

Co-evolving Memetic Algorithms: A review and progress report

J.E. Smith

Abstract—Co-Evolving Memetic Algorithms are a family of meta-heuristic search algorithms in which a rule-based representation of local search is co-adapted alongside candidate solutions within a hybrid evolutionary system. Simple versions of these systems have been shown to outperform other non-adaptive memetic and evolutionary algorithms on a range of problems

This paper presents a rationale for such systems and places them in the context of other recent work on Adaptive Memetic Algorithms. It then proposes a general structure within which a population of local search algorithms can be evolved in tandem with the solutions to which they are applied. Previous research started with a simple self-adaptive system before moving on to more complex models. Results showed that the algorithm was able to discover and exploit certain forms of structure and regularities within the problems. This “meta-learning” of problem features provided a means of creating highly scalable algorithms. This work is briefly reviewed to highlight some of the important findings and behaviours exhibited. Based on this analysis new results are then presented from systems with more flexible representations which again show significant improvements. Finally the current state of, and future directions for, research in this area is discussed.

I. INTRODUCTION

THE performance benefits which can be achieved by hybridising Evolutionary Algorithms (EAs) with Local Search (LS) operators, so-called *Memetic Algorithms* (MAs), have now been well documented across a wide range of problem domains such as optimisation of combinatorial, non-stationary and multi-objective problems (see [1] for a review, [2] for a collection of recent algorithmic and theoretical work and [3] for a comprehensive bibliography). Commonly in these algorithms, a Local Search improvement step is performed on each of the products of the generating (recombination and mutation) operators, prior to selection for the next population. There are of course many variants on this theme, for example one or more of the generating operators may be null, or the order in which the operators are applied may vary, but these can easily be fitted within a general syntactic framework [1].

In recent years it has been increasingly recognised that the particular choice of Local Search operator will have a major impact on the efficacy of the hybridisation. Of particular importance is the choice of move operator, which defines the neighbourhood function, and so governs the way in which new solutions are generated and tested. Points which are locally optimal with respect to one neighbourhood structure may not be with respect to another (unless of course they are globally optimal). It therefore follows that even if a population only

contains local optima, then changing the LS move operator (neighbourhood) may provide a means of progression in addition to recombination and mutation. This observation has led a number of authors to investigate and propose mechanisms for choosing between a set of predefined local search operators which may be used during a particular run of a meta-heuristic such as an EA (see Section II for a further discussion).

Previous papers have reported initial results from a system within which the definitions of Local Search operators applied within the MA may be changed during the course of optimisation. This was named the COevolving Memetic Algorithm (COMA). It maintains two populations: one of genes encoding for candidate solutions, and one of memes encoding for Local Search operators to be used within the MA. This paper places the COMA framework in the context of algorithmic advances by other authors, and then reviews the progression of research and results from the initial simple systems [4]–[6] to the more complex, truly co-evolutionary systems tested in [7].

In those investigations the emphasis had been placed on evolving the rule-based neighbourhood definition but leaving much of the rest of Local Search algorithm fixed. This restriction is then removed. Results and analysis are presented from a benchmark testing of the developed systems on a commonly used NP-Hard problems: MAX-3SAT. The rest of this paper proceeds as follows:

- Section II draws some parallels between this work and related work in different fields, in order to place this work within the more general context of studies into adaptation, development and learning.
- Section III describes the proposed approach, and how it can be used to represent a range of algorithms.
- Section IV summarises the results and analysis of a set of preliminary experiments using a simple self-adaptive model. These investigate whether, and if so how, the use of adaptive variable lengths rules benefits optimisation.
- Section V reviews experiments considering more general co-evolutionary forms of the model. It examines the effect of different pivot and pairing strategies and noise tolerance within credit assignment, and identifies two different modes of behaviour that benefit search: discovering and exploiting of regularities in the search space and continuously changing neighbourhood definitions.
- Section VI details new experiments in which more of the local search definitions are made open to adaptation. The effects of these changes are assessed in a benchmark comparison on the classic Max-3SAT problem.
- Finally, Section VII discusses the implications of these results, draws conclusions and suggests future work.

II. RELATED WORK

The COMA system can be related to a number of different branches of research, all of which offer different perspectives and means of analysing its behaviour. These range from Multi-Memetic Algorithms and the Self-Adaptation of search strategies, through co-evolutionary, learning, and developmental systems, to the evolution of rules in Learning Classifier Systems. The more important are briefly outlined below.

A. Memetic Algorithms with Multiple Local Search Operators

Although the author is not aware of other algorithms in which the LS operators used by an MA are adapted in this fashion, there are other recent examples of the use of multiple LS operators within evolutionary systems. Ong et al. [8] present an excellent recent review of work in the field of what they term “Adaptive Memetic Algorithms”. This encompasses the works of Krasnogor in “Multi-Memetic Algorithms” [9]–[12], Ong and Keane (who term their approach “metalamarkian memetic algorithms”) [13], and Hyper-Heuristics [14]–[17]. In an interesting extension to the use of a set of fixed strategies, Krasnogor and Gustafson have recently proposed a grammar for “Self-Generating MAs” which specifies, for instance, when local search takes place [18], [19]. Essentially all of these approaches maintain a pool of local search operators available to be used by the algorithm, and at each decision point make a choice of which to apply. There is a clear analogy between these algorithms and the Variable Neighbourhood Search algorithm [20], where a heuristic is used to control the order of application of a set of predefined local searchers to a single improving solution. The difference here lies in the population based nature of MAs, so that not only do we have multiple local searchers but also multiple candidate solutions, which makes the task of deciding which LS operator to apply to any given one more complex.

The classification of Ong et al. uses the terminology developed elsewhere to describe adaptation of operators and parameters in Evolutionary Algorithms [21]–[23]. They categorise algorithms according to the way that these decisions are made. One way is to use a fixed strategy, this is termed “static”. another is to use feedback of which operators have provided the best improvement recently. This is termed Adaptive, and is further subdivided into external, local (to a deme or region of search space), and global (to the population) according to the nature of the knowledge considered. Finally they note that LS operators may be linked to candidate solutions (Self-Adaptive). In this terminology, the COMA algorithm may be local-Adaptive or Self-Adaptive.

B. Self-Adaptation and Co-Evolution

As noted above, the COMA algorithm maintains two populations – one of genes and one of memes. If the populations are of the same size and selection of the two is tightly coupled then this instantiation of the COMA framework can be considered as a type of Self Adaptation. This also will be referred to as “linked” below. The use of the intrinsic evolutionary processes to adapt mutation step sizes has long been used

in Evolution Strategies [24], and Evolutionary Programming [25]. Similar approaches have been used to self-adapt mutation probabilities [26], [27] and recombination operators [28] in genetic algorithms as well as complex generating operators which combined both mutation and recombination [29].

If selection is performed separately for the two populations then COMA is a co-operative co-evolutionary system, where the fitness of the members of the meme population is assigned as some function of the relative improvement they cause in the “solution” population. Paredis has examined the co-evolution of solutions and their representations [30]. Potter and DeJong have also used co-operative co-evolution of partial solutions in situations where an obvious problem decomposition was available [31]. Both reported good results. Bull [32] conducted a series of more general studies on co-operative co-evolution using Kauffman’s static NKC model [33]. In [34] he examined the evolution of linkage flags in co-evolving “symbiotic” systems and showed that the strategies which emerge depend heavily on the extent to which the two populations affect each others fitness landscape, with linkage preferred in highly interdependent situations. He also examined the effect of different pairing strategies [35], with mixed results, although the NKC systems he investigated used fixed interaction patterns. Parker and Blumenthal’s “Punctuated Anytime Learning with samples” [36] is a recent approach to this problem of deciding how to pair members of different populations using periodic sampling to estimate fitness

There has also been a large body of research into competitive co-evolution (see [37] for an overview). Here the fitnesses assigned to the two populations are directly related to how well individuals perform “against” the other population - what has been termed “predator-prey” interactions. Luke and Spector [38] have proposed a general framework within which populations can be co-evolved under different pressures of competition and co-operation. This uses speciation both to aid both the preservation of diversity and as a way of tackling the credit assignment problem.

In all the co-operative and competitive co-evolutionary work cited above, the different populations only affect each other’s perceived fitness, unlike the COMA framework where the meme population can directly affect the genotypes within the solution population. This raises the question of whether the modifications arising from Local Search should be written back into the genotype (Lamarckian Learning) or not (Baldwinian Learning). Although the pseudo-code and the discussion below, assumes Lamarckian learning, this is not a prerequisite of the COMA framework. However, even if a Baldwinian approach was used, COMA differs from the co-evolutionary systems above because there is a selection phase within the local search, so that if all of the neighbours of a point defined by the meme’s rule are of inferior fitness, then the point is retained unchanged within the population. If one was to discard this criterion and simply apply the rule (possibly iteratively), the system could be viewed as a type of “developmental learning” akin to the studies in Genetic Code e.g. [39] and the “Developmental Genetic Programming” of Keller and Banzhaf [40], [41]

III. A RULE-BASED MODEL FOR THE ADAPTATION OF MOVE OPERATORS

A. Specifying Local Search

The Local Search step can be illustrated by the pseudo-code in the first box, and has three principal components. The first is the choice of pivot rule, which can be *Steepest Ascent* or *Greedy Ascent*. In the former the “termination condition” is that the entire neighbourhood $n(i)$ has been searched, ($counter = |n(i)|$), whereas the latter stops as soon as an improvement is found; i.e. the termination condition is ($counter = |n(i)| \vee (best \neq i)$). Note that some authors resort to only considering a randomly drawn sample of size $N \ll |n(i)|$ if the neighbourhood is too large to search.

The second component is the “depth” of the Local Search, i.e. the “iteration condition”. This lies in the continuum between only one improving step being applied ($iterations = 1$) to the search continuing to local optimality ($(counter = |n(i)|) \wedge (best = i)$). Considerable attention has been paid to studying the effect of changing this parameter within MAs e.g. [42]. Like the choice of pivot rules it has been shown to have an effect on the performance of the Local Search algorithm, both in terms of time taken and of quality of solution found.

```

Local_Search(i) :
Begin
/* given a starting solution i */
/* and a neighbourhood function n */
set best = i;
set iterations = 0;
Repeat Until ( iteration condition is satisfied )
Do
set counter = 0;
Repeat Until ( termination condition satisfied )
Do
generate the next neighbour j ∈ n(i);
counter = counter +1;
If (f(j) is better than f(best)) Then
set best = j;
Fi
Od
set i = best;
set iterations = iterations +1;
Od
End.

```

The third, and primary factor that affects the behaviour of the LS is the choice of neighbourhood generating function. This can be thought of as defining a set of points $n(i)$ that can be reached by the application of some move operator to the point i . One representation is as a graph $G = (v, e)$ where the set of vertices v are the points in the search space, and the edges relate to applications of the move operator i.e. $e_{ij} \in G \iff j \in n(i)$. The provision of a scalar fitness value, f , defined over the search space means that we can consider the graphs defined by different move operators as “fitness landscapes” [43]. Merz and Freisleben [44] present a number of statistical measures which can be used to characterise fitness landscapes, and have been proposed as potential measures of problem difficulty. They show that the choice of move operator can have a dramatic effect on the efficiency and effectiveness of the Local Search, and hence of the resultant MA.

B. Adapting the Specification of Local Search

The aim of this work is to provide a means whereby the definition of the LS operator used within a MA can be varied over time, and then to examine whether evolutionary processes can be used to control that variation so that a beneficial adaptation takes place. Accomplishing this aim requires the provision of five major components, namely:

- A means of representing a LS operator in a form that can be processed by an evolutionary algorithm, i.e. a meme.
- A means for initialising a population of memes, and a set of variation operators to be applied to them.
- A means of assigning fitness to the meme population
- A choice of population structures and sizes, selection and replacement methods for managing the meme population.
- A set of experiments, problems and measurements to permit evaluation and analysis of the system.

The representation chosen for the memes is a tuple $\langle Iterate_Condition, Terminate_Condition, Pairing, Move \rangle$. Once this representation is chosen, this leads naturally to the choice of variation operators to act on that representation within an evolutionary setting, and can readily encompass all of the other requirements identified above. The pseudo-code in Figure 1 illustrates the algorithm which was used to undertake the research reported in this paper. Note that although this pseudo-code assumes equal sized populations and synchronous evolution, this need not in general be the case.

The two conditions in the tuple have been described above, and can be easily mapped onto an integer or cardinal representation as desired, and manipulated by standard genetic operators. The element *Pairing* indicates how the choice of which members of the two populations to combine for evaluation is managed. The purpose of this element is to allow the system to be varied between two extremes. One is a fully unlinked system, in which although still interacting the two populations evolve separately. The other is a fully linked, self-adapting system. Here the memes are effectively extra genetic material inherited and varied along with the problem representation. As is illustrated in the *If.Else* section of the pseudo-code, this range of behaviours can be achieved by encoding the pairing as a value taken from the set $\{Linked, Random, Fitness_Based\}$. This value governs the choice of parents of the meme to be used to perform Local Search on a given offspring candidate solution.

The representation chosen for the move operators was as *condition:action* pairs, which specify a pattern to be looked for in the problem representation, and a different pattern it should be changed to. Although this representation at first appears very simple, it has the potential to represent highly complex moves via the use of symbols to denote not only single/multiple wild-card characters (in a manner similar to that used for regular expressions in Unix) but also the specifications of repetitions and iterations. Further, permitting the use of different length patterns in the *condition* and *action* parts of the rule gives scope for *cut* and *splice* operators working on variable length solutions.

```

COevolving Memetic Algorithm for Binary Coded Problems :
Begin
/* given a starting population P of  $\mu$  solutions and a same-sized population M of memes */
initialise P with randomly selected binary genes;
initialise M with randomly selected memes;
set generations = 0;
set evaluations = 0;
Repeat Until ( run_termination condition is satisfied )
Do

/* Create  $\mu$  offspring in each population */
For child := 1 To child =  $\mu$  Do
/* Create offspring by selection, recombination and mutation, storing the parents id's */
set FirstParent[child] = Select_One_Parent(P);
set SecondParent[child] = Select_One_Parent(P);
set Offspring[child] = Recombine(FirstParent[child],SecondParent[child]);
Mutate(Offspring[child]);

/* Select parents of the new meme */
set Pairing = Get_Pairing(M,child);
If (Pairing = Linked) Then
set FirstMemeParent[child] = FirstParent[child];
set SecondMemeParent[child] = SecondParent[child];
Fi
Else If (Pairing = Fitness_Based) Then
set FirstMemeParent[child] = Select_One_Parent(M);
set SecondMemeParent[child] = Select_One_Parent(M);
Fi
Else
set FirstMemeParent[child] = RandInt(1, $\mu$ );
set SecondMemeParent[child] = RandInt(1, $\mu$ );
Esle

/* Create new meme from these parents using recombination and mutation */
set NewMemes[child] = Recombine(FirstMemeParent[child],SecondMemeParent[child]);
Mutate(NewMemesffspring[child]);

/* Finally apply local search to Offspring Using Memes */
set original_fitness = Get_Fitness(Offspring[child]);
/* Applying the rule part of a meme to an offspring produces a set of neighbours */
set Neighbours = Apply_Rule_To_Offspring(Offspring[child],NewMemes[child]);
Evaluate_Fitness(Neighbours);
/* Pivot rule of meme determines choice of neighbour */
set Offspring[child] = Apply_Pivot_Rule_To_Neighbours(Neighbours,NewMemes[child]);
/* Finally update meme fitness according to increase in solution fitness */
set  $\Delta$ fitness = Get_Fitness(Offspring[child]) - original_fitness;
Update_Meme_Fitness(NewMemes[child],  $\Delta$ fitness);
set child = child +1;
Od
set evaluations = evaluations +1 + |Neighbours|;
set P = Offspring;
set M = NewMemes;
Od
End.

```

Fig. 1. Pseudo-Code Definition of COMA algorithm

IV. INITIAL PROOF OF CONCEPT: ANALYSIS OF DIFFERENT FIXED AND VARYING SIZED RULES

A. Initial Implementation

Initial experiments were restricted to considering a simple system, and examining its behaviour on a well understood set of test problems. They were designed to find out whether the system was able to evolve useful rules when the rule length naturally matched the structure of the problem, and then further to that whether the system was able to adapt to an appropriate rule length for different problems. For this reason it was decided to avoid the various issues concerning population management, pairing strategies and credit assign-

ment, and instead work with a single improvement step, a fully linked self-adaptive system and a greedy pivot rule. These choices were coded into the chromosomes at initialisation, and variation operators were not used on them.

The initial systems only used rules where the *condition* and *action* patterns were of equal length and were composed of values taken from the set of permissible allele values of the problem representation, augmented by a “don’t care” symbol (#) which is allowed to appear in the *condition* part of the rule but not the *action*. The neighbourhood of a point i then consists of i itself, plus all those points where the substring denoted by *condition* appears in the representation of i and is replaced by the *action*.

To give an example, a rule $1\#0 \rightarrow 111$ matches the binary string 1100111000 in the first, second, sixth and seventh positions, and the neighbourhood is the set $\{1100111000, 1110111000, 1111111000, 1100111100, 1100111110\}$. Note that the string is not treated as toroidal, and the neighbours are evaluated in a random order so as not to introduce positional bias into the local search when greedy ascent is used.

In practise, each rule was augmented by a value *rule_length* specifying the number of positions in the pattern string to consider. This permitted not only the examination of the effects of different fixed rule sizes, but also the ability to adapt via the action of mutation operators on this value. This representation for the rules allows the use of “standard” genetic operators (uniform/1 point crossover, point mutation) to vary this part of the memes’ chromosomes.

B. The Test Suite and Experimental set-up

In order to examine the behaviour of the system it was used with a set of variants of test functions whose properties are well known. The first of these was a sixty four bit problem composed of 16 subproblems of Deb’s 4-bit fully deceptive function [45]. The fitness of each subproblem i is given by its unitation $u(i)$, that is the number of bits set to “one”:

$$f(i) = \begin{cases} 0.6 - 0.2 \cdot u(i) & : u(i) < 4 \\ 1 & : u(i) = 4 \end{cases} \quad (1)$$

In addition to this “concatenated” version (4-Trap), a second “distributed” version (Dist-Trap) was used in which the subproblems were interleaved i.e. sub-problem i was composed of the genes $i, i + 16, i + 32, i + 48$. This separation ensured that in a single application even the longest rules allowed in these experiments would be unable to alter more than one element in any of the sub-functions. A third variant of this problem (Shifted-Trap) was designed to be more “difficult” than the first for the COMA algorithm, by making patterns which were optimal in one sub-problem, sub-optimal in all others. Since unitation is simply the Hamming distance from the all-zeroes string, each sub-problem can be translated by replacing $u(i)$ with the Hamming distance from an arbitrary 4 bit string. There were 16 sub-problems so the binary coding of each ones’ index was used as basis for its fitness calculation.

A generational genetic algorithm, with deterministic binary tournament selection for parents and no elitism was used. One Point Crossover (with probability 0.7) and bit-flipping mutation (with a bitwise probability of 0.01) were used on the problem representation. These choices were taken as “standard” from the literature, and no attempt was made to tune them to the particular problems at hand. Mutation was applied to the rules with a probability of 0.0625 of selecting a new allele value in each locus (the inverse of the maximum rule length allowed to the adaptive version).

For each problem, 20 runs were made for each population size $\{100, 250, 500\}$. Each run was continued until the global optimum was reached, subject to a maximum of 1 million evaluations. Two performance metrics were considered; the Success Rate (SR) which is the number of runs finding the global optimum, and the Average Evaluations to Success

(AES) which is the mean time taken to locate the global optimum on successful runs. The reason for the large cut-off value was to try and avoid skewing results as can happen with an arbitrarily chosen lower cut-off, rather than to be indicative of the amount of time available for a “real world” problem. Note that since one iteration of a local search may involve several evaluations, this allows more generations to the GA, i.e. algorithms are compared strictly on the basis of the number of calls to the evaluation function.

The algorithms used (and the abbreviations which will be used to refer to them hereafter) are as follows:

- A “vanilla” GA with no Local Search (GA).
- A simple MA using a bit-flipping neighbourhood, with one iteration of greedy ascent (SMA).
- A version of COMA using a random rule in each application, i.e. with the learning disabled (RandCOMA).
- Variants of COMA using rules of fixed lengths in the range $[1,10]$ (1-COMA, ..., 10-COMA),
- An adaptive version of COMA (A-COMA), in which the rule lengths are randomly initialised in the range $[1,16]$. During mutation, a value of ± 1 is randomly added with probability 0.0625, subject to staying in range.

C. Results on “4-Trap” function

The results obtained showed that the GA, SMA, and 1-COMA algorithms frequently failed to find the optimum unlike the other COMA variants, which always did. On these problems there was a clear benefit to using adaptive neighbourhood local search, although since the RandCOMA algorithm found the optimum on every run, the Success Rate metric did not provide conclusive evidence that learning was taking place.

Considering the AES, the GA, SMA and 1-COMA algorithms took longer to locate the optimum, particularly for the smaller population sizes. For all population sizes there was greater variance in the performance of these three algorithms. Applying Tamhane’s T2 test pairwise to the AES results showed that performance of the GA, SMA and 1-COMA algorithms was significantly worse than the rest with 95% confidence for a population of 100 or 250. For a population of 500 2-COMA joined the significantly slower group.

In short, it could be observed that for fixed rule lengths of between 3 and 9, and for the adaptive version, the COMA system derived performance benefits from evolving LS rules according to both metrics on this function.

D. Results on Variants of Trap Function

For the “Shifted” trap function, the performances of the GA and SMA were not significantly different from those on the unshifted version. This was because these algorithms solved the sub-problems independently and so were “blind” to whether the optimal string for each was different.

The COMA results exhibited the same pattern of behaviour noted above; fast, reliable problem solving for all but 1-COMA and 2-COMA, and even for these two the AES results were statistically significantly better than GA or SMA.

Considering Dist-Trap, the GA, SMA and Rand-COMA failed to solve the function to optimality in any run, regardless

of population size. While the Success Rate for COMA was less than for the other problems (typically 10-15/20 for pop. size 100 and 15-20/20 for pop. size 250), the same pattern was observed of better performance (SR and AES) for the adaptive version and fixed rule lengths in the range 3-5, tailing off at the extremes of the length range.

E. Evolution of Rule Base

The results above are promising from the point of view of improved optimisation performance, but require some analysis and explanation. The deceptive functions used were specifically chosen because GA theory suggests they are best solved by finding and mixing optimal solutions to sub-problems. Thus the GA failed to solve the function when the crossover operator was not suited to the representation (Dist-Trap).

Considering the action of a single bit-flipping LS operator on these “trap” subproblems, a search of the Hamming neighbourhood of a solution will always lead towards the sub-optimal solution when the unitation is 0,1 or 2, regardless of pivot rule. Additionally, the greedy search of the neighbourhood will lead towards the deceptive optimum 75% of the time when the unitation is 3. This explains the poor results of the SMA, and 1-COMA algorithms.

The behaviour of the A-COMA algorithm was examined by plotting the population mean against time of the rule length, the specificity of the condition (the proportion of values set to #), and the unitation of the action.

For the 4Trap function, the system rapidly evolved medium length (3 – 4), general (specificity < 50%) rules whose action was to set all the bits to 1 (mean unitation 100%). Closer inspection of the evolving rule-base confirmed that the optimal subproblem string was being learnt and applied.

For the Shifted-Trap function, where the optimal sub-blocks are all different, the rule length decreased more slowly from its initial mean value of 8. The specificity also remained higher, and the mean unitation remained at 50%, indicating that different rules were being maintained. This was borne out by closer examination of the evolved rule sets.

The behaviour on Dist-Trap was similar to that on 4Trap, albeit over a longer time-scale. The algorithm could not possibly be learning specific rules about sub-problems, since no rule was able to affect more than one locus of any subproblem. Rather, the system learnt the general rule of setting all bits to 1. The rules were generally shorter than for 4Trap, which means that the number of potential neighbours was higher for any given rule. The high incidence of #s meant that the rule length defined a maximum radius in Hamming space for the neighbourhood, rather than a fixed distance from the original solution. These two observations, together with the longer times to solution, suggest that when the system was unable to find a single rule that matched the problems’ structure, a more diverse search took place using a more complex neighbourhood which slowly adapted itself to the state of the current population of solutions. Full details of these experiments and analysis may be found in [4].

F. Benchmark Test: Protein Structure Prediction

Although these results were promising in themselves, the test problems used deliberately leaned heavily towards the “building block hypothesis” school of thought in their design. Accordingly in [5] and [6] the A-COMA algorithm was benchmarked against the standard GA and SMA on a well known combinatorial optimisation problem. The Protein Structure Prediction (PSP) problem concerns the prediction of the “native” three-dimensional form of a protein from knowledge of the sequence of its constituent amino-acid residues. Dill’s HP model [46] provides an estimate of the free energy of a fold of a given instance, based on the summation of pairwise interactions between the amino acid residues. It is a “virtual residue” model, that is to say that each amino acid residue is modelled by a single atom, whose properties are reduced to a quality of being hydrophobic (H) or hydrophilic (P). Thus a sequence of l amino acid residues is represented by $s \in \{H, P\}^l$, and the space of valid conformations is restricted to self-avoiding paths on a selected lattice, with each amino acid located on a vertex. Hydrophobic units that are adjacent in the lattice but non-adjacent in the sequence add a constant negative factor to the energy level. All other interactions are ignored, and a fixed penalty is added to infeasible folds. Despite its apparent simplicity, finding the global minimum of the HP model for a given sequence has been shown to be NP-complete on various lattices [47], and EAs have been widely applied since [48].

Twenty instances and parameter settings were taken from [10], which use a two-dimensional triangular lattice. For each combination of algorithm and instance, 25 runs were made, each continued until the global optimum was reached, subject to a maximum of 1 million evaluations. Following [49], the representation used a *relative* encoding where alleles come from the set $\{\textit{leftback}, \textit{leftforward}, \textit{front}, \textit{rightforward}, \textit{rightback}\}$ and represent the direction of the next move on the lattice from the point of view of the head of the growing chain. The generational genetic algorithm used (500+500) selection. One Point Crossover was applied with probability 0.8 and a Double Mutation (which has the effect of causing the mutation point to act as a pivot) was made with probability 0.3. All other settings for mutation probabilities etc. were as above.

In addition to the GA, and the simple Memetic Algorithm (SMA), we tested versions of COMA using a randomly created rules (i.e. with the learning disabled) and steepest (SRand) or greedy (GRand) pivot rules, and fully linked adaptive versions of COMA with the two pivot rules (CLS, CLG). These results are analysed according to effectiveness (SR) and efficiency (AES) as before, plus the mean best value found (MBF).

G. Results on PSP

From a total of 500 runs, the Success Rates in descending order were CLS (337), SMA (202), CLG (127), GRand (120), GA (83) and SRand (44). Using a non-parametric Friedman’s test for k-related variables shows that the differences in success rate between algorithms was significant. A series of paired t-tests confirmed that the results for the CLS algorithm were significantly better than any of the others with over 95%

confidence. This difference was particularly noticeable on the longer instances. Of the other results, the simple MA (SMA) performed well on the shorter instances, and the CLG and GRandom results were surprisingly similar, possibly because the noise inherent in the greedy ascent mechanism created problems for CLG’s credit assignment mechanism. Significantly, whatever the form of the local search, all but one of the Memetic Algorithms perform much better than the simple GA.

The least successful algorithm was SRand, and even when it was successful, it took far longer than all of the other algorithms. Like GRandom, it used randomly created rules to define the neighbourhood for each solution in each generation. However, unlike the GRandom it searched the whole of each neighbourhood. Since many of the random rules would be short or have quite low specificity, this causes large neighbourhoods and a consequent increase in the AES values for the same number of evolved candidate solutions considered. It is possible that left to run for longer, the Success Rate of the SRand would have been improved.

Of the others, the GA was always fastest, followed by the SMA. The greedy COMA algorithms were faster than their steepest ascent counterparts. A two way Analysis of Variance showed that both instance and algorithm were significant factors. Post-hoc analysis using Tamhane’s T2 test confirmed the ordering $GA < \{SMA, CLG\} \leq \{GRand, CLS\} < SRand$, with 95% confidence where $<$ is used and 90% confidence where \leq is used between groups. Ordering within groups is by changing values but these were not significant at these confidence levels.

Most algorithms found the global optimum for the shorter instances. Therefore when comparing performance on the basis of the quality of the best solutions found, i.e., mean best fitness (MBF), only results for the longer and harder instances were considered. It was observed that CLS reached consistently higher values and with a smaller variance in performance than the others, and that the SRand algorithm was correspondingly worse. A two-way ANOVA test on the values for the best solution found in each run, with instance number and algorithm as the factors confirmed the significance of the algorithm in determining the performance. Post-hoc testing using Tamhane’s T2 test at the 95% confidence level gave the groupings $CLS > \{CLG, SMA, GRand\} > GA > SRand$.

H. Analysis of Meme Evolution on PSP

A number of test runs of CLS were made in which the evolving memes were saved at regular intervals. These showed a strong tendency towards short rules of the form $\{\#\#\rightarrow leftback rightback\}$ or $\{\#\#\rightarrow leftback leftforward\}$. On a two-dimensional triangular lattice both of these rules act to bring residues i and $i+2$ into contact, and these patterns could be thought of as the two-dimensional equivalent of representing a single turn of an alpha helix. Experimentation on a square two-dimensional lattice showed a tendency towards rules of the form $\{\#\#\#\rightarrow lll\}$ or $\{\#\#\#\rightarrow rrr\}$, also the shortest path that brings two residues into contact.

The use of the word “tendency” should be noted here: in most cases the rule-set continued to contain a number of

different rules of varying lengths. It has been argued above that in addition to the extra scalability attained by identifying and re-applying regular structural motifs, the presence of a diverse, evolving rule-set means that the neighbourhood structure defining which points around the current population are examined, is continuously changing. Thus, even if the population is converged to a single point, which is locally optimal according to most neighbourhood structures, eventually a rule may evolve for which the neighbourhood of that point contains a fitter solution. This can be thought of as continually testing new search landscapes to look for “escape routes” from local optima.

Looking back to the results for the GRandom algorithm, in which the rules defining neighbourhoods were created at random, this “changing landscape” effect was noticeable in the superior success rates to the SMA. The fact that the CLS algorithm was the best performer according to both Success Rate and MBF metrics points to both modes of operation having a positive effect.

V. EXTENSION TO TRUE CO-EVOLUTION

Having established the basic principle of evolving memes which coded for LS rules as a means of enhancing optimisation performance in MAs, the next series of experiments used a full co-evolutionary model. The aims were to explore the effects of different pivot rules and linkage strategies, and to test the hypothesis of the different modes of operation suggested above. Again the benchmarks for comparison used a GA, and a Simple Memetic Algorithm (SMA) employing a bit-flipping greedy hill-climber. COMA with adaptive rule lengths was used with all possible combinations of greedy or steepest ascent with linked, random or fitness-based pairing strategies. In the latter case memes were selected to be used via binary tournaments. The fitness of each meme taken to be $\Delta fitness$, i.e., the fitness improvement caused when it was last applied to a candidate solution. The implications of this credit assignment strategy will be discussed later. These algorithms are referred to as CXY where X comes from the set $\{L(inked), R(andom), T(ournament)\}$ and Y is one of G (Greedy) or S (Steepest). In order to tease out statistical significance on the harder functions used here, 50 runs were made of each algorithm on each problem instance, with a population size of 500. All other experimental settings were the same.

A. Exploiting search space regularities gives scalability

The results summarised above suggested that COMA was able identify and utilise regularities in the problem space. This was investigated further using multiple length variants of two well understood test functions. The first of these comprised multiple concatenated copies of (1) with lengths in the range $\{40, 60, 80, \dots, 200\}$.

As expected from above, the results for SMA were extremely poor. The next worst algorithm was CRS. Out of the 50 runs for each size, the SR steadily decreased 50 at length 40 to 5 at length 100 and zero above that. All the other algorithms showed SR of 49 or 50 up to length 160, but only the CLS (39) CLG (50) and CTG (50) solved the 200-bit problem.

The AES results were revealing. The GA was faster than Grand and SRand, but the increase in AES with length was worse than linear. The AES results of the successful COMA variants, and analysis of the evolving rule bases, supported the hypothesis of discovering and exploiting regularities. In this case it meant identifying a rule giving the optimal solution to the sub-problems, and then applying it to each sub-problem in the string in successive generations. CLG was the fastest algorithm, followed by CTS and CLS, and all three were near-linear. For example, a linear regression of AES to length for CLG fitted the data with a correlation co-efficient of 0.97.

The second test function was Watson’s H-IFF function, a highly epistatic problem designed to examine the virtues of recombination. At the bottom level, fitness is awarded to matching pairs of adjacent bits in a solution s , i.e.

$$f_1 s = \sum_{i=0}^{l/2-1} 1 - XOR(s_{2i}, s_{2i+1}) \quad (2)$$

and this process is applied recursively, so that a problem of size $l = 2^k$ has k levels. In each ascending level the number of blocks is reduced by a factor of two, and the fitness awarded for each matching pair is increased by a constant factor, in our case 2. This problem has a number of Hamming sub-optima, and two global optima corresponding to the $u(i) \in \{0, 1\}$. Problem sizes $l \in \{8, 16, \dots, 512\}$ were used, corresponding to 3 to 9 levels. Note that for $l > 16$ the length of the blocks to be identified and matched at the highest levels far exceeded the maximum rule length.

All of the MAs had higher Success Rates than the GA, and again the same three variants of COMA exhibited the best performance. For example, out of 50 runs with $l = 128$ the SR values were 0 (GA, CRG, CRS), 2 (CTG), 4 (MA), 38 (CLG), 43 (CLS) 49 (CTS). Only the CLG (10) and CTS (11) variants solved the 256 bit problem. Considering different pivot rules, the same pattern was observed as on other problems: the greedy ascent versions found the optimum faster (lower AES) than the equivalent steepest ascent versions but not as reliably (lower SR). ANOVA on the MBF results confirmed that the performance was statistically significantly different with 95% confidence. Post-hoc analysis showed that the CLG, CLS and CTS variants had a higher mean best fitness than all other algorithms but did not significantly differ.

Analysis of the evolving rule base suggested that for the H-IFF problem the improved scalability arose from the system to making a decision between the “ones” and “zeroes” blocks and then applying this throughout the string. The choice between 1s and 0s appeared to occur with equal probability.

B. Escaping local optima by changing neighbourhoods

The performance improvements exhibited above clearly arise from a situation in which the adjacent epistatic interactions within the problem give rise to patterns in the search space which can be exploited by COMA. The two 64 bit variants of the 4Trap function described above were used to examine the behaviour when this is not the case. As reported above, the SMA and GA completely failed to solve the DistTrap functions, and this situation did not change by

allowing more runs. The only algorithms which ever located the global optimum were the CLG, CLS and CTS, all three of which always did so. The AES ranking was $CLG < CTS < CLS$, these differences significant with 99.9% confidence.

On the Shifted-Trap function, the SMA found the optimum 45 times out of the 50 runs, and the random-pairing steepest ascent version of COMA (CRS) 39 times. The GA and all other variants of COMA always located the global optimum in the time allowed. It might be expected that attempting to reuse a pattern on different sub-problems would hinder the progress of the COMA algorithms. However in fact the mean solution time was not significantly different to that of the GA for all but SMA and the CRS variant, and there was a noticeable reduction in the variability of time to solution.

C. Choice of Pairing and Pivot Strategies in Coevolution

The results presented above showed that the choice of pivot and pairing strategies is crucially important and intertwined.

Unsurprisingly, the greedy variants almost always used less evaluations than the steepest ascent equivalents on successful runs. One obvious explanation for this is that when a “good rule” (e.g. #### \rightarrow 1111 for 4Trap) is applied to a candidate solution, it will often match in a number of places, and cause the same fitness improvement in several. Unlike memes with a greedy pivot rule, which stop evaluating neighbours as soon as the first improvement is found, those with a steepest pivot rule will evaluate the whole neighbourhood. Thus the number of evaluations used to achieve the same improvement, and hence the AES, will be higher.

In the case of random pairing, there is no selective pressure in the meme population, so the rule base will remain diverse until genetic drift causes an eventual convergence. For the Shifted-Trap function this was not a problem, since it is desirable to maintain different *condition* parts of the rules for different sub-problems, and CRG performed well. However for CRS, and CRG on the other other problems it prevented identification and use of appropriate rules, and a corresponding decrease in performance was observed.

In the linked variants, the selective pressure towards the evolution of good rules is created implicitly via a continued association with fit solutions, and CLG outperformed CLS on the binary problems (but not PSP).

By contrast, for some problems (but not all) the extra noise introduced by using a greedy ascent was sufficient to “fool” the simple credit assignment mechanism used in these experiments. Thus a good rule will only get a low fitness if the first match only leads to a small improvement, whereas larger improvement (and hence fitness) might be seen if it was applied elsewhere in the solution. Another source of noise is the choice of partner. It is possible that a more sophisticated method such as Paredis Life Time Fitness Evaluation (LTFE) (in which a running average of the last twenty pairings is used), or Parker and Blumenthal’s PAL with Sampling may well provide a more stable and robust credit assignment mechanism whilst retaining the speed-benefits of a greedy ascent pivot rule. However these would not be simple to use in our situation with memes evolving at the same rate as the solutions unless the selection pressure were much greater.

VI. EXTENDING ADAPTABILITY AND BENCHMARK COMPARISON 2: MAX-3SAT

A. Problem Description

The Max-SAT problem is a classical combinatorial optimisation problem, consisting of a number of Boolean variables and a set of clauses built from those variables. A full description and many examples can be found in [50]. For each length $\{50,100,250\}$ the first 25 were taken from the sets of uniformly randomly created satisfiable instances around the phase transition (in terms of hardness) where there are approximately 4.3 clauses per variable. The same experimental set up was used as for the previous experiments, with a maximum of 500,000 evaluations per run. Each algorithm was run ten times on each instance, giving 250 runs for each combination of algorithm and length.

As well as the GA and simple MA with steepest (SMA-S) and greedy (SMA-G) ascent, COMA variants with linked or fitness-based pairing were used with fixed pivot strategies as before, and also with the pivot condition in the rules randomly initialised and subject to mutation at a rate 0.0625 (CLA/CTA). In addition each algorithm was run with and without the ability to use a # symbol in the action part of the rule, which was taken to denote that the symbol in the solution should be inverted. This is denoted as CXY-I and provides the ability to create the rule $\# \rightarrow \#$ - equivalent to a simple MA.

It should be noted at the outset that the random way in which these instances are created does not provide any structural regularities for the COMA algorithms to exploit, so the second “changing neighbourhood” modus operandi might be expected.

B. Success Rate

Table I shows the number of success from 250 runs. No algorithm found a solution for the problems with 250 variables, so it is perhaps worth re-iterating that no time was spent tuning the underlying GA.

As can be seen, for the 50 variable instances the simple MAs have the highest success rates, followed by the variants of COMA with linked rule pairing. For the longer instances all methods are much less successful, and many instances are not solved by any algorithms. SMA-G and CLG-I show the same performance

For the shorter instances the steepest ascent strategy is better, but for the longer instances the cost of searching the entire neighbourhood every iteration becomes prohibitive, so that SMA-S and CLS solve no instances. It is immediately apparent that the adaptive variants CLA and CTA appear to perform on a par with whichever of the S or G variants is better for each length, suggesting successful adaptation.

The effect of the ability to use inversion is less clear cut, but it yields a slight improvement on the longer instances.

C. Efficiency

Figure 2 shows the mean time to solution analysed by algorithm and problem size. ANOVA on just the COMA results showed that the ability to use inversion was not a

Algorithm	Length 50	Length 100
GA	125	21
SMA-S	154	0
SMA-G	153	25
CLS	141	0
CLS-I	141	3
CLG	135	21
CLG-I	136	25
CLA	144	8
CLA-I	142	8
CTS	112	10
CTS-I	131	12
CTG	117	19
CTG-I	118	20
CTA	119	14
CTA-I	126	19

TABLE I
NUMBER OF RUNS (OUT OF 250) IN WHICH A SOLUTION WAS IDENTIFIED
FOR DIFFERENT LENGTH MAX-3SAT PROBLEMS.

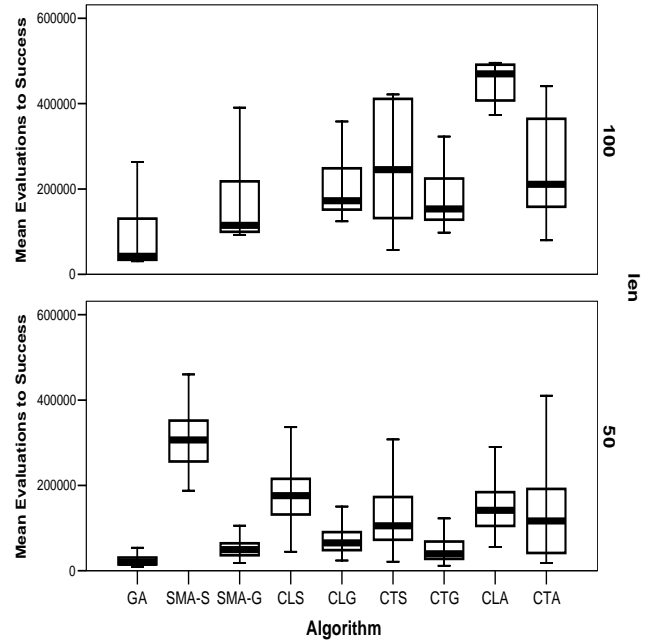


Fig. 2. Box plots of Average Evaluations to Success for 10 runs on each of 25 MAX-3SAT instances with 100 (top) and 50 (bottom) variables.

significant factor on the AES, so these results are omitted from the figure for clarity. As can be seen the GA is the fastest algorithm followed by a close grouping of SMA-G, CLG and CTG, with the CLS and CTS variants taking more time and having a higher variance. The adaptive pivot variants both fall between their respective greedy and steepest counterparts, both in terms of mean and variance. The fitness based pairing is faster than the linked counterpart in every case. A two-way ANOVA with problem size and algorithm as fixed factors showed that the algorithm was a determining factor in AES. Post-hoc testing using Tamhane’s test at the 95% confidence level showed that the GA was significantly faster, and the SMA-S significantly slower than the other algorithms, but otherwise the grouping was not well defined, most algorithms being not significantly different to three or more others.

D. Mean Best Fitness

Figure 3 shows the mean best fitness analysed by algorithm for the longer instances. As noted above, the performance of the steepest ascent versions of any of the algorithms was worse than the greedy or adaptive versions, and so these are omitted for clarity. Analysis show no significant difference between the means for linked and fitness based pairings (CTG–CLG, CTA–CLA etc.) for this performance indicator, although the fitness-based pairing algorithms tend to have higher variance than their linked pairing counterparts. Again the use of inversion was not a significant factor for these results.

Overall, adding the possible of an “invert” symbol (#) to the action has clearly improved the success rate, and not been detrimental to the other measures. The variants with adaptive pivot rules are outperformed by the better of steepest or greedy ascent, but this is often marginal. In particular, if the most important criteria to the algorithm designer is reliability (high SR), then letting the pivot rules evolve is clearly preferable to making a fixed choice. On some problems steepest was better, on others greedy, and on the MAX-3SAT problems the choice is size-dependant. In contrast to this, if the desired properties are speed (low AES) and consistently good results (high MBF), then the fixed greedy strategy is preferable on the MAX-3SAT, although again this does not always hold for other problems.

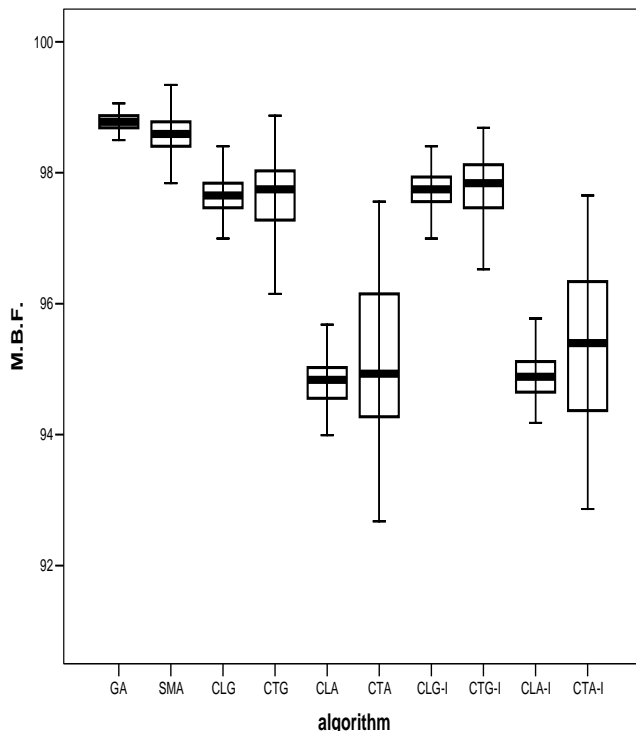


Fig. 3. Box plots of best values found 10 runs on each of 25 MAX-3SAT instances with 250 variables.

VII. CONCLUSIONS

The premise that evolutionary optimisation algorithms can be improved by incorporating an appropriate local search

mechanism is now widely accepted, but it is increasingly recognised that the choice of move operator, and hence neighbourhood function, used in the local search is crucial to delivering success.

This paper reviewed a framework which offers the promise for creating robust scalable optimisation techniques, based on the concept of co-evolving memes encoding definitions of Local Search operators. Results were reviewed showing the development of algorithms fitting into this framework with progressively more co-evolutionary natures and more degrees of freedom available to the evolution of memes. These results are highly competitive on a variety of different classes of problem. These experiments have used a number of different performance metrics to compare between algorithms, as different intended uses of EAs have different goals. In some repeated tasks, reliably fast and reasonably accurate behaviour (i.e. low AES and high MBF) may be more important than finding the true global optimum at whatever cost (e.g. high SR and AES), but the reverse may be true in e.g. design situations.

The results obtained illustrate that performance improvement can arise from different mechanisms. If the representation of the rules is able to capture regular repeated features within the problem space, then highly scalable behaviour is exhibited - for example the linear speed-up on the 4trap function. This arises from the rapid evolution of the system to a rule-set which captures and represents knowledge about how to solve the problem. It was noted that in order for this to occur it is necessary to maintain sufficient accurate selection pressure within the population of Local Searchers.

In contrast to this, when there is not sufficient selective pressure for evolution a “fall-back” position is observed. This might occur for example when a “good” pattern only applies to one position in the solution, or when the rule representation cannot possibly capture the regularities present in the space. In these cases improved reliability is observed, but at the expense of speed of solution. Essentially what happens is that even if the solution population converges to a local optimum for its crossover and mutation operators, the continued evolution of memes means that eventually a Local Search landscape is discovered in which the solutions are not locally optimal and improvements can occur. This is akin to Variable Neighbourhood Search or other Adaptive Memetic Algorithms, but with the advantage that it is not necessary to specify, or be bound by, a fixed set of neighbourhood functions.

These results from a series of benchmark tests highlight this problem of choosing the appropriate local search operator which provided the original rationale for the development of COMA. For example, although the Memetic Algorithm with a simple bit-flipping hill climber had the highest Success Rates and Mean Best Fitness on the Max-3SAT problems, it’s performance on the other problems was derisory, and frequently worse than the simple GA.

The COMA variants using “linked” pairing, which effectively self-adapt the memes, exhibited better performance than the GA or SMA over a wide range of problems, according to different metrics. The results on MAX-3SAT were comparable with SMA, but this problem is perhaps unusual in having no structure to be exploited. Other authors have suggested

methods for re-ordering MAX-SAT representations so as to maximise local gene interactions, which would clearly aid the rule-based COMA algorithms, as probably would allowing more evaluations to try different neighbourhoods.

When the behaviour of the truly co-evolutionary models with fitness-based selection of memes (CTS and CTG) is considered, the picture is less clear. On the H-IFF and multiple 4Trap problems the method works well, particularly when used with the steepest ascent pivot rule. However on the MAX-3SAT problem the situation is reversed, and the greedy ascent rule appears to work better. As suggested, this may be because the neighbourhoods are so large, or because the fitness measure used for memes, was too simple. The use of adaptive pivot rules goes a long way towards resolving the issue of choosing a pivot rule, but there is clearly scope for future work here, and it is worth considering a change from what Ong et. al. would term local to global fitness measures, such as running averages etc.

Clearly there remains much work to be done analysing the possibilities of this framework. It would be fatuous to claim that COMA represents some fabulous all-purpose problem solver, however promising these results. Immediate priorities are the investigation of alternative methods for credit assignment within the meme population, and the extension to more generic representations of conditions and actions. However the two modes of operation noted above, coupled with the ability of the algorithm to explicitly represent the information that it has learnt and is using to solve the problem at hand, would seem to offer much potential.

ACKNOWLEDGMENT

The author would like to thank Natalio Krasnogor for many fruitful discussions during the initial stages of this work.

REFERENCES

- [1] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.
- [2] W. Hart, N. Krasnogor, and J. Smith, Eds., *Recent Advances in Memetic Algorithms*. Springer, Berlin, Heidelberg, New York, 2004.
- [3] P. Moscato, "Memetic algorithms" home page, visited July 2003: http://www.densis.fee.unicamp.br/~moscato/memetic_home.html.
- [4] J. Smith, "Co-evolution of memetic algorithms: Initial investigations," in *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, J. M. Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, Eds., no. 2439. Springer, Berlin, Heidelberg, New York, 2002, pp. 537–548.
- [5] —, "Protein structure prediction with co-evolving memetic algorithms," in *2003 Congress on Evolutionary Computation (CEC'2003)*. IEEE Press, Piscataway, NJ, 2003, pp. 2346–2353.
- [6] —, "The co-evolution of memetic algorithms for protein structure prediction," in *Recent Advances in Memetic Algorithms*, W. Hart, N. Krasnogor, and J. Smith, Eds. Springer, Berlin, Heidelberg, New York, 2004, pp. 105–128.
- [7] —, "Co-evolving memetic algorithms: A learning approach to robust scalable optimisation," in *2003 Congress on Evolutionary Computation (CEC'2003)*. IEEE Press, Piscataway, NJ, 2003, pp. 498–505.
- [8] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," *IEEE Transactions on Systems Man and Cybernetics Part B*, vol. 36, no. 1, 2006.
- [9] N. Krasnogor, "Coevolution of genes and memes in memetic algorithms," in *Proceedings of the 1999 Genetic and Evolutionary Computation Conference Workshop Program*, A. Wu, Ed., 1999.
- [10] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Morgan Kaufmann, San Francisco, 2000, pp. 987–994.
- [11] —, "Emergence of profitable search strategies based on a simple inheritance mechanism," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, Eds. Morgan Kaufmann, San Francisco, 2001, pp. 432–439.
- [12] N. Krasnogor, "Studies in the theory and design space of memetic algorithms," Ph.D. dissertation, University of the West of England, 2002.
- [13] Y. Ong and A. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.
- [14] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," *Lecture Notes in Computer Science*, vol. 2079, pp. 176–95, 2001.
- [15] E. Burke and A. Smith, "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 122–128, 2000.
- [16] G. Kendall, P. Cowling, and E. Soubeiga, "Choice function and random hyperheuristics," in *Proceedings of Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL)*, 2002, pp. 667–671.
- [17] E. Burke, G. Kendall, and E. Soubeiga, "A tabu search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, 2003.
- [18] N. Krasnogor, "Self-generating metaheuristics in bioinformatics: The protein structure comparison case," *Genetic Programming and Evolvable Machines. Kluwer academic Publishers*, vol. 5, no. 2, pp. 181–201, 2004.
- [19] N. Krasnogor and S. Gustafson, "A study on the use of "self-generation" in memetic algorithms," *Natural Computing*, vol. 3, no. 1, pp. 53–76, 2004.
- [20] P. Hansen and N. Mladenović, "An introduction to variable neighborhood search," in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Proceedings of MIC 97 Conference*, S. Voß, S. Martello, I. Osman, and C. Roucairol, Eds. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998.
- [21] J. Smith and T. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, vol. 1, no. 2, pp. 81–87, 1997.
- [22] R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: A survey," in *Proceedings of the 1997 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1997.
- [23] A. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.
- [24] H.-P. Schwefel, *Numerical Optimisation of Computer Models*. Wiley, New York, 1981.
- [25] D. Fogel, "Evolving artificial intelligence," Ph.D. dissertation, University of California, 1992.
- [26] T. Bäck, "Self adaptation in genetic algorithms," in *Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, F. Varela and P. Bourguine, Eds. MIT Press, Cambridge, MA, 1992, pp. 263–271.
- [27] J. Smith and T. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," in *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*. IEEE Press, Piscataway, NJ, 1996, pp. 318–323.
- [28] J. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*, J. Grefenstette, Ed. Lawrence Erlbaum, Hillsdale, New Jersey, 1987, pp. 36–40.
- [29] J. Smith and T. Fogarty, "Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm," in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds., no. 1141. Springer, Berlin, Heidelberg, New York, 1996, pp. 441–450.
- [30] J. Paredis, "The symbiotic evolution of solutions and their representations," in *Proceedings of the 6th International Conference on Genetic Algorithms*, L. Eshelman, Ed. Morgan Kaufmann, San Francisco, 1995, pp. 359–365.
- [31] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimisation," in *Proceedings of the 3rd Conference on Parallel*

- Problem Solving from Nature*, ser. Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds., no. 866. Springer, Berlin, Heidelberg, New York, 1994, pp. 248–257.
- [32] L. Bull, “Artificial symbiology,” Ph.D. dissertation, University of the West of England, 1995.
- [33] S. Kauffman, *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, NY, 1993.
- [34] L. Bull and T. Fogarty, “Horizontal gene transfer in endosymbiosis,” in *Proceedings of the 5th International Workshop on Artificial Life : Synthesis and Simulation of Living Systems (ALIFE-96)*, C. Langton and K. Shimohara, Eds. MIT Press, Cambridge, MA, 1997, pp. 77–84.
- [35] L. Bull, “Evolutionary computing in multi agent environments: Partners,” in *Proceedings of the 7th International Conference on Genetic Algorithms*, T. Bäck, Ed. Morgan Kaufmann, San Francisco, 1997, pp. 370–377.
- [36] G. Parker and H. Blumenthal, “Varying sample sizes for the co-evolution of heterogeneous agents,” in *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*. IEEE Press, 2004, pp. 766–771.
- [37] J. Paredis, “Coevolutionary algorithms,” in *Handbook of Evolutionary Computation*, T. Bäck, D. Fogel, and Z. Michalewicz, Eds. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1998.
- [38] S. Luke and L. Spector, “Evolving teamwork and coordination with genetic programming,” in *Proceedings of the 1st Annual Conference on Genetic Programming*, J. Koza, D. Goldberg, D. Fogel, and R. Riolo, Eds. MIT Press, Cambridge, MA, 1996, pp. 141–149.
- [39] H. Kargupta and S. Ghosh, “Towards machine learning through genetic code-like transformations,” Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Tech. Rep. TR-CS-01-10, 2001.
- [40] R. Keller and W. Banzhaf, “Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes,” in *Proceedings of the 1st Annual Conference on Genetic Programming*, J. Koza, D. Goldberg, D. Fogel, and R. Riolo, Eds. MIT Press, Cambridge, MA, 1996, pp. 116–122.
- [41] —, “The evolution of genetic code in genetic programming,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, and R. Smith, Eds. Morgan Kaufmann, San Francisco, 1999, pp. 1077–1082.
- [42] W. Hart, “Adaptive global optimization with local search,” Ph.D. dissertation, University of California, San Diego, 1994.
- [43] T. Jones, “Evolutionary algorithms, fitness landscapes and search,” Ph.D. dissertation, The University of New Mexico, Albuquerque, NM, 1995.
- [44] P. Merz and B. Freisleben, “Fitness landscapes and memetic algorithm design,” in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. McGraw Hill, London, 1999, pp. 245–260.
- [45] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1997.
- [46] K. Dill, “Theory for the folding and stability of globular proteins,” *Biochemistry*, vol. 24, p. 1501, 1985.
- [47] B. Berger and T. Leight, “Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete,” in *Proc. 2nd Annual Intl. Conf. Computational Molecular Biology RECOMB98*, 1998.
- [48] R. Unger and J. Moult, “Genetic algorithms for protein folding simulations,” *Journal of Theoretical Biology*, vol. 231, no. 1, pp. 75–81, 1993.
- [49] N. Krasnogor, W. Hart, J. Smith, and D. Pelta, “Protein structure prediction with evolutionary algorithms,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, and R. Smith, Eds. Morgan Kaufmann, San Francisco, 1999, pp. 1596–1601.
- [50] “Satlib: <http://www.satlib.org>.”