Protein Structure Prediction with Co-evolving Memetic Algorithms

J. E. Smith
Faculty of Computing, Engineering and Mathematical Sciences
University of the West of England
Bristol, U.K.
james.smith@uwe.ac.uk

Abstract- This paper presents 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 then 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 scaleable algorithms.

1 Introduction

This paper presents a co-evolutionary learning-optimisation approach to Protein Structure Prediction which uses a Memetic Algorithm as its underlying search method. Within this Coevolutionary Memetic Algorithm (COMA) system, instance-specific knowledge can be learned, stored and applied 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. The rest of this paper proceeds as follows:

- In Section 2 describes the particular application, namely Protein Structure Prediction using Dill's HP model [1].
- Section 3 provides a brief introduction to Memetic Algorithms to set a context for the algorithmic developments in this paper.
- Section 4 describes the proposed approach to adaptive Memetic Algorithms, and the simplified model implemented here. It also summarise the results of initial investigations published elsewhere.
- In Section 5 are presented 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.
- In Section 6 the implications of these results are discussed, before conclusions are drawn and future work suggested.

2 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 [2] 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 [3].

The HP model for PSP [1] 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 [5]. In [6] 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 [7, 8]. The work described here extends this by not relying on a fixed set of move operators encoding domain-specific knoweldge, but rather evolving a set of move operators, thus *learning* that domain-specific knowledge.

3 Memetic Algorithms

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 (see [9] 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 not be used, or the order in which the operators are applied may vary.

There are three principal components which affect the workings of a Local Search. The first is the choice of pivot rule, which can be *Steepest Ascent* or *Greedy Ascent*. In the former the entire neighbourhood $n(i)$ of the current solution i is searched and the best neighbour used, whereas the latter stops exploring the neighbourhood as soon as an improvement is found. The second component is the depth of the local search. This can vary from a minimum of only one improving step being applied, through a fixed number of iterations, to the extremum where the local search is iteratively applied until a local optimum is reached. Considerable attention has been paid to studying the effect of changing this parameter within MAs e.g. [10]. Equally the choice of pivot rules 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 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. 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" [11]. A number of statistical measures can used to characterise fitness landscapes, and have been proposed as potential measures of problem difficulty by various authors. Merz and Freisleben [12] discuss a number of these in the particular context of MAs, and 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 [12, pp254–258], but also dependent on the state of the evolutionary search [13]. This result is not surprising when it is considered 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.

Krasnogor and Smith described a "MultiMemetic Algorithm" , in which a gene was added to the end of each chromosome indicating which of a fixed set of static LS opera-

tors ("memes") should be applied to the individual solution [13]. Variation was provided during the mutation process, by randomly resetting this value with a low probability. The results demonstrated that this system was 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 [14]. 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 [15], 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.

4 COMA: A Rule-Based Model for the Adaptation of Move Operators

4.1 The Model

The aim of this work is to provide a means whereby the definition of the local search 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. The approach taken is to use a coevolutionary system with two populations, one of candidate solutions, and one of Local Search Operators (LSOs) which act on the members of the first population.

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

The element *Pairing* efectively 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*$\}$.

For the *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 populations 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. An examination of the issues surrounding fitness-based pairing can be seen in an accompanying paper in these proceedings [16].

The first two elements in the tuple have been described above, and like the pairing can be easily mapped onto an integer or cardinal representation as desired, and manipulated by standard genetic operators. 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 components listed above provide basis for a major body of research, which is well beyond the scope of this paper. For these initial investigations the LSOs were restricted to one of greedy or steepest ascent, a single improvement step, and *linked* pairing. These choices are coded into the LSO chromosomes at initialisation, and variation operators are not used on them. The 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* (but not the *action*) part of the rule. 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, given a solution represented by the binary string 1100111000 and a rule 1#0:111, then this matches the first, second, sixth and seventh positions, and the neighbourhood is the set $\{$1110111000, 11111111000, 1100111100,1100111110$\}$. Note that in this work the string is not considered to be toroidal (although this will be considered in later work), and that 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.

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 representation for the rules means that "standard" genetic operators (uniform/1 point crossover, point mutation) can be used to vary this part of the LS chromosome.

The inclusion of an adaptive *rule_length* permits the ability to adapt via the action of mutation operators on this value. In [17] the effects of different fixed rule sizes were examined, and a version which had the ability to adapt via the action of mutation operators on this value. The results obtained demonstrated that the version with adaptive rule lenths was able to quickly adapt to using the appropriate optimal size rule for the different problems.

4.2 Initial Results

The results of initial investigations using this system were reported in [17]. The test suite was problems made out of a number of subfunctions either interleaved or concatenated. Two different classes of subfunction 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, and GRand algorithms described below. It was shown that a version of the system with adaptive rule lengths was able to perform better than these three. It also performed comparably with variants of GComa that had the optimal fixed rule-lengths for the different problems. Analysis showed that these GCOMA algorithms discovered and used problem specific information (such as optimal patterns for different sub-problems), and subsequent work has shown them to be highly scalable with respect to problem length. Further results and analysis can be seen elsewhere in these proceedings [16].

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 [18], were used, which use a two-dimensional triangular lattice. The generational genetic algorithm used (500+500) selection and a *relative* encoding. In this encoding 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 [5], where alleles specify directions to move relative to an external frame of reference. Results presented in [6] have suggested that this relative encoding is preferable, not least because the absence of a "back" move means that all conformations that can be represented are one-step self-avoiding.

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 can be analysed in number of ways, of which we will focus on three: firstly in terms of reliability (i.e. of the number of runs in which the global optimum was found), secondly in terms of the average number of evaluations taken to find the optimum in those successful runs, and thirdly in terms of the mean performance (i.e. the best value found in the maximum time alloted, averaged over 25 runs). Figure 1 summarises the results of these experiments in terms of the first two criteria – reliability and speed. The vertical bars show the mean time to the optimum, with error bars at +/- one standard deviation from the top of the bar. Where there is an annotation above the bar, this represents the number of runs in which the global optimum was found (out of 25). No annotation indicates that all 25 runs found the optimum, and a missing column indicates that no runs found the global optimum on that instance.

As can be seen, the SComa algorithm is notably more successful than the other algorithms, and the successful runs use comparable numbers of evaluations as those of the faster of the other algorithms. For the instances 17-19, only the SComa algorithm was able to find the global optimum, (doing so 8,1 and 1 times respectively). No algorithm solved instance 20 in the time allowed. The success rate is quite high for the shorter instances, and this is reflected in similar performance in terms of the metric of best value found. The difference in mean best value becomes more marked for the longer instances as is shown in Figure 2, where the superior performance of the S-Coma algorithm is clear.

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 Tamhanes 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 1. 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. SComa is omitted as it is significantly better than all others according to both tests.

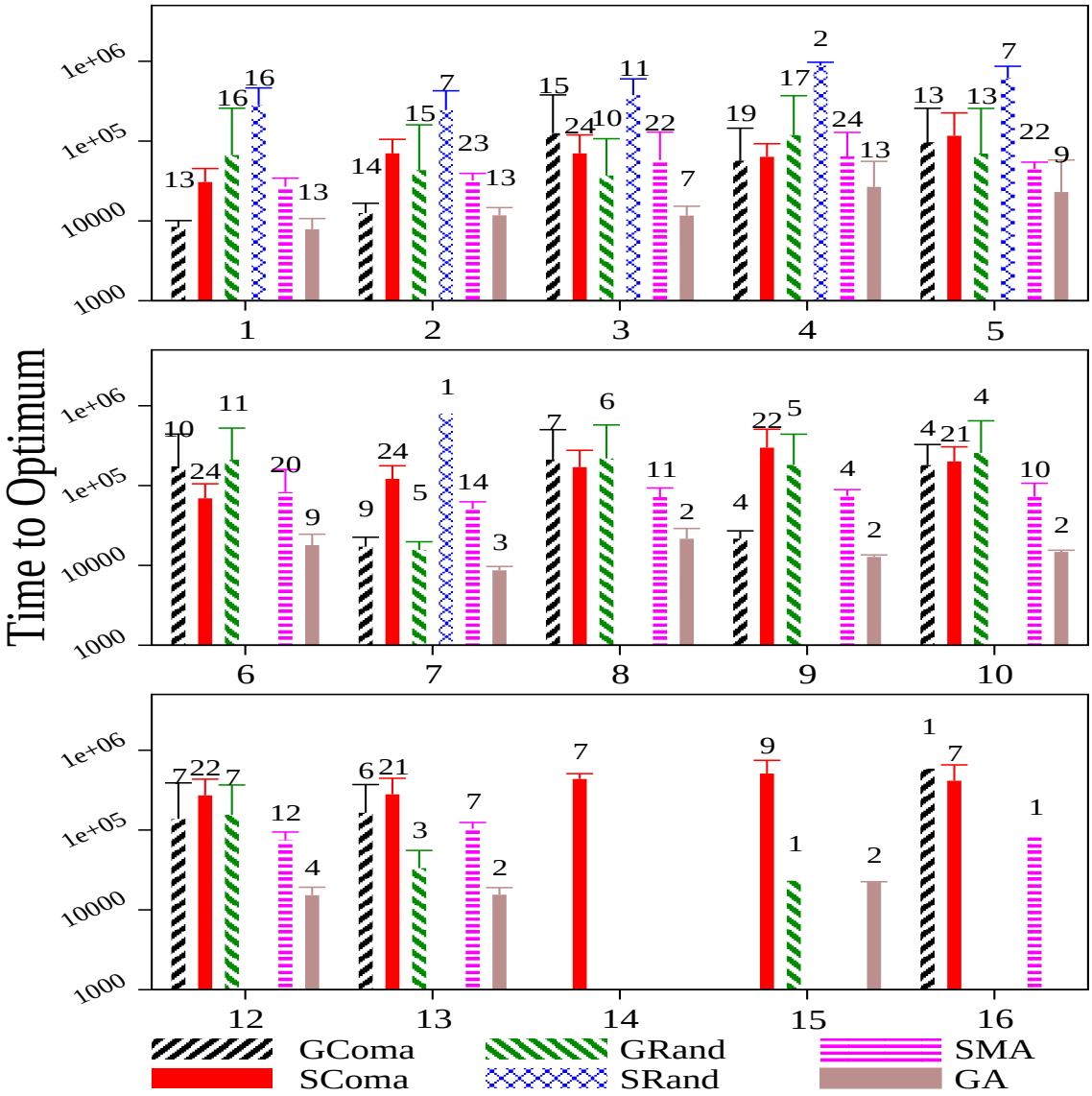

Figure 1: Times to reach optimum by algorithm, instances 1-5 (top), 6-10(middle), and 11-16 (bottom). Error bars represent +/- 1 std. dev. from top of bar. Annotations represent number of runs where optimum found, if less than 25.

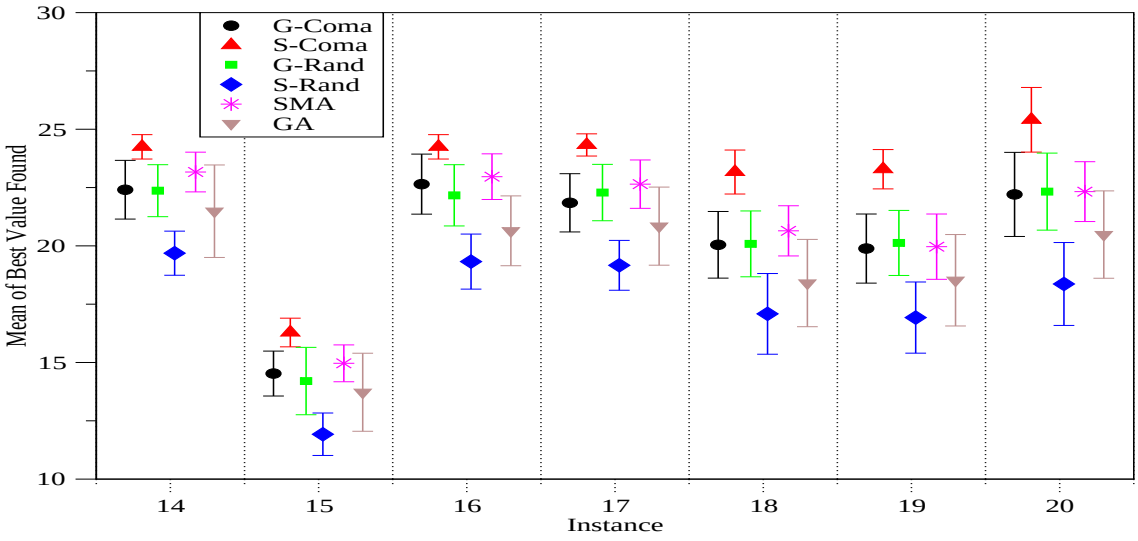

Figure 2: Mean and std deviation of best values found for instances 14-20, analysed by algorithm

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

Table 1: Statistical significance of pairwise comparisons between algorithms on basis of best values found. SComa omitted as it is significantly better than all others. – 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.

5.2 Restricting the Search to Feasible Solutions

In [19] we have reported results 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.

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.

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.

These results are shown in Figure 3 for the GA, SMA and SComa. Tests were also made with variants of the memetic algorithms in which the local search phase was continued until a local optimum rather than just a single step (SMA* and SComa*).

As can be seen from a comparison of Figures 1 and 3 these results demonstrate that improved reliability arisies from restricting the variation operators to using feasible solutions. These differences were found to be statistically significant.

The effects of changing the depth of local search were less conclusive. Considering only the SComa and SComa* algorithms: a statistical analysis of the best individual found in each run with the instance and depth (a binary choice between one step/to optimum) as the variables did not show a significant difference, although the interaction term was significant. From Figure 3 it can be seen that the number of evaluations taken to find a solution is comparable for the two approaches on each instance. Given that one might expect an overhead in terms of the number of neighbours tested in an iterative search, this is perhaps surprising.

Analysis of the form of the evolving rules showed that there was a strong tendency towards 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 we are working in a two-dimensional plane 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.

We would like to highlight the use of the word "tended" here: in most cases the rule-set continued to contain a number of different rules of varying lengths. We have argued elsewhere [16] that in addition to the extra scalability attained by identifying and re-applying regular structural motifs, the presence of a diverse, evolving rule-set gives increased possibilities of escaping from local optima to COMA, by continually testing new search landscapes.

6 Discussion and Conclusions

As can be seen from the results section above, the S-Coma algorithm provides better performance according to both metrics (reliability and mean best fitness) than the GA, MA or a comparable system with the rule-learning turned off (SRand, GRand). These results are especially noticable 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.

These results are improved still further by the restricting the variation operators to producing feasible solutions. Changing the depth of the local search appeared to make things slightly worse for the standard MA, and did not make any statistically significant difference for COMA, suggesting that the extra computational effort is not worthwhile in this case.

The reliability results are also 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.

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 we feel that the approach outlined in this paper represents

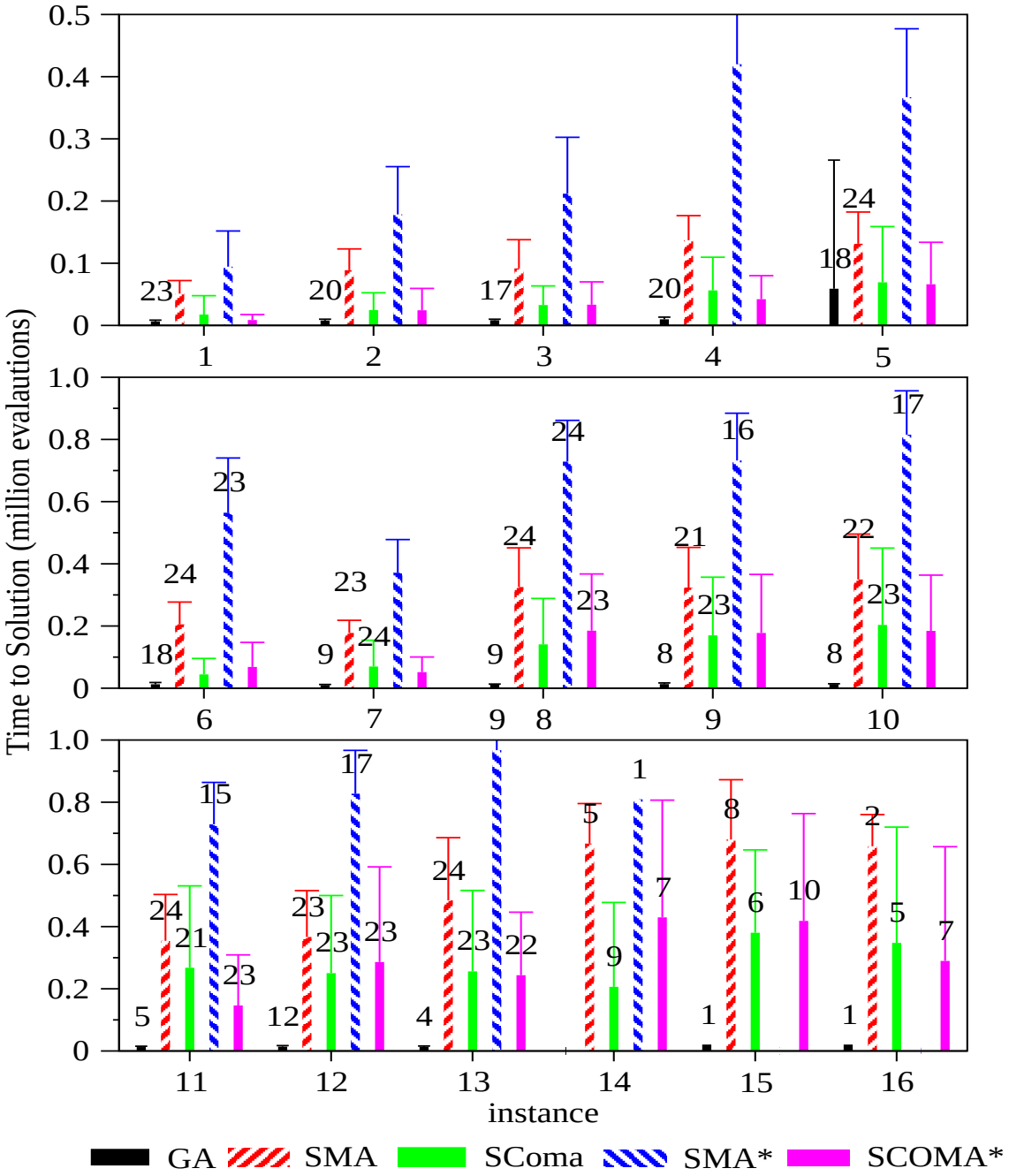

Figure 3: Times to reach optimum by algorithm, instances 1-5 (top), 6-10(middle), and 11-16 (bottom). Error bars represent +/- 1 std. dev. from top of bar. Annotations represent number of runs where optimum found, if less than 25.

a highly promising prospect given its ability to discover and *explicitly represent* structural motifs. 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 processs 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.

7 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.

Bibliography

[1] K. Dill, *Biochemistry*, vol. 24, p. 1501, 1985.

[2] A. Kolinski and J. Skolnick, "Assembly of protein structure from sparse experimental data: An efficient monte-carlo method," *Proteins: Structure Function and Genetics*, vol. 32, pp. 475–494, 1998.

[3] 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*, vol. 41, pp. 86–97, 2000.

[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] R. Unger and J. Moult, "A genetic algorithm for 3D protein folding simulations," in *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest, Ed. Morgan Kaufmann, San Francisco, 1993, pp. 581–588.

[6] 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, 1999, pp. 1596–1601.

[7] N. Krasnogor, "Studies in the theory and design space of memetic algorithms," Ph.D. dissertation, University of the West of England, 2002.

[8] N. Krasnogor, B.P. Blackburne, E.K. Burke and J. D. Hirst, "Multimeme algorithms for protein structure prediction," 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, 2002, pp. 769 –778.

[9] P. Moscato, "Memetic algorithms' home page," http://www.densis.fee.unicamp.br/˜moscato/memetic_home.html, 2002.

[10] W. E. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, University of California, San Diego, 1994.

[11] T. Jones, "Evolutionary algorithms, fitness landscapes and search," Ph.D. dissertation, The University of New Mexico, Albuquerque, NM, 1995.

[12] 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, 1999, pp. 245–260.

[13] N. Krasnogor and J. Smith, "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, 2001, pp. 432–439.

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

[15] P. Hansen and N. Mladenovič, "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. H. Osman, and C. Roucairol, Eds. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998. [Online]. Available: http://www.crt.umontreal.ca/ pierreh/pub-en.html or http://www.wkap.nl/book.htm/0-7923-8369-9

[16] J. Smith, "Co-evolving memetic algorithms: A learning approach to robust scalable optimisation," University of the West of England, Tech. Rep., 2003, to appear in proceedings of IEEE CEC03.

[17] ——, "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, 2002, pp. 537–548.

[18] 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, 2000, pp. 987–994.

[19] S. Duarte-Flores and J. Smith, "Study of fitness landscapes for the HP model of protein structure prediction," University of the West of England, Tech. Rep., 2003, to appear in CEC03 Special Session on Bioinformatiocs.