Genetic Programming and Evolvable Machines, 5, 181–201, 2004

Self Generating Metaheuristics in Bioinformatics: The Proteins Structure Comparison Case

N. KRASNOGOR Natalio.Krasnogor@nottingham.ac.uk
Automated Scheduling, Optimisation and Planning Group, School of Computer Science & IT, University of Nottingham, NG81BB, Nottingham, UK

Submitted February 13, 2003; Revised November 7, 2003

Abstract. In this paper we describe the application of a so called "Self-Generating" Memetic Algorithm to the Maximum Contact Map Overlap problem (MAX-CMO). The maximum overlap of contact maps is emerging as a leading modeling technique to obtain structural alignment among pairs of protein structures. Identifying structural alignments (and hence similarity among proteins) is essential to the correct assessment of the relation between proteins structure and function. A robust methodology for structural comparison could have impact on the process of rational drug design.

The Self-Generating Memetic Algorithm we present in this work evolves concurrently both the solutions (i.e. proteins alignments) and the local search move operators that it needs to solve the problem instance at hand. The concurrent generation of local search strategies and solutions allows the Memetic Algorithm to produce better results than those given by a Genetic Algorithm and a Memetic Algorithm with human-designed local searchers. The approach has been tried in four different data sets (1 data set composed of randomly generated proteins and the other 3 data sets with real world proteins) with encouraging results.

Keywords: memetic algorithms, bioinformatics, MAX-CMO, self-generation, co-evolving memetic algorithms

1. The comparison of protein structures

The comparison of proteins structures is at the core of today's biomedical research. Although a variety of structure comparison methods have been proposed and used in classification servers, such as SCOP [38], DALI [19], LGA [52, 53], structure comparison is still considered an open problem. Identifying structural similarities is essential to the correct assessment of the relation between structure and function in proteins. Without the ability to perform reliable and efficient structural matchings rational drug design becomes a more difficult task.

When sequence similarities cannot be used to infer relationships among proteins, distant evolutionary membership to protein families can still be assessed with structural measures. Researchers interested in comparing and evaluating function, structures and sequences have at their fingertips more than 30 genomes fully sequenced and available through the Internet.[1] To understand a protein's biological function, biologists usually resort to the comparison of a target protein with known ones. That comparison, if the sequences are more than 25% identical, can be made at the sequence level. Below that 25% similarity threshold, relationships between proteins are made based on structure. This approach takes the name of structural genomics. There is yet another important role for structural matching algorithms: the evaluation of *ab-initio*, *threading or homology modeling* structure predictions. In the words of Zemla et al. [53]:

> "With the significant expansion of activity in the structure prediction field, processing and subsequent analysis of predictions has become increasingly complex... One of the lessons learned from CASP is that analyzing the effectiveness of prediction methods is not a trivial matter... To evaluate predictions, first we need an analytical approach to identify what in a prediction worked and what failed. Second we need a comparative approach, using both general and specialized techniques, to identify which methods work best, and which address a specific aspect of prediction most successfully."

To assess the quality of a prediction, a target structure must be compared with the predicted structure (the model) in order to determine which regions of the later closely resemble the former. Hence, any advancement in structural matching will also have an impact on structure prediction and its evaluation methodologies.

Several researchers proposed different methodologies to assess structure similarities. Those methodologies range from dynamic programming [49], comparisons of distance matrices [19], maximal common sub-graph detection [1] and geometrical matching [51], to name a few. There are, however, several problems associated with the most common approaches. These problems are often related to the fact that methods implicitly accept that a suitable scoring function can be defined for which optimum values correspond to the best possible structural match between two structures. Also, approaches that employ *root-mean-square-distances*, e.g., [7, 34], and *differences of distance matrices*, e.g., [20], present numerical instabilities problems; other algorithms cannot produce a proper ranking due to an ambiguous definition of structural similarity or the fact that they neglect alternative (different) solutions. More recent approaches that partially address these problems can be found in [21, 31, 53].

In [17] Goldman et al. present the following list of desirable properties for a structural similarity metric:

- it should not penalize too heavily insertions and deletions,
- it should be reasonably robust, in that small perturbations of the definition should not make too much difference in the measure,
- it should be easy to compute (or at least rigorously approximated),
- it should be able to discover both local and global alignments,
- it should be able to discover hydrophilic-hydrophobic alignments,
- it should take into account the self-avoiding nature of a protein,
- it should be subject to empirical studies on Protein Data Base (PDB) data to validate its success in capturing structural similarity, and
- even if one comes up, from a theoretical standpoint, with a "perfect" measure, it will be difficult to displace entrenched measures, used for years by protein scientists. Acceptance in the field is thus a further desideratum.

The authors go on to argue that contact map overlap is the only measure that comes close to realizing the mentioned list of desirable properties. Thus, one of the emerging techniques for solving this problem is the *Alignment of Contact Maps*.

1.1. The maximum contact map overlap problem

In its simplest form, a contact map is a matrix of all pairwise distances within a protein[6, 33, 36]. It is a minimalist representation of a protein's native three dimensional structure.

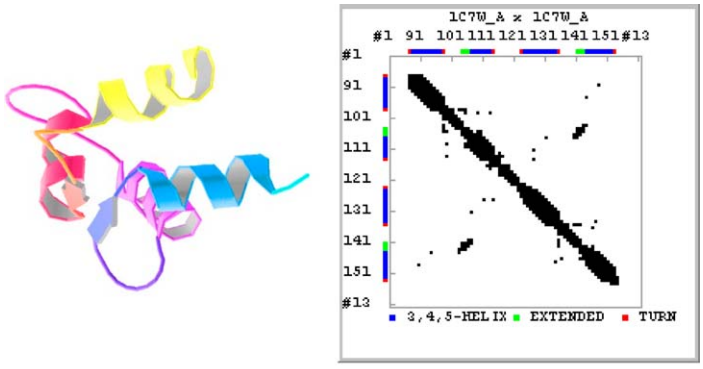
In its most abstract formulation, it takes the form of:

$$S_{i,j} = \begin{cases} 1 & \text{if residue } i \text{ and } j \text{ are in contact} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Residues i and j are said to be in contact if they are closer than a threshold of, for instance, R Angstroms. In this version of the contact map, distances are not explicitly represented (usual values for R are between 2 and 9 Angstroms), rather a Boolean value is assigned to a matrix cell specifying whether two residues are considered neighbors or not.

Contact maps are used in two main flavors;[2] either the distance considered to specify R is that of the C_α atoms of the residues involved, or the minimum distance between *any* two atoms belonging to those residues. In more specific formulations, the entries in $S_{i,j}$ will be positive real numbers specifying the interresidue distances. Several combinatorial problems associated with the alignment of distance matrices were reported in the literature. See [19, 22, 39] and references therein for a thorough discussion.

As mentioned before, the contact map captures the three dimensional structure of proteins; for example, in matrix $S_{i,j}$, α-helices are represented by wide bands on its main diagonal, while *β-sheets* manifest themselves as bands parallel or perpendicular to it. Figure 1(a) shows the three dimensional structure of protein 1C7W while Figure 1(b) shows the matrix representation of $S_{i,j}$ for this protein. Several software packages, some of them in the public domain, permit the user to compute and display contact maps[3] (see [43, 48] and references therein). A contact map can also be represented as an undirected graph. In this graph, each residue is a node and there exists an edge between two nodes if they are neighbors in the sense described before. The graph representation, rather than the matrix one, is exploited in the Maximum Contact Map Overlap Problem (MAX-CMO). An *alignment* between two contact maps is an assignment of residues in the first contact map to residues on the second contact map. Residues that are thus aligned are considered equivalent. The value of an alignment between two contact maps is the number of contacts in the first map whose end-points are aligned with residues in the second map that, in turn, are in contact (i.e. the



(a) (b)

Figure 1. Protein structure and its contact map (binary) graphical representation for protein 1C7W which mainly contains alpha-helix features.

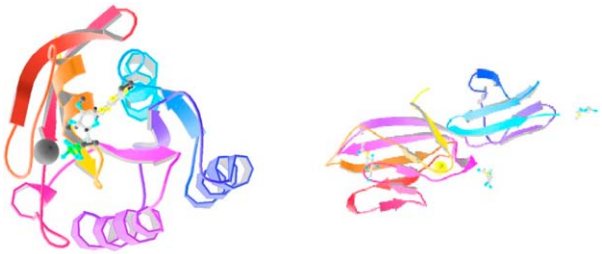


Figure 2. Three dimensional native structures for proteins 1AA9 (a) and 1HNF (b).

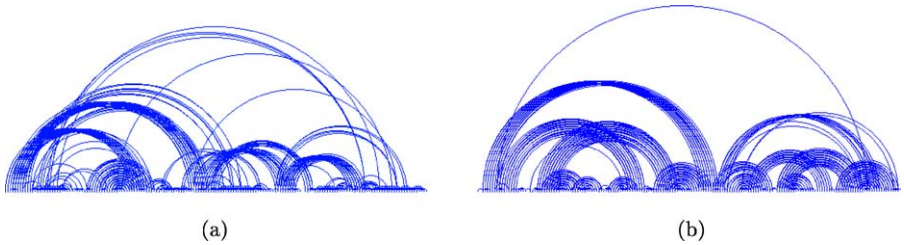


Figure 3. Two snapshots of our code to plot the contact map of 3d structures. Proteins 1AA9 (a) and 1HNF (b) from the PDB are shown.

number of size 4 undirected cycles that are made between the two contact maps and the alignment edges). This number is called the *overlap* of the contact maps and the goal is to maximize this value. The Max CMO problem was introduced in [15] and proved NP-hard in [17] and later in [25].

Figure 2 presents the native structures for two proteins and in Figure 3 we can see their respective contact maps. Note the long range interactions of residues induced by the beta-sheet features. These residues are far away in the sequence but close in the three dimensional structure. A candidate alignment for the two contact maps is shown in Figure 4 (equivalent residues are identified by (slanted) vertical alignments). Note that although some residues are considered to be equivalent by this alignment, they do not necessarily present identical local contact maps. An alignment cannot always satisfy all the constraints that the two contact maps induce, hence a solution is a compromise isomorphism between sub-graphs of the global contact maps.

Once the contact maps of two proteins are computed the task that remains, automatically comparing them,[4] is the most difficult. Eidhammer in [12] said:

> "In structure comparison, we do not even have an algorithm that guarantees an optimal answer for pairs of structures. . . "

However, after recent work in [5, 32], the optimal comparison of structures is possible by means of the overlap of contact maps. The first rigorous approach to contact maps overlap was introduced by Lancia et al. [32]. This approach is based on an effective integer programming (IP) formulation of protein structures contact map overlaps and the development of a branch and cut strategy that uses lower bounding heuristics at the branch

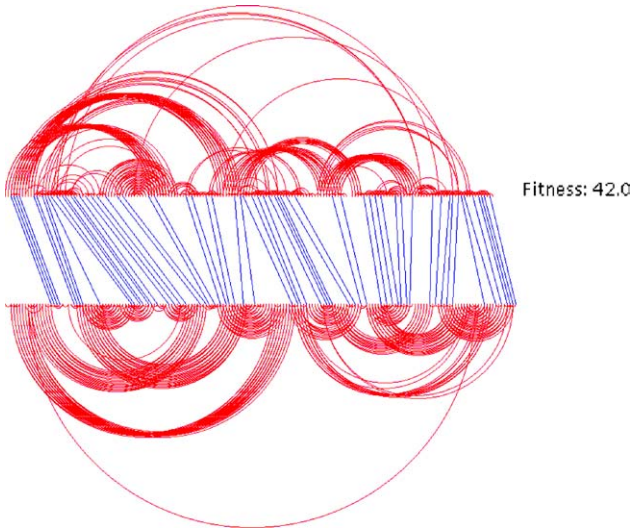


Figure 4. A candidate alignment of value 42 for the contact maps of proteins 1A99 and 1HNF.

nodes. Also, a linear programming (LP) formulation exists that provides upper bounds on the value of the optimal alignments. These upper bounds allow us to compare the results produced by our algorithms (lower bounds) with those of the LP formulation. Having the upper and the lower bounds for the value of the resulting structural overlap of two proteins is a strong certificate of quality for the alignment, and an indication of how similar two protein structures really are.

2. Self-generating memetic algorithms

Several of the most successful metaheuristics for hard combinatorial problems are rooted in the appropriate coordination of low level heuristics (e.g. local search operators, constructive methods, genetic operators, etc) to produce one or more solutions to a specific problem instance. Among the most popular examples of metaheuristics for hard combinatorial problems are Variable Neighborhood Search (VNS) [18], Memetic Algorithms (MAs) [35], Tabu Search [14], GRASP [42], Ant Colony Optimization (ACO) [10], Simulated Annealing [23] and various forms of Evolutionary Algorithms [2]. One of the simplest of these is VNS where at any one time during the optimization process only *one* solution to the problem instance is produced. The low level heuristics (LLHs), also called move operators, are applied in a well defined *fixed* schedule one after the other to improve the current solution. On the other hand, MAs,[5] employ a *population* of solutions and a variety of low level heuristics (also called in this context genetic and local search operators). The idea behind using a population of solutions, instead of just one (as used in VNS, GRASP or Tabu Search), is to produce a more accurate sampling of the solution space, rendering the learning and optimization process more robust.

As the No Free Lunch theorem shows [50], no one metaheuristic is best across all instances and all problems. Moreover, a sort of *conservation of competence* applies, in the sense that

the better one algorithm is solving a specific instance (class) the worst it is solving a different instance (class) [44]. Hence, it is simply incorrect to a priori assume that the most complex (or for that purpose the simplest) metaheuristic is best suited for all possible situations.

A vast number of very successful applications of Memetic algorithms (MAs) have been reported in the literature in the last years for a wide range of problem domains. See Moscato's web-based bibliography on Memetic Algorithms for an up-to-date overview of the field [37]. The majority of the papers dealing with MAs are the result of the combination of highly specialized **pre-existing** local searchers and usually purpose-specific genetic operators.

In order to have greater applicability, robustness and re-use, variants of current metaheuristics are needed that do not require a priori detailed knowledge of the instances to be solved or a subtle understanding of the interplay between the different low level operators. What it is needed are metaheuristics that can **self-generate** [25] the specific kind of local search, constructive method, genetic operator, etc., which are best suited to the classes of instances the metaheuristic is being presented with.

In [25, 27, 46, 47] it was proposed and demonstrated that the concept of Self-Generating Memetic algorithms (called Co-evolutionary Memetic Algorithms in [46])can be implemented by following more closely the dual inheritance dynamics of a coupled gene-meme evolutionary system [11, pp. 186] where two-replicators (i.e. genes and memes) co-exist. Gabora [13] mentions three phenomena that are unique to cultural (i.e. memetic) evolution, namely, *Knowledge-based operators*, *imitation* and *mental simulation*. It is these three phenomena that our Self-Generating Memetic Algorithm implements, and by virtue of which, can produce its own local searchers.

3. Self-generating memetic algorithms for MAX-CMO

In [32] Lancia et al. used a standard GA with specially tailored genetic operators to provide solutions and lower bounds for the pairs of proteins that were bench-marked. Later a Multimeme algorithm was proposed that, while using the same representation and genetic operators as the GA [32], also included a set of local searchers that were shown to improve the quality of the alignments (i.e. better overlap values) [5]. In this paper we will use the same representation and genetic operators as were used in the mentioned papers and we will compare our results with those in [5].

In an evolutionary algorithm for *Max CMO* a chromosome is represented by a vector $c \in [0, \ldots, m]^n$, where m is the size of the longer protein and n the size of the shorter. A position j in c, $c[j]$, specifies that the j^{th} residue in the longer protein is aligned to the $c[j]^{th}$ residue in the shorter. A value of -1 in that position will signify that residue j is not aligned to any of the residues in the other protein (i.e., a structural alignment gap). Unfeasible configurations are not allowed. That is, if $i < j$ and $c[i] > c[j]$ or $i > j$ and $c[i] < c[j]$ (e.g., a crossing alignment) then the chromosome is discarded. It is simple to define genetic operators that preserve feasibility based on this representation. Two-point crossover with boundary checks was used to mate individuals and create one offspring. Although both parents are valid alignments, the newly created offspring can result in invalid (crossed) alignments. After constructing the offspring, feasibility is restored by deleting any alignment that crosses other alignments. The mutation move employed in the experiments is called a sliding mutation. It selects a consecutive region of the chromosome vector and adds (slides right) or subtracts (slides left) a small number. The phenotypic effect produced

is the tilting of the alignments. Due to space limitations we invite the reader to consult [5] and [32] for details on these genetic operators. In our previous work [5] we employed a multimeme algorithm that also had a set of 6 human-designed local search operators. Four of the local searchers implemented were parameterized variations of the sliding operator. The direction of movement, left or right sliding, and the tilting factor, i.e., the number added or subtracted, were chosen at random in each local search stage. The size of the window was taken from the set {2, 4, 8, 16}. Two new operators were defined: a "wiper" move and a "split" move. At every iteration of a wiper move operator, two alignments are chosen and re-aligned onto various different residues, keeping two (of the four) anchoring points fixed. The best alignment was chosen as the next solution. The split operator was used to redistribute consecutive alignments by introducing gaps in the (partial) isomorphism encoded by a solution. Details of these human-designed local searchers can be found in [5].

3.1. Memes description

As mentioned in previous sections, we seek to produce a metaheuristic that creates from scratch the appropriate local searcher to use under different circumstances. The embodiment of a local searcher is performed by *memeplexes* [4]. A memeplex is a co-adapted complex of memes. A meme represents one particular way of doing local search. Memes can adapt through changes in their parameter set or through changes in the actions they perform. The local search involved can be very complex and composed of several phases and processes. In the most general case we want to be able to explore the space of all possible memes. One can achieve this by using a formal grammar that describes memeplexes and by letting a Genetic Programming [24] based system evolve sentences in the language induced by that grammar [25]. Sentences represent syntactically valid complex local searchers that are the instructions used to implement specific search behaviors and strategies. The grammar below characterizes the space of potential memeplexes that the Self-Generating Memetic Algorithm in this paper explores:

- Memeplex = $< Meme > \mid < Meme >:< Memeplex >$
- Meme = $(< Where >, < When >, < How >, < Frequency >)$
- Where = $< Location > Crossover \mid < Location > Mutation \mid < Location >$ *Update Function*
- Location = *After* | *Before*
- When = $< BooleanCondition > \mid < Probability >$
- How = *GeneralHillClimber'* $< Move > \rightarrow < Move >$ ' $< Strategy >< Iterations > \mid$ *BoltzmannAdaptiveHillClimber* $< Move > < Strategy > < Iterations >$
- Frequency = *Always*|*For* $< Iterations >$ |*Never*|...
- Move = $< Num > \mid < Num > - < Move >$
- Strategy = *FirstAscent* $< Size >$ |*SteepestAscent* $< Size >$ |*MiniMax* $< Size >$ |*MaxiMin* $< Size >$ |*XAware*|*MAware*
- Size = $< Num >$
- Iterations = $< Num >$

In this grammar for memeplexes, $< Num >$ and $> Probability >$ are assumed to be integer and real numbers respectively (with the later within the [0, 1] interval). An example of a sentence in the language induced by the previous grammar, i.e. a particular local search strategy, is the following:

memeplex' = (*Before Crossover*, 1.0, *BoltzmannAdaptiveHillClimber* '$2 \rightarrow 2$' *FirstAscent* 50 3, *Always*): (*After Mutation*, *If_Not_Feasible*, *GenerallHillClimber* '$1-2-3 \rightarrow 2-4-6$' *SteepestAscent* 100 1, *Always*)

This example specifies a memeplex for local search with the following characteristics: Two local search processes (memes) are scheduled to occur, one of them just before crossover takes place and the other immediately after mutation takes place.[6] These two local searchers are of a different nature. The local searcher associated with crossover is executed with a probability of 1.0 (i.e. every individual goes through a process of local search improvement) by using a Boltzmann adaptive hill-climber [29]. The hill-climber's move operator is defined by '$2 \rightarrow 2$'. This representation is explained in detail later on. The best of the sampled points are accepted following a first ascent strategy (i.e. the first solution that improves the current one is accepted). The acceptance strategy can sample, using move operator '$2 \rightarrow 2$', at most 50 points and the process is repeated 3 times. The local search represented by this meme is always applied, meaning in this context, in every generation of the memetic algorithm. On the other hand, the meme for the local search associated with mutation specifies that it should only be applied if the current solution is unfeasible. In that case, it will use a general hill climber with move operator '$1-2-3 \rightarrow 2-4-6$', using a steepest ascent strategy that samples at most 100 candidate points. This process is repeated only once for each candidate structure. The meme is activated always, that is, in each generation.

To understand the representation used to evolve the move operator itself we resort to a few examples. In Figure 5 we can see two contact map alignments produced by our algorithm. All the contact maps represented in the figure have a very specific pattern of contacts among their residues, in the present example, with a certain probability a residue is connected to

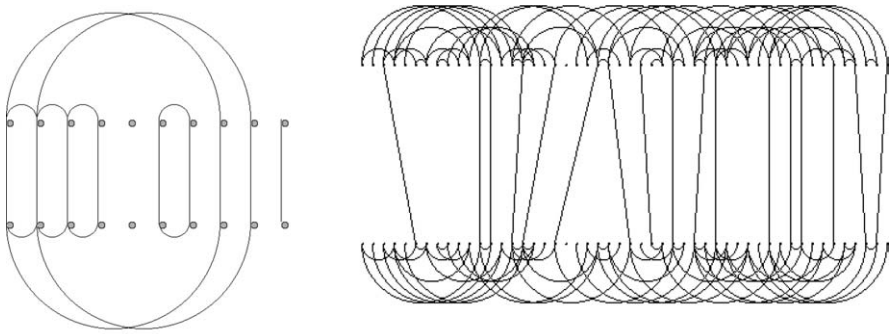


Figure 5. Two contact map alignment snapshots. In (a) the two randomly generated proteins have 10 residues, while in (b) the patterns of contacts are maintained but the protein is 50 residues long.

its neighbor, to a residue that is 7 residues away in the protein sequence or to both. In Figure 5(a) a protein 10 residues long is aligned with itself, while in 5(b) this is done for a 50 residues long protein.

This contact pattern can be represented by the string 1–7, meaning that the residue which occupies the ith position in the protein sequence is in contact in the native state with residues $(i + 1)$th and $(i + 7)$th. An appropriate move operator for a local searcher acting in any of the contact maps on Figures 5(a) and (b) would be one that iterates through every residue in one of the contact maps, checking which residues on the lower contact map fulfills the pattern of connectivity, and makes a list of them. The same procedure would be applied to the top contact map producing a second list of residues. The local searcher would then pair residues of one list with residues of the second list, and thus producing a new and correct alignment. The number of residues that verifies the pattern in each list puts an upper bound on how expensive the local search move operator can be. If the size of the first list is L_1, and that of the second L_2, and without loss of generality we assume that $L_1 \leq L_2$ then there are at most $\sum_{i=1}^{i=L_1} (\frac{L_1!}{(L_1-i)!*i!} \frac{L_2!}{(L_2-i)!*i!})$ pairings. That is, each L_1 taken by i residues in the shortest list can be paired with L_2 taken by i residues of the second (and longest) list.[7] Clearly this number is too big to be searched exhaustively, this is why the previous grammar allows for the adaptation of the sample size. Moreover, although it is well known that real proteins present these contact patterns [8, 41], it is impossible to know a priori *which* of these patterns will provide the best fitness improvement for a particular pair of protein structures. Hence, the Self-Generating MA needs to discover this by itself. As a further example consider the contact maps in Figure 6.

In this figure the contact maps to be aligned have the same number of residues as before but the contact patterns are different. In this case, each residue could be in contact only with one other residue 5 positions away. The representation of the move operator for our first example is '$1 - 7 \rightarrow 1 - 7$', while for the second example it is '$5 \rightarrow 5$'. If the contact maps of the two proteins to be aligned were different (in the previous cases a protein was aligned with itself for the sake of clarity), then a move operator able to account for that variation in patterns must be evolved. In the examples that appear in Figure 7 one possible move operator would be '$1 - 3 \rightarrow 1 - 4$'.

The move operator induces a neighborhood for every feasible alignment. If an alignment s is represented as above and L_1, L_2 are the list of vertices that matches the move operator,

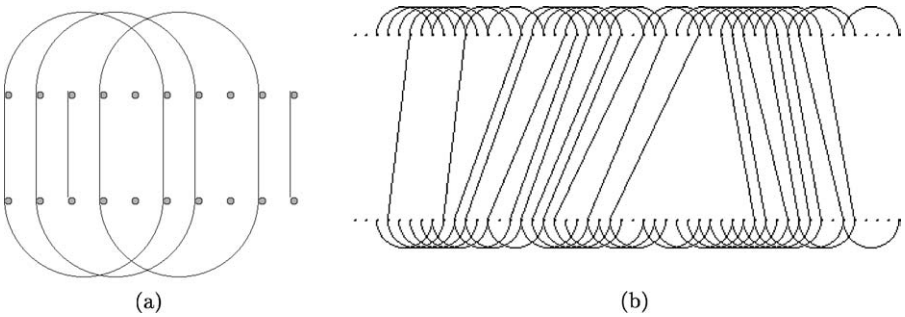


Figure 6. Two contact map alignment snapshots. In (a) the two randomly generated proteins have 10 residues, while in (b) the patterns of contacts are maintained but the protein is 50 residues long.

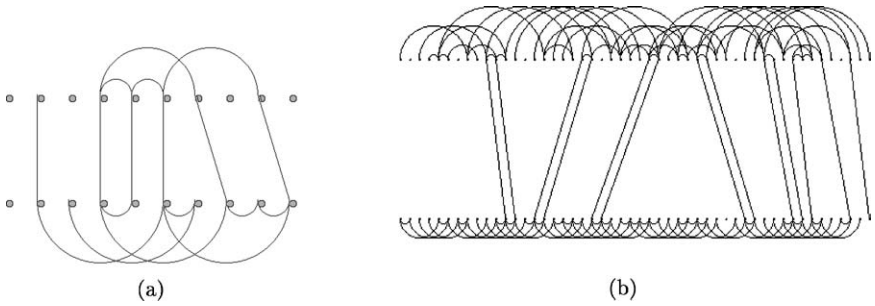


Figure 7. Two contact map alignment snapshots. In (a) the two randomly generated proteins have 10 residues, while in (b) the patterns of contacts are maintained but the protein is 50 residues long.

then every *feasible* solution that can be obtained by adding to s one or more alignments of vertices in L_1 with vertices on L_2 is a neighbor of s. The other components of a meme will then decide how to sample this neighborhood and which solutions to accept as the next one. As this paper is an account of the initial investigations we performed on the use of SGMA, we fixed several aspects of the memes that could otherwise be evolved. That is, in this paper all memes employ first improvement ascent strategy and they are applied after crossover. In addition, the evolved local searcher used a sample size of either 50 or 500 and it was iterated 2 times.

As we mentioned previously, there were three memetic processes: imitation, innovation and mental simulation. Upon reproduction, a newly created offsprings inherited the meme of one of its parents accordingly to the simple inheritance mechanism described in [30]. In addition to this mechanism, and with a certain probability (called "imitation probability"), an individual could choose to override its parental meme by copying the meme of some successful individual in the population to which it was not (necessarily) genetically related. In order to select from which individual to imitate a search behavior, a tournament selection of size 4 was used among individuals in the population and the winner of the tournament was used as role model and its meme copied. Innovation was a random process of mutating a meme's specification by either extending, modifying or shortening the pattern in a meme (either before or after the $\rightarrow$). If during 10 consecutive generations no improvement was produced by either the local search or the evolutionary algorithm a stage of mental simulation was started. During mental simulation, each individual (with certain probability) will intensively mutate its current meme, try it in the solution it currently holds, and if the mutant meme produces an improvement, both the newly created solution and the meme will be accepted as the next state for that individual. That is, mental simulation can be considered as a guided hill-climbing on the memetic space. If ten mental simulation cycles finished without improvements, then metal simulation was terminated and the standard memetic cycle resumed.

4. Experimental settings

The following sections describe several experiments to investigate whether Self-Generating Memetic Algorithms are at least as competent as human-designed ones. That is, *is the*

co-evolution of local searchers within MAs at least as productive in terms of quality of solutions as hand-made local searchers?. With this goal in mind we conducted two family of experiments, one based on randomly generated contact maps and the other with contact maps obtained from real proteins.

4.1. Experiments with random contact maps

We designed a random instance generator with the purpose of parameterizing the complexity of the contact map overlap problems to be solved. The input to the random instance generator is a list of the form:

$$r\ d\ n\ p_1\ pr_1\ p_2\ pr_2 \ldots p_n\ pr_n,$$

where r is the number of residues in the randomly generated contact map, d is the density of random edges (i.e. noise) and n is the number of patterns in the contact map. For each of the n patterns two numbers are available, p_i and pr_i. P_i specifies that a residue j is connected to residue $j + p_i$ with probability pr_i for all $i \in [1, n]$. Every pattern occurs with certain probability in each residue, thus an upper bound on the expected number of contacts is given by $r * d + r * \sum_{i=1}^{i=n} pr_i \leq r * (n + d)$. In our experiments $r \in \{10, 50, 100, 150, 200, 250\}$, $d = 0.01$ and $n \in \{1, 2, 3, 4\}$. Contact maps as short as 10 residues and as long as 250 residues were considered. For each contact map length, every possible number of patterns was used, which gives rise to 24 pairs of (r, n) values. For each pair, 5 random instances were generated spanning from low density contact maps to high density contact maps.[8] A total of 120 instances were generated. From all the possible parings of contact maps, we randomly choose a total of 96 pairs to be aligned by means of 10 runs each.

We present next comparisons of the performance of a Genetic Algorithm versus that of the SGMA. We were able to reproduce the results of [5, 32] and consider our implementations as equivalent to the earlier ones.

In Figures 8(a) and (b) and 9(a) and (b) we compare the overlap values[9] against the *first hitting times*. First hitting time (FHT) is the time (in number of fitness evaluations) at which the best value of a run was encountered. Each graphs presents the results for 1, 2, 3 and 4 patterns respectively and for a range of contact maps sizes. The particular parameters used in the GA and the SGMA for these experiments are shown in Table 1.

The Figures in 8 and 9 are representative of the results obtained with these two types of algorithms. Under a variety of changes to the values in Table 1 the results remain equivalent to those shown here.

From the figures we can see that the Self-Generating Memetic Algorithm produces a much better amortized overlap value than the simple GA. If enough time is given to the SGMA, it will sooner or later discover an appropriate local searcher move that will supply new building blocks. In turn, this will deliver an order of magnitude better overlaps than the Genetic Algorithm. Also, it seems that the GA is oblivious to the size (i.e. residues number) of the contact maps as it seems to produce mediocre local optima solutions even when given the maximum CPU time allocation (in these experiments 2×10^5 fitness evaluations) for the whole range of 10 to 250 residues. The GA converges very soon into local optima, this

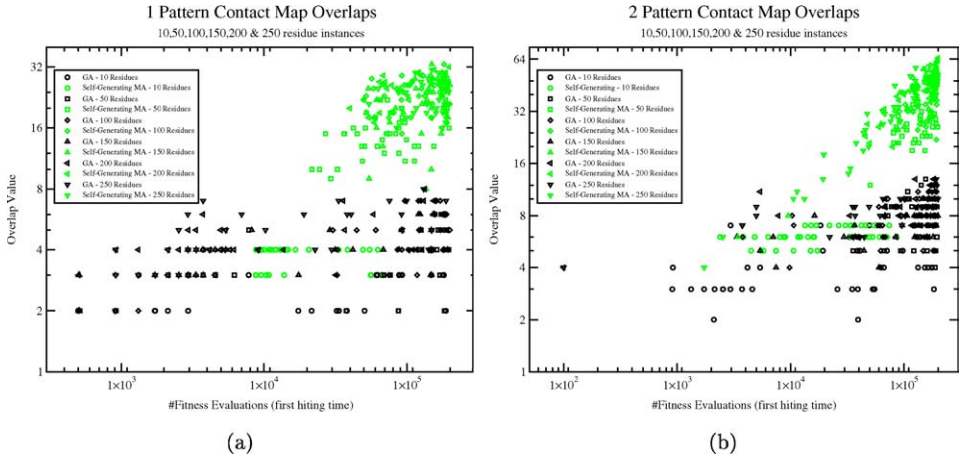


Figure 8. Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number. Contact maps present one pattern (a) and two patterns (b).

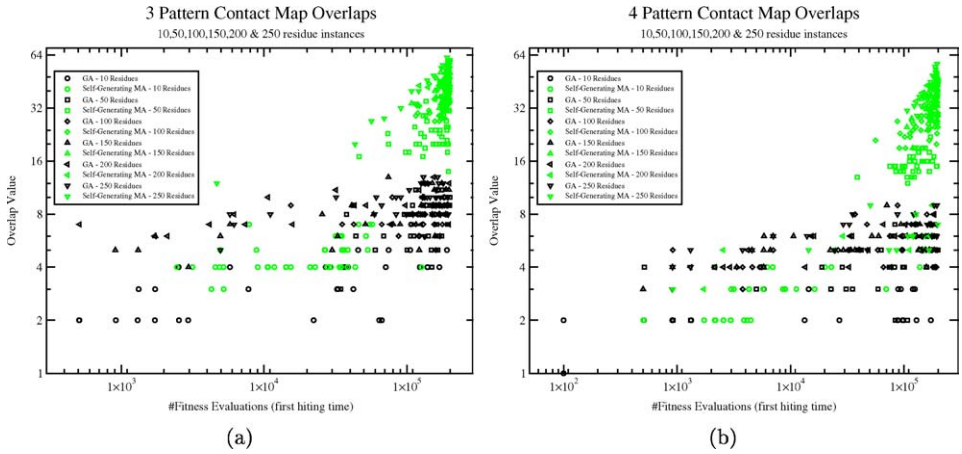


Figure 9. Comparison of the first hitting times and the quality of overlaps obtained for GA and SGMA on increasingly difficult randomly generated instances. Complexity increases as a function of residues number. Contact maps present three patterns (a) and four patterns (b).

is seen in the graphs by bands parallel to the *x-axis* over the range of energy evaluations for low overlap values. On the contrary, as the SGMA continuously improves its solutions, it is not until very late in the execution (i.e. to the right of the *x-axis*) that the best solutions are found. In contrast to the GA, the SGMA (as expected) is sensitive to the number of residues in the contact maps involved. Longer contact maps require larger CPU time to come up with the best value of the run (which is seen in the graph in the clustering patterns for the different residues number). Another important aspect to note is that both the *x-axis* and the *y-axis* are represented in logarithmic scales. Taking this into consideration it is evident that the quality of the overlaps produced by the SGMA are notoriously better than those produce by the

Table 1. Value of parameters for the GA and the SGMA

Parameter	GA	Self-generating MA
Mutation prob.	0.15	0.15
Crossover prob.	0.75	0.75
μ	50	50
λ	75	75
Local search prob.	NA	1.0
Imitation prob.	NA	1.0
Imitation neigh. size	NA	4
Mental sim. prob.	NA	1.0
Innovation prob.	NA	1.0
Iteration number	NA	2
LS CPU budget	NA	50
Mental effort	NA	50

GA. As it is evident from the graphs, for sufficiently small instances (e.g. all the 10 residues long and some of the 50 residues long) it is not worth using the SGMA as it requires more CPU effort to produce the same quality of overlaps as the GA. On the other hand, as the number of residues increases beyond 50, then the instances become sufficiently complex to allow for the emergence of suitable local searchers in time to overtake and improve on the GA results. Also, as the number of patterns that are present in the instances increases both algorithms, as expected, require larger amounts of CPU to come up with the best solution of a run. However, the GA is still insensitive to the number of residues, while the SGMA is clustered in the upper right corner (of Figure 9(b)). This indicates that during the SGMA execution the algorithm is making progress toward better and better solutions, the best of which is to be found near the end of the run. Moreover, this behavior indicates that the SGMA is not prematurely trapped in poor local optima as the GA. The ability of the SGMA to overcome local optima comes from the fact that the evolved local searchers will introduce good building-blocks that match the particular instance. This supply of building-blocks is essential for a synergistic operation of both the local searcher and the genetic operators.

4.2. Real proteins

In this section we compare the performance of the Self-Generating Memetic Algorithm with the results reported in [5] (which in turn are better than those of [32]). We used 18 pairs of protein structures taken from the PDB [3]. These proteins represent a mix of alpha, beta and alpha-beta types. Although in general residues equivalences will be sought for similar (i.e. structurally related) proteins, in some cases it might be necessary to obtain alignments (and structural similarity assessments) of proteins that are unrelated. The evaluation of *ab initio*, threading or homology modeling structure predictions [9, 45] is an example of these situations. During the assessment of structure prediction quality, a (possibly large) collection of candidate structures are evaluated against a target structure. It is not known a priori whether the candidate structures have or have not common features with the target one.

Table 2. Value of parameters for the SGMA that employs imitation, innovation and mental simulation. SGMA-1 and SGMA-2 differ in the local search probability and the CPU budget given to each local search phase

Parameter	SGMA-1	SGMA-2
Mutation prob.	0.3	0.3
Crossover prob.	0.75	0.75
μ	300	300
λ	300	300
Local search prob.	1.0	0.5
Imitation prob.	1.0	1.0
Imitation neigh. size	4	4
Mental sim. prob.	1.0	1.0
Innovation prob.	1.0	1.0
Iteration number	2	2
LS CPU budget	50	500
Mental effort	50	50
Total CPU budget	5×10^6	5×10^6

For each pair of the proteins that appear in Table 3 we executed a SGMA 10 times and report the best overlap value obtained from the 10 runs. There were two sets of 10 runs each. Each set employed slightly different parameters (which appear in Table 2) and the results are shown in Table 3.

The columns named GA and MMA were produced in our previous publications [5, 25] where both algorithms were given up to 5×10^6 function evaluations. The same budget of function evaluations were given to the SGMAs in this paper.

The multimeme algorithm (column MMA) was able to match 4 of the optimum bounds produced by the LP. In the four instances where the GA and the multimeme achieved similar results, i.e., pairs 1a8o-1f22, 1avy-1bct, 1df5-1f22 and 1utr-1wdc, the values obtained are below the LP bounds. However, we speculate that for the pairs 1a8o-1f22 and 1df5-12f22 the alignments obtained by the four methods are indeed optimal and that the LP program is able to obtain higher values for the alignments by using fractional solutions in the linear model that cannot possibly be of physical relevance. Also, it is important to note that the gap between the results by all the memetic algorithms and the LP bounds is in all cases smaller than 4 (except in the case of the pair 1c9o-1kdf for which the gap is 6) and within the logarithmic integrality gap of the IP-LP models in [32]. If we compare the values obtained by the Self-Generating Memetic Algorithms and the best of the values given by either the GA or the MMA we can observe that SGMA-1 produces lower overlap values in 7 protein pairs, better overlaps in 1 and equivalent values in 10 pairs. On the other hand, SGMA-2 produces lower overlap values in 7 protein pairs, better overlaps in 3 and equivalent values in 8 pairs. That is, overall, the SGMAs produce comparable results in terms of the overlap values than a human-designed local searcher for a memetic algorithm as 11 out of 18 pair

Table 3. Maximum Contact Map Overlap values for several protein pairs. A GA, a multimeme algorithm (MMA) as devised in [5] and two Self-Generating Memetic Algorithms (SGMA1, SGMA2). The value of the LP results are also displayed to the right

Instance	GA	MMA	SGMA-1	SGMA-2	LP
1a8o-1f22	25	25	25	25	28
1avy-1bct	22	22	22	24	25
1b6w-1bw5	23	24	24	24	24
1bct-1bw5	17	20	19	18	20
1bct-1f22	16	21	20	21	22
1bct-1ilp	18	19	20	20	23
1c7v-1c7w	62	62	62	62	62
1c9o-1kdf	31	34	32	31	40
1df5-1f22	24	24	24	24	27
1hlh-1hrf	20	22	22	22	24
1hlh-1nmf	22	23	23	21	27
1kst-2new	22	23	22	21	26
1nmf-2new	23	25	25	24	27
1nmg-1wdc	18	19	19	19	23
1pfn-1svf	16	16	16	16	16
1utr-1wdc	26	26	25	27	28
1vnb-1bhb	19	23	22	21	27
2new-3mef	23	22	23	21	26

values are equal or better than the MMA for SGMA-1 and 11 out of 18 pair values are equal or better than the MMA for SGMA-2.[10]

We compared the root mean square deviations (RMSD) obtained by a state of the art software for structure alignment, LGA [52, 53], which is available as a web server in the Protein Structure Prediction Center at Los Alamos National Laboratories. The parameters user to run LGA were $-4\ -sia\ -o2\ -d\_4.0$, meaning that a sequence independent alignment was performed with a distance cutoff of 4 Angstroms. For details on how LGA works please refer to the mentioned references. Experiments were performed on a subset of the pairs in Table 3. The results are shown in Table 4 where the RMSDs are shown for each pair. From the table we can see that our algorithms provides results of equivalent quality to those of LGA.

We also ran our algorithm and the LGA on two other data sets. One of the data sets (called "Mixed") consisted of protein structures with alpha, beta, and alpha-beta topological fingerprints. The second data set consisted of alpha-beta type of proteins (called "Alpha-Beta"). We converted the alignments produced by LGA to MAX-CMO overlap values in order to be compared with the overlap values produced by our algorithm (rather than using RMSD as in Table 3). The results are shown in Table 5 for the Alpha-Beta data set and in Table 6 for the Mixed data set.

Table 4. Comparison of the RMSD obtained with the Self-Generating Memetic Algorithms and state of the art structural alignment sofware (LGA) for a selected subset of pairs from Table 3

Pair	LGA	SGMA
1a8o-1f22	2.70	1.90
1avy-1bct	0.88	2.12
1bct-1bw5	2.01	2.11
1bct-1f22	2.40	2.20
1df5-1f22	2.35	2.04
1bct-1ilp	2.01	2.22

Table 5. Contact map overlap values for the alignments obtained with LGA and SGMA on a set of Alpha-Beta proteins

Protein pair	LGA	SGMA
1aa9-1ct9	80	131
1gnp-1ct9	64	180
1gnp-6q21	360	132
1qra-1aa9	344	110
1qra-1ct9	91	127
1qra-6q21	369	169
5p21-1aa9	334	114
5p21-1ct9	96	133
6q21-1ct9	62	138

Table 6. Contact map overlap values for the alignments obtained with LGA and SGMA on a set of Mixed proteins

Protein pair	LGA	SGMA
1aa9-1hnf	20	66
1cnp-1aa9	42	86
1cnp-1eca	54	120
1cnp-1hnf	18	53
1cnp-1jhg	73	82
1eca-1aa9	84	165
1eca-1hnf	20	63
1jhg-1aa9	22	100
1jhg-1eca	78	165
1jhg-1hnf	11	55

An inspection of these two tables shows evidence that the protein structure alignments produced by SGMA are of comparable (and in some cases better) quality to those delivered by LGA in the two data sets.

5. Conclusion

In this paper we introduced the concept of "Self-Generating Metaheuristics" and we exemplified its use in a bioinformatics domain: the comparison of protein structures. The particular implementation of Self-Generating Metaheuristics used in this paper was based on Memetic Algorithms. Unlike commonly held views on Memetic Algorithms and Hybrid GAs, we do not resort here to human-designed local searchers but rather we allowed the SGMA to discover and assemble *on-the-fly* the local searcher that best suits the particular situation. The SGMA was compared to a Memetic Algorithm that used human-designed problem specific local search operators and we found both to be of equivalent quality. One of the reasons for the success of the SGMA is that the evolved local searchers act as a (low and medium order) building block supplier. These continuous supply of building blocks aids the evolutionary process to improve solutions continuously by producing a more synergistic operation of the local searchers and the genetic operators. Thus, *the local searcher as a building blocks supplier* is proposed as a design principle for competent memetic algorithms. An inspection of the evolved local searchers reveals that the patterns that specify the move operator to use are tightly related to the number (i.e. density) and detail of the contact maps in which the local searchers are used. For example, if a map had a distribution of contacts with a large number of $i + 1$ followed by $i + 5$, for instance, and with the presence of some $i + 10$ contacts then the evolved local searchers will more likely be represented by $i + 1$ and $i + 5$ rather than by $i + 10$ features. These observations, which motivated our discussion on "designability" in [26], can be related to the concepts of building blocks scaling and cross-talk [16].

The obtained technology was tested under two different circumstances. A random problem generator was produced and the behavior of the SGMA analysed on a large set of randomly generated instances. It was found that the SGMA was capable to evolve the correct local searcher without incurring on computational overhead. That is, a Genetic Algorithm that used the same number of function evaluations as the SGMA produce considerably poorer results. On the other hand, the SGMA was compared to a human-design local search based Memetic Algorithm on 18 pairs of real world protein pairs taken from the Protein Data Bank. The former was found to produce results of equivalent quality to the later. The SGMA was also compared to one of the state of the art in protein structure alignment servers called LGA on two other data sets. SGMA produced alignment results with similar root mean square deviation and overlap values to those generated with LGA.

5.1. *Future research*

There are several research issues that are being considered. The self-generated local searchers produced a good supply of low and medium order building blocks (BB) which is suitable to align proteins with a high content of α-helices. However, if β-sheets are considered they cannot be modeled using low order BB, as they are represented in contact maps

by long edges with a regularly varying pattern (unlike α-helices which can be modeled with fix patterns). We are extending our memes grammar to account for varying patterns. In [28] we proposed a similarity metric for protein structures that can be harnessed to store and retrieve newly discovered memeplexes on a behavioral database. The data mining of such a database could be a rich source of problem domain information. Protein structural features that are preserved across protein families and captured by the memeplexes can be exploited by alignment and classification algorithms. We are working toward the creation of a user-friendly graphical interface for a structure comparison web-server where contact maps could be easily computed for a variety of conditions and their maximum overlap calculated using both the SGMA and the LP model. A rigorous benchmarking of our algorithm on a much larger set of proteins is under way and we expect to make our results public in the near future. From a strictly memetic algorithms engineering point of view we will be investigating whether the newly proposed design principle for memetic algorithms (i.e. the local searcher as a building block supplier) is of use on other domains where MAs are routinely applied.

Acknowledgments

I would like to acknowledge R.D. Carr, M.J. Collins, S. Gustafson, W.E. Hart, J.D. Hirst, G. Lancia, D.A. Pelta, J.E. Smith and B. Wallencz for our collaborations. Parts of this work were done during the author's visit to *Centro de Ciencias Matematicas. Universidade Da Madeira*, Madeira, July/August 2002. Some of the experiments for this paper were run on the cluster of the Computational Chemistry Unit, School of Chemistry, University of Nottingham. The author wants to thank A. Zemla for his support running the LGA server and the anonymous reviewers for their valuable comments.

Notes

1. Visit the NCBI site ftp://ncbi.nlm.nih.gov/genbank/genomes
2. However, other approaches exist, e.g., R can be calculated based on the side chains' center of mass.
3. A java based program compatible with PDB format for proteins and that produces several types of contact maps is available from the author by request.
4. The reader should not confuse the alignment of protein *structures*, which is the object of study in this paper, with that of aligning protein *sequences*. For details on the later see for example [40].
5. We concentrate here on MAs as they are the metaheuristic used in this paper.
6. Please consult [25] for details on scheduling local search operators in memetic algorithms.
7. This is an upper bound as many of these pairings will be crossing, and hence invalid, alignments.
8. The program to generate random contact maps was written in java 1.1:8 and is available by request from the author.
9. A higher overlap value means a better structural alignment.
10. Other experiments were performed with different genetic operators, like DPX crossover and different mutation moves, but the results were similar to the ones mentioned here, hence they are omitted.

References

1. P. J. Artimiuk, A. R. Poirrette, D. W. Rice, and P. Willett, "The use of graph theoretical methods for the comparison of the structure of biological macromolecules," Topics of Current Chemistry, vol. 174, pp. 73–103, 1995.

2. T. Back, D. B. Fogel, and Z. Michalewicz, Handbook of Evolutionary Computation, IOP Publishing Ltd and Oxford Unversity Press, 1997.
3. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissing, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," Nucleic Acids Research, vol. 28, pp. 235–242, 2000.
4. S. Blackmore, The Meme Machine, Oxford University Press, 1999.
5. R. D. Carr, W. E. Hart, N. Krasnogor, E. K. Burke, J. D. Hirst, and J. E. Smith, "Alignment of protein structures with a memetic evolutionary algorithm," in GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference, W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska (eds.), 2002.
6. H. S. Chan and K. A. Dill, "Origins of structure in globular proteins," Proc. National Academy of Science, USA, vol. 97, pp. 6388–6392, 1990.
7. F. E. Cohen and M. J. E. Sternberg, "On the prediction of protein structure: The significance of the root mean square deviation," Journal of Molecular Biology, vol. 138, pp. 321–333, 1980.
8. T. E. Creighton (ed.), Protein Folding, W. H. Freeman and Company, 1993.
9. D. Fischer, L. Rychlewski, R. L. Dunbrack, A. R. Ortiz, and A. Elofsson, "Cafasp3: The third critical assessment of fully automated structure prediction methods," Proteins—Suppl. 6, 2003 (in press).
10. M. Dorigo and L. M. Gambardela, "Ant colony system: A cooperative learning approach to the travelling salesman problem," IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, 1997.
11. W. H. Durham, Coevolution: Genes, Culture and Human Diversity, Stanford University Press, 1991.
12. I. Eidhammer, I. Jonasses, and W. R. Taylor, "Structure comparison and structure patterns," Journal of Computational Biology, vol. 7, pp. 685–716, 2000.
13. L. M. Gabora, "Meme and variations: A computational model of cultural evolution," in 1993 Lectures in Complex Systems, L. Nadel and D. L. Stein (eds.), Addison Wesley, 1993, pp. 471–494.
14. F. Glover, E. Taillard, and D. de Werra, "A user's guide to tabu search," Annals of Operations Research, vol. 41, pp. 3–28, 1993.
15. A. Godzik, J. Skolnick, and A. Kolinski, "A topology fingerprint approach to inverse protein folding problem," Journal of Molecular Biology, vol. 227, pp. 227–238, 1992.
16. D. E. Goldberg, "The Design of Innovation: Lessons from and for Competent Genetic Algorithms, Kluwer Academic Publishers, 2002.
17. D. Goldman, S. Istrail, and C. Papadimitriou, "Algorithmic aspects of protein structure similarity," in Proceedings of the 40th Annual Symposium on Foundations of Computer Sciences, 1999, pp. 512–522.
18. P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," European Journal of Operational Research, vol. 130, pp. 449–467, 2001.
19. L. Holm and C. Sander, "Protein-structure comparison by alignment of distance matrices," Journal of Molecular Biology, vol. 233, pp. 123–138, 1993.
20. L. Holm and C. Sander, "Mapping the protein universe," Science, vol. 273, pp. 595–602, 1996.
21. T. J. P. Hubbard, "Rms/coverage graphs: A qualitative method for comparing three-dimensional protein structure predictions," PROTEINS: Structure, Function, and Genetics Suppl, vol. 3, pp. 15–21, 1999.
22. R. K. Kincaid, "A molecular structure matching problem," Computers Ops Res., vol. 24, pp. 25–35, 1997.
23. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671–680, 1983.
24. J. R. Koza, F. H. Bennet, D. Andre, and M. A. Keane, Genetic Programming III, Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers, 1999.
25. N. Krasnogor, in Studies on the Theory and Design Space of Memetic Algorithms. Ph.D. Thesis, University of the West of England, Bristol, United Kingdom, 2002. http://www.cs.nott.ac.uk/∼nxk/papers.html.
26. N. Krasnogor, "Using polynomial local search and kolmogorov complexities to better understand evolutionary algorithms," DIMACS Workshop on Complexity and Inference, June 2–6,2003, Rutger University, USA, 2003.
27. N. Krasnogor and S. Gustafson, "Toward truly "memetic" memetic algorithms: discussion and proof of concepts," in Advances in Nature-Inspired Computation: The PPSN VII Workshops, D. Corne, G. Fogel, W. Hart, J. Knowles, N. Krasnogor, R. Roy, J. E. Smith, and A. Tiwari (eds.), PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9, 2002.

28. N. Krasnogor and D. A. Pelta, "Measuring the similarity of protein structures by means of the universal similarity metric," to appear in the Journal of Bioinformatics, 2003.
29. N. Krasnogor and J. E. Smith, "A memetic algorithm with self-adaptive local search: Tsp as a case study," in GECCO 2000: Proceedings of the 2000 Genetic and Evolutionary Computation Conference, D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and Hans-Georg Beyer (eds.), Morgan Kaufmann, 2000.
30. N. Krasnogor and J. E. Smith, "Emergence of profitable search strategies based on a simple inheritance mechanism," in GECCO 2001: Proceedings of the 2001 Genetic and Evolutionary Computation Conference, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshj, M. H. Garzon, and E. Burke (eds.), Morgan Kaufmann, 2001.
31. P. Lackner, A. Koppensteiner, F. S. Domingues, and M. J. Sippl, "Automated large scale evaluation of protein structure predictions," PROTEINS: Structure, Function, and Genetics Suppl, vol. 3, pp. 7–14, 1999.
32. G. Lancia, R. Carr, B. Walenz, and S. Istrail, "101 optimal pdb structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem," in Proceedings of The Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001, 2001.
33. S. Lifson and C. Sander, "Antiparallel and parallel beta-strands differ in amino acid residue preferences," Nature, vol. 282, pp. 109–111, 1979.
34. V. N. Maiorov and G. M. Crippen, "Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins," Journal of Molecular Biology, vol. 235, pp. 625–634, 1994.
35. P. Merz, "Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies," Ph.D. Thesis, Parallel Systems Research Group. Department of Electrical Engineering and Computer Science. University of Siegen, 2000.
36. L. Mirny and E. Domany, "Protein fold recognition and dynamics in the space of contact maps," Proteins, vol. 26, pp. 391–410, 1996.
37. P. Moscato, http://www.densis.fee.unicamp.br/∼moscato/memetic_home.html.
38. A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "Scop: A structural classification of proteins database for the investigation of sequences and structures," Journal of Molecular Biology, vol. 247, pp. 536–540, 1995.
39. D. A. Pelta, "Algoritmos heuristicos in bioinformatics," Ph.D. Dissertation. Departamento de Ciencias de la Computacion e Inteligencia Artificial, Universidad de Granada, 2002.
40. P. A. Pevzner, Computational Molecular Biology: An Algorithmic Approach, The MIT press, 2000.
41. P. L. Privalov, "Physical basis of the stability of the folded conformations of proteins," in Protein Folding, T. E. Creighton, (ed.), W. H. Freedman and Company, 1999.
42. M. Resende and T. Feo, A grasp for satisfiability, 1996.
43. I. N. Shindyalov and P. E. Bourne, "Wpdb a pc-based tool for analyzing protein structures," Journal of Applied Crystallography, vol. 28, no. 6, pp. 847–852, 1995.
44. C. Shumacher, M. D. Vose, and L. D. Whitley, "The no free lunch and problem description length," in GECCO 2001: Proceedings of the Genetic and Evolutionary Computation Conference, L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. K. Burke (eds.), Morgan Kaufmann, 2001.
45. N. Siew, A. Elofsson, L. Rychlewsky, and D. Fischer, "Maxsub: an automated measure for the assessment of protein structure prediction quality," Bioinformatics, vol. 16, no. 9, pp. 776–785, 2000.
46. J. E. Smith, "Co-evolution of memetic algorithms: Initial results," in Parallel Problem Solving from Nature—PPSN VII, LNCS 2439, Beyer Fgenandez-Villacans Merelo, Adamitis and Schwefel (eds.), Springer Verlag, 2002.
47. J. E. Smith, "The co-evolution of memetic algorithms for protein structure prediction," in Advances in Nature-Inspired Computation: The PPSN VII Workshops, D. Corne, G. Fogel, W. Hart, J. Knowles, N. Krasnogor, R. Roy, J. E. Smith, and A. Tiwari (eds.), PEDAL (Parallel, Emergent and Distributed Architectures Lab). University of Reading. ISBN 0-9543481-0-9, 2002.
48. E. L. L. Sonnhammer and J. C. Wooton, "Dynamic contact maps of protein structures," Journal of Molecular Graphics and Modelling, vol. 16, pp. 1–5, 1998.
49. W. R. Taylor, "Protein structure comparison using iterated double dynamic programming," Protein Science, vol. 8, pp. 654–665, 1999.
50. D. Wolpert and W. Macready, "No free lunch theorems for optimization," IEEE Transactions in Evolutionary Computation, pp. 67–82, 1997.

51. T. D. Wu, S. C. Schmidler, T. Hastie, and D. L. Brutlag, "Regresion analysis of multiple protein structures," Journal of Computational Biology, vol. 5, pp. 585–595, 1998.
52. A. Zemla, "Lga program: A method for finding 3-d similarities in protein structures," http://PredictionCenter.llnl.gov/local/lga, 2000.
53. A. Zemla, C. Venclovas, J. Moult, and K. Fidelis, "Processing and analysis of casp3 protein structure predictions," PROTEINS: Structure, Function, and Genetics Suppl, vol. 3, pp. 22–29, 1999.