Genetic Threading

J. Yadgari[1], A. Amir[1†], R. Unger[2*‡]

July 2000

[1]Department of Mathematics and Computer Science
[2]Department of Life Sciences
Bar-Ilan University
Ramat-Gan, 52900, Israel

[*] Corresponding author: ron@biocom1.ls.biu.ac.il
Tel: 972-3-5318124
Fax: 972-3-5351824

[†] Partially supported by NSF grant CCR-96-10170 and BSF grant 96-00509.
[‡] Partially supported by ISF grant 569/98.

Abstract

The biological function of proteins is dependent, to a large extent, on their native three dimensional conformation. Thus, it is important to know the structure of as many proteins as possible. Since experimental methods for structure determination are very tedious, there is a significant effort to calculate the structure of a protein from its linear sequence. Direct methods of calculating structure from sequence are not available yet. Thus, an indirect approach to predict the conformation of protein, called threading, is discussed. In this approach, known structures are used as constraints, to restrict the search for the native conformation. Threading requires finding good alignments between a sequence and a structure, which is a major computational challenge and a practical bottleneck in applying threading procedures. The Genetic Algorithm paradigm, an efficient search method that is based on evolutionary ideas, is used to perform sequence to structure alignments. A proper representation is discussed in which genetic operators can be effectively implemented. The algorithm performance is tested for a set of six sequence/structure pairs. The effects of changing operators and parameters are explored and analyzed.

Key Words: Protein Folding, Protein Threading, Genetic Algorithms

1. INTRODUCTION

1.1 Background

One of the most important open problems in biochemistry is predicting the three dimensional structure of a protein from its amino acid sequence. Proteins are key molecules in all life processes. The function of proteins is directly related to their three dimensional structure. Thus, knowing and understanding the structure of proteins will have a tremendous impact on understanding biological processes, medical discoveries, and biotechnological inventions.

By the time you have read the first paragraph many billion proteins have been folded into their three dimensional conformation in your body. Yet this routine process is not well understood, and can not yet be simulated by any algorithm on any computer. Here we report on the status of our work on using genetic algorithms to address the protein folding problem by the threading approach. The principles of our work and some preliminary results have been published recently in [Yadgari *et. al.*, 1998]. Here we present threading in the wider context of the protein folding problem, and present additional results especially with regard to the selection of the specific genetic operators that are suitable for threading.

This paper is organized in the following way: First we describe the underlying biological aspects of protein structure and protein folding. Then, we shortly discuss the current computational approaches to protein folding. Next, we describe the inverse folding approach which is, in a sense, utilization of the current known structures as constraints to form prediction of unknown structures. An example of this approach is a method known as threading in which a sequence of one protein is threaded through a template representing the structure of another protein. Threading is a difficult computational problem (shown to be NP-hard), and hence should be addressed by effective heuristics. The Genetic Algorithms model, an optimization method based on evolutionary ideas, turned out to be such a heuristic. We discuss the genetic algorithms method, and then describe our implementation of a genetic algorithm procedure suitable for protein threading and discuss the design of efficient genetic operators. We present data demonstrating the ability of the method. We conclude with a discussion suggesting what kind of improvements to the method are still needed.

1.2 Protein Structure and Folding

A protein is built up from a chain of amino acids linked by peptide bonds. See Fig 1 for a schematic view of a protein fragment. There are 20 amino acids that can be divided into several classes based on size and other physical and chemical properties. The main classification is into hydrophobic residues that don't like to interact with the solvating water molecules, and hydrophilic residues which have the ability to form hydrogen bonds with water. Each amino acid (also called a residue) consists of a common main chain part, containing the heavy atoms N, C, O, C_α that form the so called amide plane, and a side chain which, depending on the specific side chain, is built from zero (glycine, the smallest amino acid) to ten (tryptophan, the largest amino acid) additional atoms. The amino acids are concatenated through a "peptide bond" connecting the main chain C atom of one residue to the main chain N atom of the next.

The main degrees of freedom in forming the three dimensional conformation of protein molecules are two dihedral angles ϕ and ψ that allow significant three dimensional flexibility around each C_α. The side chains have additional degrees of freedom that enable them to adjust their local conformation to their environment. Proteins contains runs of amino acids that adopt similar main chain dihedral angle values. These are called "secondary structure" elements, a repetitive local arrangement of segments, in the form of helices, strands, and turns.

In living cells, proteins are synthesized on ribosomes as a linear chain from the so called N-terminal (the first residue) to the C-terminal (the last residue). The blueprint of the protein primary sequence is encoded in the DNA. Each triplet (three nucleic acids) in coding regions of DNA is translated into one of the twenty amino acids. Proteins range in size from a few dozen to a few hundred amino acids and even longer. After synthesis, the linear chain is folded to its unique three dimensional functional form, the so called "native structure". The time scale of the folding process *in vitro* is on the order of seconds, whereas folding in the complex environment of a living cell is much more complicated and may depend on additional factors.

Currently there are two experimental methods to determine the three dimensional structure (i.e. the 3D coordinates of each atom) of a protein. The first method is x-ray crystallography: The protein has to be first isolated and highly purified. Then, a series of physical techniques and a lot of patience is used to grow a crystal containing as many as 10^{14} identical protein molecules ordered on a regular lattice. The protein is then exposed to X-ray radiation and the pattern of reflections is recorded. From these reflections it is possible to deduce the actual three-dimensional electron density of the protein and thus to solve its structure. The second method is NMR (Nuclear Magnetic Resonance) which is currently applicable only to relatively small proteins. The underlying principle is that by exciting one nucleus and measuring the coupling effect on a neighboring nucleus one can estimate the distance between these nuclei. A series of such measured pairwise distances is used to reconstruct the full structure. The two methods probe the structure in two different environments, the crystal system and the solvent system. Still, in the few cases where the same protein was solved by both methods the results were virtually identical [Bax, 1989], reinforcing the claim that a protein has a unique stable native conformation.

While the experimental methods are too tedious to determine the structure of all or most proteins, the known structures have enabled us to gain an insight into the important common features of protein structures. Most soluble proteins tend to have a globular shape, forming a relatively compact core of mainly hydrophobic residues and a surface containing mainly polar and charged residues. Proteins use standard secondary structure elements which reflect specific types of local hydrogen bond patterns and local arrangements of the backbone chain. Usually four categories are considered: Helices, extended strands that are associated to form sheets, specific kinds of turns, and random coils. Different amino acids have different propensities to be included in different secondary structure elements. Another important conclusion is that proteins tend to share common structural motifs, even if their primary sequences are quite different. These principles are the basis for the threading approach that we describe below.

The exquisite 3D arrangement of proteins makes it clear that folding is a process driven into low free energy conformations where most of the amino acids can participate in favorable interactions according to their chemical nature. It should be emphasized that a protein and its surrounding solvent (i.e. thousands of water molecules) constitute a system, and that the low free energy conformation should be achieved for the whole system. Thus, the aim of most computational approaches for solving the protein folding problem is to look for the conformation with the lowest free energy. This computational approach is hindered by two fundamental problems: First, we do not know exactly what is the form of the energy function that describes the protein system. Second, we do not have an effective search algorithm that can find the conformation with minimal energy.

Two principal methods are currently in use for computing the lowest energy conformation of a protein, Molecular dynamics and Monte Carlo. In molecular dynamics [Karplus and Weaver, 1976], an all atom description is usually used. Forces acting on each atom at a particular state of the system are calculated using an empirical force field. Atoms are then allowed to move with the accelerations resulting from forces according to Newton's second law. Once the atoms have moved far enough for the forces to have changed significantly, the forces are recalculated and new accelerations applied. In practice, forces have to be recalculated approximately every 10^{-15} of a second. Even with powerful super computers only very short time periods (currently about 10^{-9}) can be simulated, much shorter than the actual folding process. Hence, this method can be used, at this time, only to describe some sub-folding events (e.g. the initiation of the process, or re-folding after a slight perturbation) rather than the whole process.

The other search technique is called Monte Carlo [Kirkpatrick *et. al.*, 1983]. It is usually used with a simplified model of a protein, for example representing each amino acid by one atom. The procedure starts with an initial conformation and makes a random "move" to another conformation. The energy of the new conformation is compared with the energy of the old one. If the new conformation is better (i.e. it has a lower free energy) the new conformation replaces the old one. If the new conformation has a higher energy, it is subject to a non-deterministic decision based on the amount of the energy gained, such that a larger energy gain makes the acceptance less likely. If the new conformation is not accepted, the old conformation is retained. Then, the current conformation is subject to another random change and the procedure iterates. Monte Carlo methods have been applied in many protein studies for different tasks with different levels of success [Covell and Jernigan, 1990; Skolnick and Kolinski, 1990; Shin and Jhon, 1991]. Yet, as a search method, even on simple models, the method is usually not powerful enough [Shakhnovich *et. al.*, 1991] to find, starting from a random conformation the lowest free energy conformation.

1.3 Threading

After many years of efforts that led to much progress in our understanding of the protein folding phenomena, a direct computation of structure from sequence is still not possible. Facing this reality, another approach to fold recognition without direct computations of the native structure was suggested [e.g. Jones *et. al.*, 1992]. The following observation was of special interest: Chothia [1992] estimated that the number of different protein folds is around 1,000 to 1,500. Later estimation by Orengo *et. al.*, [94] yielded a much higher number of protein folds, but still much smaller than the number of sequences. At the current rate of structure determination, it is reasonable to suggest that a representative of most of the folds that exist in Nature will be observed in the foreseeable future. Since there are many more different sequences than different structures, many different sequences must fold to similar conformations. Indeed, it is often the case that protein structures determined by X-ray crystallography or NMR are similar to previously observed conformations that a special effort is being made, as part as the "structural genomics" [Burley *et. al.*, 1999] initiative, to locate and solve structure of proteins that may have a novel conformation.

For a new sequence whose structure is unknown, the idea is to assume that it will adopt a conformation that was observed before, so the search is constrained to regions of the conformational space that are "around" the observed conformations, which will dramatically reduce the size of the relevant search space.

This is why threading by genetic algorithm is a good model of a biological problem where constraints supplied by the template structure are used by the algorithm in both implicit and explicit ways (for example in the control on the validity and quality of the outcome of the crossover operation) as a way to prune the huge search space.

The caveat of the whole approach is that if the new sequence at hand has a new structure (i.e. a conformation that was not observed before), then there is no way to predict its conformation. Yet, it is estimated that more than 50% of new sequences [Rost and Sander, 1996] have known conformations, and this percentage is going to increase further as more conformations are observed.

Threading means mapping a given sequence to a given structure. Based on this mapping one can estimate the compatibility of a sequence to a structure. To assign a structure to a sequence one would then need to thread the sequence through all known conformations (or to a representative subset of those), estimate the compatibility of the sequence with each one of the known conformations and assign the sequence to the conformation with which it is most compatible.

Thus, any threading procedure must contain two major components, an alignment algorithm to position a sequence on a structure and a score function to evaluate the "energy" of the sequence in the given conformation. The alignment is important since the sequence and the structure can be of different lengths, and experience shows that there are differences of insertions and deletions between sequences that share the same conformation. Thus, it is imperative to determine the best structure position of each residue in the sequence by allowing insertions and deletions to introduce gaps in the alignment. An example of such an alignment is shown in Figure 2. The energy function is then used to evaluate the quality of the fit of the sequence and the structure, reflecting criteria like positioning of hydrophobic residues in the core of the protein, hydrophilic residues on the surface, bringing to contact residues that prefer to interact with each other and separating residues that need to avoid each other. Formally, the optimization question is: given a sequence, a structure, and an energy function, find the alignment for which the energy of the sequence threaded through the structure value is optimal.

This optimization problem is hard, and has been shown formally to belong to the class of NP-hard problems [Lathrop, 1994]. Current solutions [for a review see Torda, 1997] are very limited in their performance, to quote from this review: "this is one area in which there may be some consensus: current methods do not perform well". In many cases these methods involve putting additional, not biologically necessary, restrictions on the problem, like allowing insertions and deletions only in specific pre-defined regions of the structure.

In the two meetings in which protein folding predictions were objectively evaluated (Meeting on the Critical assessment of Techniques for Protein Structure Prediction (CASP1 1994 [Moult *et. al.*, 1995], CASP2 1996 [Dunbrack *et. al.*, 1997]) it became clear that threading is a promising technique, and that in many cases it was possible to predict the correct fold for sequences of proteins whose structure was not known from experiments. It also became clear that there is a need to improve the method. Problems were identified in the two major components of the threading procedures: The energy functions were not discriminate enough to tell right from wrong structures, and the alignment algorithms often failed to identify the correct alignments. It is interesting to note that even in cases where the correct folds were predicted, the actual alignments were often quite wrong. As insight into the function of proteins can often be gained from knowledge about the location of specific residues, wrong alignments can be quite misleading. Even with these known limitations, threading became a significant tool in protein structure prediction.

1.4 Genetic Algorithms

The objective of our study is to develop an efficient application of Genetic Algorithms for threading. The Genetic Algorithm (GA) paradigm is based on the observation that natural systems have evolved to adopt very efficiently to their environment. Natural systems use replications, mutations and crossovers of their genetic material to "search" for the optimal solutions. Holland [1975] has pointed out that similar operators can be used in computational search and optimization problems. The method works in the following general way: Usually solutions are represented by strings. A population of solutions is maintained going through successive generations of evolution. Solutions can be replicated to the next generation. Solutions can mutate to form other solutions and solutions can crossover to form hybridized solutions. The crossover operation is very important as it allows creations of combined features. Better solutions have higher probability of being selected to participate in these operations, and thus the system has a selective pressure to promote better solutions. Still the population must have enough diversity to enable many properties to be included in the genetic pool. Genetic algorithm were shown to be good optimizers in a variety of problems [Goldberg, 1989]. Genetic algorithms are effective in problems where the correct solution of the full problem is composed from correct solutions to subproblems which can be recognized and promoted via the evolutionary process. Many biological problems tend to be of that nature. As a consequence, in the last few years genetic algorithms were used and shown to be valuable in many applications in computational biology [e.g. Unger and Moult 1993a, and a review in [Pedersen and Moult, 1996].

2. METHODS

Threading is a process in which a new sequence is computationally put (or threaded) on a known structure (see figure 2), and the quality of the match (the level of compatibility) between them is assessed. The algorithm threads the sequence of one protein through the known structure of another, looking for a reliable alignment with the lowest free energy value. The design of a threading procedure should include two main components: a) An algorithm to align the residues of the sequence with a structure, and b) a score (or energy) function to evaluate the quality of the alignment.

2.1 Energy Function

Our current work concentrates on the algorithmic part of the method. Currently, it does not include construction of a new energy function, rather we use one of the available energy functions. The common energy functions [e.g. Sippl 1995, Bryant and Lawrence, 1993] are based on the principle that the frequency of an occurrence of a certain feature in a large enough sample of known structures is correlated with the energy contribution of that feature: The more frequent a feature is, the more favorable in energy term it must be. More specifically, energy functions are calculated by extracting from the structural database frequencies of interactions between pairs of residues as a function of amino acids types and distance. For example, if Arg – Asp pairs are found in a distance of about 5 Angstroms in a much higher frequency than expected, then it must be a favorable interaction; the actual values are calculated as the log of the ratio of found to expected frequencies according to Boltzmann's law. Similarly, the tendency of certain hydrophobic residues to be buried inside the structure and the tendency of hydrophilic residues to be on the surface can be approximated by an energy term related to each position [Bowie et. al., 1991]. Thus, when a sequence is threaded through a given fold and residues of the sequence are assigned to three dimensional positions, it is possible to compute the free energy of the sequence in this fold by summing the singleton and pairwise energy terms. The singleton terms reflect the quality of positioning residues into the structural environment (i.e. surface/core) and the pairwise terms, where the amino acid types are taken from the sequence and the distance is taken from the fold, reflect the quality of the match between residues that are brought together by the fold.

2.2 Sequence to Structure Alignments

The calculation of energy is based on the assignment of the amino acids in the sequence into the three dimensional positions dictated by the structure. Often, inserting or deleting few residues can change the energy score dramatically. In addition, in most cases, the length of the sequence and the template structure are not the same, and hence insertions or deletions are necessary. The alignment problem is thus how to find the optimal assignment, i.e. to decide where to insert or delete residues. The problem is difficult because there is no known algorithm to search the exponentially large number of ways of selecting these insertions and deletions.

Finding the optimal sequence to structure alignment is a hard problem: Protein folding is known to be a hard problem and formal models of protein folding were shown to be NP-complete [e.g. Unger and Moult, 1993b]. There was a hope that by inverting the folding problem and formulating it as a threading problem, its formal complexity will be reduced. Combinatorially, threading is similar to the following questions: How many combinations exist to distribute k identical objects into n different cells, where a cell may contain more than one object. In threading k is the sequence length and n is the structure length. Thus the size of the search space is:

$$\frac{(n-1+k)!}{(n-1)!k!} \tag{1}$$

Search in this space was proved to belong to the NP-complete class of difficult problems [Lathrop, 1994]. Thus a deterministic algorithm that is guaranteed to find the optimal threading in all cases is very unlikely. To make the alignment calculation more feasible, many threading algorithms divide each structure into core and non-core regions. This classification is usually based on considering the secondary structure elements of alpha helices and beta sheets as rigid cores and the loop regions connecting them as non-cores. No insertions and deletions are allowed in the core regions or, in some variants, only limited changes are allowed in these regions. The score function is usually calculated based on residues assigned to core elements while loop regions are ignored or taken care of by a correction term. This makes the calculation much simpler, no gap penalties are needed. In this case the combinatorial size of the problem is

$$\frac{((m-1)+(k-c))!}{(m-1)!(k-c)!} \tag{2}$$

where k is the length of the sequence, m is the number of loops (and $m-1$ is the number of core elements), the total number of residues in core elements is c (and hence the number of residues in loops is $k-c$). It is clear that by considering only core elements the search space becomes considerably smaller, but the problem is still exponential.

While it is true that secondary structure elements are more conserved than loop regions, it is clear that loop regions do carry significant structural information. It is also clear that insertions and deletions have been observed between similar proteins even inside corresponding secondary structure elements. Threading methods that can handle full alignments without the somewhat arbitrary restrictions to core elements will thus have an important advantage. See [Thilele *et. al.*, 1999] for a recent discussion of the core restriction issue. Note that in full threading, where insertions/deletions are possible anywhere, gap penalties for creating them must be part of the score function.

Current alignment methods are based on: Enumeration [Bryant and Lawrence, 1993] which is trying all possible insertions and deletions and is possible only for very small examples; double dynamic programming approximations [Taylor and Orengo, 1989] in which an approximated alignment is calculated in the first pass, and then refined in a second pass; a recursive dynamic programming [Thiele *et. al.*, 1999] in which the regions where the sequence can be significantly matched against the structure are recursively identified, a "frozen" approximation [Godzik *et. al.*, 1992] in which a structural template library is used to match the sequence; Monte Carlo variants like Gibbs sampling [Madej *et. al.*, 1995] which sample the space of all possible alignments; Branch and bound search [Lathrop and Smith, 1996] which is exponential in nature but is reported to perform efficiently on many examples; Monte Carlo methods using a move set of small changes in the alignments [Mirny and Shakhnovich, 1998].

2.3 Representing Threading for Genetic Algorithms

Genetic Algorithms are parallel computational tools that are based on the principle of diversity and selection. Solutions are represented as strings. These strings are maintained as a population that undergoes evolutionary process via genetic operators such as: Replication (i.e. copying a string from one generation to the next); Mutation (i.e. changing a bit of bits in a string); and Crossover (i.e. concatenating a prefix of one string with the suffix of another). An evaluation function attaches fitness value to a solution as an indicator to its performance. Strings which represent solutions with higher values (i.e. better solutions) have a higher chance to participate in genetic operations. Performing this process for successive generations leads, in most cases, to the emergence of solutions with very high fitness [Goldberg, 1989, Holland, 1975].

The most important step in applying Genetic Algorithms to a given problem is choosing the appropriate representation of the solutions as strings. This is usually done by binary strings using the alphabet of $\{0,1\}$. Representing sequence/structure alignments as strings is not simple, and we suspect that it is the reason why GA application for threading were not implemented before. A good representation should enable an efficient validation that the alignment it represents is valid, should enable an efficient implementation of the genetic operators (i.e. mutations and crossovers), and should enable an efficient calculation of the energy function. We suggest using fixed length strings to represent the individual solutions, and to use an alphabet of numbers $\{0,1,2,3,...,K\}$ to represent the solutions (where K is the sequence length). Each 1 in the string represents a residue from the sequence aligned consecutively to a position in the structure. Each 0 represents that no residue is aligned to that position in the structure (this can be called either structure deletion or sequence insertion). Numbers bigger than 1 represent the number of residues that are not aligned to any structure position (sequence deletion or structure insertion). An example is shown in Figure 2.

In this form of representation it is very simple to validate the generated solutions. The total sum of the numbers that appear in the representation string should be equal to the length of the threaded sequence, while the fixed length of the string should be equal to the length of the structure. Strings that do not meet this condition do not represent a valid alignment, and any string that meets this condition is a valid alignment.

2.4 Implementing Genetic Operators

Mutations: In the context of threading, mutations mean introducing or deleting a gap or insertion in the sequence. Technically mutations are performed by increasing or decreasing randomly the value of a number in the string and offsetting it by the opposite change in the same amount in other positions. For example the threading in Figure 2 which is represented by the string 11111100111311 can be mutated to 11111100211211, i.e the insertion of three amino acids in position 12 was reduced to insertion of only two residues while another insertion of two residues was introduced in position 9. Still, a large variability in implementing this operator is possible. For example, (1) Selecting the number of changes to be performed in each string; (2) The number of residues that can be changed in each event; (3) Whether an increase in one position must be compensated by a change in a neighboring position or can the correction be spread over all of the string, etc.

An additional type of mutations can be done by selecting a small substring (e.g. 5 positions) and reversing it. This mutation is interesting since it is clear that the validity of the string as a threading solution is maintained.

Crossovers: Crossovers represent a combination of two alignments, i.e. taking the mapping of sequence to structure in the first part (the N-terminal) from one solution, and the mapping of the last part (the C-terminal) from another. Crossovers are performed by randomly choosing a position and building two new offsprings by concatenation of the prefix of one, up to the chosen position, to the suffix of the other one. Since an arbitrary fusion of two alignments is not guaranteed to represent a valid alignment, each offspring is validated. If its sum does not match the sequence length anymore then a random position is chosen near the crossover point and its value is changed accordingly.

For example, a crossover between the string 11201120111111 and the string 11111100111311 in position 8 would yield the combined string 11201120111311. Note that this string does not represent a valid threading since the its sum is 16 rather than 14. The correction mechanism might change this string around position 8 to yield 11201120001311.

Again, a large variability is possible here in the way this operation is implemented, with regard to the cutting point, and the mechanism of correcting incompatible fragments.

To calculate the energy function of a given alignment represented by a string, it is necessary to find the three dimensional position of each contributing residue in the sequence. In linear time, the string is converted back into three dimensional coordinates for which the energy can be calculated. The energy function of [Bryant and Lawrence, 1993] is used throughout this study. The fitness function that is used to select the solutions that participate in the genetic operations is obtained by simply taking the energy values of the population and normalizing them such that solutions with lower energy have fitness values that make them more likely (in reasonable proportion) to be chosen for the genetic operations.

Early convergence is a well known problem in genetic algorithms in which one individual solution takes over the populations and thus makes additional search useless. One way to prevent it is to introduce high mutation rates to constantly maintain the diversity of the population. Another possible solution is not to allow creation of solutions that appears too many times in the population. We have explored [Yadgari *et. al.*, 98] using the trie data structure [Sedgewick, 1988] to efficiently eliminate duplicate solutions. This technique was not used in the experiments we describe in this paper.

2.5 Threading without Core Restrictions

Most current threading algorithms limit the alignments only to core regions, defined either as regions of helices and sheets or as regions buried inside the protein. This limitation is mainly introduced for computational efficiency and is only partly justified by the data. A recent analysis of approximations made by threading algorithms [Zhang *et. al.*, 1997] indicates that the difference in the amount of conserved residues between core and non-core regions is not dramatic. Threading by Genetic Algorithms does not include any built-in core region limitations, since the algorithm is powerful enough to find good alignments without relying on such limitations. However, it is possible in our representation to impose such constraints simply by marking some regions of the strings as inaccessible to genetic manipulations. Indeed in the experiments described here, no core restrictions were used.

3. RESULTS

The experiments described here were done on six sequence/structure pairs that were taken from a database of structural alignments [Godzik, 1998] where the sequence of one protein was threaded through the structure of the other, see Table 1. In these cases the structures of both proteins are known, so the reference alignment, which is based on a structural super-position of the two structures, is quite reliable. However, structural alignments do not necessary lead to the optimal score of the energy function for sequence/structure alignments. This is mainly due to the well known fact that the current energy functions are far from being accurate enough to single out the optimal alignment over the huge number of alternatives [Todra, 1997].

As we have mentioned above, our goal here is not to design or test energy functions. Rather, we use an available energy function to explore the performance of the algorithms. However, we can use the structural alignments as a reference point to be compared with the results of the threading: We should require that the threading algorithm find alignments with energy scores that are as good or better than the structural alignments. Still, since the structural alignments are based on actual similarity between the two proteins, it is of interest to check whether the threading alignments are somewhat similar to the structural alignments.

Table 1 shows the results of running genetic algorithms threading for the six sequence/structure pairs, each pair was run for five multiple runs. We show results for running the algorithm for 1000 generations with population of 300 solutions, and also for 2000 generations with population of 150 solutions. The population of one generation was changed into the next by applying mutations to 25% of the strings, making 60% by crossovers, and the additional 15% were taken by simple replications. Strings were selected to participate in these genetic operations according to their fitness as explained above. Mutations were done in the following way: 5% of the positions of the string were randomly selected, and for each selected position the corresponding value was randomly changed (either increased or decreased) by up to 5. Compensating changes (see below) were done in random positions along the sequence. Table 1 shows the lowest energy score achieved in each run, averaged for the five different runs. The structural energy score was calculated by applying the energy function to the alignment implied by the structural similarity. Constant gap penalties of 3 energy units were applied for each sequence deletion in all cases. It is clear that in all cases, the threading procedure was able to find alignments that score much better than the reference structural alignments.

To asses the statistical significance of the performance of the algorithm, we run one of these examples (2pf1 on 1kdu) for a large number (100) runs. The average score was -102 with a standard deviation of 15.9. The best threading score in these runs was -135 and the worse score was -60. These results show that, as expected from a stochastic algorithm challenged by a difficult problem, there is a range of results produced by the algorithm. Thus, in practice using multiple runs (as many as your computing power allows) and choosing the best result is the best strategy.

Convergence rate of the algorithm can be seen in Figure 3 which shows the progress of the lowest energy score during the run. There is a very sharp decrease in the first 100 generations, additional significant decrease for about another 100 generations, and then it takes a long time to find a new, lower, solution. This behavior is similar for the 5 different runs we analyzed.

Figure 4 shows the average energy of the populations during a typical run. The average drops significantly in the first 200 generations to a score of about -150. After that, the average of the population fluctuates between -150 and -100, indicating that the diversity of solutions in the population is maintained.

In order to check the similarity between the threading alignments and the structural alignment we had to design a suitable representation that would enable such comparison. It is not simple to compare between two alignments (of the same sequence/structure pair) since a Hamming distance type calculation might score poorly on very similar alignments that are only slightly shifted relative to each other. We designed the following representation. For each alignment the following series is built: For each position in the structure, a running sum of the number of sequence insertions minus the number of sequence deletions is calculated. Comparing these series, either numerically or graphically, gives an indication of the level of similarity between the alignments. The advantages of this representation are clear: Small local changes between the alignments will tend to cancel out, while more significant changes will persist. Results of such comparison are shown for two pairs in Figure 5. The series of the structural alignment is shown in a solid line. From the five threading runs, we show in a dashed line the most similar one and, for comparison purposes, in a dotted line we show the least similar one. In both cases (1bbh-2ccy and 1ash-1cca), there is a result of the threading procedure that is quite similar to the structural alignments. Actually, in the case of 1bbh-2ccy even the least similar threading result is very close to the structural alignment in the C-terminal region.

Since genetic algorithms performance is dependent to a certain degree on the specific selection of the relevant operators and parameters we further explore some of these options.

We address the following seven issues:

1. The linear trade-off between population size and the number of generations.
2. Optimal level of mutation rate.
3. Locality of the mutation operator.
4. Locality of the crossover operator.
5. Regular mutations versus reverse mutations.
6. Magnitude of the mutation operation.
7. Quality control of the crossover operation.

To answer these questions we ran the genetic algorithm on the examples mentioned above, varying the relevant parameters each time and measuring the performance by comparing the average energy score of five runs.

The first two questions were addressed for other proteins in our previous study [Yadgari *et. al.*1998], the other questions are discussed here for the first time.

The running time of a GA depends linearly on the number of solutions in the population (i.e. population size) and also depends linearly on the number of generations the process is repeated. This means that for fixed computational resources, there is a trade-off between population size and number of generations. It is clear that running genetic algorithms on a population of one (or very few) element for a long time is meaningless, as is running the algorithm for very few generations on a huge population. We have run the algorithm on different combinations of population size versus number of generations, all other conditions being equal, and found that for our examples the optimal performance is achieved with population size of 300 solutions and duration of 1000 generations, a combination of 150 solutions for 2000 runs was only a little less effective, see Table 1.

A low level of mutations might reduce the divergence in the population and thus lead to early convergence. On the other hand, a high mutation rate might decrease the stability of the population and thus prevent efficient search. Varying the mutation rate, i.e. the number of strings that are subject to change between generations, the optimal rate of mutations is 0.25 to 0.3 of the populations. These two results obtained for these sets of proteins is consistent with the results we have previously obtained [Yadgari *et. al.*, 1998] for other proteins, so it seems that these conclusions are quite general.

Next we turn to inspect the selection of the genetic operators themselves. First we address the issue of the locality of the mutation operation: When a value is changed in a position in the string that represents a solution, then a compensating change must be done in other position/s. For example, if 1102114101 is changed to 110$\underline{4}$114101 then the additional sequence insertion must be compensated by deleting two residues elsewhere in the sequence. Does it make a difference if the correction is local (e.g. 110$\underline{400}$4101) or more remote (e.g. 110$\underline{4}$11$\underline{30}$01)? We have analyzed this question for the 1bbh-2ccy alignment. After 600 generations, local mutations yield an average score of -173, while non-local mutations yield a score of -174. Running the simulation to completion (2000 generations) the local mutations yield a score of -180, while the non-local mutations yield a score of -184. Thus, it seems that there is an advantage (although very small) in not restricting the mutations to be only local.

A similar question can be asked for the crossover operation. When the crossover operation is used and two alignments are fused, it is often the case that the hybrid string is not valid, (i.e. the sum of values in all positions does not match the sequence length) and a correction must be made to fix the string. Does it make a difference whether the correction is made close to the fusion point or can it be made anywhere in the string? These two alternative were tried for the set of proteins. The results, shown in table 2 are not conclusive and show different preference for different pairs of proteins.

Since regular mutation operations need a correction, we decided to try out a type of mutations that would not need such a correction. This can be achieved by randomly selecting a substring and inverting it. For example, 11$\underline{02114}$101 to 11$\underline{41120}$101. It is clear that the validity of the solution is maintained since the sum of values is kept fixed. The results are shown in Table 2, and it seems as if the reverse substring operation is significantly better than the regular mutation operation.

When a mutation is performed, the value in the position is changed. The question is, does the magnitude of the change make a difference. Is it better to change the value by one each time, or is it more effective to allow a bigger change? To explore this subject, the maximal magnitude of the change was varied in a series of experiments (namely, a maximal change of 5 means that a change of a value between 1 and 5 is selected uniformly). Note that not every change is possible, e.g. there is no way to decrease a value of 2 by 4, and hence only possible changes are considered. The results of these experiments are shown in figure 6, which clearly indicate that small mutations are better than larger ones.

The last question we have addressed is about "quality control" of the crossover operation. Crossover is a major genetic operation which fuses together two unrelated solutions and can lead to solutions that would be significantly worse than the parents solutions. While the "natural selection" inherent in genetic algorithms can take care and eventually eliminate bad solutions, it was found in some applications [e.g. Unger and Moult, 1993a] that a direct "quality control" of offsprings is useful. Namely, offsprings that perform worse that their parents are not considered for the next generation, and crossovers are repeatedly made until enough fit offspring are produced. We have tried two variants of this idea. In the first, an offspring is accepted only if its score is better than at least one of its parents. In the second, a “bad” offspring can still be accepted if it passes a Metropolis test, i.e. its score is not too much worse than its parents. The results are shown in table 3. It seems as if this policy yields a better solution much quicker, i.e. in early generations, but in the long run quality control in this application has a negative effect on the performance of the algorithm.

Running time of the algorithm are shown in Table 1 for runs of 300 solutions for 1000 generations. Times are given for a single Dec Alpha processor 255 Mhz. Running time vary from 13 minutes in the case of 2pf1-1kdu to 120 minutes in the case of 1rtc-1apa. As expected, running time scales as the square of the structure length, since the evaluation function is dominated by a quadratic pairwise summation. This dependence on time is in some sense optimal, since the evaluation of the energy function is an ”atomic” procedure in the algorithm. Clearly, with a more powerful processor and with some code optimizations, faster times can be achieved.

4. DISCUSSION

The data we have presented here indicate that the Genetic Algorithms method is a feasible and efficient approach to threading. Since the problem is very complicated and the search space is huge, it is very difficult to prove that the algorithm is capable of finding consistently good, almost optimal, solutions. Nevertheless, the ability of the algorithm to find alignments which consistently score better than the structural alignment is a positive indication. It is especially encouraging that the threading alignments are quite similar, qualitatively, to the structural alignments. We assume that the ability of the genetic algorithm to find alignments which score better than the "correct" structural alignment is due to its ability to take advantage of the peculiarities of the energy function we used. This is probably a general problem in threading procedures, which should be attributed to the energy function component of the procedure rather than to its algorithmic part. The approximate nature of any residue level based potential, even much better than the one we currently use, would enable finding solutions that score better than the "correct" ones.

Regarding to the specific questions we have studied here, the general conclusion is that genetic algorithms threading is quite robust and is not overly dependent on the particular selection of parameter or operators. Still, some observations can be made: Changing the locality of the mutation and crossover operations does not show a consistent change in the performance of the algorithm.

It seems as if doing a major change in a single position (mutations with high magnitude) is counterproductive, probably because changes between the template and the assigned structure do not tend to concentrate in single positions.

Using crossover under strict quality control was shown not to be effective, probably since the quality control mechanism is inherent in the genetic process itself.

The success of the reverse mutation is quite surprising and should be further explored. This is especially promising as this operation maintains the validity of the solution, and is very efficient to implement.

The algorithm we used here has a problem of local convergence. When using repeated runs, a variation is found between the solutions. As indicated by the results shown in figure 5, the alignments are generally similar to each other and to the structural alignment, but differ on some of the details. A local optimization, either by direct minimization or by small scale genetic operators can be suggested to find the minimum energy as an additional final step of the process.

The results we have achieved so far are encouraging. However, threading algorithms should be tested on their ability to assign a conformation for new and unknown sequence. Thus, the next stage in our project is to implement the genetic algorithm in a complete threading package, with all the necessary components and to test it in a realistic prediction setup.

acknowledgments Amihood Amir acknowledges support from NSF grant CCR-96-10170 and BSF grant 96-00509. Ron Unger is supported by grant 569/98 from the Israel Science Foundation.

REFERENCES:

Bax, A.D. (1989).
Two-dimensional NMR and protein structure.
Ann. Rev. Biochem. **58**, 223-256.

Bowie, J. U., Luthy, R., Eisenberg, D. (1991).
A method to identify protein sequences that fold into a known three dimensional structure.
Science **253**, 164-170.

Bryant, S. H., Lawrence, C. E. (1993).
An empirical Energy function for threading protein sequence through folding motif.
Proteins **16**, 92-112.

Burley, S. K., Almo, S. C., Bonanno, J. B., Capel M., Chance, M. R., Gaasterland, T., Lin, D., Sali, A., Studier, F. W., Swaminathan, S. (1999)
Structural genomics: beyond the human genome project
Nature Genetics, **23**, 151-157.

Chothia, C. (1992).
One thousand families for the molecular biologist.
Nature, **357**, 543-544.

Covell, D. G., Jernigan, R. L. (1990).
Conformation of folded proteins in restricted spaces.
Biochemistry **29**, 3287-3294.

Dunbrac,k R. L. Jr, Gerloff, D. L., Bower, M., Chen, X., Lichtarge, O., Cohen, F. E. (1997).
Meeting review: the Second meeting on the Critical Assessment of Techniques for Protein Structure Prediction (CASP2).
Fold Des **2** R27-42.

Godzik, A., Kolinski, A., Skolnick, J. (1992)
Topology fingerprint approach to the inverse protein folding problem.
J Mol Biol **227**, 227-238.

Godzik, A. (1998)
Website: http://cape6.scripps.edu/adam/service/alignbase.html.

Goldberg, D. E. (1989).
Genetic Algorithms in Search, Optimization, and Machine Learning.
Addison-Wesley, Reading, MA.

Holland, J. H. (1975).
Adaptation in Natural and Artificial Systems.
The University of Michigan Press, Ann Arbor.

Jones, D. T., Taylor, W. R., Thornton, J. M. (1992).
A new approach to protein fold recognition.
Nature **358**, 86-89.

Karplus, M., Weaver, D. L. (1976).
Protein folding dynamics.
Nature **260**, 404-406.

Kirkpatrick, S., Gellat, C. D., Vecchi, M. P. (1983).
Optimization by simulated annealing.
Science **220**, 671-680.

Lathrop, R. H. (1994).
The protein threading problem with sequence amino acid interaction preferences is NP-complete.
Protein Eng **7**, 1059-1068.

Lathrop, R. H., Smith, T. F. (1996).
Global optimum protein threading with gapped alignment and empirical pair score functions. J Mol Biol **255**, 641-665.

Madej, T., Gibrat, J. F., Bryant, S. H. (1995).
Threading a database of protein cores.
Proteins **23**, 356-369.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E. (1953).
Equation of state calculations by fast computing machines.
J. Chem. Phys. **21**, 1087-1092.

Mirny, L. A., Shakhnovich, E. I. (1998).
Protein Structure Prediction by Threading. Why it Works and Why it Does Not.
J Mol Biol **283**, 507-526.

Moult, J., Pedersen, J. T., Judson, R., Fidelis, K. (1995)
A large-scale experiment to assess protein structure prediction methods.
Proteins, **23** ii-v.

Orengo, C. A., Jones, D. T., Thornton J. M. (1994)
Protein superfamilies and domain superfolds.
Nature, **372**, 631-634.

Pedersen, J. T., Moult, J. (1996).
Genetic algorithms for protein structure prediction.
Curr Opin Struct Biol **6**, 227-231.

Rost, B., Sander, C. (1996).
Bridging the protein sequence-structure gap by structure predictions.
Annu Rev Biophys Biomol Struct **25** 113-136.

Sedgewick, R. (1988).
Algorithms.
Addison-Wesley, MA.

Shakhnovich, E., Farztdinov, G., Gutin, A. M., Karplus, M. (1991).
Protein folding bottlenecks: a lattice Monte Carlo simulation. Phys. Rev. Let. **67**, 1665-1668.

Shin, J. K., Jhon, M. S. (1991).
High directional Monte Carlo procedure coupled with the temperature heating and annealing as a method to obtain the global energy minimum structure of polypeptides and proteins.
Biopolymers **31** 177-185.

Sippl, M. J., (1995).
Knowledge-based potentials for proteins.
Curr Opin Struct Biol **5**, 229-235.

Skolnick, J., Kolinsky, A. (1990).
Simulations of the folding of globular proteins.
Science **250**, 1121-1125.

Taylor, W. R., Orengo, C. A. (1989).
Protein structure alignment.
J Mol Biol **208**, 1-22.

Thiele, R., Zimmer, R., Lengauer, T. (1999)
Protein Threading by Recursive Dynamic Programming.
J Mol Biol **290** 757-779.

Torda, A. E, (1997).
Perspectives in protein-fold recognition.
Curr Opin Struct Biol **7** 200-205.

Unger, R., Moult, J. (1993a).
Genetic algorithms for protein folding simulations.
J Mol Biol **231**, 75-81.

Unger, R., Moult, J. (1993b).
Finding the lowest free energy conformation of a protein is a NP-Hard problem: proof and implications. Bull. Math. Biol. **55**, 1183-1198.

Yadgari, j., Amir, A., Unger, R. (1998).
Genetic algorithms for Protein threading.
Proc. of the Sixth ISMB, Glasgow, J., *et. al.* EDS. 193-202.

Zhang, B., Jaroszewski, L., Rychlewski, L., Godzik, A. (1997).
Similarities and differences between nonhomologous proteins with similar folds: evaluation of threading strategies.
Fold Des **2**, 307-317.

Legends

Table 1: The results of threading algorithms for six pairs of proteins. The structural alignment score was calculated by applying the energy score on the alignment induced by the structural super-position of the two proteins. The technical details of the genetic algorithms used here are given in the text. In all cases, the alignments produced by genetic algorithms score better than the structural alignments.

Table 2: Exploring variation of the genetic operators. The standard conditions for the runs are described in the text. In "nonlocal crossovers", the correction of fused solutions is done all over the string, not only near the cutting point. In "reverse mutations", values in the string are not changed randomly, rather a substring of 5 positions is randomly chosen and its order is reversed. Results are shown for runs with 150 solutions (for 2000 generations), and 300 solutions (for 1000 generations). The results for the nonlocal crossovers are not consistent for all pairs. Reverse mutations are advantageous in all cases.

Table 3: Controlling the quality of the crossover operation. Offspring solutions are accepted into the next generation of the population only if they pass a quality control check. Two policies were tested. Minimization, where the offspring has to be better than at least one of its parents, and Metropolis where it has to pass a non-deterministic comparison with its parents based on the Metropolis test [Metropolis, 1953]. In both cases, quality control produces good solutions very fast, but in the long run it is less effective then regular runs.

Figure 1: A schematic figure of a short fragment of a protein. Three amino acids are shown: alanine (A), Aspartic Acid (D) and phenylalanine (F).

Figure 2: A schematic view of the threading process. The structure is presented as by the gray trace through the circles that represent the structural positions (14 in the example). The sequence on the top left is threaded through this structure, i.e. putting amino acids

from the sequence into structural positions. In this alignment two structural positions were deleted (the shortcut by the dashed line), and two sequence residues were deleted (the G and the A which are looped out). The specific alignment is encoded by the string in the bottom where values of zero indicate structure deletion and values larger than one indicate sequence deletion. Note that the length of the string is equal to the length of the structure, and the sum of all values in the string is equal to the length of the sequence. In this way the structure is used as a template to constrain the possible arrangements of the threaded sequence.

Figure 3: The minimal energy of threading the sequence of 1bbh on the structure of 2ccy. The plot shows five runs (300 solutions for 1000 generations). For each generation, the lowest energy score found until that generation is shown (data is shown each 25 generations). The data show that there is a sharp decrease in the lowest energy found in the first 200 generation, after which further improvement is slow. Also note that the results of the five runs are quite similar suggesting that the algorithm converged to a similar alignment.

Figure 4: The average energy of the population during threading the sequence of 1bbh on the structure of 2ccy. For a run with 300 solutions for 1000 generations, the average energy of the population is shown (data points every 25 generations). After an initial drop in energy, the average energy fluctuate indicating that the variability in the populations is maintained.

Figure 5: A comparison between structural alignment and the alignments obtained by genetic algorithms. To enable comparison, alignments are encoded as series in which each position reflect the difference between sequence deletions and structure deletions up to that position. The solid line represents the structural alignment, the dashed line the threading alignment which is most similar to it, and the dotted line the threading alignment which is the least similar to it. The result indicate that threading alignments do resemble the structural alignment.

Figure 6: Changing the magnitude of mutations. The magnitude of the maximal change that can be done in a single mutation was varied from 1 to 10 in the alignment of 2ccy to 1hbb.

The solid line shows the average score of five runs (150 solutions for 2000 generations) after 600 generations, the dashed line shows the score after 2000 generations. It is clear that using large mutations is counterproductive.

Table 1: Threading results

Protein Family	Seq. .	Length	Struct.	Length	Struct. score	Threading (150 sol.)	Threading (300 sol.)	Time (Min.)
Four helix bundles	1bbhA	131	2ccyA	127	-123	-170	-175	29
Globins	1ash	147	1eca	136	-84	-109	-111	33
Kringle domains	2pf1	121	1kdu	80	98	-103	-94	13
Immunoglobulin fold	2rhe	114	1tlk	103	-20	-122	-122	20
Cytochromes c	1ccr	111	451c	82	35	-30	-37	14
Antiviral protein/ricin	1rtc	268	1apa	261	-436	-460	-491	120

Table 2: Effectiveness of genetic operators

Seq.	Struct.	Standard		Nonlocal crossovers		Reverse mutations	
		150	300	150	300	150	300
1bbhA	2ccyA	-170	-175	-163	-177	-179	-186
1ash	1eca	-109	-111	-112	-116	-127	-131
2pf1	1kdu	-103	-94	-72	-91	-132	-135
2rhe	1tlk	-122	-122	-113	-121	-133	-125
1ccr	451c	-30	-37	-38	-36	-57	-55
1rtc	1apa	-460	-491	-468	-481	-511	-518

Table 3: Quality Control: 1bbh-2ccy

Number of Generations	Standard	Minimization	Metropolis
25	-30	-130	-142
100	-62	-138	-153
600	-158	-156	-155
1000	-178	-161	-163

Figure 1: A schematic view of a polypeptide chain

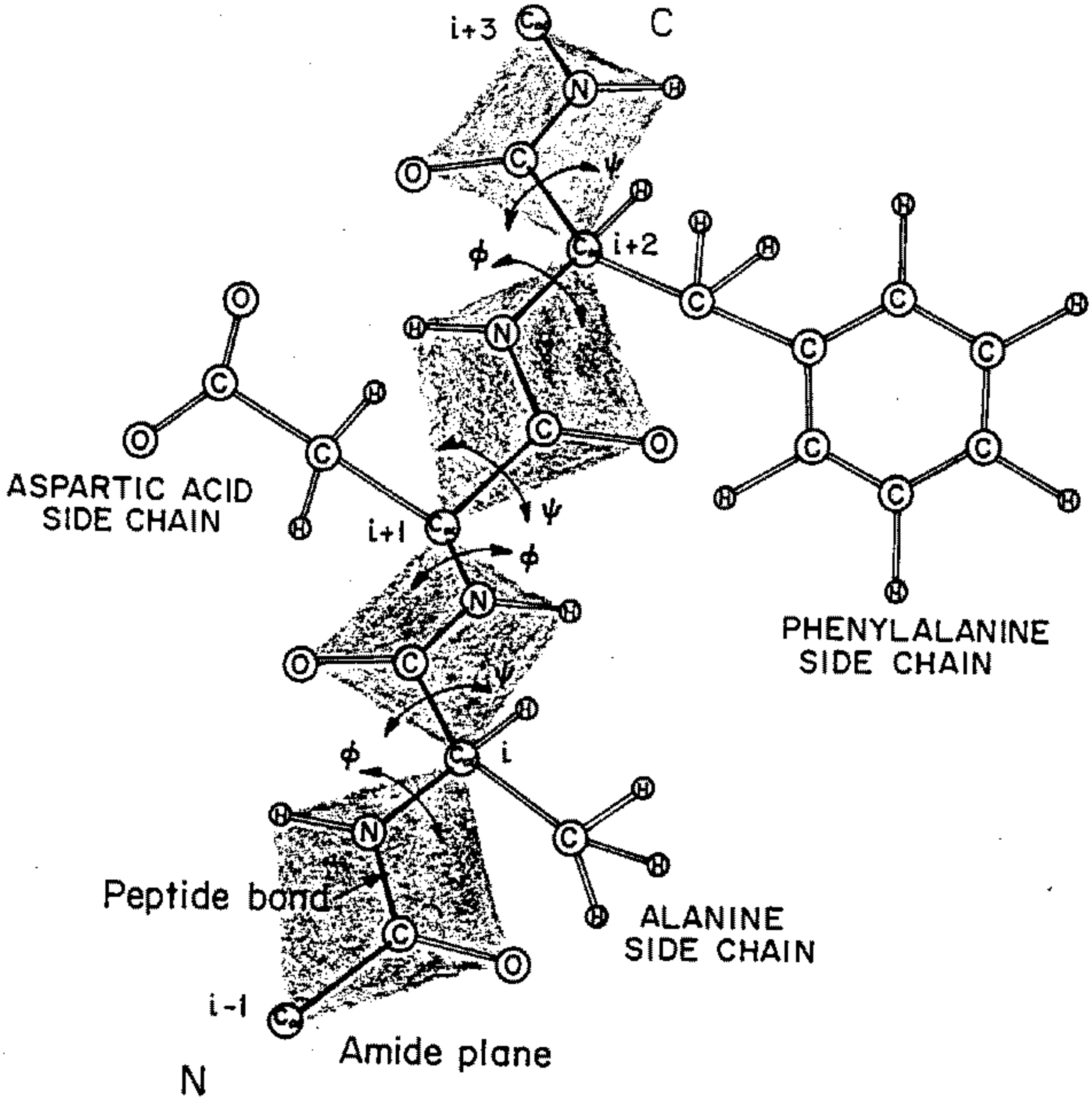


Figure 2: Threading and its string representation

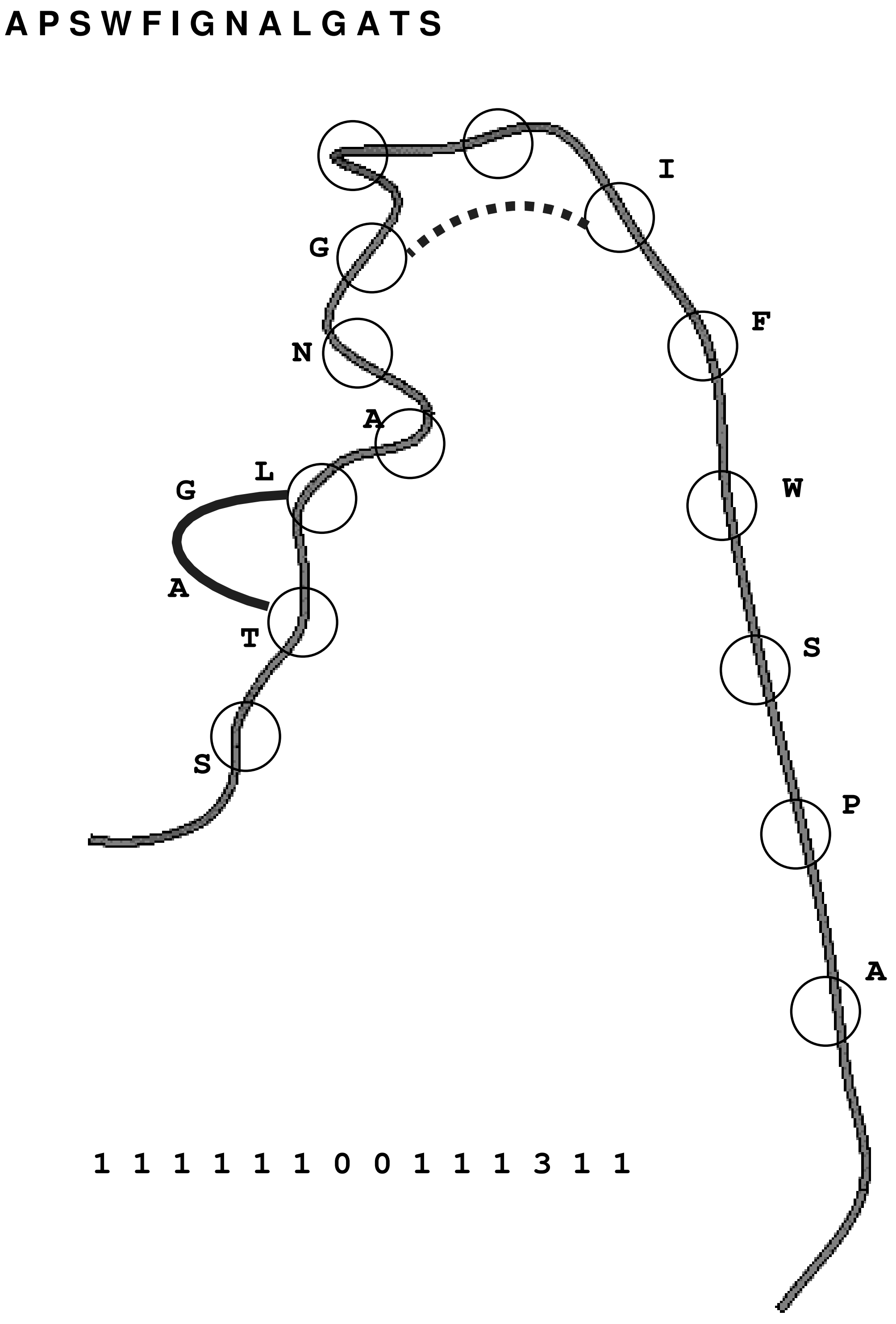


Figure 3: Minimal energy of threading 1bbh−2ccy

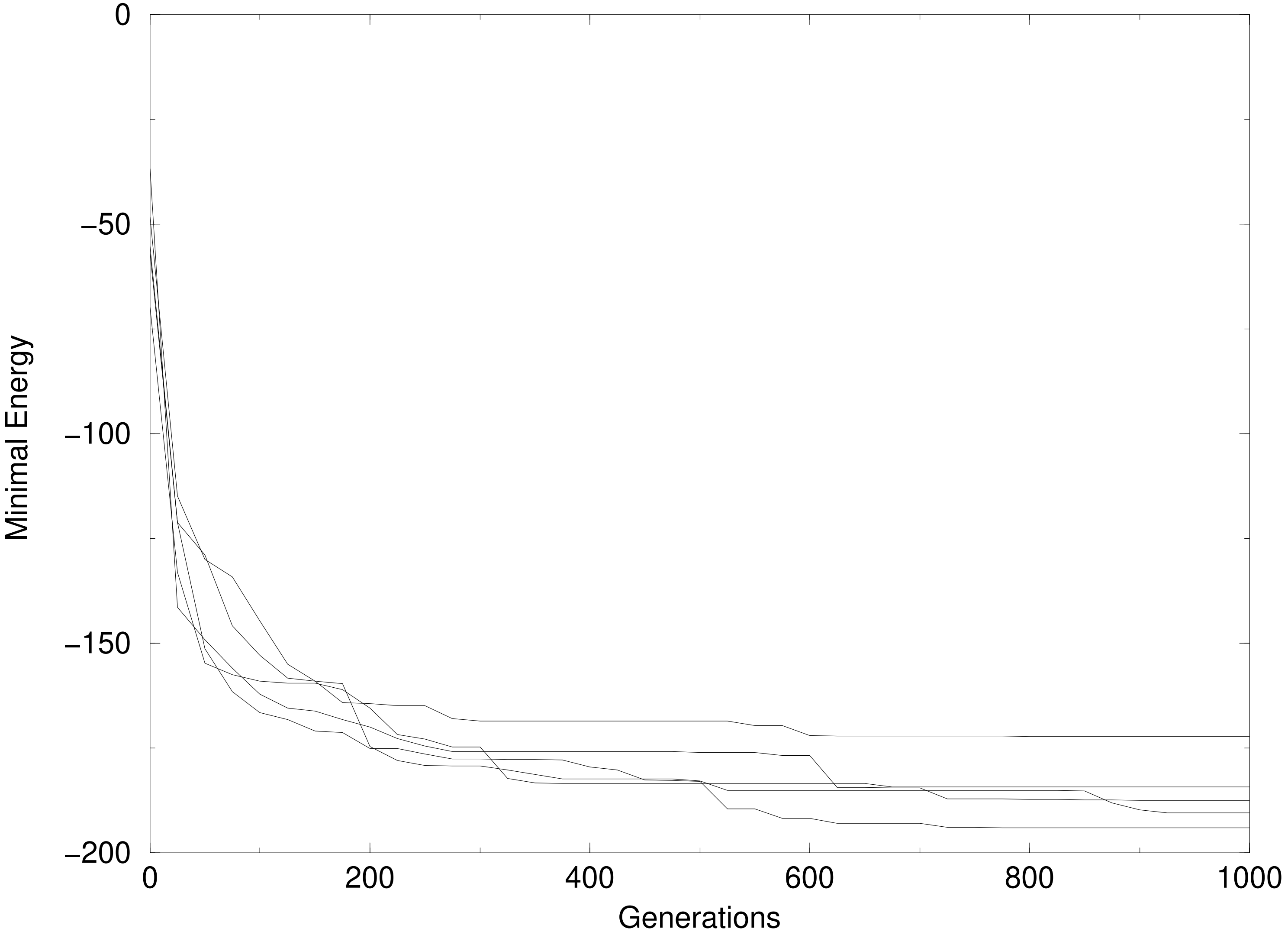


Figure 4: Average energy of threading 1bbh−2ccy

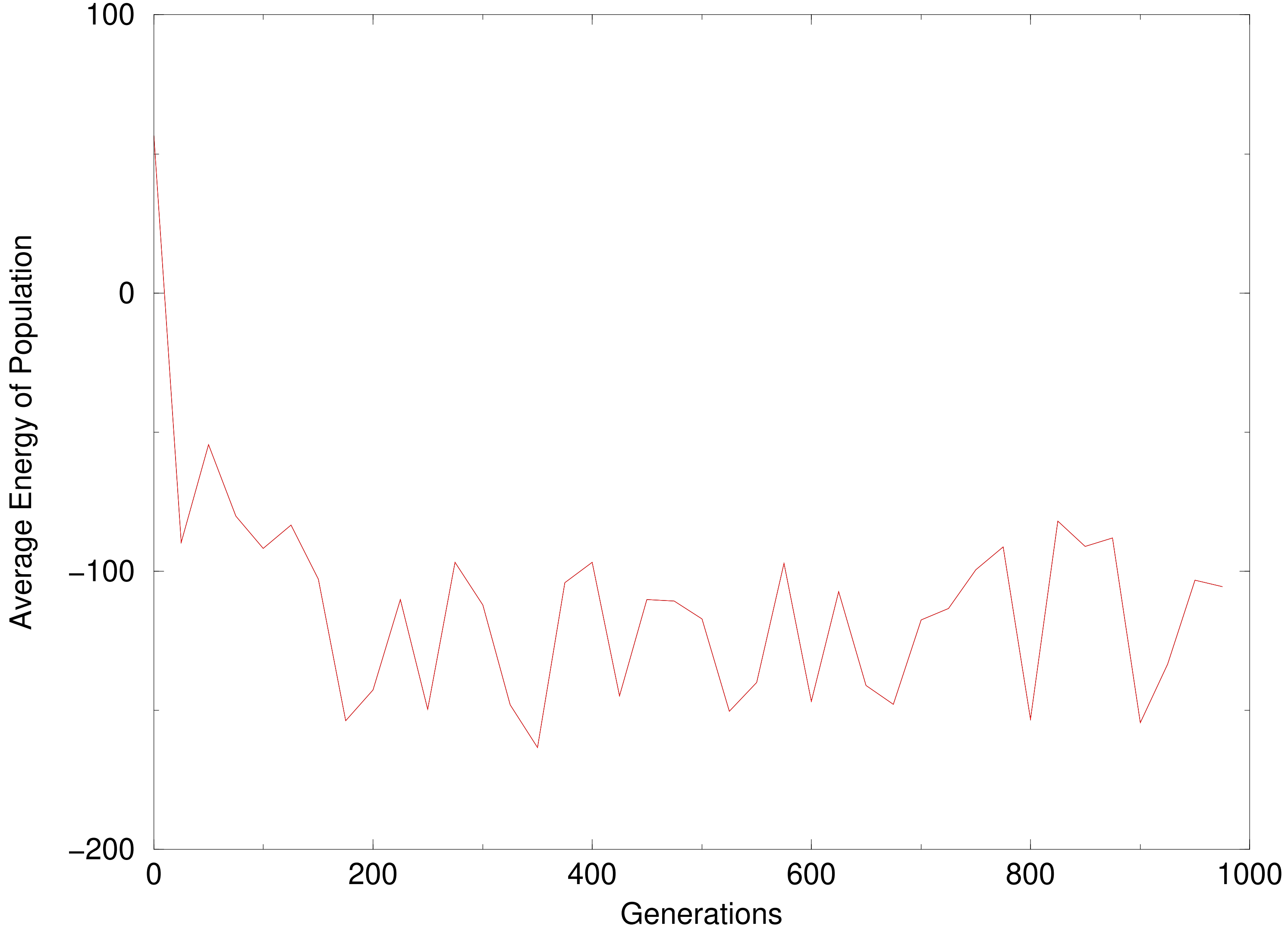
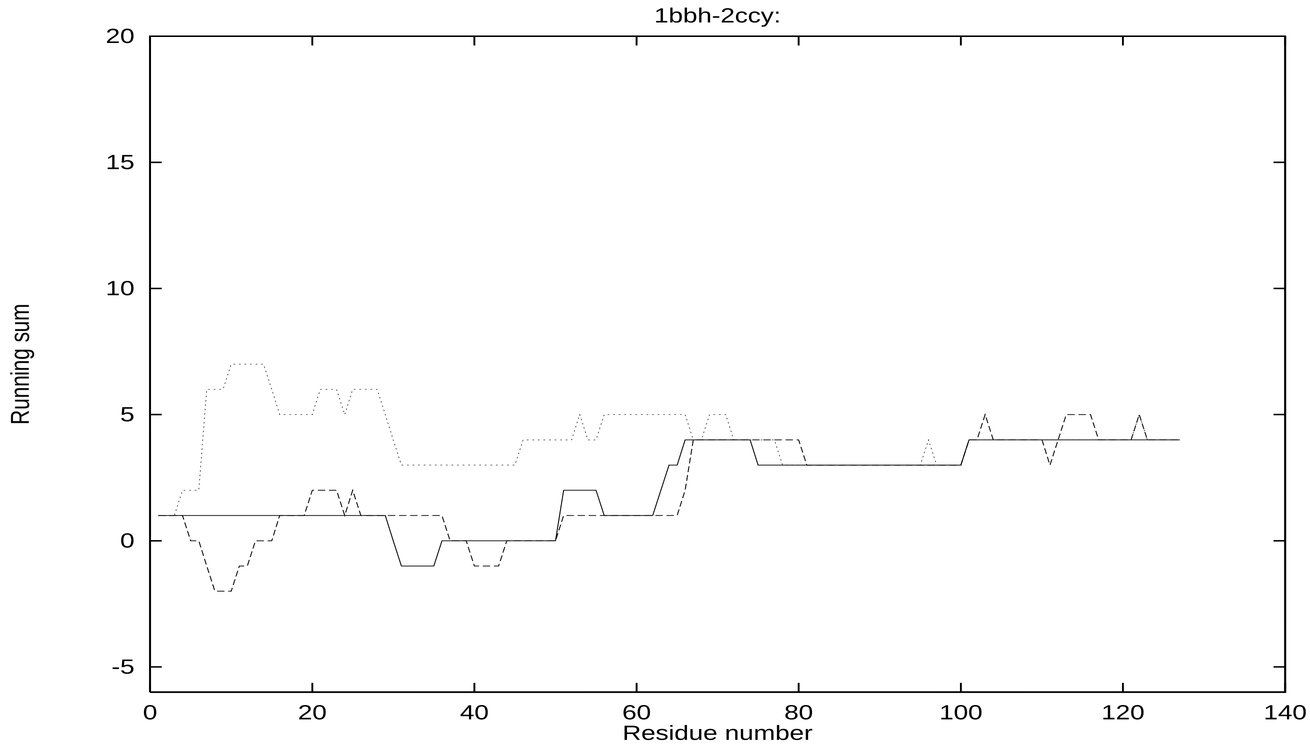


Figure 5: Structural alignment versus threading



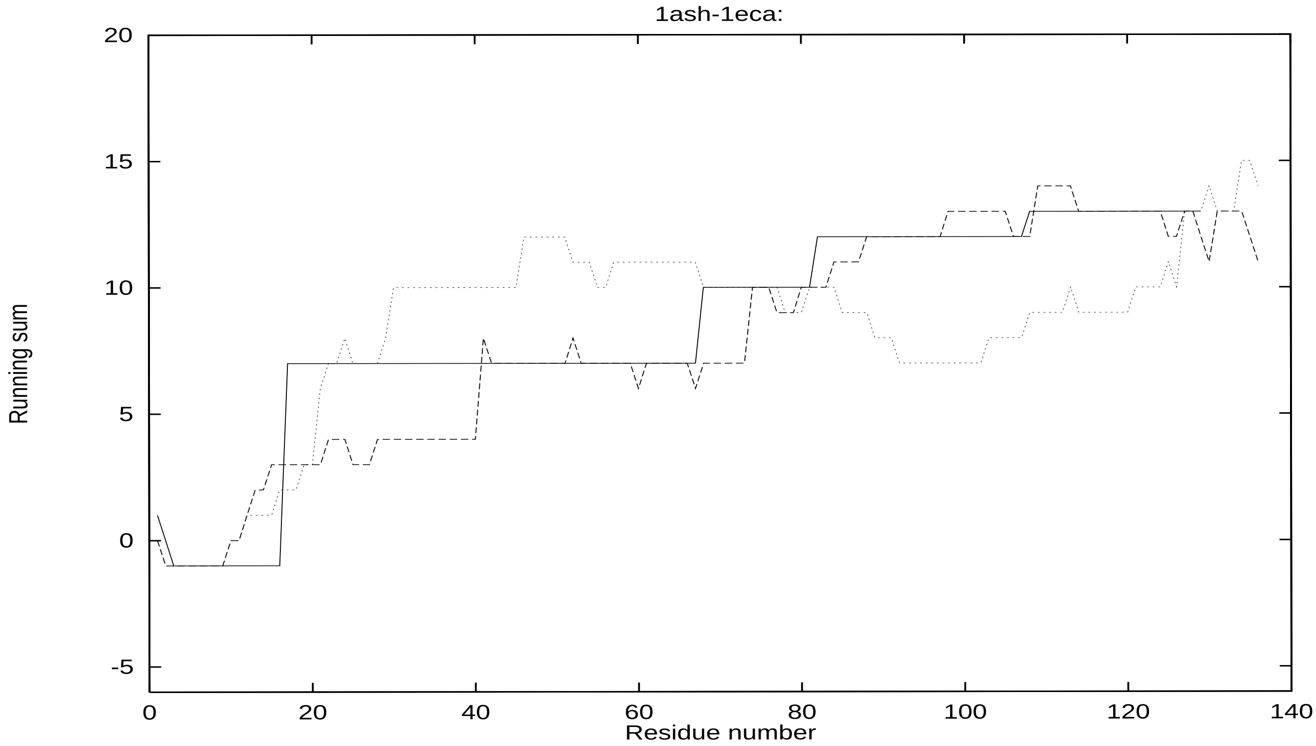


Figure 6: Changing the magnitude of mutations