The Co-Evolution of Memetic Algorithms for Protein Structure Prediction

J.E. Smith

Faculty of Computing, Engineering and Mathematical Sciences,
University of the West of England,
Bristol BS16 12QY, U.K.
james.smith@uwe.ac.uk
http://www.cems.uwe.ac.uk/~jsmith

Summary. This paper describes a co-evolutionary learning-optimisation approach to Protein Structure Prediction which uses a Memetic Algorithm as its underlying search method. Instance-specific knowledge can be learned, stored and applied by the system in the form of a population of rules. These rules determine the neighbourhoods used by the local search process, which is applied to each member of the co-evolving population of candidate solutions.

A generic co-evolutionary framework is proposed for this approach, and the implementation of a simple Self-Adaptive instantiation is described. A rule defining the local search's move operator is encoded as a $\{condition : action\}$ pair and added to the genotype of each individual. It is demonstrated that the action of mutation and crossover on the patterns encoded in these rules, coupled with the action of selection on the resultant phenotypes is sufficient to permit the discovery and propagation of knowledge about the instance being optimised.

The algorithm is benchmarked against a simple Genetic Algorithm, a Memetic Algorithm using a fixed neighbourhood function, and a similar Memetic Algorithm which uses random (rather than evolved) rules and shows significant improvements in terms of the ability to locate optimum configurations using Dill's HP model. It is shown that this "meta-learning" of problem features provides a means of creating highly scalable algorithms.

1 Introduction

The performance benefits which can be achieved by hybridising evolutionary algorithms (EAs) with local search operators, so-called *Memetic Algorithms* (MAs), have now been well documented across a wide range of problem domains such as combinatorial optimisation [27], optimisation of non-stationary functions [42], and multi-objective optimisation [20] (see [29] 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 absent, or the order in which the operators are applied may vary. The local search step can be illustrated by the pseudo-code below:

```
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 is satisfied ) Do
      generate the next neighbour $j \in n(i)$;
      set $counter = counter + 1$;
      If ($f(j)$ is better than $f(best)$) Then
        set $best = j$;
      endIf
    endDo
    set $i = best$;
    set $iterations = iterations + 1$;
  endDo
End.
```

There are three principal components which affect the workings of this local search. 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, i.e. $counter = |n(i)|$, 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 $N << |n(i)|$ if the neighbourhood is too large to search.

The second component is the depth of the local search, i.e. the iteration condition which 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. [14]. Along with the choice of pivot rule, it can be shown to have an effect on the performance of the Local Search algorithm, both in terms of time taken, and in the quality of solution found.

The third, and primary factor that affects the behaviour of the local search 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. An equivalent 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" [15]. Merz and Freisleben [28] 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.

In some cases, domain specific information may be used to guide the choice of neighbourhood structure within the local search algorithms. However, it has recently been shown that the optimal choice of operators can be not only instance specific within a class of problems [28, pp254–258], but also dependent on the state of the evolutionary search [26]. This result is not surprising when we consider that 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). Thus if a set of points has converged to the state where all are locally optimal with respect to the current neighbourhood operator, then changing the neighbourhood operator may provide a means of progression, in addition to recombination and mutation. This observation forms the heart of the *Variable Neighbourhood Search* algorithm [49].

This paper describes one mechanism whereby the definitions of local search operators applied within the MA may be changed during the course of optimisation, and in particular how this system may usefully be applied to a simplified model of the Protein Structure Prediction Problem. This system is called Co-evolving Memetic Algorithms (COMA). The rest of this paper proceeds as follows:

- Section 2 discusses some previous work in this area, describes the proposed approach, and the development of a simplified model within that framework. It also summarises the results of initial investigations published elsewhere.
- Section 3 draws some parallels between this work and related work in different fields, in order to place this work within the context of more general studies into adaptation, development and learning.
- Section 4 details the particular application under concern, namely Protein Structure Prediction using Dill's HP model [8].
- Section 5 presents the results and analysis of a set of preliminary experiments designed to investigate whether the use of adaptive rules is able to benefit the optimisation process.
- Section 6 goes on to investigate the benefits of restricting the search to feasible solutions, rather than using a penalty function approach.
- Section 7 presents some analyses of the behaviour of the evolving rule-bases, and then Section 6 discusses the implications of these results, before drawing conclusions and suggesting future work.

2 A Rule-Based Model for the Adaptation of Move Operators

2.1 The Model

The aim of this work is to provide a means whereby the definition of the local search operator (LSO) 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 LSO in a form that can be processed by an evolutionary algorithm
- Intimately related to this, a set of variation operators, such as recombination and mutation that can be applied to the LSO representation, and a means for initialising a population of LSO operators.
- A means of assigning fitness to the LSO population members
- A choice of population structures and sizes, along with selection and replacement methods for managing the LSO population
- A set of experiments, problems and measurements designed to permit evaluation and analysis of the behaviour of the system.

The representation chosen for the LSOs is a tuple *<Pivot_Rule, Depth, Pairing, Move, Fitness>*.

The first two elements 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* effectively co-ordinates the evolution of the two populations. When a candidate solution is to be evaluated, a member of the LSO population is chosen to operate on it, hopefully yielding improvements. The fitness of the candidate solution is thus affected by the choice of LSO to operate on it, and the fitness assigned to the LSO is in turn affected by the candidate solution to which it is applied.

Values for *Pairing* are taken from the set {*linked, fitness_based, random*}. The purpose of this element is to allow the system to be varied between the extremes of a fully unlinked system, in which although still interacting the two populations evolve separately, and a fully linked system in which the LS operators can be considered to be self-adapted. The different values have the following effects:

- For a *linked* pairing strategy, the LSOs can be considered to be extra genetic material which is inherited and varied along with the problem representation. Thus if the k^{th} candidate solution is created from parents i and j, then a LSO is created by the actions of recombination and mutation on members i and j of the current LSO population. This new LSO is used to evaluate the new candidate solution and becomes the k^{th} member of the next LSO population. Note that this assumes the two population are the same size. The fitness is assigned to the new LSO is immaterial since selection to act as parents happens via association with good members of the solution population.

- For a *fitness-based* pairing strategy, when a candidate solution requires evaluation, a LSO is created and put into the next LSO population as above. However the two LSOs which acts as parents for recombination are now chosen using a standard selection mechanism acting on those members of the current LSO population which do not have *Pairing = linked.* A number of methods can be used to define the fitness of an LSO.
- For a *random* pairing strategy, the same process occurs as for the fitness-based method, except that parents are selected randomly from the unlinked members of the LSO population, without regard to their fitness.

Although the long-term goal is to examine a "mixed-economy" of paring strategies, for the purposes of this paper the system is restricted to the situation where the whole population uses the same value, initially *Pairing = linked.*

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 wildcard 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.

In themselves, the degrees of freedom afforded by the five components listed above provide basis for a major body of research, and the framework described above is intended to permit a full exploration of these issues which is currently underway [37, 36].

This paper presents results from a simplified instantiation of this framework, focusing on the benefits of knowledge discovery and re-use. In order to achieve this focus, some of the adaptive capabilities are restricted, i.e., the LSOs always use one of greedy or steepest ascent, a single improvement step, and full linkage. These choices are coded into the LSO chromosomes at initialisation, and variation operator are not used on them. This restriction to what are effectively self-adaptive systems provides a means of dealing with the credit assignment and population management issues noted above

The COMA system is also restricted to considering only rules where the *condition* and *action* patterns are of equal length and are 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*, although this could be interpreted as "leave alone". The neighbourhood of a point i then consists of all those points where the substring denoted by *condition* appears in the representation of i and is replaced by the *action*. The neighbourhood of i therefore potentially includes i itself, for example by means of a rule with identical *condition* and *action* parts.

To give an example, if a solution is represented by the binary string 1100111000 and a rule by 1#0:111, then this rule matches the first, second, sixth and seventh positions, and the neighbourhood is the set {1110111000, 11111111000, 1100111100,1100111110}. In practice a random permutation is used to specify the order in which the neighbours are evaluated, so as not to introduce positional bias into the local search when greedy ascent is used. Note that in this work the string is not considered as toroidal (although this will be considered in later work).

In practice, each rule was implemented as two 16 bit strings, and was augmented by a value *rule_length* which detailed the number of positions in the pattern string to consider. This permits 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 means that "standard" genetic operators (uniform/1 point crossover, point mutation) can be used to vary this part of the LSO chromosome.

2.2 Initial Results

The results of initial investigations using this system were reported in [37]. The test suite was problems made out of a number of sub-functions either interleaved or concatenated. Two different classes of sub-function were used which posed either entropic (Royal Road) or fitness (Deceptive) barriers to the discovery of the global optimum. Greedy versions of the COMA (GComa) algorithm were tested against the GA,MA, GRand algorithms described below, and it was shown that a version of the system with adaptive rule lengths was able to perform better than these three, and comparably with variants of GComa with optimal fixed rule-lengths for the different problems. Analysis showed that these algorithms discovered and used problem specific information (such as optimal patterns for different sub-problems).

Subsequent work [36] has shown them to be highly scalable with respect to problem length on problems where there are repeated patterns in the regions of the search space corresponding to high quality solutions. This behaviour arises from the discovery and re-use of knowledge about these patterns. It was also shown that in the absence of such patterns, the systems still displays better performance (both in terms of mean best fitness and the reliability of locating the global optimum). In this case the improved performance arose from the maintenance of a diverse set of move operators, and hence from the examination of multiple search landscapes, which provides a better means of escaping local optima.

3 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 it's behaviour. These range from MultiMemetic Algorithms and the Self-Adaptation of search strategies, through co-evolutionary, learning and developmental systems, to the evolutionary search for generalised rules as per Learning Classifier Systems. Space precludes a full discussion of each of these, so the more important are briefly outlined below.

Although the authors are not aware of other algorithms in which the LSOs used by an MA are adapted in this fashion, there are other examples of the use of multiple LS operators within evolutionary systems. Krasnogor and Smith [26] describe a "MultiMemetic Algorithm", in which a gene is added to the end of each chromosome indicating which of a fixed set of static LS operators ("memes") should be applied to the individual solution. Variation is provided during the mutation process, by randomly resetting this value with a low probability. They report that their systems are able to adapt to use the best meme available for different instances of TSP.

Krasnogor and Gustafson have extended this and proposed a grammar for "Self-Generating MAs" which specifies, for instance, where in the evolutionary cycle local search takes place [22]. Noting that each meme potentially defines a different neighbourhood function for the local search part of the MA, we can also see an obvious analogy to the Variable Neighbourhood Search algorithm [49], where a heuristic is used to control the order of application of a set of local searchers (using different, fixed, neighbourhood structures) to a single improving solution. The difference here lies in the population based nature of COMA, so that not only do we have multiple candidate solutions, but also multiple adaptive neighbourhood functions in the memes.

As noted above, if the populations are of the same size, and are considered to be linked, then this instantiation of the COMA framework can be considered as a type of Self Adaptation. The use of the intrinsic evolutionary processes to adapt step sizes governing the mutation of real-valued variables has long been used in Evolution Strategies [35], and Evolutionary Programming [11]. Similar approaches have been used to self-adapt mutation probabilities [2, 39] and recombination operators [34] in genetic algorithms as well as complex generating operators which combined both mutation and recombination [38]. This body of work contains many useful results concerning the conditions necessary for strategy adaptation, which could be used to guide implementations of COMA.

If the two populations are not linked, then COMA is a co-operative coevolutionary system, where the fitness of the members of the LSO 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 [31], and Potter and DeJong have also used co-operative co-evolution of partial solutions in situations where an obvious problem decomposition was available [33], both with good reported results. Bull [5] conducted a series of more general studies on co-operative co-evolution using Kauffmann's static NKC model [17]. In [7] 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 [6], with mixed results, although the NKC systems he investigated used fixed interaction patterns, whereas in the systems used here are more dynamic in nature.

There has also been a large body of research into competitive-coevolution, (an overview can be seen in [32]) whereby 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.

In both 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 LSO population can directly affect the genotypes within the solution population. A major source of debate and research within the community has focused around the perception that this phase of improvement by LS can be viewed as a kind of lifetime learning. This has lead naturally to speculation and research into whether the modified phenotype which is the outcome of the improvement process should be written back into the genotype (Lamarkian Learning) or not (Baldwinian Learning). Note that although the pseudo code of the local search, and the discussion above assumes Lamarkian learning, this is not a prerequisite of the framework. However, even if a Baldwinian approach was used, the principal difference between the COMA approach and the co-evolutionary systems above lies in the fact that there is a selection phase within the local search, that is to say that if all of the neighbours of a point defined by the LSO 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. [16] and the "Developmental Genetic Programming" of Keller and Banzhaf [18, 19]

Finally, and perhaps most importantly, it should be considered that if a rule has an improving effect on different parts of a solution chromosome over as number of generations, then the evolution of rules can be seen as learning generalisations about patterns within the problem representation, and hence the solution space. This point of view is akin to that of Learning Classifier Systems. For the case of unlinked fitness-based selection of LS operators, insight from this field can be used to guide the credit assignment process.

It is tempting to draw a further generalisation which would see the *conditions* as representing schema and the *actions* as representing higher fitness (and possibly higher order) alternatives, but this is a more dubious analogy as the conditions are allowed to match anywhere within the string, i.e. even a fully specified rule of length l matches $L - l$ schema within a string of length L.

4 Dill's HP model of Protein Structure Prediction

The problem of Protein Structure Prediction (PSP), i.e. the prediction of the "native" three-dimensional form of a protein from knowledge of the sequence of its constituent amino-acid residues is one of the foremost challenges facing computational biology. Current approaches to PSP can be divided into three classes; comparative modelling, fold recognition, and *ab initio* methods. The first two explicitly search the ever-growing databases of known structures for similar sequences (homologues) and sub-sequences. In contrast, the third approach represents the "last chance" scenario of trying to predict the tertiary structure by minimising a free energy model of the structure. Approaches that make use of existing knowledge currently represent the state of the art (and are likely to remain so), however ab initio approaches are important for two main reasons. The first of these relates to the situation where a sequence does not correspond to any known fold. The second, and more fundamental reason is that the development of true ab initio methods can give greater insight into the relationship between different fold families, and to the dynamical process of folding.

Current approaches to ab initio PSP can be divided according to two criteria, namely the nature of the choice of energy function, and the number of degrees of freedom in the conformation, as exemplified by the granularity (all atom models vs. virtual atom) and locational constraints (e.g. lattice based models vs. off-lattice models). Although most lattice based models are physically unrealistic, they have proved a useful tool for exploring issues within the field. Some of the more complex models, e.g. SICHO [21] have been shown to be capable of accurate predictions of the conformations of simple proteins, especially when used in conjunction with techniques for subsequent refinement to an all-atom model [10].

The HP model for PSP [8] provides an estimate of the free energy of a fold of a given instance, based on the summation of pair-wise 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 or hydrophilic, thus simplifying the energy calculations still further. Hydrophobic residues avoid interacting with the water molecules of the solvent, whereas hydrophilic (or polar) residues are able to form hydrogen bonds with the water molecules. Thus, polar residues are often found at the surface of the protein and hydrophobic residues are normally found buried in the inner part, or core, of the protein. The HP model captures this behaviour, despite its extreme simplicity. In the model, a sequence of l amino acid residues is represented by $s \in \{H,P\}^l$, where H represents a hydrophobic amino acid and P represents a hydrophilic one. The space of valid conformations is restricted to self-avoiding paths on a selected lattice, with each amino acid located on a vertex. The torsion angles of the peptide bonds between residues are thus restricted by a finite set determined by the shape of the lattice. The first amino acid of the sequence is located on a randomly selected vertex, and an orientation is assumed for it. From there, according to the orientation, the chain grows, placing every subsequent amino acid either ahead of the previous one, at 90 degrees to the left or at 90 degrees to the right (assuming a square lattice). 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. In some cases, to make feasible conformations more attractive, the infeasible folds suffer penalisation in the form of adding a substantial positive factor to their energy levels. In this way, the model reflects the tendency of hydrophobic amino acids to form a hydrophobic core. Despite the apparent simplicity of this model, the search for the global energy minimum in the space of possible conformations of a given sequence has been shown to be NP complete on various lattices [4].

Evolutionary algorithms (in particular Genetic Algorithms) have been applied, with some success, to the PSP using the HP and all-atom off-lattice models, by a number of authors since [41, 40]. In [23] the effect of different encoding schemes and constraint management techniques were examined, and a modified fitness function was developed which extends the basic HP model to permit the allocation of reward for non-adjacent pairs of Hydrophilic residues. More recent work has demonstrated the use of self-adaptation within a memetic algorithm to permit the selection from amongst a fixed set of predetermined local search strategies, using different move operators such as local "stretches", reflections etc [25, 30]. The work described here extends this by not relying on a fixed set of move operators encoding domain-specific knowledge, but rather evolving a set of move operators, thus *learning* that domain-specific knowledge.

5 Experimental Results

5.1 The Test Suite and Experimental set-up

In order to investigate the value of this approach, 20 instances and parameter settings from [24], were used, which use a two-dimensional triangular lattice. These instances are detailed in Table 1.

The representation used is a *relative* encoding. In this, where the alleles come from the set {*leftback, leftforward, front, rightforward, rightback*} and represent the direction of the next move on the lattice from the point of view of the head of the growing chain. This is an alternative to the *absolute* encoding used by Unger and Moult [41], where alleles specify directions to move relative to an external frame of reference. Results presented in [23] have suggested that this relative encoding is preferable, not least because the absence of a "back"

Table 1. HP instances used in these experiments

Id	Sequence	Length	Optimum
1	HHPHPHPHPHPH	12	11
2	HHPPHPHPHPHPHP	14	11
3	HHPPHPPHPHPHPH	14	11
4	HHPHPPHPPHPPHPPH	16	11
5	HHPPHPPHPHPHPPHP	16	11
6	HHPPHPPHPPHPPHPPH	17	11
7	HHPHPHPHPHPHPHPHH	17	17
8	HHPPHPPHPHPHPPHPHPHH	20	17
9	HHPHPHPHPHPPHPPHPPHH	20	17
10	HHPPHPPHPHPPHPHPPHPHH	21	17
11	HHPHPPHPPHPHPHPPHPPHH	21	17
12	HHPPHPHPHPPHPHPPHPPHH	21	17
13	HHPPHPPHPHPHPPHPPHPPHH	22	17
14	HHHPHPHPHPHPHPHPHPHPHHH	23	25
15	HHPPHPPHPPHPPHPPHPPHPPHH	24	17
16	HHHPHPHPPHPHPHPHPHPHPHHH	24	25
17	HHHPHPHPHPPHPHPHPHPHPHHH	24	25
18	HHHPPHPPHPPHPPHPHPPHPHPPHPPHHH	30	25
19	HHHPPHPPHPPHPHPPHPHPPHPPHPPHHH	30	25
20	HHHPPHPPHPPHPHPHPPHPPHPPHPPPPHPHPHHH	37	29

move means that all conformations that can be represented are one-step self-avoiding.

The generational genetic algorithm used (500+500) selection. One Point Crossover was applied with probability 0.8 and a Double Mutation was made with probability 0.3. Viewed from an external frame of reference the mutation operator has the effect of causing the mutation point to act as a pivot, about which one half of the structure is rotated through some multiple of $\pi/6$ (for a triangular lattice). 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).

For each combination of algorithm and instance, 25 runs were made, each run continued until the global optimum was reached, subject to a maximum of 1 million evaluations. 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 GA i.e. with no use of Local Search (GA).
- A simple MA using a bit-flipping neighbourhood, with one iteration of greedy ascent (SMA).

- Versions of COMA using a randomly created rule in each application, i.e. with the learning disabled. One iteration of steepest (SRand) or greedy (GRand) ascent local search was applied.
- Adaptive versions of COMA with the two pivot rules (SComa and GComa). In these the rule lengths are randomly initialised in the range [1,16]. During mutation, a value of $+/-1$ is randomly chosen and added with probability 0.0625.

These results are analysed according to three different performance criteria: firstly the Success Rate (the number of runs in which the global optimum was found), secondly in terms of efficiency, as measured by the average number of evaluations to solution (AES) in those successful runs, and thirdly in terms of the mean performance measured in terms of the best value found in the maximum time alloted, averaged over 25 runs.

5.2 Success Rate

Table 2 shows the Success Rate for each algorithm itemised by instance and in total. Using a non-parametric Friedman's test for k-related variables shows that the differences in success rate between algorithms is significant, and a series of paired t-tests confirms that the results for the SComa algorithm are significantly better than any of the others with over 95% confidence. This difference is particularly noticeable on the longer instances. Of the other results, the simple MA (SMA) performs well on the shorter instances, and the GComa and GRand results are surprisingly similar. This may well be due to the noise inherent in the greedy ascent mechanism making it hard for the credit assignment mechanism to function properly as was previously noted in [36]. Significantly, whatever the form of the local search phase, all but one of the Memetic Algorithms perform much better than the simple GA. The least reliable algorithm was SRand, and possible reasons for this will be discussed further in the following section.

5.3 Efficiency

Figure 1 shows the Average Evaluations to Solution (i.e., the globally optimal conformation) for the runs in which algorithms were successful. Immediately we can see that even when it is successful, the SRand algorithm is far slower than all of the other algorithms. Like the more successful GRand algorithm, it is using a randomly created rule to define the neighbourhood for each solution in each generation. However, unlike the GRand algorithm it is searching the whole of each neighbourhood, and the increase in the AES values suggests that the neighbourhoods are generally quite large. This suggests the frequent use of short, low rules of low specificity, i.e. with lots of #'s. It is possible that left to run for longer, the Success Rate of the SRand algorithm would have been improved.

Table 2. Number of runs (out of 25) in which the minimum energy conformation was identified

	algorithm					
instance	GComa	SComa	GRand	SRand	SMA	GA
1	13	25	16	16	25	13
2	14	25	15	7	23	13
3	15	24	10	11	22	7
4	19	25	17	2	24	13
5	13	25	13	7	22	9
6	10	24	11	0	20	9
7	9	24	5	1	14	3
8	7	25	6	0	11	2
9	4	22	5	0	4	2
10	4	21	4	0	10	2
11	5	21	7	0	7	2
12	7	22	7	0	12	4
13	6	21	3	0	7	2
14	0	7	0	0	0	0
15	0	9	1	0	0	2
16	1	7	0	0	1	0
17	0	8	0	0	0	0
18	0	1	0	0	0	0
19	0	1	0	0	0	0
Total	127	337	120	44	202	83

Of the others, the GA is always fastest, followed by the SMA. The rest of the picture is less clear, although the greedy versions are usually faster than their steepest ascent counterparts. A two way Analysis of Variance, with instance and algorithm as factors, shows that both are significant, and post-hoc analysis using the Least-Significant Difference test shows that the ordering GA < SMA < {GRand,GComa} < SComa < SRand is significant with 95% confidence. If we do not assume equal variance, Tamhane's T2 test shows that the GA is significantly faster, but under these more cautious assumptions the SMA is only significantly faster than GRand with 93% confidence and is not significantly faster than GComa. Similarly GRand and SComa are no longer significantly different in speed of finding solutions.

5.4 Mean Best Fitness

As was evidenced in Table 2 it is not hard to find solutions 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

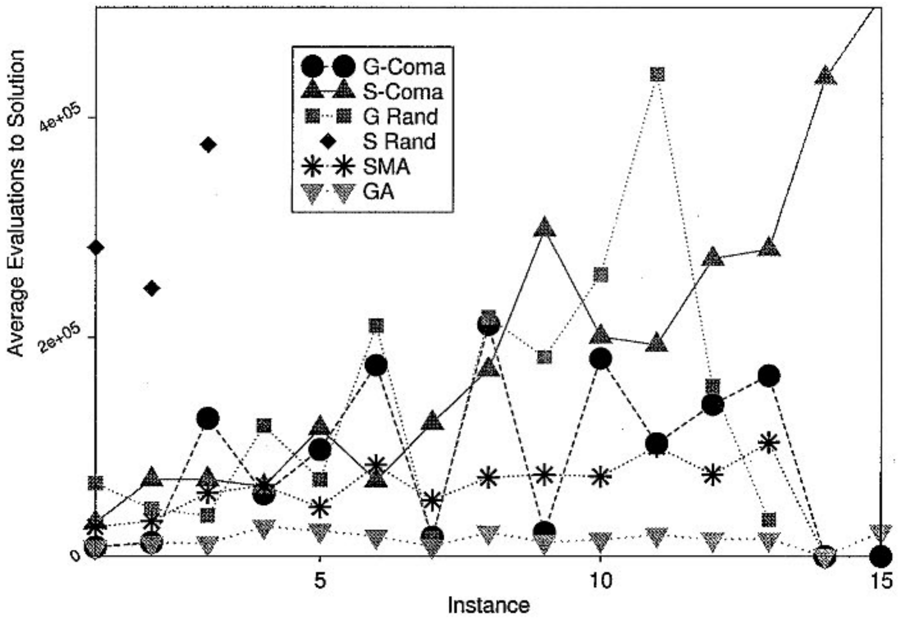

Fig. 1. Average Evaluations to Solution (when found) by algorithm.

the longer and harder instances 14-20 have been considered. Figure 2 shows these results graphically for each algorithm, sorted by instance. From these it is clear that the SComa reaches consistently higher values and with a smaller variance in performance than the others, and that the SRand algorithm is correspondingly worse.

In order to investigate the statistical significance of these results, a two-way ANOVA test was performed on the values for the best solution found in each run, with instance number and algorithm as the factors. This confirmed the significance of the algorithm in determining the performance, and so two sets of post-hoc tests were performed to analyse the differences between pairs of algorithms. These were Least-Significant Difference, and Tamhane's T2 test (the latter is more conservative as it does not make any assumptions about the samples having equal variances). The results of these tests are summarised in Table 3. An entry r or R indicates that the algorithm indicated by the row index was significantly better than the one indicated by the column index, with 95% confidence according to the LSD or T2 test respectively. Similarly an entry of c or C indicates that the column algorithm is better than the row algorithm with 95% confidence according to the LSD or T2 test respectively.

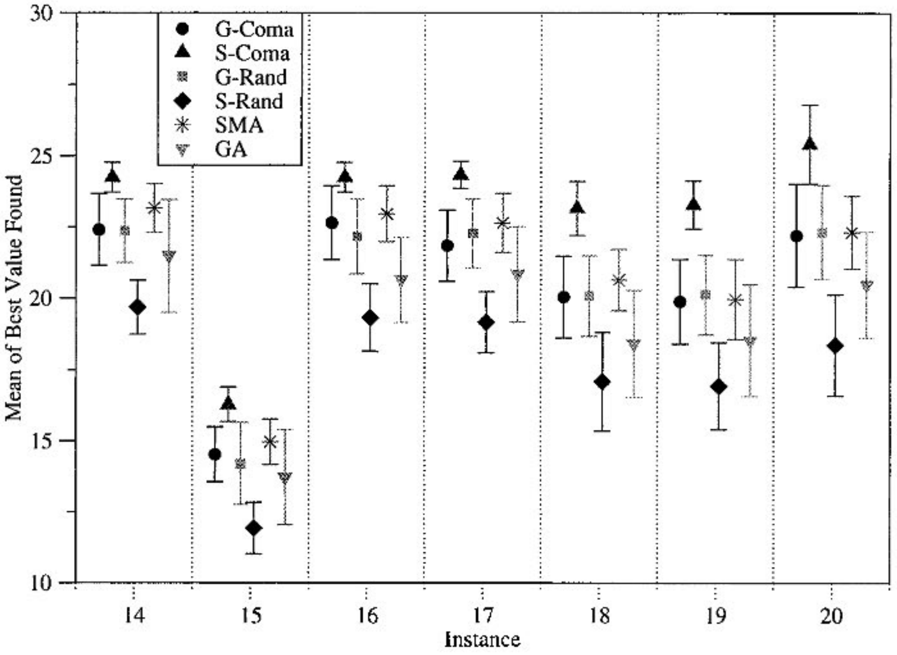

Fig. 2. Mean and std deviation of best values found for instances 14-20, analysed by algorithm

Table 3. Statistical significance of pairwise comparisons between algorithms on basis of best values found. – indicates no significant difference. r[c] denotes algorithm indicated by row[column] is better with 95% confidence. Lower triangle (lower case) is for LSD test, upper quarter (upper case) is for Tamhane's T2 test.

SComa	-	R	R	R	R	R
GComa	c	-	R	-	-	R
SRand	c	c	-	C	C	C
GRand	c	-	r	-	-	-
SMA	c	r	r	r	-	R
GA	c	c	r	c	c	-
Algorithm	SCOMA	GComa	SRand	GRand	SMA	GA

6 Restricting the Search to Feasible Solutions

In [9] results are reported from a detailed study of the fitness landscape of HP model proteins which suggests that the feasible regions of the search space are more highly connected than has previously been thought, and that correspondingly there may be performance advantages arising from a restriction of the search process to only considering feasible solutions.

In order to investigate this, the crossover and mutation operators were modified so that they only produced feasible offspring. This process is less lengthy than it would first appear since in practice infeasible offspring can almost always be quickly identified during the path growth process and the evaluation stopped. However no attempt was made to restrict the initial population to feasible solutions, as the infeasible ones are quickly weeded out by selection, and preliminary experimentation revealed that creating a feasible initial population by random generation of values takes an extremely long time.

The mutation operator still applied one double mutation - a random permutation of the loci was generated, and for each of these a random permutation of the possible changes was created. Offspring were produced and tested in this order until a feasible one was created. The crossover operator was modified similarly: if the offspring produced using a given crossover point was infeasible the operator next tested all of the different possible orientation of the two substrings by varying the allele value in the locus corresponding to that crossover point, before moving on to trying the next.

6.1 Success Rate

Table 4 shows the results from running the GA, SMA and SComa algorithms with the modified crossover and mutation operators, alongside those for the unmodified versions. As can be seen (and statistical testing confirms) there is far better reliability for the GA-F and SMA-F algorithms than their unrestricted counterparts. The results for the SComa are less clear - if anything the performance is better for short instances and worse for long ones, but the difference is not statistically significant.

6.2 Efficiency

Figure 3 shows the efficiency (AES) comparisons for the same set of algorithms, again restricted to successful runs. As when comparing Success Rates, there is little difference between the SComa and SComa-F algorithms, but under this metric the performance of the GA and GA-F algorithms are not significantly different, i.e., the GA is still very efficient on those runs when it does find the optimum, and with the restricted operators it does so far more often. In contrast to this, the SMA algorithm exhibits much greater AES values when restricted to feasible solutions, despite being more successful.

6.3 Mean Best Fitness

As evidenced in Table 4, restricting the search to feasible solutions makes it even easier to find solutions for the shorter instances. Therefore when comparing performance on the basis of the quality of the best solutions found,

Table 4. Effect on Success Rate of restricting search to feasible solutions. Results for GA, SMA and SComa algorithms are shown alongside those using modified crossover and mutation (indicated by –F)

instance	algorithm					
	GA	GA-F	SMA	SMA-F	SCOMA	SCOMA-F
1	13	23	25	25	25	25
2	13	20	23	25	25	25
3	7	17	22	25	24	25
4	13	20	24	25	25	25
5	9	18	22	24	25	25
6	9	18	20	24	24	25
7	3	9	14	23	24	24
8	2	9	11	24	25	25
9	2	8	4	21	22	23
10	2	8	10	22	21	23
11	2	5	7	24	21	21
12	4	12	12	23	22	23
13	2	4	7	24	21	23
14	0	0	0	5	7	9
15	2	1	0	8	9	6
16	0	1	1	2	7	5
17	0	0	0	4	8	0
18	0	0	0	0	1	0
19	0	0	0	0	1	0
total	83	173	202	328	337	332

i.e., mean best fitness (MBF), only results for the longer and harder instances 14-20 have been considered again. Figure 4 shows these results graphically for each algorithm, sorted by instance.

In order to investigate the statistical significance of these results, a two-way ANOVA test was performed on the values for the best solution found in each run, with instance number and algorithm as the factors. This confirmed the significance of the algorithm in determining the performance, and so two sets of post-hoc tests were performed to analyse the differences between pairs of algorithms. These were Least-Significant Difference, and Tamhane's T2 test (the latter is more conservative as it does not make any assumptions about the samples having equal variances). The results of these tests are summarised in Table 5. An entry r or R indicates that the algorithm indicated by the row index was significantly better than the one indicated by the column index, with 95% confidence according to the LSD or T2 test respectively. Similarly an entry of c or C indicates that the column algorithm is better than the row algorithm with 95% confidence according to the LSD or T2 test respectively.

In general it is plain that the rank order is GA < GA-F < SMA< SMA-F < SComa-F < SComa. These differences are generally statistically significant

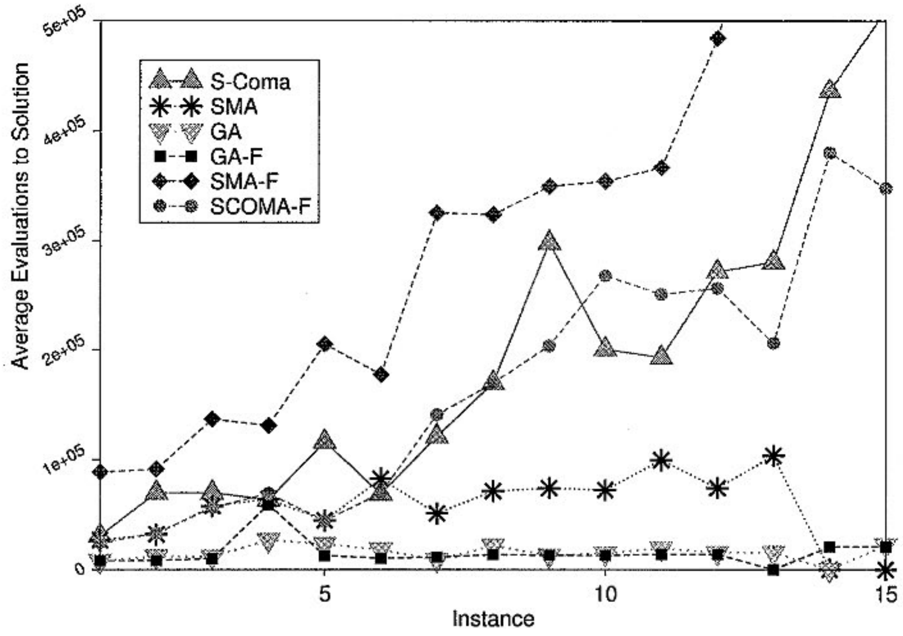

Fig. 3. Effect on efficiency of restricting search to feasible solutions. Plot shows Average Evaluations to Solution for successful runs of GA, SMA, SComa and their restricted counterparts (indicated by –F).

according to both tests, although it should be noted that this depends to some extent on the choice of instances considered. If we include all instances, then the general success on the shorter ones makes the differences less significant, whereas if we restrict ourselves to only considering a few harder instances, the significance increases.

Table 5. Statistical significance of pairwise comparisons between algorithms on basis of best values found. – indicates no significant difference. r[c] denotes algorithm indicated by row[column] is better with 95% confidence. Lower triangle (lower case) is for LSD test, upper quarter (upper case) is for Tamhane's T2 test.

GA	-	C	C	C	C	C
GA-F	r	-	-	C	C	C
SMA	r	-	-	C	C	C
SMA-F	r	r	r	-	C	-
SComa	r	r	r	r	-	-
SComa-F	r	r	r	-	-	-
Algorithm	GA	GA-F	SMA	SMA-F	SComa	SComa-F

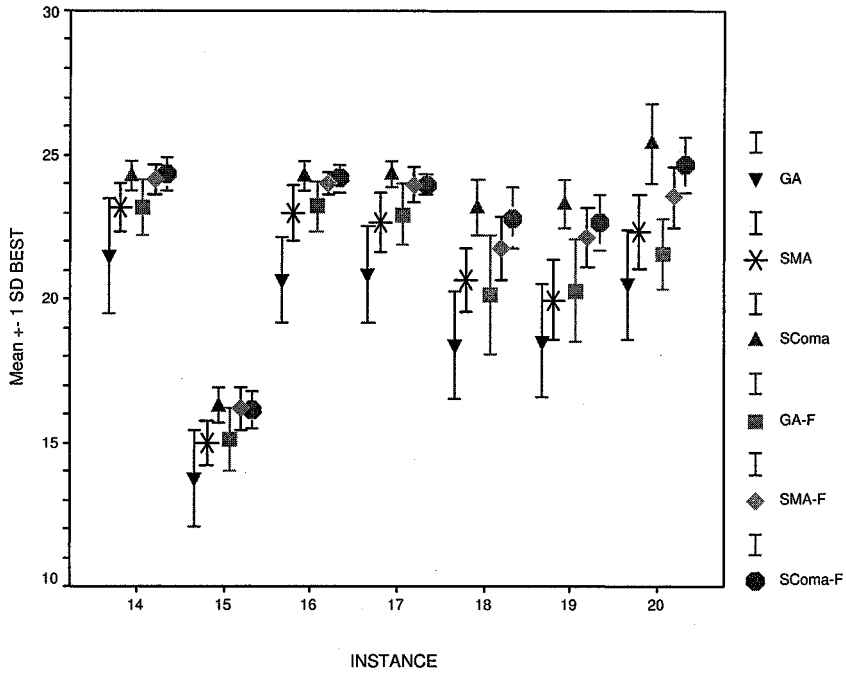

Fig. 4. Mean and std deviation of best values found for instances 14-20, analysed by algorithm

7 Analysis of LSO Evolution

In order to gain a greater understanding of the behaviour of the SComa algorithm, a number of test runs were made in which the contents of the LSO population were output to file at regular intervals.

Examination of the form of the evolving LSOs showed that there was a strong tendency towards short rules of the form $\#\# \rightarrow lr$ or $\#\# \rightarrow lL$. Here l = *leftback*, r = *rightback*, and L = *leftforward* relative to the previous direction of growth. Both of these rules act to bring residues i and $i + 2$ into contact, via causing a torsion angle of $\Pi/6$ at residue $i + 1$.

Given that the system is evolving conformations in a two-dimensional plane, these patterns these could possibly 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 that the rules which evolved on a number of instances tended to have length three and be of the form $\#\#\# \rightarrow lll$ or $\#\#\# \rightarrow rrr$ which is the shortest path that can be made bringing two residues into contact.

The use of the word "tended" 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 elsewhere [36] 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 GRand algorithm, in which the rules defining neighbourhoods are created at random, this "changing landscape" effect is noticeable in the superior success rates to the SMA. The fact that the SComa algorithm is the best performer according to both Success Rate and MBF metrics points to both modes of operation having a positive effect.

8 Discussion and Conclusions

As can be seen from the results section above, the S-Coma algorithm provides better performance according to Success Rate and Mean Best Fitness metrics than the GA, MA or a comparable system with the rule-learning turned off (SRand, GRand). These results are especially noticeable for the longer instances where the COMA system is able to learn and then exploit regularities within energetically favourable conformations, corresponding to secondary structural motifs. This happens at some expense of speed - the AES results show that the addition of any local search to a GA slows down the rate of discovery of globally optimal solutions, and that searching the whole neighbourhood (steepest ascent) rather than stopping once a better neighbour is found (greedy ascent) also imposes a cost. Nevertheless it must be emphasised that the results for the GA and the greedy algorithms come from many fewer successful runs. In other words, when the genetic search is able to find the optimum, it does so quickly, but it is prone to premature convergence.

Restricting the crossover and mutation operators to producing feasible solutions has mixed results. The Success Rate and Mean Best Fitness are much improved for the GA and SMA, and for the SComa on the shorter problems but if anything is slightly worse for SComa on the long instances. It was suggested in the previous section that the SComa had two modes of operation, re-use of secondary structural motifs, and continuously changing neighbourhoods. These results suggests that possibly the former mode is enhanced by the restriction to feasible solutions, but that the latter, which permits escape from local optima on the longer instances, is inhibited. Clearly this warrants further attention. Considering the efficiency with which solutions are found, this is not significantly changed for the GA or SComa, but is much worse for the SMA algorithm.

There is a clear place for the use of expert knowledge in the design of search algorithms, and its encapsulation in the form of carefully designed move operators. Nevertheless the approach outlined in this paper represents a highly promising prospect given its ability to discover and *explicitly represent* structural motifs. As an example, the reliability results reported above are better, especially for the longer instances, than those reported elsewhere using a self-adaptive multi-memetic algorithm, with the meme set especially designed after a comprehensive study of the literature and extensive experimentation [24]. This suggests that there is a clear role for adaptation of some kind within the specification of memes, rather than using a fixed set. The results presented here and elsewhere suggest that evolution may well be a suitable way of achieving that adaptation.

One obvious path for future work would be to examine the effects of seeding the rule population with expert-designed rules. Another, perhaps more pressing path is to examine the behaviour on more complex lattices and for different energy functions. As indicated above, these results are only the beginning of a process of investigation, clearly more analysis of the evolving rule-sets is needed, as well as a thorough investigation of the other algorithmic possibilities. It seems likely however that this represents a promising direction for the future development of scalable optimisation techniques which may yield new insights into the energy landscapes of the HP and other lattice models of proteins.

9 Acknowledgements

The author would like to thank Natalio Krasnogor for many fruitful discussions during the initial stages of this work, and for introducing him to the Protein Structure Prediction problem.

References

1. ., editor. *2003 Congress on Evolutionary Computation (CEC'2003)*. IEEE Press, Piscataway, NJ, 2003.
2. Thomas Bäck. Self adaptation in genetic algorithms. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the 1st European Conference on Artificial Life*, pages 263–271. The MIT Press, Cambridge, MA, 1992.
3. W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*. Morgan Kaufmann, 1999.
4. B. Berger and T. Leight. Protein folding in the hydrophobic-hydrophilic (hp) model is NP-complete. In *Proc. 2nd Annual Intnl. Conf. Computational Molecular Biology RECOMB98*, 1998.

5. Larry Bull. *Artificial Symbiology.* PhD thesis, University of the West of England, 1995.
6. Larry Bull. Evolutionary computing in multi agent environments: Partners. In Th. Bäck, editor, *Proceedings of the 7th International Conference on Genetic Algorithms*, pages 370–377. Morgan Kaufmann, San Francisco, 1997.
7. Lawrence Bull and Terence C. Fogarty. Horizontal gene transfer in endosymbiosis. In Christopher G. Langton and Katsunori Shimohara, editors, *Proceedings of the 5th International Workshop on Artificial Life : Synthesis and Simulation of Living Systems (ALIFE-96)*, pages 77–84, Cambridge, May 16–18 1997. MIT Press.
8. K. Dill. *Biochemistry*, 24:1501, 1985.
9. S. Duarte-Flores and J.E. Smith. Study of fitness landscapes for the HP model of Protein Structure Prediction. In . [1], page to appear.
10. M. Feig, P. Rotkiewicz, A. Kolinski, J. Skolnick, and C. Brooks. Accurate reconstruction of all-atom protein representations from side-chain-based low-resolution models. *Proteins:Structure Fucntion and Genetics*, 41:86–97, 2000.
11. David B. Fogel. *Evolving Artificial Intelligence.* PhD thesis, University if California, 1992.
12. J.J. Merelo Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacanas, and H.-P. Schwefel, editors. *Proceedings of the 7th Conference on Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science. Springer, Berlin, 2002.
13. P. Hansen and N. Mladenovič. An introduction to variable neighborhood search. In S. Voß, S. Martello, I. H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and trends in local search paradigms for optimization. Proceedings of MIC 97 Conference.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
14. W. E. Hart. *Adaptive Global Optimization with Local Search.* PhD thesis, University of California, San Diego, 1994.
15. Terry Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, The University of New Mexico, Albuquerque, NM, 1995.
16. Hillol Kargupta and Samiran Ghosh. Towards machine learning through genetic code-like transformations. Technical Report TR-CS-01-10, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, 2001.
17. S.A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, New York, NY, 1993.
18. Robert E. Keller and Wolfgang Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo, editors, *Proceedings of the 1st Annual Conference on Genetic Programming*, pages 116–122. MIT Press, 1996.
19. Robert E. Keller and Wolfgang Banzhaf. The evolution of genetic code in genetic programming. In Banzhaf et al. [3], pages 1077–1082.
20. Joshua Knowles and David Corne. A comparative assessment of memetic, evolutionary and constructive algorithms for the multi-objective D-MSAT problem. In *Gecco-2001 Workshop Program*, pages 162–167, 2001.
21. A. Kolinski and J. Skolnick. Assembly of protein structure from sparse experimental data: An efficient Monte-Carlo method. *Proteins: Structure Function and Genetics*, 32:475–494, 1998.

22. N. Krasnogor and S. Gustafson. Toward truly "memetic" memetic algorithms: discussion and proofs of concept. In David Corne, Gary Fogel, William Hart, Joshua Knowles, Natalio Krasnogor, Rajkumar Roy, Jim Smith, and Ashutosh Tiwari, editors, *Advances in Nature-Inspired Computation: The PPSN VII Workshops*, pages 9–10, Reading, UK, 2002. PEDAL (Parallel, Emergent & Distributed Architectures Lab), University of Reading.
23. N. Krasnogor, W. Hart, J.E. Smith, and D. Pelta. Protein structure prediction with evolutionary algorithms. In Banzhaf et al. [3], pages 1596–1601.
24. N. Krasnogor and J. Smith. A memetic algorithm with self-adaptive local search: TSP as a case study. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 987–994. Morgan Kaufmann, 2000.
25. Natalio Krasnogor. *Studies in the Theory and Design Space of Memetic Algorithms.* PhD thesis, University of the West of England, 2002.
26. Natalio Krasnogor and Jim Smith. Emergence of profitiable search strategies based on a simple inheritance mechanism. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 432–439. Morgan Kaufmann, 2001.
27. Peter Merz. *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Efective Search Strategies.* PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.
28. Peter Merz and Bernd Freisleben. Fitness landscapes and memetic algorithm design. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 245–260. McGraw Hill, 1999.
29. Pablo Moscato. Memetic algorithms' home page. Technical report, http://www.densis.fee.unicamp.br/~moscato/memetic_home.html, 2002.
30. N. Krasnogor, B.P. Blackburne, E.K. Burke and J. D. Hirst. Multimeme algorithms for protein structure prediction. In Guervos et al. [12], pages 769 –778.
31. Jan Paredis. The symbiotic evolution of solutions and their representations. In L.J. Eshelman, editor, *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 359–365. Morgan Kaufmann, San Francisco, 1995.
32. Jan Paredis. Coevolutionary algorithms. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation.* Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1998.
33. M. A. Potter and K.A. DeJong. A cooperative coevolutionary approach to function optimisation. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature*, number 866 in Lecture Notes in Computer Science, pages 248–257. Springer, Berlin, 1994.
34. J.David Schaffer and Amy Morishima. An adaptive crossover distribution mechanism for genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*, pages 36–40. Lawrence Erlbaum Associates, 1987.
35. H.-P. Schwefel. *Numerical Optimisation of Computer Models.* John Wiley and Sons, New York, 1981.
36. J.E. Smith. Co-evolving memetic algorithms: A learning approach to robust scalable optimisation. In . [1], page to appear.

37. Jim Smith. Co-evolution of memetic algorithms : Initial investigations. In Guervos et al. [12], pages 537–548.
38. Jim Smith and T.C. Fogarty. Adaptively parameterised evolutionary systems: Self adaptive recombination and mutation in a genetic algorithm. In Voigt et al. [43], pages 441–450.
39. Jim Smith and T.C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation*, pages 318–323. IEEE Press, Piscataway, NJ, 1996.
40. R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Theoretical Biology*, 231(1):75–81, 1993.
41. Ron Unger and John Moult. A genetic algorithm for 3D Protein Folding Simulations. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 581–588. Morgan Kaufmann, San Francisco, 1993.
42. F. Vavak, T.C Fogarty, and K. Jukes. A genetic algorithm with variable range of local search for tracking changing environments. In Voigt et al. [43], pages 376–385.
43. H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors. *Proceedings of the 4th Conference on Parallel Problem Solving from Nature*, number 1141 in Lecture Notes in Computer Science. Springer, Berlin, 1996.