Local_Search(...) after the mutation stage. The selection strategy used was a generational GA with 1-elitism (the worst individual of the new population is replaced with the best of the previous one) and the algorithm also used some specially designed operators. The authors reported what, at the time the paper was written, were exciting results.

Fleurent and Ferland [50] studied a number of MAs for MGC based on the hybridization of a standard steady-state GA with problem-specific local searchers and TS. The improvement stage was used instead of the mutation stage of the standard GA. The authors also ran experiments with a problem-specific crossover. The pseudocode employed in this paper is omitted because of its similarity with Genetic_Hybrid_Algorithm(...) already discussed.

In [51], Dorne and Hao' proposed an MA for the MGC. This MA used a new crossover, based on the union of independent sets, which is itself a kind of local searcher. The mutation stage was replaced by the powerful TS. With this MA, the authors were able to improve over the best known results of some large instances of the famous *Dimacs benchmarks*. Their algorithm is as follows.

```
GL_for_Coloring
Begin
  /* f, F*: fitness function and */
  /* best value encountered so far */
  /* s*: best individual encountered so
  far */
  /* best(P): returns the best
  individual */
  /* of the population P */
  i = 0;
  generate(P_0);
  s* := best(P_0);
  f* := f(s*);
  While (f* > 0 and i < maxIter) Do
    P'_i := crossing(P_i, T_x);
    /* using specialised crossover */
```

```
    P_{i+1} = mutation(P'_i);
    /* using Tabu search */
    If (f(best(P_{i+1})) < f*) Then
      s* := best(P_{i+1});
      f* := f(s*);
    Fi
    i := i + 1;
  Od
End.
```

E. MAs for the PSP

Protein structure prediction is one the most exciting problems that computational biology faces today. In the words of Smith [52]:

> Although we understand how genes specify the sequence of amino acids in a protein, there remains the problem of how the one-dimensional string of amino acids folds up to form a three-dimensional protein... it would be extremely useful to be able to deduce the three-dimensional form of a protein from the base sequence of the genes coding for it; but this is still beyond us.

Because "all-atom" simulations are extremely expensive researchers often resort to a simplified model of the *PSP*. One well-studied example is Dill's HP model [53]. Despite being a simplification, variations of this model have been shown NP-hard, see for example [54]–[56]. It may be defined as follows.

HP-model of Protein Structure Prediction
Instance: A simplified protein sequence of length l, i.e., a string $s \in \{H, P\}^l$.
Solution: A self-avoiding path p which embeds s into a two or three dimensional lattice (i.e., Z^2 or Z^3)
This defines a *Distance Matrix*, D, of inter-residue distances.
Measure: Potential energy, of the sequence in that fold, approximated by the number of pairs of H-type residues, which are not sequence-adjacent, but are at distance 1 in p
$E(p) = -\sum_{i=1}^{l=2}\sum_{j=i+2}^{l} D_{ij} | (D_{ij} = 1) \wedge (s_i = s_j = H)$
Aim: Minimum energy solution $p^* : \forall p \neq p^* \; E(p) > E(p^*)$.

In [57], we applied the following MA to the PSP.

```
PF_MA
Begin
  Random initialize population Parents;
  Repeat Until (Finalization_criteria_met) Do
    Local_Search(Parents);
    mating_pool=Select_mating(Parents);
    offsprings:=Cross(mating_pool);
    Mutate(offsprings)
    Parents:=Select(Parents+offsprings)
  Od
End.
```

This algorithm was able to find optimum configurations for 19 out of 20 protein instances of moderate size, out performing a GA with identical architecture except for the use of `Local_Search(...)`. In this MA, a $(\mu + \lambda)$ replacement strategy was used, together with fitness-proportionate selection for mating. In contrast to all the previous MA, in this scheme `Local_Search(...)` is considered a "first class" operator. It receives the entire population and applies with probability p_{ls} a complex local search algorithm to each individual. Under this scheme, solutions are improved during all their life span. In [58], several MAs for other molecular conformation problems are briefly commented on. In [59], a comparison of SA against GA and LS hybrids is presented for the closely related *drug docking* domain. In [60], a coevolutionary memetic approach is introduced, while in [61] the authors introduce a memetic crossover for the PSP.

V. Syntactic Model for MAs

A. Syntactic Model for EAs

Following [62], the EA can be formalized within a "generate-and-test" framework by the following:

$$GA = (P^0, \delta^0, \lambda, \mu, l, F, G, U).$$

- $P^0 = (a_1^0, \ldots, a_\mu^0) \in I^\mu$: Initial population.
- $I = \{q_1, \ldots, q_n\}^l$: n-ary finite-discrete problem representation.
- $\delta^0 \subseteq \Re$: Initial parameter set for operators.
- $\mu \in \mathbf{N}$: Population size.
- $\lambda \in \mathbf{N}$: Number of offspring.
- $l \in \mathbf{N}$: Length of representation.
- $F : I \mapsto \Re^+$: Fitness function.
- $G : I^\mu \mapsto I^\lambda$: Generating function.
- $U : I^\mu \times I^\lambda \mapsto I^\mu$: Updating function.

Note that in this model, we consider the effects of survivor selection at the end of one generation, and parent selection at the start of the next, to be amortized into a single function U, which is responsible for updating the working memory of our algorithm. The MAs' literature, as does the general EA literature, contains examples of the incorporation of diversity-preservation measures into U. These have included implicit measures, such as the imposition of spatial structure on the population (e.g., [42], [63], and [64]) or explicit measures such as duplicate prevention (e.g., [65] and [66]). This issue will be discussed in more depth in Section VII.

Examples of G as generating functions are mutation and crossover operators. A recombination operator has as its signature[1] $R : I^\mu \times \delta \mapsto I$ and a mutation operator $M : I \times \delta \mapsto I$. The initial values for parameters of the operators used (e.g., mutation probabilities) are represented by δ^0. If $O \in I^\lambda$ denotes the set of offspring, then an iteration of the GA is

$$\begin{aligned} O_i^t &= M\left(R(P^t, \delta^t), \delta^t\right) \; \forall i \in \{1, \ldots, \lambda\} \\ P^{t+1} &= U(O^t \cup P^t) \end{aligned} \tag{1}$$

where t is the time step.

[1]Note that the use of the superscript μ permits the modeling of crossover operators with variable arity, e.g., Moscato's K-mergers.

Although the formalization above assumes a finite-discrete problem representation with each element of the representation having the same arity, this is done simply for the sake of clarity, and the framework permits the use of any desired representation via suitable redefinition of I.

B. Extension to MAs

We will need to extend this notation to include local search operators as new generating functions. We define these to be members of a set, $\mathcal{L} = \{L_1, \ldots, L_m\}$, of local search strategies available to the MA.

Examples of so called "multimeme algorithms" where the local search phase has access to several distinct local searchers (i.e., $m > 1$) can be found in [20] and [67]. The signature of each member of the set $\mathcal{L}$ is $L_j : I^{c_1} \times \zeta \mapsto I^{c_1}$, where ζ is a strategy_specific parameter (with a role equivalent to δ), j is an index into the set $\mathcal{L}$, and c_1 is a constant that determines how many solutions the local searcher uses as its argument and how many solutions it returns. In general, we will assume that $c_1 = 1$ and, consequently, drop the subscript for the sake of clarity, but as an example of a local searcher with $c_1 = 2$, the reader might consider Jones' Crossover Hill Climber [68].

As can be seen from the pseudocode in the previous sections, the local search stage can happen before or after crossover, mutation, and selection, or in any imaginable combination, and the local searchers are members of a (potentially) large set of alternative heuristics, approximate or exact algorithms with which solutions could be improved.

To model this, we define entities called *schedulers* which are higher order functions.[2] An early example of the application of higher order functions to MAs see [69], where the authors implement Radcliffe and Surry's formalism [19] in a functional language.

C. Coordinating Local Search With Crossover and Mutation

The *fine-grain scheduler (fS)* coordinates when, where, and with which parameters local searchers from $\mathcal{L}$ will be applied during the mutation and crossover stages of the evolutionary cycle. It has the following signature:

$$fS : (I^{c_1} \times \delta \mapsto I) \times \mathcal{L} \times I^{c_1} \times \delta \times \zeta \mapsto I.$$

The fS receives three arguments. The first is a generating function with signature $I^{c_1} \times \delta \mapsto I$, that is, recombination (with $c_1 = \mu$) or mutation (with $c_1 = 1$). The second is a set of local searchers to be scheduled, which have signatures $I^{c_2} \times \zeta \mapsto I^{c_2}$. Usually c_2 will have the value 1: for example, in most of the examples above, local search is applied *after* recombination or mutation. However, our model should not rule out other possibilities—for example, doing local search on the parents *before* recombination, in which case $c_2 = c_1$. Finally, it receives a set of solutions by means of the I^{c_1} and operators and two sets of strategy specific parameters δ and ζ. In the simplest case, there will be a mutation (fS_M) and a recombination (fS_R) schedulers with the following signatures:

$$\begin{aligned} fS_M &: (I \times \delta \mapsto I) \times \mathcal{L} \times I \times \delta \times \zeta \mapsto I \\ fS_R &: (I^{\mu} \times \delta \mapsto I) \times \mathcal{L} \times I^{\mu} \times \delta \times \zeta \mapsto I. \end{aligned}$$

To illustrate this point, consider the case where a single local search method is used: $fS_M(M, L_1, i, \delta, \zeta_1)$, where i is an individual, M is a mutation operator, and L_1 is the local searcher with parameters ζ_1. Note that we are not specifying how M and L_1 will be used "inside" the scheduler. As examples of how the scheduler might operate, consider a simple case where mutation M is applied to i and the result of this operation is given as an argument to L_1. The symmetric case is equally valid, i.e., applying mutation M to the result of improving i with L_1. More complex scenarios can be imagined, it is up to fS to organize the correct pairing of inputs/outputs to functions.

A similar case can be stated for $fS_R(R, L_2, Q, \delta, \zeta_2)$, where in this case we are receiving as actual parameter a population of individuals Q (usually, a subset of P) rather than a single individual.

An illustration of this can be found in [46], where the authors argue in favor of encapsulating a $2-\texttt{Opt}(\ldots)$ local searcher into an algorithm with Nagata's and Kobayashi's *edge assembly crossover* [45]. The latter is a good example of an "intelligent" crossover operator which uses information about edge lengths to construct an offspring by connecting subtours common to both parents. However, in both the Nagata and Kobayashi's original algorithm, and Watson *et al.*'s "improved" version the crossover operator is used to generate a single offspring. In the original paper, *iterative child generation* is used, i.e., the scheduler repeatedly applies the crossover operator until a good solution is found, and in Watson's version, a 2-opt local search is applied **after** the crossover operator.

An even clearer example can be found in [70], where a new crossover for the *job-shop scheduling problem* is proposed. In this case, the crossover is a local search procedure that uses a two-solutions-based neighborhood. In other words, we can make a clear distinction between a generating operator that only ever considers one or two potential offspring, (even if it "intelligently" uses heuristics such as edge distances in its construction phase) and one that constructs and evaluates several solutions before returning an offspring. The latter case clearly is that of a scheduled local search, where the "neighborhood" of a (pair of) point(s) is defined by the action of a generating operator.

D. Coordinating Local Search With Population Management

An alternative model, as illustrated in Section IV-E, is to coordinate the action of local search with the population management and updating functions. A *coarse-grain scheduler (cS)* is defined by

$$cS : (I^{\mu} \times I^{\lambda} \mapsto I^{\mu}) \times \mathcal{L} \times I^{\mu} \times I^{\lambda} \times \delta \times \zeta \mapsto I^{\mu}.$$

In this scheduler, the formal parameters stand for the updating function U, the set of the local searchers $\mathcal{L}$, the sets of parents and offspring (I^{μ} and I^{λ}, respectively), and the operator specific parameter sets δ and ζ. The goal of this scheduler is to organize the application of a local searcher to either the set of parents, the set of offspring, or to their union. The difference between

[2] A higher order function or functional is a function whose domain is itself a set of functions, e.g., the indefinite integral of a function is a higher order function.

a coarse-grain scheduler and a fine-grain scheduler is that the former can provide population statistics to its local search operator, while the fine-grain scheduler knows just one individual at a time (or two for the one associated with crossover).

By the introduction of the local search schedulers, we can simulate any of the algorithmic combinations above. Also, by using a **set** of local searchers by the schedulers, we can model powerful multioperator hybrid strategies like those described in [71]–[73]. We can also include the approaches discussed in [74] and [75], where partial Lamarckianism or "sniffs" rather than complete local searches are allowed and allocated dynamically during the search. Further, it is possible to model the local search methods described in [26], where statistics from the population are used to apply local search selectively. Another interesting example of the use of coarse-grain schedulers can be seen in [76], where a hybrid metaheuristic is introduced which uses concepts of both EAs and gradient search. Under this scheme, potentially all the individuals in the populations are continuously "learning" since each stage of LS may be truncated rather than continue to local optimality.

E. Incorporating Historical Information

The natural extension to this model is to introduce a *meta scheduler (mS)* with the following signature:

$$mS : \mathcal{L} \times H_P^t \times I^\mu \times \delta \times \zeta^t \mapsto I^\mu$$

where $H_P^t \subseteq P_1 \cup P_2 \cup \ldots \cup P_{t-1}$.

The meta scheduler is able to use information from previous populations to influence its search by means of ζ and the elements of $\mathcal{L}$, hence a kind of evolutionary memory is introduced into the evolutionary search mechanisms. Note that in these cases the parameter sets ζ associated with the schedulers may now represent complex data structures rather than simple probability distributions.

With the introduction of this scheduler, a new class of metaheuristics is available given by the many possible instantiations of

$$\begin{aligned} O_i^t &= fS_M\left(M, \mathcal{L}, fS_R(R, \mathcal{L}, P^t, \delta^t, \zeta^t), \delta^t, \zeta^t\right) \\ &\quad \forall i \in \{1, \ldots, \lambda\} \\ P^{t+1} &= mS\left(\mathcal{L}, H_P^t, cS(U, \mathcal{L}, P^t, O^t, \delta^t, \zeta^t), \delta^t, \zeta^t\right) \end{aligned} \quad (2)$$

where the use of superscripts t recognizes that the several parameters may be time-dependant. We have not found this kind of MAs in the literature, yet they represent a novel, qualitatively different and perhaps powerful family of MAs. As an example of its use, one can imagine that the elements of $\mathcal{L}$ are based on TS and that the metascheduler uses the information of ancient populations to update their tabu lists, thus combining global and local information **across time**. An advantage of considering metaschedulers which affect ζ, is that by setting all elements of $\mathcal{L}$ to the identity function, it is possible to include within our model the work on adaptive GAs which use a *history* of previous results to update the probabilities of applying genetic operators, such as those described in [77]–[79]. Furthermore, the more recent approach to optimization called "hyperheuristics," in particular, those described in [80] and [81] can be considered to be multimeme algorithms, where the set of low-level operators (i.e., local searchers and constructive heuristics) are adaptively applied to one solution ($\lambda = \mu = 1$) by the metascheduler (called hyper-manager in hyperheuristics terminology).

VI. Taxonomy for MAs

A. Scheduler-Based Taxonomy

With the use of (2), it is possible to model the vast majority of the MAs found in the literature, capturing the interaction between local search and the standard evolutionary operators (mutation, crossover, selection). From this syntactic model, a taxonomy of architectural classes can be naturally derived based on an index number $D(A)$ which can be ascribed to any MA (A).

$D(A) = b_{mS}b_{cS}b_{fS_R}b_{fS_M}$ is a 4 bit binary number with each b_i taking the value 0 or 1 according to whether scheduler i is absent, respectively present, in A. To understand the ordering of the bits, note that the least significant bit is associated to the scheduler that receives as one of its arguments at most one solution, the next bit to the one that receives at most μ solutions, the next 2 bits are assigned to the schedulers that employ at most $\mu + \lambda$ or $|2^{P_1 \cup \ldots \cup P_{t-1}}|$ solutions, respectively, in their arguments.

To illustrate this point with examples from the review above, the algorithm `Genetic_Local_Search(...)` has an index $D = 2$ because just the fine-grain scheduler associated with crossover and meme is present, while `GLS_Based_Memetic_Algorithm(...)` has $D = 3$, since the mutation and crossover schedulers are used.

Table I classifies the various methods discussed in Section III accordingly to their D number, but it will rapidly be seen that only a small fraction of the alternative MAs were employed and investigated, and that the pattern is inconsistent across different problem types. Of particular interest are the frontiers for $D \geq 4$ and $D \geq 8$. Although throughout this paper we have concentrated mainly on single-objective problems, we have included in this table a reference to [21]. In that paper, the authors tackle a multiobjective problem using a MA with what they call a "nondominated Pareto archive" as an evolutionary memory. This work represents a clear example of an MA that resides above the frontier $D \geq 8$. It is clear from visual inspection of this table that there are plenty of alternative MAs waiting to be investigated for these problems.

B. Relationship to Other Taxonomies

The taxonomy presented here complements the one introduced in [91] by Calegary *et al.* who provide a comprehensive taxonomic framework for EAs. They define a "Table of Evolutionary Algorithms" (TEA), where the main features of the design space of EAs are placed in the columns of the table. In a related and complementary work, Talbi [92] provides a general classification scheme for metaheuristics based on two different aspects of a metaheuristic: its design space and its implementation space. He then develops a hierarchical organization for each one. Specifically, for the design space of hybrid metaheuristics, he identifies what he called low-level-relay hybrids (LRH), low-level-cooperative hybrids (LCH), high-level-relay

TABLE I
CLASSIFICATION OF ALGORITHMS DISCUSSED IN SECTION III ACCORDING TO PROBLEM AND D

9 - 15							
8							[21]
7						[76]	[82]
6		[16]		[47]			
5							[83]
4	[57]				[57], [59]	[26], [74], [84], [85]	
3	[13], [38], [44]					[86]	[87], [88]
2	[37], [42], [43], [45], [46]	[9]	[50], [51]				
1			[49]				[89], [90]
0							
D	TSP	QAP	MGC	BPQ	*PFP* and *Protein Docking*	General Studies	Other Applications

hybrids (HRH), and high-level-cooperative hybrids(HCH). Regarding those two works, MAs can be considered to be represented in Calegary *et al.* work with a TEA, where the column associated to an improving algorithms always receives a value of "yes," while in Talbi's taxonomy an MA could be placed within the LCH class.

Our approach categorizes the architecture of a subclass of the algorithms both of the previous taxonomies include. In that way, a more refined classification is obtained for the subclass of EAs and hybrid metaheuristic that are MAs. Of course such a syntactic model and taxonomy is of little interest to the practitioner unless it in some way aids in the conceptualization and design process. In the following sections, we shall move on to show how the model may be used during the design of an algorithm.

C. Distinguishing Types of Local Search Strategies

Making the separation into two sets of objects (candidate solutions in the EA's population, and local search heuristics), with interactions mediated by a set of schedulers facilitates a closer examination of the potential nature of the elements of $\mathcal{L}$. In keeping with the name given to this class of algorithms, so coincidentally very much in the spirit of his idea, we will adapt Dawkins' original definition and call each $L \in \mathcal{L}$ a "meme." Metaphorically speaking, memes can be thought of as representing alternative improvement strategies that could be applied to solutions, where these strategies may be imitated, improved, modified, etc.

The model presented in (2) already allows us to distinguish and define three cases.

- If $L^t = L^0 \forall t$, then we call L a static meme.
- If L^t adapts through changes in its parameters ζ^t as t increases, then we call L an *adaptive meme*.
- If L^t adapts through changes in L itself, e.g., by evolving under a GP approach, (and possibly in ζ^t also), then we call L a *self-adaptive meme*.

It is important to realize that it is sufficient for any $L \in \mathcal{L}$ to be adaptive to make the whole set of memes into an adaptive meme set. In the same way, if just one L^t is self-adaptive then the entire $\mathcal{L}$ is self-adaptive.[3]

To the best of the authors' knowledge the only MAs that scheduled more than one static local searcher at a time are those described in [71]–[73]. Almost all the papers studied in this work use single static memes with the exceptions of the algorithms described in [76] (if the momentum term is included into the model described therein), `GLS_Based_Memetic_Algorithms(...)` and `PF_MA(...)`. As examples of self-adaptive memes, we refer the reader to the more recent in [93]–[98].

The extension to considering a *set* of adaptive or self-adaptive memes, rather than a single local search method, gives rise to an extra level of complexity in the schedulers. The simplest case uses static memes and requires that ζ is enlarged to include a probability distribution function (pdf) for the likelihood of applying the different memes, in addition to their operational parameters. More complex cases might involve a different pdf for each scheduler.

The simplest adaptive case requires that ζ is time-dependent, with the scheduler becoming responsible for adapting the pdf. In more complex scenarios, it might be necessary to store a different pdf for each member of the population—i.e., individual rather than population level adaptation in the terminology of [99] and [100]. Allowing for adaptivity within, MAs makes it necessary to couple the adaptation over time of ζ and $\mathcal{L}$ to the evolutionary (2).

VII. DESIGN ISSUES FOR "COMPETENT" MAS

In [33], Goldberg describes "competent" GAs as:

> Genetic algorithms that solve hard problems quickly, reliably, and accurately.

As we have described above, for a wide variety of problems, MAs *can* fulfil these criteria better than traditional EAs. However, the simple inclusion of a given local search method is not enough to increase the competence of the underlying EA. Rather, the design of "competent" MAs raises a number of important issues. It is now appropriate to revisit these issues, in the light of our syntactic model and taxonomy, in order to see what new insights can be gained. While we are not suggesting that all implementations of MAs should follow the scheduler-based viewpoint, we would argue that it is certainly beneficial to consider this perspective to inform design decisions.

To recap, some of the principal design issues are,

- What local search operator should be used?
- Which fitness landscape(s) is the MA navigating?
- With what local optima is the MA operating?

[3]For the sake of clarity of the model, we have left out the minor signature modifications that are needed to reflect the fact that a meme might change its arguments or change itself.

- Where, and when, should local search be applied within the evolutionary cycle?
- Is a Baldwinian or Lamarckian model to be preferred?
- How can the genetic operators best be integrated with local search in order to achieve a synergistic effect?
- How can we engineer MAs that can efficiently traverse large neutral plateaus and avoid deep local optima?
- Which individuals in the population are to be improved by local search and how do we choose among them?
- How much CPU budget will be allocated to the local search?

We now discuss these items according to the grouping above.

A. Choice of Local Search Operators

The reader will probably not be surprised to find that our answer to the first question is "*it depends*." In [67], we showed that even within a single problem class (in that case TSP) the choice of which single LS operator gave the best results when incorporated in an MA was entirely instance-specific. Furthermore, studies of the dynamic behavior of various algorithms (including multi-Meme MAs) showed that in fact the choice of which LS operator yielded the biggest improvements was also time-dependent.

It is well known that most metaheuristics suffer from getting trapped in local optima. It is also trivially true that a point which is locally optimal with respect to one operator may not be with respect to another (unless it is globally optimal). Taking these points together has motivated recent work into metaheuristics such as *variable neighborhood search* [101], which utilize multiple local search operators.

In earlier sections, we have listed a number of papers in the recent MA literature which use multiple LS operators, and we would certainly argue that faced with a choice of operators, a sensible design approach would be not to decide *a priori* but to incorporate several. Given such an approach, for the sake of efficiency, it is worth considering methods to avoid spending time utilizing nonproductive operators, which implies at least some way of adapting the operator probabilities in ζ. This in turn implies a coarse-grain or metascheduler is present. It is perhaps worth noting that in [95], it was shown that while coarse-grain adaptation of ζ was sufficient for a *steepest-ascent* LS, the extra noise inherent in an *first-ascent* approach gave worse results. It was suggested that in such a case using a "history" of relative performance gains, as per Paredis' LTFE would be beneficial—in other words a metascheduler.

Related to this point are the two more theoretical issues concerning landscape and local optima. Merz *et al.* in [10], [102], and [103] employ the concept of fitness landscape distance correlation to assess the behavior of MAs. Although the correlation measures discussed in those papers can provide very valuable indications on the likely performance of MAs, they can sometimes be misleading. In particular, as a fitness distance correlation is measured based on one particular move operator (e.g., local searcher), if any of the schedulers in (2) has access to more than one local searcher, then different fitness landscapes will need to be considered. This fact was recognized by Jones' "one operator, one landscape" axiom [104].

B. Integration Into EA Cycle

We have grouped together the next three issues in our list as they are intimately related, and there has been some confusion in the previous literature.

Some researchers [18] consider that when the LS operator is applied *before* crossover and mutation, then the MA is a "Lamarckian" algorithm, and when the LS operator is applied *after* crossover and mutation, it is a pure "Darwinian" algorithm. This is an erroneous interpretation of Lamarckian versus Baldwin learning. In both cases, local search is used to improve (if possible) the fitness of the candidate solution, thus changing its selection probabilities. The difference is simply that in the case of Lamarckian (but not Baldwinian) learning the modifications are also assimilated into the individual—in other words, the fitter neighbor *replaces* the original candidate solution. As is clearly illustrated in the case of `GLS_Based_Memetic_Algorithm(...)`, Lamarckian learning in MAs can happen before or after the application of the other genetic operators. However, there is little point in applying Baldwinian search after parent selection but before recombination and mutation, since the resultant offspring will need to be reevaluated anyway.

If a Lamarckian local search is continued to optimality, then on average the recombination and mutation are likely to reduce the fitness of a solutions which were previously locally optimal. The hoped-for synergy in such an MA is that the use of genetic variation operators will produce offspring which are more likely to be in the basin of attraction of a high-quality local optimum than simply randomly selecting another point to optimize. Clearly, in order to achieve this synergy, i.e., to avoid selection discarding these new points, it is a good idea to perform local search on these offspring prior to selection. In other words, an algorithm "select-local search—probabilistically recombine—probabilistically mutate-select..." makes little sense.

In practice, most recent work has tended to use a Lamarckian approach, and the papers cited by Merz and Freisleben are typical in their (highly successful) advocacy of running the local search to optimality. However, as noted in Section IV, simply incorporating one or more powerful local searchers into an EA can lead to a rapid loss of diversity if steps are not taken to prevent this during the design phase. This has clear implication for the likelihood of the algorithm getting stuck in local optimum, or "stagnating" on a plateau.

The use of coarse-grain schedulers provides a simple means of avoiding this by monitoring population convergence statistics. In `QAP_MA(...)`, this is done by monitoring convergence, then applying vigorous mutation to the whole population. An alternative approach can be seen in [20] and [57] which utilizes a Boltzmann criteria in the pivot rule of local search, with the inverse of the population fitness range determining the temperature and, hence, the likelihood of accepting a worse neighbor in a local search.

C. Managing the Global-Local Search Tradeoff

The majority of MAs in the literature apply local search to every individual in every generation of the EA, our model makes

it clear that this is not mandatory. In [26] and [74], the authors explore these issues and suggest various mechanisms by which individuals are chosen to be optimized by local search, the intensity of local search, and the probability of performing the local optimization. They achieve this by providing sophisticated coarse-grain schedulers that measure population statistics and take them into consideration at the time of applying local search.

In [74], Land addresses the problem of how to best integrate the local search operators with the genetic operators. He proposes the use of fine-grain schedulers, both for mutation and crossover, that "sniff" (sample) the basin of attraction represented by a solution. That is, instead of performing a complete local search in every solution generated by the evolutionary operators, a partial local search is applied; only those solutions that are in promising basin of attraction will be assigned later (by the coarse-grain scheduler) an extended CPU budget for local search. In a similar spirit, Krasnogor in [20] proposes "crossover-aware" and "mutation-aware" local searchers.

In [57] and [71], the issue of large neutral plateaus and deep local optima is addressed by providing modified local searchers that can change their behavior accordingly to the convergence state of the evolutionary search. As we have noted above, a different approach to avoid getting trapped in a local optimum is to use various local searchers simultaneously in the population. In [67], [72], and [73] the authors resort to that technique to improve the robustness of the MAs.

VIII. Conclusion and Further Work

In this paper, we committed ourselves to the study of several works on MAs, coming from different sources, with the purpose of designing a syntactical model for MAs. In contrast with [19] where an *a priori* formal MA is given, ours is an *a posteriori* formalization based on the papers cited here and several others.

The syntactical model obtained allowed for the definition of an index number D into the taxonomy of MAs implicit in (2). When plotting the D index for a number of papers, we were able to identify classes of MAs that had received a lot of attention and other classes that were little explored from a theoretical and practical point of view. For example, we found no representatives of MAs (for mono objective optimization) with $D > 7$, although their counterparts are well known in the GA literature and in the multiobjective MA literature.

Furthermore, our syntactical model suggests the existence of a novel class of metaheuristic in which four schedulers interact. The reader should note that while a higher D value implies a more complex algorithm, it does not necessarily result in a better algorithm; all things being equal, a lower D algorithm should be preferred to one with a larger D.

We were able to identify two kinds of helpers, static and adaptive, and to generalize a third type: self-adaptive helpers. While examples were found of the first two types, the third type was just recently explored [93]–[98] suggesting another interesting line of research.

The adaptation of the index D to reflect the kind of helpers being used by the schedulers is straightforward.

Another important avenue of research is the study of which kind of MA, defined by its D index, is suitable for different types of problems. As shown in the second graph, just a few MAs' architectures were studied for each of the problem surveyed, it remains to be seen whether there are structural or merely historical reasons for the grouping observed. Our taxonomy complements the taxonomies in [91] and [92]. Both the syntactic model and the taxonomy aids our understanding of the design issues involved in the engineering of MAs.

Finally, we were able to revisit a list of important design questions and reconsider them in the light of our model, which offered us some new insights and ways of seeing common threads in disparate successful MAs. While we are not suggesting that all implementations of MAs should follow this scheduler-based viewpoint, we would argue that it is certainly beneficial to consider this perspective to inform design decisions.

Acknowledgment

N. Krasnogor would like to acknowledge insightful discussions with S. Ahmadi, S. Gustafson, D. Pelta, and W. Hart. The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] P. Moscato, "On evolution, search, optimization, GAs and martial arts: toward memetic algorithms," California Inst. Technol., Pasadena, CA, Tech. Rep. Caltech Concurrent Comput. Prog. Rep. 826, 1989.
[2] R. Dawkins, *The Selfish Gene*. New York: Oxford Univ. Press, 1976.
[3] L. Davis, *Handbook of Genetic Algorithms*. New York: Van Nostrand, 1991.
[4] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[5] J. Culberson, "On the futility of blind search: An algorithmic view of "no free lunch"," *Evol. Comput.*, vol. 6, no. 2, pp. 109–128.
[6] D. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in *Proc. Genetic Evol. Comput. Conf.*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, Eds., 1999, pp. 220–228.
[7] L. He and N. Mort, "Hybrid genetic algorithms for telecommunications network back-up routing," *BT Technol. J.*, vol. 18, no. 4, pp. 42–56, 2000.
[8] M. Vazquez and L. Whitley, "A hybrid genetic algorithm for the quadratic assignment problem," in *Proc. Genetic Evol. Comput. Conf.*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., 2000, pp. 135–142.
[9] C. Fleurent and J. Ferland, "Genetic hybrids for the quadratic assignment problem," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. Providence, RI: American Math. Soc., 1993.
[10] P. Merz, "Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies," Ph.D. dissertation, Parallel Syst. Res. Group, Dept. Elec. Eng. Comput. Sci., Univ. Siegen, Siegen, Germany, 2000.
[11] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a lamarkian genetic algorithm and an empirical binding free energy function," *J Comput. Chem.*, vol. 14, pp. 1639–1662, 1998.
[12] K. Ku and M. Mak, "Empirical analysis of the factors that affect the Baldwin effect," in *Lecture Notes in Computer Science*, 1998, Proc. Parallel Problem Solving From Nature—PPSN-V, pp. 481–490.
[13] P. Moscato, "Memetic algorithms: A short introduction," in *New Ideas in Optimization*, D. Corne, F. Glover, and M. Dorigo, Eds. New York: McGraw-Hill, 1999, pp. 219–234.
[14] W. Hart, N. Krasnogor, and J. Smith, Eds., *Recent Advances in Memetic Algorithms*. Berlin, Germany: Springer-Verlag, 2004.

[15] J. Smith, W. Hart, and N. Krasnogor, "Special issue on memetic algorithms," *Evol. Comput.*, vol. 12, no. 3, pp. 273–353, 2004.

[16] P. Merz and B. Freisleben, "A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem," in *Proc. Int. Congr. Evol. Comput.*, Washington DC, 1999, pp. 2063–2070.

[17] E. Marchiori, "Genetic, iterated and multistart local search for the maximum clique problem," in *Applications of Evolutionary Computing, LNCS 2279*. Berlin, Germany: Springer-Verlag, 2002, pp. 112–121.

[18] T. Ibaraki, "Combination with local search," in *Handbook of Genetic Algorithms*, U.K. Ibaraki, T. Back, D. Fogel, and Z. Michalewicz, Eds. London: Inst. Physics, Oxford Univ. Press, 1997, pp. D3.2-1–D3.2-5.

[19] N. Radcliffe and P. Surry, "Formal memetic algorithms," in *Lecture Notes in Computer Science*, T. Fogarty, Ed. Berlin, Germany: Springer-Verlag, 1994, vol. 865, Proc. Evol. Comput.: AISB Workshop, pp. 1–16.

[20] N. Krasnogor, "Studies on the theory and design space of memetic algorithms," Ph.D. dissertation, Univ. West of England, Bristol, U.K., 2002. [Online]. Available: http://www.cs.nott.ac.uk/~nxk/papers.html.

[21] J. Knowles and D. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *Proc. Congr. Evol. Comput. (CEC2000)*, vol. 1, 2000, pp. 325–332.

[22] ——, "A comparison of diverse approaches to memetic multiobjective combinatorial optimization," in *Proc. Genetic Evol. Comput. Conf., 1st Workshop Memetic Algorithms, Workshop Program* , A. Wu, Ed., 2000, pp. 103–108.

[23] ——, "A comparative assessment of memetic, evolutionary, and constructive algorithms for the multiobjective D-MST problem," in *Proc. 2001 Genetic Evol. Comput. Conf., 2nd Workshop Memetic Algorithms, Workshop Program* , R. Heckendorn, Ed., 2001, pp. 162–167.

[24] E. Talbi, M. Rahoud, M. Mabed, and C. Dhaenens, "New genetic approach for multicriteria optimization problems," in *Lecture Notes in Computer Science*, E. Zitzler, K. Deb, L. Thiele, C. Coello, and D. Corne, Eds. Berlin, Germany: Springer-Verlag, 2001, vol. 1993, Proc. 1st Int. Conf. Evol. Multicriterion Optimization, pp. 416–428.

[25] C. C. Coello, "List of references on evolutionary multiobjective optimization," [Online]. Available: http://www.lania.mx/~ccoello/EMOO/EMOObib.html.

[26] W. E. Hart, "Adaptive Global Optimization With Local Search," Ph.D. dissertation, Univ. California, San Diego, CA, 1994.

[27] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary search of approximated n-dimensional landscapes," *Int. J.Knowledge-Based Intell. Eng. Syst.*, vol. 4, no. 3, pp. 172–183, 2000.

[28] K. Liang, X. Yao, and C. Newton, "Lamarckian evolution in global optimization," in *Proc. of the 26th IEEE Int. Conf. Ind. Electron., Control, Instrumentation and the 3rd Asia-Pacific Conf. Simulated Evolution and Learning*, 2000, pp. 2975–2980.

[29] Y. S. Son and R. Baldick, "Hybrid coevolutionary programming for Nash equilibrium search in games with local optima," *IEEE Trans. Evol. Comput.*, vol. 8, no. 4, pp. 305–315, Aug. 2004.

[30] Y. S. Ong and A. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[31] T. Ray and K. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.

[32] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evol. Comput.*, vol. 12, no. 3, pp. 273–302, 2004.

[33] D. Goldberg, *The Design of Innovation: Lessons From and for Competent Genetic Algorithms*. Norwell, MA: Kluwer, 2002.

[34] P. Moscato. Memetic Algorithms' Home Page. [Online]. Available: http://www.densis.fee.unicamp.br/~moscato/memetic_home.html

[35] P. Crescenzi and V. Kann. (1999) A compendium of NP optimization problems. [Online]. Available: http://www.nada.kth.se/viggo/problemlist/compendium.html

[36] D. Johnson, C. Papadimitriou, and M. Yannakakis, "How easy is local search," *J. Comput. Syst. Sci.*, vol. 37, pp. 79–100, 1988.

[37] E. Aarts and G. Verhoeven, "Chapter g9.5: Genetic local search for the traveling salesman problem," in *Handbook of Evolutionary Computation*, T. Back, D. Fogel, and Z. Michalewicz, Eds. London, U.K.: Oxford Univ. Press, 1997, pp. G9.5:1–G9.5:7.

[38] D. Holstein and P. Moscato, "Memetic algorithms using guided local search: A case study," in *New Ideas in Optimization*, D. Corne, F. Glover, and M. Dorigo, Eds. New York: McGraw-Hill, 1999, pp. 235–244.

[39] C. Voudouris and E. Tsang, "Guided local search," *Eur. J. Oper. Res.*, vol. 113, no. 2, pp. 469–499, 1999.

[40] G. Reinelt. Tsplib (mirror site: gopher://softlib.rice.edu/11/softlib/tsplib). [Online]. Available: http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/tsplib95/tsplib.html

[41] A. Mariano, P. Moscato, and M. Norman, "Arbitrarily large planar ETSP instances with known optimal tours," *Pesquisa Operacional*, vol. 15, pp. 89–96, 1995.

[42] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 616–621.

[43] ——, "New genetic local search operators for the traveling salesman problem," in *Lecture Notes in Computer Science*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1141, Proc. 4th Conf. Parallel Problem Solving From Nature—PPSN IV, pp. 890–900.

[44] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1997, pp. 159–164.

[45] Y. Nagata and S. Kobayashi, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in *Proc. 7th Int. Conf. Genetic Algorithms*, T. Back, Ed., 1997, pp. 450–457.

[46] J. Watson, C. Ross, V. Eisele, J. Denton, J. Bins, C. Guerra, D. Withley, and A. Howe, "The traveling salesrep problem, edge assembly crossover, and 2-opt," in *Lecture Notes in Computer Science*, Proc. Parallel Problem Solving From Nature—PPSN-V. Berlin, Germany, 1998, pp. 823–832.

[47] P. Merz and B. Freisleben, "Genetic algorithms for binary quadratic programming," in *Proc. Int. Genetic Evol. Comput. Conf.*, Orlando, FL, 1999, pp. 417–424.

[48] J. Beasley. (1990) Or-Library. [Online]. Available: http://graph.ms.ic.ac.uk/info.html

[49] D. Costa, A. Hertz, and O. Dubouis, "Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs," *J. Heuristics*, vol. 1, pp. 105–128, 1995.

[50] C. Fleurent and J. Ferland, "Genetic and hybrid algorithms for graph coloring," *Ann. Oper. Res.*, vol. 63, pp. 437–461, 1997.

[51] R. Dorne and J. Hao, "A new genetic local search algorithm for graph coloring," in *Parallel Problem Solving From Nature V*. Amsterdam, The Netherlands: N. Holland, 1998, vol. 1498, pp. 745–754.

[52] J. M. Smith, *Shaping Life: Genes, Embryos and Evolution*: Weidenfeld and Nicolson, 1998.

[53] K. A. Dill, "Theory for the folding and stability of globular proteins," *Biochemistry*, vol. 24, pp. 1501–1509, 1985.

[54] J. Atkins and W. E. Hart, "On the intractability of protein folding with a finite alphabet," *Algorithmica*, vol. 25, no. 2–3, pp. 279–294, 1999.

[55] B. Berger and T. Leight, "Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete," in *Proc. 2nd Annu. Int. Conf. Comput. Molecular Bio.*, 1998, pp. 30–39.

[56] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis, "On the complexity of protein folding," in *Proc. 2nd Annu. Int. Conf. Comput. Molecular Bio.*, 1998, pp. 51–62.

[57] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proc. Genetic Evol. Comput. Conf.*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., 2000, pp. 987–994.

[58] P. Moscato, "Memetic algorithms for molecular conformation and other optimization problems," *Newsletter of the Commission for Powder Diffraction, of the International Union of Crystallography*, no. 20, 1998.

[59] C. D. Rosin, S. Halliday, W. E. Hart, and R. K. Belew, "A comparison of global and local search methods in drug docking," in *Proc 7th Int. Conf. Genetic Algorithms*, T. Baeck, Ed., San Francisco, CA, 1997, pp. 221–228.

[60] N. Krasnogor, "Coevolution of genes and memes in memetic algorithms," in *Proc. Genetic Evol. Comput. Conf. Workshop Program*, A. Wu, Ed., 1999, p. 371.

[61] N. Krasnogor, P. M. Lóopez, E. de la Canal, and D. Pelta, "Simple models of protein folding and a memetic crossover," in *Exposed at INFORMS CSTS, Computer Science and Operations Research: Recent Advances in the Interface Meeting*, 1998.

[62] J. Smith, "Self adaptation in evolutionary algorithms," Ph.D. dissertation, Univ. West of England, England, U.K., 1998.

[63] M. Gorges-Schleuter, "ASPARAGOS: An asynchronous parallel genetic optimization strategy," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., 1989, pp. 422–427.

[64] P. Moscato and F. Tinetti, "Blending Heuristics with a population-based approach: A memetic algorithm for the traveling salesman problem," Universidad Nacional de La Plata, La Plata, Argentina, Rep. 92-12, 1992.

[65] S. Ronald. (1995) Preventing diversity loss in a routing genetic algorithm with hash tagging. Complexity International. [Online]. Available: http://journal-ci.csse.monash.edu.au/ci/vol02/sr_hash/

[66] C. Fernandez, R. Tavares, C. Munteanu, and A. Rosa, "Using assortative mating in genetic algorithms for vector quantization problems," in *Proc. ACM Symp. Appl. Comput.*, 2001, pp. 361–365.

[67] N. Krasnogor and J. Smith, "Emergence of profitable search strategies based on a simple inheritance mechanism," in *Proc. Genetic Evol. Comput. Conf.*, L. Spector, E. Goodman, A. Wu, W. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshj, M. Garzon, and E. Burke, Eds., 2001, pp. 432–439.

[68] T. Jones, "Crossover, macromutation, and population based search," in *Proc. 6th Int. Conf. Genetic Algotihms*, M. Kauffman, Ed., 1995, pp. 73–80.

[69] N. Krasnogor, P. Mocciola, D. Pelta, G. Ruiz, and W. Russo, "A runnable functional memetic algorithm framework," in *Proc. Argentinian Congr. Comput. Sci.*, Universidad Nacional del Comahue, 1998, pp. 525–536.

[70] J. Sakuma and S. Kobayashi, "Extrapolation-directed crossover for job-shop scheduling problems: Complementary combination with JOX," in *GECCO-2000: Proc. Genetic Evol. Comput. Conf.*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds., 2000, pp. 973–980.

[71] N. Krasnogor and D. Pelta, "Fuzzy memes in multimeme algorithms: A fuzzy-evolutionary hybrid," in *Fuzzy Sets Based Heuristics for Optimization*, J. Verdegay, Ed. Berlin, Germany: Springer-Verlag, 2002, pp. 49–66.

[72] R. Carr, W. Hart, N. Krasnogor, E. Burke, J. Hirst, and J. Smith, "Alignment of protein structures with a memetic evolutionary algorithm," in *Proc. Genetic Evol. Comput. Conf.*, W. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. Potter, A. Schultz, J. Miller, E. Burke, and N. Jonoska, Eds., 2002, pp. 1027–1034.

[73] N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst, "Multimeme algorithms for protein structure prediction," in *Lecture Notes in Computer Science*, 2002, Proc. Parallel Problem Solving From Nature—VII, pp. 769–778.

[74] M. Land, "Evolutionary algorithms with local search for combinatorial optimization," Ph.D. dissertation, Univ. California, San Diego, CA, 1998.

[75] C. Houck, J. Joines, M. Kay, and J. Wilson, "Empirical investigation of the benefits of partial Lamarckianism," *Evol. Comput.*, vol. 5, no. 1, pp. 31–60, 1997.

[76] R. Salomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Trans. Evol. Comput.*, vol. 2, no. 2, pp. 45–55, Jul. 1998.

[77] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. Grefenstette, Ed., 1989, pp. 61–69.

[78] D. Corne, P. Ross, and H. Fang, "Fast practical evolutionary timetabling," in *Proc. AISB Workshop Evol. Comput.*, T. Fogarty, Ed., 1994, pp. 251–263.

[79] B. Julstrom, "What have you done for me lately?," in *Proc. 6th Int. Conf. Genetic Algorithms*, J. Grefenstette, Ed., 1995, pp. 81–87.

[80] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Lecture Notes in Computer Science*, E. Burke and W. E. Editors, Eds. Berlin, Germany: Springer-Verlag, 2001, Proc. Theory of Automated Timetabling PATAT 2000, pp. 176–190.

[81] ——, "Hyperheuristics: A tool for rapid prototyping in scheduling and optimization," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2002, Proc. 2nd Eur. Conf. Evol. Comput., EvoCop, pp. 1–10.

[82] F. Vavak, T. Fogarty, and K. Jukes, "A genetic algorithm with variable range of local search for tracking changing environments," in *Lecture Notes in Computer Science*, H. Voigt, I. Rechenberg, and H. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1141, Proc. Parallel Problem Solving From Nature—PPSN IV, pp. 376–385.

[83] G. Dozier, J. Bowen, and A. Homaifar, "Solving constraint satisfaction problems using hybrid evolutionary search," *IEEE Trans. Evol. Comput.*, vol. 2, no. 1, pp. 23–33, Apr. 1998.

[84] S. Tsutsui and A. Ghosh, "Genetic algorithms with a robust solution searching scheme," *IEEE Trans. Evol. Comput.*, vol. 1, no. 3, pp. 201–208, Sep. 1997.

[85] T. Murata, H. Ishibuchi, and M. Gen, "Specification of local search directions in genetic local search algorithms for multiobjective optimization problems," in *Proc. Genetic Evol. Comput. Conf.*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, Eds., 1999, pp. 441–448.

[86] P. Merz and B. Freisleben, "On the effectiveness of evolutionary search in high-dimensional NK-landscapes," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 741–745.

[87] B. Dengiz, F. Altiparmak, and A. Smith, "Local search genetic algorithm for optimal design of reliable network," *IEEE Trans. Evol. Comput.*, vol. 1, no. 3, pp. 179–188, Sep. 1997.

[88] J. Berger, M. Sassi, and M. Salois, "A hybrid genetic algorithm for the vehicle routing problem with time windows and itinerary constraints," in *Proc. Genetic Evol. Comput. Conf.*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, Eds., 1999, pp. 44–51.

[89] E. Burke and J. Newall, "A multistage evolutionary algorithm for the timetable problem," *IEEE Trans. Evol. Comput.*, vol. 3, no. 1, pp. 63–74, Apr. 1999.

[90] E. Marchiori and C. Rossi, "A flipping genetic algorithm for hard 3-sat problems," in *Proc. Genetic Evol. Comput. Conf.*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakaiela, and R. Smith, Eds., 1999, pp. 393–400.

[91] P. Calegary, G. Coray, A. Hertz, D. Kobler, and P. Kuonen, "A taxonomy of evolutionary algorithms in combinatorial optimization," *J. Heuristics*, vol. 5, pp. 145–158, 1999.

[92] E. Talbi, "A taxonomy of hybrid metaheuristics," *J. Heuristics*, vol. 8, no. 5, pp. 541–564, 2002.

[93] N. Krasnogor and S. Gustafson, "Toward truly "memetic" memetic algorithms: discussion and proof of concepts," in *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, D. Corne, G. Fogel, W. Hart, J. Knowles, N. Krasnogor, R. Roy, J. E. Smith, and A. Tiwari, Eds., 2002, pp. 9–10. PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9.

[94] J. Smith, "The coevolution of memetic algorithms for protein structure prediction," in *Recent Advances in Memetic Algorithms*, W. Hart, N. Krasnogor, and J. Smith, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 105–128.

[95] ——, "Coevolving memetic algorithms: a learning approach to robust scalable optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 498–505.

[96] ——, "Protein structure prediction with coevolving memetic algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 2346–2353.

[97] N. Krasnogor, "Self-generating metaheuristics in bioinformatics: The protein structure comparison case," *Genetic Program. Evolvable Mach.*, vol. 5, no. 2, pp. 181–201, 2004.

[98] N. Krasnogor and S. Gustafson, "A study on the use of "self-generation" in memetic algorithms," *Natural Comput.*, vol. 3, no. 1, pp. 53–76, 2004.

[99] P. Angeline, "Adaptive and self-adaptive evolutionary computations," in *Computational Intelligence*. Piscataway, NJ: IEEE Press, 1995, pp. 152–161.

[100] J. Smith and T. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Comput.*, vol. 1, no. 2, pp. 81–87, 1997.

[101] P. Hansen and N. Mladenovic, "An introduction to variable neighborhood search," *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, pp. 433–458, 1999.

[102] P. Merz and B. Freisleben, "Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning," *J. Evol. Comput.*, vol. 8, no. 1, pp. 61–91, 2000.

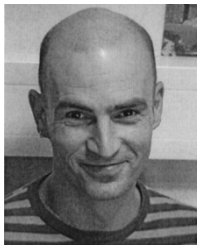
[103] ——, "On the effectiveness of evolutionary search in high-dimensional nk-landscapes," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1998, pp. 741–745.

[104] T. Jones, "Evolutionary algorithms, fitness landscapes and search," Ph.D. dissertation, Univ. New Mexico, Albuquerque, NM, 1995.

Natalio Krasnogor did his systems analysis and informatics studies at the Universidad Nacional de La Plata, La Plata, Argentina, and received the Ph.D. degree from the University of the West of England, England, U.K.

He did Postdoctoral Research at the University of Nottingham, Nottingham, U.K., where he became a Lecturer in 2002. He has established and Co-Chaired the series of International Workshops on Memetic Algorithms (WOMA). He is also a co-editor of *Recent Advances in Memetic Algorithms* (Berlin, Germany: Springer-Verlag), the the first book dedicated exclusively to memetic algorithms. He has served as a Reviewer for various international conferences and prestigious journals (e.g., *Journal of Heuristics*, *Journal on Evolutionary Computation*, *Journal of Fuzzy Sets and Systems*, *Bioinformatics*, etc.). In 2004, he was a Guest Editor of a Special Issue of the *Evolutionary Computation Journal* (MIT Press) dedicated to memetic algorithms. He is Guest Editor of a Special Issue of the *Journal of Fuzzy Sets and Systems* dedicated to bioinformatics. He is a member of the Editorial Board of the *International Journal of Computational Intelligence*, the *Journal of Evolutionary Computation*, and committee member of the Society for the Study of Artificial Intelligence and the Simulation of Behavior and the International Society for Genetic and Evolutionary Computation. He is principal investigator or co-investigator in grants totaling in excess of 1 million pounds from the British Research councils and the European Union.

Dr. Krasnogor has served as Reviewer for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is Guest Editor of an upcoming Special Issue on Memetic Algorithms to appear in the IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS (2005/2006).



Jim Smith received the Electrical Sciences Tripos from Cambridge University, Cambridge, U.K., the M.Sc. degree (with distinction) in communicating computer systems, and the Ph.D. degree from the University of the West of England, England, U.K., in 1993 and 1998, respectively.

He has been employed first as a Research Fellow in Adaptive Systems, and currently as a Senior Lecturer in the School of Computer Science, University of West of England. He has been awarded a number of industrial, U.K. governmental agency, and EU funded projects. He has authored a book on evolutionary computing, and has recently edited a book and a Special Issue of the *Journal on Evolutionary Computation* on Memetic Algorithms (a class of hybrid search heuristics). He has been researching in the field of heuristic optimization, machine learning and data mining since 1994.

Dr. Smith is Co-Founder and Chair of the International Workshops on Memetic Algorithms, and was Co-Chair for the Eighth International Conference on Parallel Problem Solving From Nature 2004.