A Hybrid Genetic Algorithm for the 2D HP Protein Folding Problem

Alena Shmygelska
University of British Columbia
Computer Science Department
oshmygel@cs.ubc.ca.

Rosalía Aguirre-Hernández
University of British Columbia
Institute of Applied Mathematics
rosalia@cs.ubc.ca

April 15, 2002

Abstract

A hybrid between local search heuristics and genetic algorithms has shown to be effective approach for global optimization on a number of combinatorial problems. In this paper we consider a new Hybrid Genetic Algorithm (GA) for the protein folding problem using the 2D HP Model. We study the impact of several algorithmic factors on the given problem: the use of local search, the location of the starting folding point, elitist versus non elitist strategy during the selection of individuals to the next generation, the effect of the crossover operation, and non-random mating versus random mating. Further, we compare the obtained results with the results from Ant Colony Optimization (ACO) implemented for the Stochastic Search Algorithms class [14]. Both of these algorithms have not been studied in the context of the protein folding problem.

1 Introduction: Protein Folding Problem

The protein folding problem is one of the most challenging in current biochemistry and molecular biology, and is a very rich source of interesting problems in local and global optimization. After the successful deciphering of the genetic code that defines how the amino acid sequences of proteins are coded in the DNA, one of the major missing steps in understanding the chemical basis of life is the protein folding problem: the task of understanding and predicting how the information coded in an amino sequence of proteins at the time of their formation translates into the 3-dimensional structure of the biologically active protein. Since it is already known how to use genetic engineering to produce proteins with a given amino acid sequence, knowledge of how such a protein would fold would allow one to predict its chemical and biological properties. If we were able to solve the protein folding problem, it would greatly simplify the task of interpreting the data collected by the Human Genome Project, understanding the mechanism of hereditary and infectious diseases, designing drugs with specific therapeutic properties, and growing biological polymers with specific material properties.

Currently, biochemists use techniques such as MRI (magnetic resonance imaging) and X-ray crystallography on protein crystals in order to determine the conformation of the protein. These techniques are expensive in terms of equipment, computation and time. Additionally, they require isolation, purification and crystallization of the target protein. Therefore, the study of various computational models is widely employed.

2 Motivation and Background

The objective of this project is as follows: given the sequence of amino acids of a protein, predict its structure on a simple two dimensional Hydrophobic - Polar lattice model (2D HP) using a Hybrid GA approach, a combination of GA and a local search. We will study the behavior of the Hybrid GA algorithm, and try to find how to improve its performance based on what we learn about the search space of this problem during the experimental phase of the project. Specifically, we are interested in studying the effect that different variants of the genetic operators (mutation, macro-mutation, crossover and selection) have on the behaviour of the Hybrid GA algorithm. In order to study the advantages that GA provides, we will compare the running time of a GA plus local search with the local search alone. Finally, we will evaluate the performance of the algorithm and compare it with the performance described in literature (Krasnogor *et al.* [6], Patton *et al.* [9] and Unger & Moult [15]), and also compare with the Ant Colony Optimization (ACO) that has been implemented for the Stochastic Search Algorithms class [14].

We chose to study GA because it has been shown that it produces good results for the 2D HP protein folding problem. Our goal was to develop new Hybrid GA and to conduct a complete empirical study of the GA, that will help in development of better algorithms for the protein folding problem.

2.1 The 2D HP Lattice Model

The Hydrophobic-Polar model (HP model) was proposed by Dill [8]. In the HP model the sequence is "folded" on a lattice. The feasible protein folds are represented as self-avoiding walks on a lattice in which vertices are labeled by the amino acids. The amino acids are classified as hydrophobic (H) or polar (P).

It was previously discovered using wet lab techniques that:

1. native structures of proteins are compact and have well-packed cores which are highly enriched in hydrophobic residues [10].
2. hydrophobic interaction is the driving force for protein folding [8].
3. native structures of proteins have minimal solvent-exposed non-polar surface areas [8].

4. the hydrophobicity of amino acids is the main force for development of a native conformation of small globular proteins [10].

Globular proteins tend to form one or more hydrophobic cores, i.e., the more hydrophobic amino acids are concentrated in the compact cores whereas the more hydrophilic amino acids are located on the surface of the protein.

One strategy for finding good folds is to look for configurations that maximize the number of H-H contacts. The HP model adds a value of ε (usually -1) for every pair of hydrophobic residues that form a topological contact and are not neighbors on the chain. For example, Figure 1 represents a particular conformation of the protein *hphpphhphpphphhpphph* where the black squares represent hydrophobic amino acids and the white squares represent polar amino acids. The dotted lines represents the H-H contacts. The fold in Figure 1 has an energy of -9 which is the minimum energy for this protein.

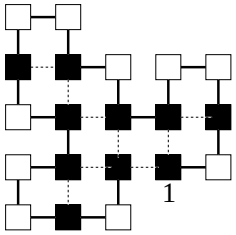

Figure 1: Optimal conformation of the protein hphpphhphpphphhpphph

The folding of a protein is encoded either in *absolute* coordinates $\in$ { U(up), D(down), R(right), L(left)} or in *relative* (to the previous amino acid) coordinates $\in$ { S(straight), L(left), R(right) } on the 2D square lattice. The absolute and relative representations for the protein in Figure 1 are (*g, r, u, u, l, d, l, u, l, u, l, d, d, r, d, l, d, r, r, u*) and (*g, g, l, s, l, l, r, r, l, r, l, l, s, l, r, r, l, l, s, l*) respectively. Note that one amino acid in absolute coordinates and two amino acids in relative coordinates are fixed (g-grounded) because it is necessary to have a reference point.

In order to make the mutations during the local search phase more efficient we used relative coordinates. It has been shown that there is a greater chance to get a valid conformation by flipping the direction coded in relative coordinates since flipping corresponds to a rotation [15].

Of course, this model is very simplistic. Interactions in a realistic protein are much more complex and are three dimensional. But this model allows us to study techniques for global optimization (in this case, GA) for simplified version of the protein folding problem. As mentioned above, this simplistic model has valid biomolecular reasons for existence.

2.2 Simple Genetic Algorithm

Before giving a description of the different types of GAs found in the literature, we will explain the characteristics of a simple GA algorithm. GAs are population based search procedures that are based on the mechanism of natural selection. GA works with a population of individuals that represent candidate solutions to a given problem. The fitness of an individual is the objective function value of a corresponding candidate solution. In each iteration, three genetic operators are applied to a population producing a new set of individuals. The genetic operators are selection, mutation and crossover. The selection operator chooses members from the population in a way that is proportional to their fitness. Thus, the fittest individuals have higher probability to appear in the next generation. Selection process intensifies the search. The crossover operator combines two individuals (parents) in a way that the resulting offspring is likely to share desirable properties of both parents while improving over their fitness. This operator diversifies the search process. The mutation operator perturbs candidate solutions using small exchange neighborhood. Mutation also diversifies the search. Crossover and mutation are usually applied with a certain probability to each individual in the population. The general outline of a GA algorithm is as follows:

procedure *GeneticAlgorithm*
  $t \leftarrow 0$
  initialize the population at time t, $P(t)$
  while *(***not** *terminate)* **do**
      $t \leftarrow t + 1$
      crossover of $P(t)$
      mutation of $P(t)$
      selection from $P(t)$
  end
end

2.3 Review of existing work

Recent computational analysis of the protein folding problem has shown that the protein folding problem is intractable (NP-hard) on simple lattice models [6]. Consequently, heuristic optimization methods have been widely used on this problem. Solutions to the 2D HP model were obtained using Simulated Annealing (SA) (variations of the Monte Carlo method) [12] and Genetic Algorithms (GA) [13]. In particular, GAs have been shown to be robust and effective global optimization techniques for protein structure prediction [6].

An early application of GAs to protein structure prediction was that of Unger and Moult [13, 15]. They presented a nonstandard GA that has characteristics of SA. Conformations were codified with absolute coordinates.

Unger's and Moult's GA only considers feasible conformations that are self-avoiding paths

on the lattice. The crossover operator tries to rejoin the sequence at a 0, 90 or 270 degree angle (in random order) and produces only one child. Figure 2 represents an example of crossover with the crossover point after residue 14 (position B in Figure 2) and with a 90 degree rotation.

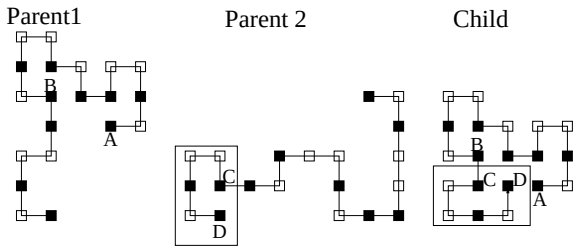

Figure 2: One point crossover mechanism.

Unger and Moult also incorporated characteristics of SA into their algorithm. After a conformation is mutated or a child is generated by a crossover operation, the new conformation is accepted if its energy is less than the energy of the old conformation. On the other hand, if the energy of the new conformation increases, it is accepted with probability:

$$exp(\frac{E_j - E_k}{c_k})$$

where E_j and E_k are the energies of the old and the new conformations respectively, and c_k is the temperature parameter. Unger and Moult fold sequences of length ranging from 20 to 64 residues (see Table 1). They obtained the optimum conformation for every sequence except for Sequence 8. In this case the best value they achieved was -37. Unger and Moult do not include an empirical evaluation of their algorithm; they only report the average number of energy evaluations performed to find the optimum.

Patton *et al.* [9] described a GA that significantly outperforms the GA used by Unger and Moult [15]. They employed a coordinate representation that uses relative directions. The authors penalize infeasible conformations instead of eliminating them from the search space. Patton *et al.* argue that preserving diversity or preventing convergence seemed to played a major role in the success of their implementation. Indeed, they used a high crowding factor and an incest reduction to preserve diversity. The crowding factor is the replacement of the closest individual in terms of the Hamming distance out of a pool of n individuals selected for replacement. The incest reduction consists of choosing an individual farthest in terms of the Hamming distance from the pool of n individuals selected for breeding.

Krasnogor *et al.* [7] implemented another GA in which the conformations are represented in relative coordinates. They introduce macro-mutations which consist of selecting two points of the sequence and modifying the directions between these two points in one of the following ways:

Seq. No	Length	Opt.	Protein
1	20	-9	hphpphhphpphphhpphph
2	24	-9	hhpphpphpphpphpphpphhh
3	25	-8	pphpphhpppphhpppphhpppphh
4	36	-14	ppphhpphhpppppphhhhhhpphhpppphhpphpp
5	48	-22	pphpphhpphhpppppphhhhhhhhhhpppppphhpphhpphpphhhhh
6	50	-21	hhphphphphhhhphpppphpppphpppphpppphpppphphhhhphphphhh
7	60	-34	pphhhphhhhhhhhpppphhhhhhhhhhphpppphhhhhhhhhhhhpppphh hhhhphhph
8	64	-42	hhhhhhhhhhhhphphpphhpphhpphpphhpphhpphpphhpphhpphp hphhhhhhhhhhh
9	20	-10	hhhpphphphpphphphpph

Table 1: Benchmark problems.

1. randomly, where random directions are chosen for each position between the two points
2. unfold the segment
3. rotate the segment by 0, 90, 180 or 270 degrees
4. horizontal/vertical reflection, see Figure 3.

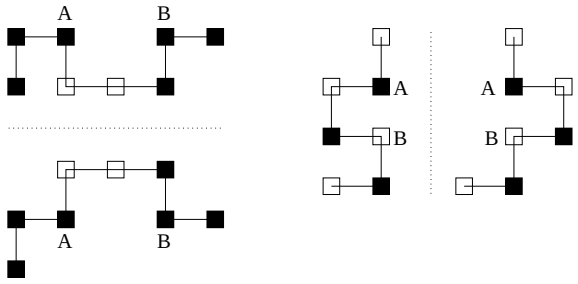

Figure 3: Macro-mutation: horizontal and vertical reflection.

Krasnogor *et al.* [7] empirically evaluated mixtures of evolutionary operators (mutations, macro-mutations, crossover) that are most useful for solving the protein folding problem for the HP model. Their experiments compared GAs that used these operators with different combinations of probabilities. For the instances they studied, the best combination of parameters had a small crossover probability and high mutation and macro-mutation probabilities. Their results suggest that (1) one point crossover along was not able to transfer building blocks and (2) the macro-mutation was more important than crossover for searching the optimum. Krasnogor *et al.* [7] found the optimum value for Sequences 1

Seq.No.	Length	Opt.	Unger Energy (no. evaluations)	Krasnogor Energy (no. evaluations)
6	50	-21	-21 (592,887)	-21 (50,085)
7	60	-34	-34 (208,781)	-33 (40,014)
8	64	-42	-37 (187,393)	-39 (20,067)

Table 2: Results from Unger and Moult and Krasnogor *et al.*.

through 6 and 9. The best value they achieved for Sequence 7 was -33 and for Sequence 8 was -39. They report less energy evaluations than Unger and Moult, see Table 2. As Unger and Moult, Krasnogor *et al.* do not include an empirical evaluation of the algorithm.

3 Our Genetic Algorithm

Our approach is based on a genetic local search algorithm, i.e., a GA combined with an additional local search method. In this algorithm, the GA effectively searches the space of local optima found by the local search method. The local search gives more search intensification which is generally needed in pure GA algorithms. The GA algorithm works with a population of individuals. In the protein folding problem, the individuals represent different conformations or folds of a given protein in the 2D HP model.

The genetic operators that have been used and evaluated in our implementation of the GA are as follows:

1. **Mutation.** We studied how the probability of mutation affects the overall performance and convergence of the algorithm. The algorithm converges when the standard deviation of the fitness of the individuals in the population is zero or close to zero.

2. **Crossover.** We implemented one point crossover, and studied the effect of the random and non-random mating strategy for choosing parents.

3. **Local Search.** In order to increase the intensification of the search and drive the solution to the local optima we used local search, specifically, we implemented iterative first improvement and compared it with the genetic local search algorithm (GA + iterative first improvement).

4. **Selection.** The selection process was implemented as a roulette wheel selection. [2] Here we studied the effect of having selection process between children and parents (intensification strategy) or just children (diversification strategy).

Infeasible conformations generated after mutation and crossover were eliminated.

4 Empirical evaluation

We used the benchmark instances listed in Table 1 that are found on William Hart and Sorin Istrail's web site [3]. The optimal energies for the first three sequences were validated by full enumeration of the energies of all valid conformations.

The methodology for the empirical analysis of the GA algorithm is based on a run-time distribution (RTD) of the known optimal solution for individual problem instances [4]. The RTDs were computed as follows: we run GA 100-200 times for each instance to ensure that the estimates for the RTD is sufficiently stable. In each run we recorded whenever the new best solution is found and the time (CPU time [sec] / search step) needed to obtain it. We sorted solution quality traces and obtained cumulative qualified RTDs. All plots are done on a semi-log graph [5] to give a better view of the distribution over its whole range. On the *x-axis* we plot log time (CPU sec) and on the *y-axis* we plot proportion of the runs that have reached the desired (optimal) solution quality. The results for larger sequences are not graphically represented since the number of times the optimum was found is too low for plotting RTD graphs. All results have been measured on a Pentium V processor with 566 MHz. At least 100 generations of GA were performed for each instance. The cut off point is when there has been no improvement of the global best energy for a set number of generations (usually a 100-1000 depending on the length of the sequence).

In the following subsections we conduct an empirical study of a number of algorithmic issues for GA: generation of the initial population, population size, mutation rate, mating strategy, local search, selection process. During the testing process we studied one variation of a single parameter at a time, since there is a number of parameters and variations that can be studied, and an exact relation between parameters is difficult to infer.

4.1 Initial Population

The initial population for Hybrid GA is generated randomly, following the generally accepted approach [16]. During the initial construction process entanglement poses a big problem for large proteins, therefore we developed a new backtracking scheme. While constructing a protein we check the feasibility of the conformation every time a new residue is added. If there are no feasible directions for the current residue we backtrack half the distance already folded (various modifications of the backtracking were tested, this particular one showed fast and reasonable performance).

All individuals in the population have the same starting point, since it simplifies operation of the crossover. Generally in literature protein is folded starting from the first residue [6, 13, 15], but we considered different starting folding points: beginning, middle and end residue, since it has been shown that proteins employ folding pathways to avoid extensively searching the whole conformation space [11]. Proteins fold by hierarchical condensation, therefore it is not unreasonable to consider various choices for the starting folding point. We fold first one part of the protein, and then the other. This idea is based on the fact that real proteins have folding nuclei [1], and it should be most efficient to start from

such a nucleus.

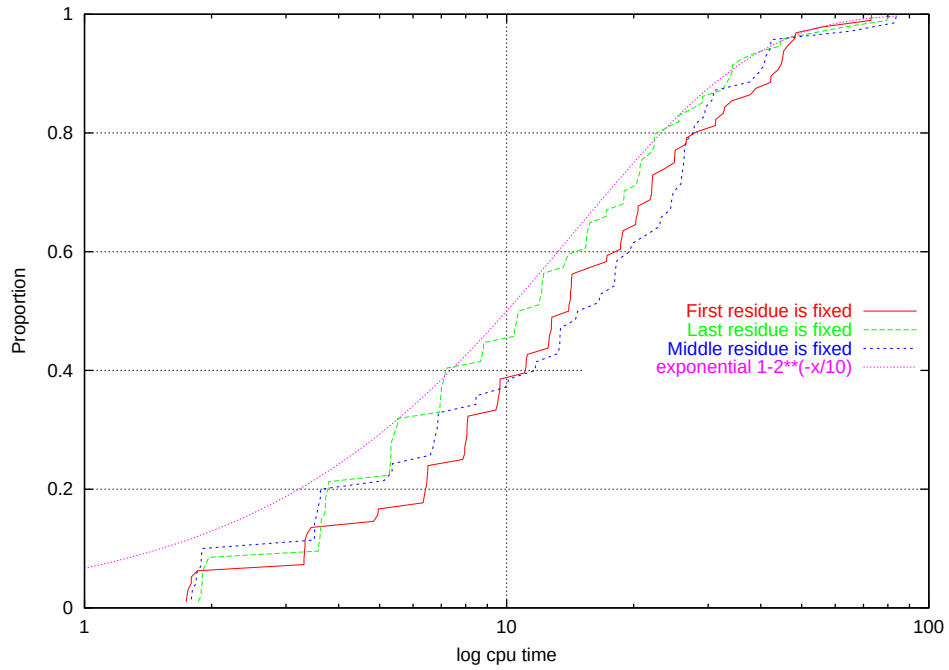

Figure 4: Effect of the position of the starting folding point, Sequence 1.

Results for Sequences 1 and 3 can be seen in Figure 4 and Figure 5. In some cases the choice for the starting point was very successful and speed up the search substantially, in others it did not. We take this observation as an indication that in various sequences the end group already provide an effective nucleation site. We also tried to grow the chain on both sides simultaneously. However, it turned out that this is not effective computationally.

4.2 Population Size

Population size is an important factor in any GA. In our Hybrid GA it plays even more important role, since if the population is too large it is inefficient to conduct the local search. We tested Hybrid GA with different population sizes ($PopSize$), see Figure 6 for results from Sequence 1. Increased population size increases diversity thereby reducing convergence speed, but at the same time requires more CPU time per generation step. When population size was set to 20 we observed premature convergence especially for larger sequences. We used much smaller population size compared to the general literature since the population in our hybrid GA consists of local optima. The population size between 50-100 was the most optimal.

In general, optimal population size was correlated with the duration of the local search.

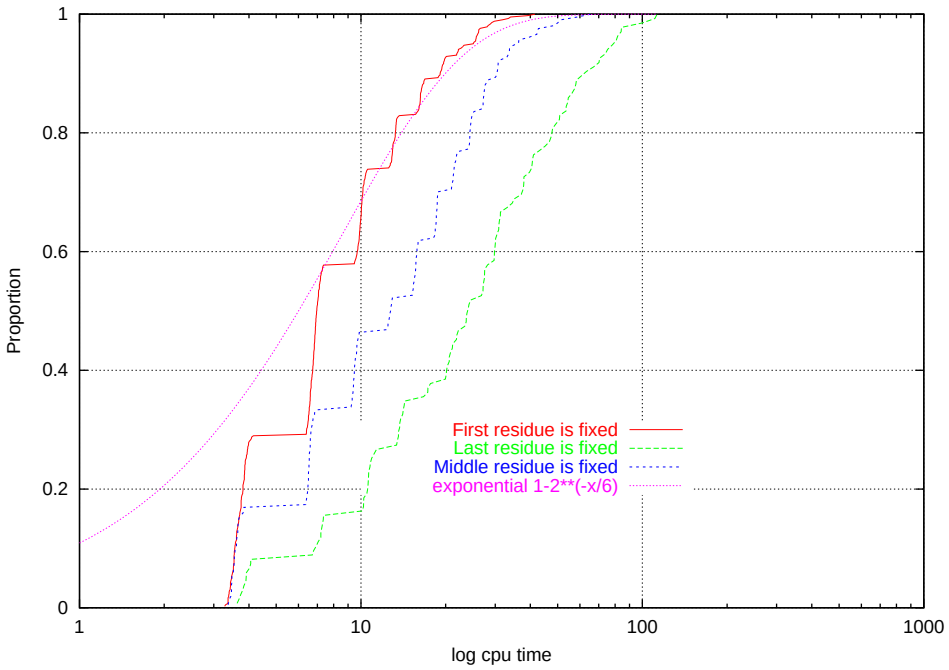

Figure 5: Effect of the fixed point, Sequence 3.

If we have a large population, local search (the time crucial step) takes longer. On the other hand, if the population size is too small, there is not enough diversification to find good solutions quickly.

4.3 Mutation

We wanted to identify the mutation parameter range that is optimal in our implementation of Hybrid GA. It was expected that the optimal mutation parameter would be higher than the generally accepted mutation rate of 0.01 since we are dealing with the population of the local optima and more diversification is possibly needed.

Two kinds of mutation operators were used in our implementation of the GA. The first mutation operator is called macro-mutation. Macro-mutation changes conformation dramatically, it is employed in the local search stage, and it was implemented as described in Krasnogor *et al.* [6] where the direction of the residues between 2 points are changed randomly. The second mutation operator is point mutation, which employs a one-exchange neighborhood, and is used in the mutation stage of the GA algorithm to diversify the search. One point mutation consists on selecting randomly a residue of the sequence and changing randomly its direction.

Each individual in the population was mutated with probability P_m (we tried various combinations, this particular one was easy to implement and worked reasonably well).

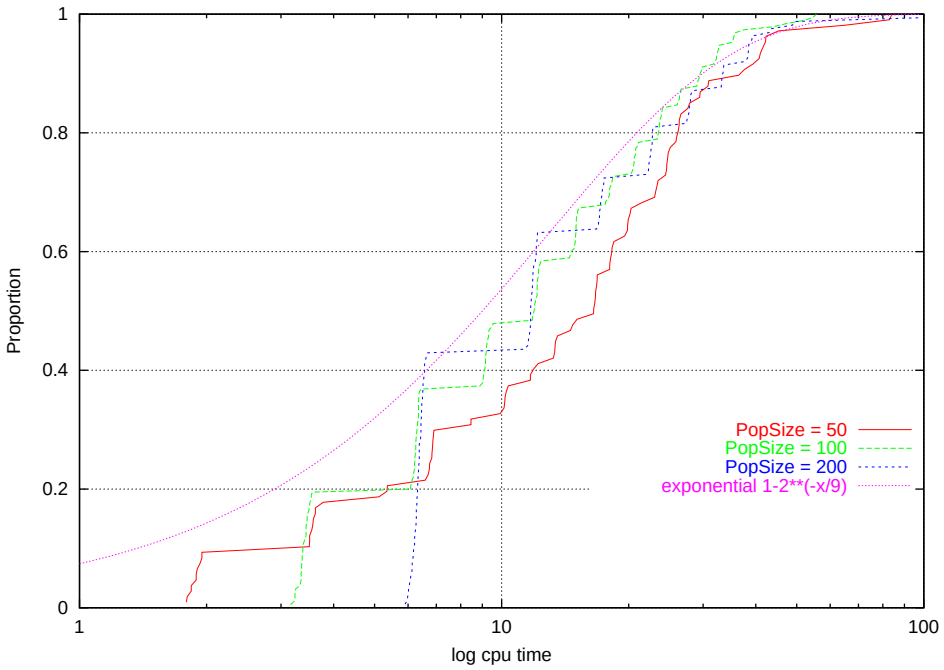

Figure 6: Run time distribution for the Hybrid GA algorithm with various population sizes on Sequence 1.

If the individual is selected for mutation, the number of one point mutations that the algorithm performs on it depends on the fitness f_i of the individual. Specifically, the algorithm performs one point mutation on a sequence while

$$\frac{1-f_i}{f_{opt}} > \text{random number}$$

where f_{opt} is the optimal energy of the given protein. The number of point mutations is bounded by the length of the protein. Note that performing mutation in this way makes individuals that have more H-H interactions (lower Gibbs free energy) have less point mutations than individuals with fewer H-H interactions.

The effect of the mutation rate on the example of Sequence 1 can be seen in Figure 7. A mutation rate of about 0.1 - 0.3 has been shown to be optimal, since the mutation operator performs "jumps" in the corresponding search spaces and most likely drives the solution away from the local optima. It is best to set the mutation parameter to a low value when the local optima are located close to the global optima (this is the case for Sequence 1 shown in Figure 7) and to a high value otherwise, but it is necessary to perform fitness-distance correlation analysis to obtain required information about the distance to the optimum for various sequences. Unger and Moult have shown that there is a single global optima for Sequence 1 by enumeration. For other sequences the search space has not been analyzed yet [15].

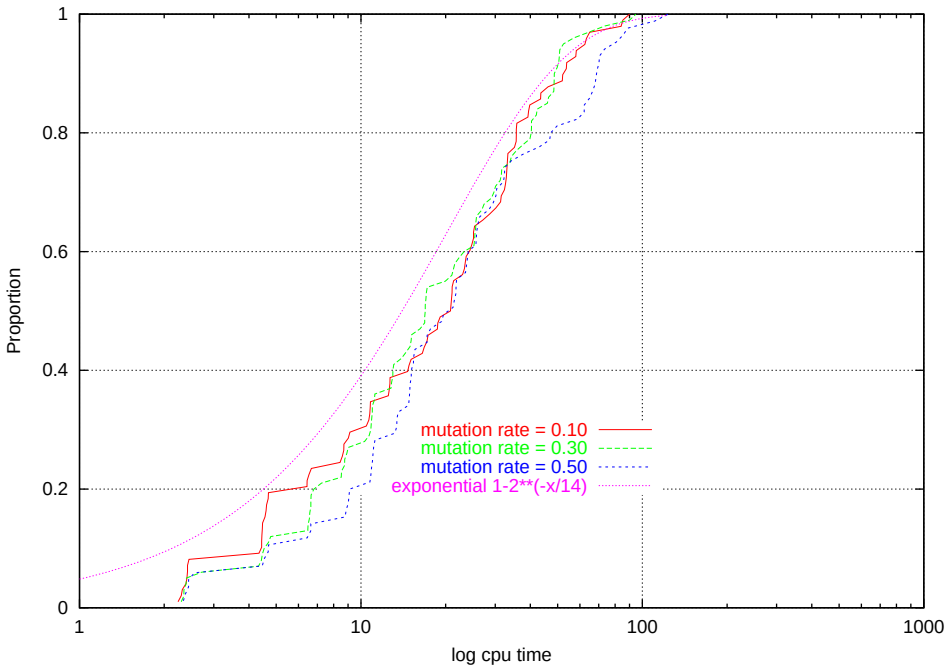

Figure 7: Effect of the probability of mutation P_m on Sequence 1.

4.4 Cross-over

Crossover is a special operator that distinguishes GA from other algorithms. Here, we studied the usefulness of a simple one-point crossover for the Hybrid GA algorithm.

4.4.1 The mechanism of crossover

During a one-point crossover we pick a pair of parents (the choice of parents will be considered next), a random crossover point is chosen, and the bond directions ("genes") are swapped up to the crossover point (if the structure is unfeasible we rejoin at 0, 90 or 270-degree angle as described by Unger and Moult). As a result of the crossover we produce one child as described in a general GA literature [16].

4.4.2 Usefulness of crossover

Crossover should enhance the ability to leave regions with a large number of H-H interactions. A major reason for the maintenance of a population in a GA is the hope of increased performance via direct communication of information between individuals through crossover. Crossover should produce offsprings of similar or even better fitness (objective fitness value); if not it is a severe disadvantage, and the GA algorithm may

not, on average, perform any better than a variety of simpler algorithms that are not population-based. Therefore, the efficiency of a GA depends in a crucial way on the proper selection of crossover rules.

The idea of crossover is to combine building blocks from two individuals into offspring whose fitness exceeds either parent. Even when there is no reason to believe that both parents have an above average chance of contributing above average material to the offspring, crossover may still be useful simply through performing macro-mutation (thus considering the higher order neighborhood) via its mechanics.

In our experiments the usefulness of the crossover was determined by testing the GA algorithm on the same instance with and without crossover. In this way we can determine whether crossover is an important operator for the behaviour of the algorithm or is better to use simpler algorithms that do not work with crossover or with a population of individuals.

The experimental results for Sequence 2 of the runs with random crossover (which will be described in Section 4.4.3) and without crossover are shown in Figure 8.

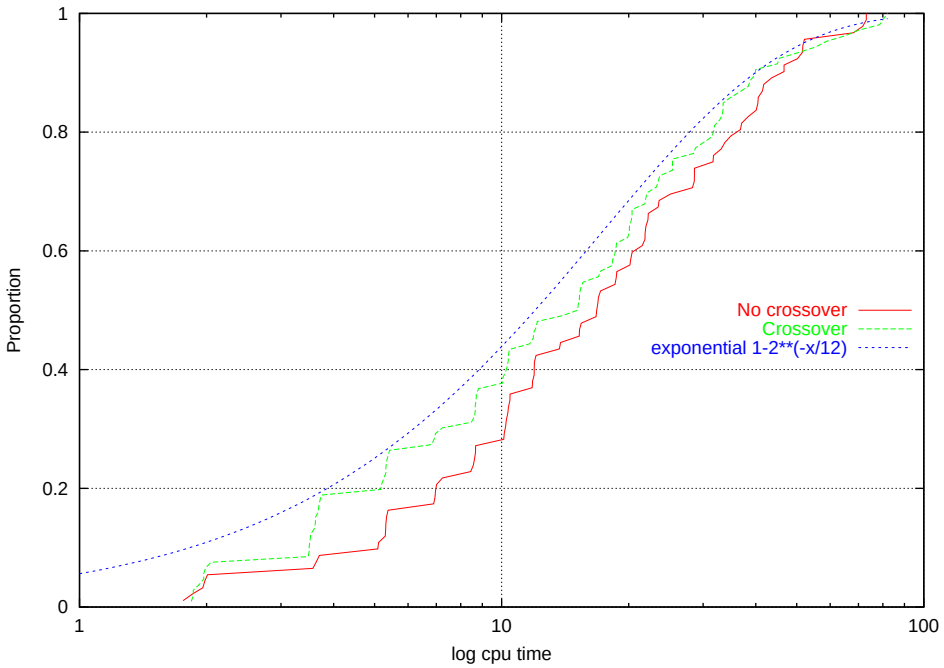

Figure 8: Crossover versus no crossover, Sequence 2.

Crossover appears to be useful (since disabling it results in worse performance), but in fact, its usefulness may be not more than a consequence of the macro-mutation it is performing since the worsening in performance is very small.

4.4.3 Non-random versus random mating strategy

Two strategies for selecting parents were implemented: (1) non-random, and (2) random.

It is known from biology that homologous exchange encourages changes of a very narrow and specific type, genetic material is exchanged in a manner that preserves the function of the all-important transcription segments (genes) [16]. Inspired by the biological metaphor we considered a non-random mating strategy between the parents. Another reason to use non-random mating is that it is very unlikely to produce feasible conformations under the crossover operation. If the 2D structure of the crossover region of two individuals is more similar, the possibility of their mating should increase due to greater chance of obtaining a feasible offspring. The measure of similarity between the two parents used was the Hamming Distance.

During non-random mating we selected the first parent randomly with the probability proportional to its fitness, in the second step we found the second parent such that the Hamming distance among residue orientations is minimal between two individuals in the population. Results are compared with the random mating strategy (that is widely used in literature) when both parents are selected randomly with the probability proportional to their fitness. See Figure 9 for results, Sequence 1.

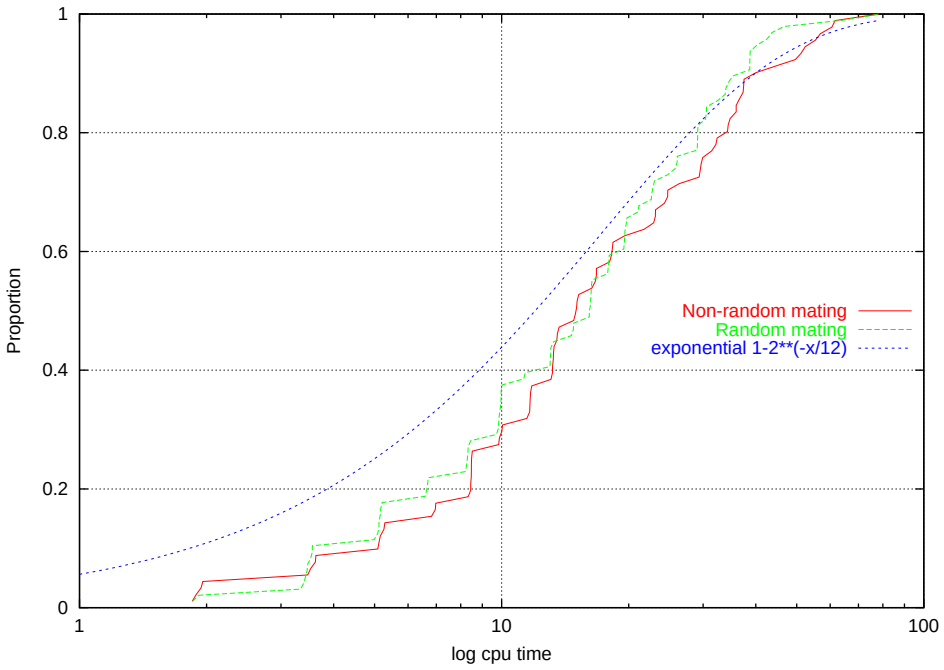

Figure 9: Non-random versus random mating strategy, Sequence 1.

Our results show a slight disadvantage of non-random mating over random mating for all sequences. This could be explained by the fact that two similar parents can produce a

child that is very similar to both of the parents, therefore crossover is no longer really an advantage since it creates individuals similar to their parents.

4.5 Selection process

In this section, we studied if the selection process responsible for the intensification of the search should be more elitist for our Hybrid GA. The selection process was implemented as a roulette selection, one of the widely accepted selection schemes [16]. The probability of an individual to be selected is $P_i = f_i / \sum f$, where f_i is the fitness of the individual and $\sum f$ is the total fitness of the individuals in the population. We use an elitist GA, in order to ensure that at least one copy of the best individual of a generation is guaranteed to survive into the subsequent generation, because otherwise the results are not competitive.

Since elitist strategy seem to work better we considered selection process between children (crossover generates $PopSize$ children) and their parents ($PopSize$) and a selection just between children (crossover generates $2 * PopSize$ children) as described in GA literature [16]. Results for Sequence 1 are given in Figure 10.

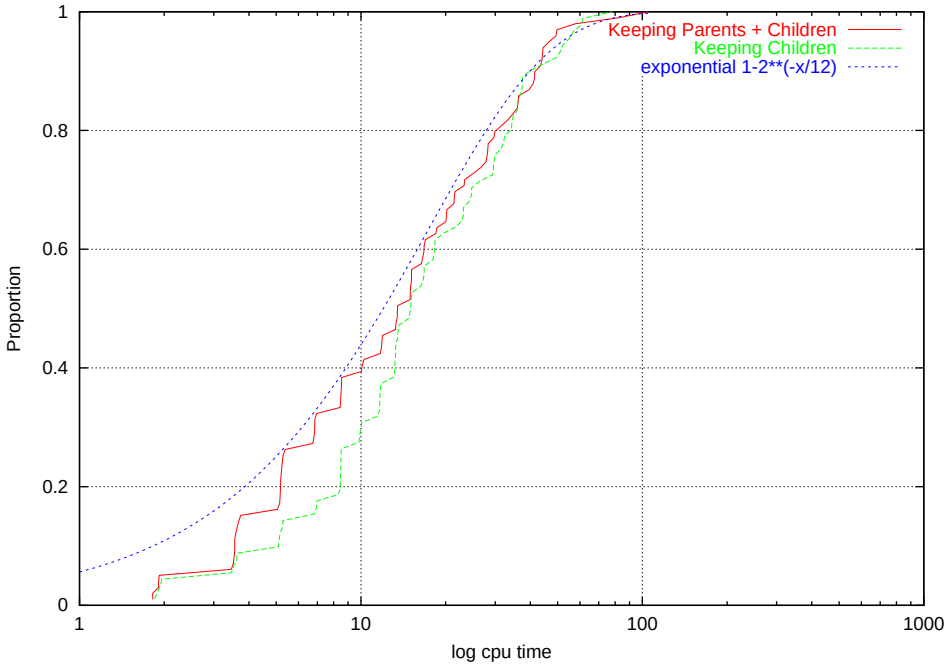

Figure 10: Selection among parents and children versus selection among children only, Sequence 1.

Carrying over parents was shown to be advantageous for all sequences, because it helps to intensify the search. Crossover and mutation, along with the local search that incorporates perturbing moves (macro-mutations) diversify the search and therefore carrying

over parents fulfills the intensification role in our implementation of Hybrid GA.

4.6 Local search - Iterative First Improvement

In order to enhance performance of the GA algorithm we incorporated a local search phase. Local search (LS) algorithms start from a complete initial solution - protein conformation, and try to find the solution with the lowest energy in an appropriately defined neighborhood of the current solution. We considered two types of neighborhoods:

1. macro-mutation neighborhood – two candidate solutions differ by one or more (up to sequence length) directions of the sequential residues.
2. point mutation neighborhood – two candidate solutions differ by a direction of a single residue.

We implemented Iterative First Improvement where the first solution (conformation) found with better fitness (lower energy) than the previous one replaces the current solution. The algorithm searches first in the macro-mutation neighborhood. During macro-mutation two points are chosen randomly, and the direction of the residues between those two points are changed randomly as described by Krasnogor *et al* [6]. The new conformation is accepted if its energy is better than the energy of the previous one. Otherwise, the algorithm continues the search in the previous conformation. Then, point mutation is performed in the following way: the algorithm first generates a permutation $p_1 p_2 \cdots p_n$ of residues, where n is the number of residues in the protein. Then, it performs one point mutation on the residue that is located in position p_1 within the protein. The new conformation is accepted if its energy is better than the energy of the previous one and the search continues until we are done scanning through the permutation. We repeat the search starting from the macro-mutation neighborhood. Note that the search alternates between the macro-mutation and the point mutation neighborhoods. These steps are repeated until no improving neighbor solutions are found for a number of iterations, the last being a parameter ($noImpr$).

The macro-mutation neighborhood is higher order neighborhood, and we observed that it yields better results than a simple point mutation neighborhood. For this reason we implemented local search that uses both neighborhoods.

Results of the local search alone versus Hybrid (GA + Local search) are provided in Table 3. Thus we can see that the GA algorithm is responsible for the smallest but the hardest part of the optimization process, since GA+LS has a significant advantage over LS alone.

In order to set the parameters right we tested the GA algorithm with different threshold values for local search (no improvement parameter $noImpr$, which indicates for how long we will continue a local search while seeing no improving steps). Results for a various number of steps with no improvement for Sequence 2 are given in Figure 11. There is a trade-off between how long one is willing to wait for an improvement and the quality

Instances			*GA + local search*			*local search*		
No	Length	Opt.	sq	n_{opt}/n_{runs}	t_{avg} CPU sec	sq	n_{opt}/n_{runs}	t_{avg} CPU sec
1	20	-9	-9	3473/3473	18.70	-9	100/258	111.43
2	24	-9	-9	741/741	38.45	-9	8/113	162.15
3	25	-8	-8	634/634	34.93	-8	44/129	125.42
4	36	-14	-14	25/151	543.21	-14	5/72	136.10
5	48	-22	-21	3/8	1555.08	-20	1/20	1780.74
6	50	-21	-20	3/81	6220.29	-18	3/18	1855.96
7	60	-34	-33	1/1	900.39	-30	2/20	1623.21
8	64	-42	-33	1/1	982.79	-28	2/9	1441.88
9	20	-10	-10	1295/1295	19.24	-10	5/202	134.57

Table 3: Comparison of the genetic local search (GA + local search) and the local search algorithms where sq is the best solution quality over multiple runs, n_{opt} is the number of runs the algorithm finds sq, n_{runs} is the number of runs, and t_{avg} is the average time required by the algorithm to find sq.

of the solution obtained. The optimal number of the *noImpr* parameter depends on the length of the sequence. For small sequences, its optimal value is 50-100, for larger ones 100-500. The higher the value of *noImpr*, the more efficient the local search is, and the less chance that we will observe stagnation behavior as seen in the case when *noImpr* is set to 30.

4.7 Comparison of the Hybrid GA with the ACO

In this section we conduct comparison of the two algorithms that we have implemented. Since some of the components used in both algorithms are the same, for example, generation of initial population, checking feasibility, and local search, this comparison will provide further indication of what approaches are successful for the protein folding problem.

The population size for all instances of GA was 50, while ACO used 15 ants.

Figures 13 and 14 show RTDs for various sequences as compared with RTDs for the ACO algorithm implemented for the Stochastic Algorithms Course. We can see that the performances of the GA and the ACO algorithms are very similar on small instances, and the cumulative run-time distribution follows the exponential distribution, which means that the chance of getting an optimal-quality solution over k runs of time t is the same as running it once for time $k * t$.

Results for the GA algorithm versus ACO algorithm are given in Table 4. From Table 4 we can see that GA found optima for Sequences 1-4 and 9 (20 - 25 amino acid long), and came really close to the optima on sequences of length 48, 50 and 60 residues. ACO found the optimal solution for all sequences (lengths ranging from 20 residues up to 60)

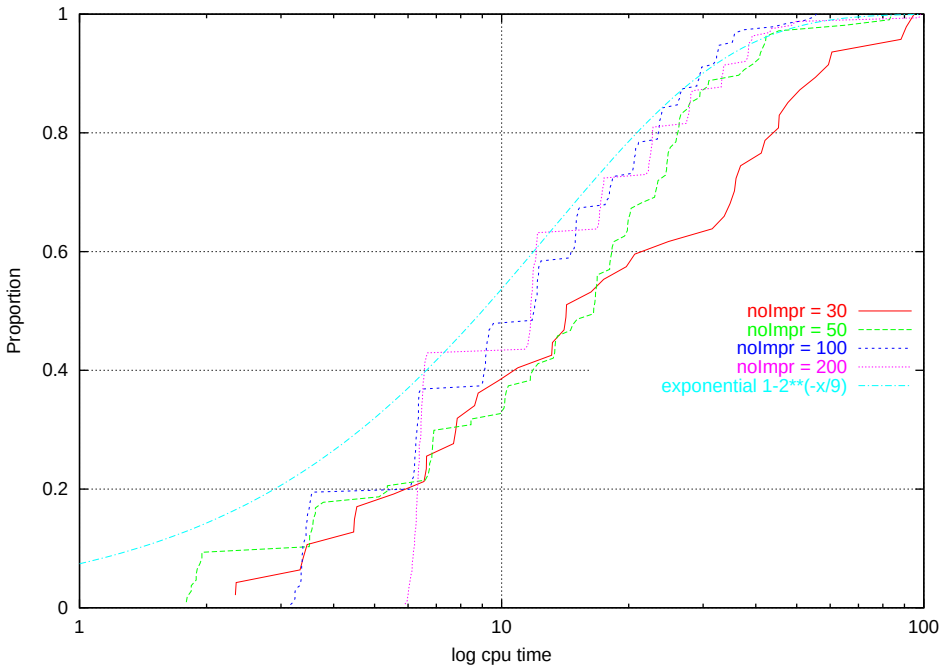

Figure 11: Effect of no improvement parameter, Sequence 2.

but failed to find the optimum for sequence of length 64. The GA algorithm was faster on all small sequences (length 20-36). Basic solution quality distribution (SQD) statistics for Sequences 1 and 3 are given in Tables 5, 6, 7 and 8. The solution quality in these tables is defined as energy of the corresponding conformation. Note that for both sequences the GA algorithm finds a better mean SQD than the ACO algorithm in less number of steps. For Sequence 1, the mean SQD is 7.716 in step 10 for the ACO algorithm and 8.156 for the GA algorithm. For Sequence 3, the mean SQD is 8.116 in step 10 for the ACO algorithm and 8.922 for the GA algorithm. It should also be noted that it takes a very long time for both GA and ACO to find the optimum/suboptimum for sequences of length 36 - 60, and the success rate is not very high (cut off was the number of no improvements for 1000 iterations).

The stagnation behavior was observed for large sequences for both GA and ACO. Poor performance on larger sequences can be due to the fact that better initial solutions, more diversification, and more aggressive local search is required; while performance on the smaller sequences follows the exponential distribution even in the tail section, as shown in Figure 15 and 16.

The largest sequence that the GA algorithm could fold optimally is Sequence 4 of 36 amino acids. On the other hand, the largest sequence that the ACO algorithm fold optimally is Sequence 7 of 60 residues long. The optimal conformation is given in Figure 12. The large hydrophobic core is prominent.

Due to the fact that both algorithms considered are population-based we can also obtain some information about the algorithms by studying the mean solution quality of the whole population of individuals and their standard deviation over time, Figures 17 and 18. Note that the solution quality is defined as energy of the corresponding conformation. The GA algorithm mean solution quality tends to slightly improve over the run and the standard deviation tends to slightly decrease (the optimal solution quality was obtained before stagnation occurred). ACO mean solution quality is not as stable as GA mean solution quality, it tends to oscillate, and standard deviation tends to stay the same or even increase, which indicates a need for intensification.

A general conclusion from the RTD-based analysis of the GA and the ACO algorithms is that the GA takes an initial time to find good solutions for small to medium sequences, but after a certain time it tends to outperform the ACO algorithm or give similar solutions, while the ACO algorithm tends to quickly find optimal solutions at the beginning, but is rather slow later. It is also interesting to note that the GA algorithm gets trapped in the local optima for large sequences and therefore doesn't reach the optimum, while the ACO algorithm is able to escape but the success rate is still rather low.

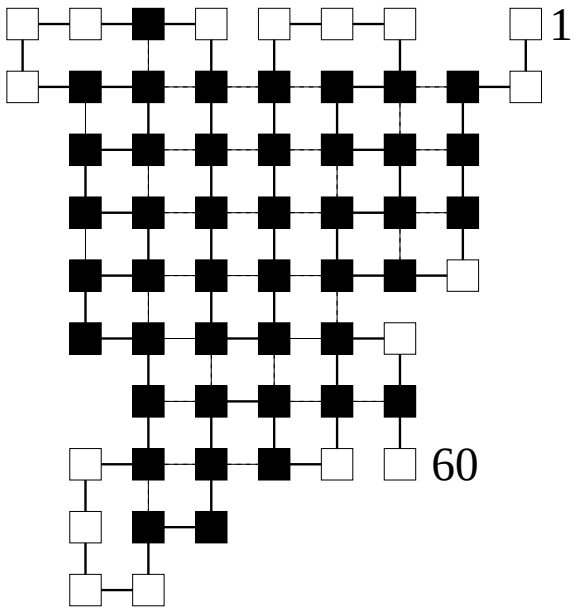

Figure 12: Optimal conformation for Sequence 7 (energy = -34).

Instances			*GA*			*ACO*		
No	Length	Opt.	sq	n_{opt}/n_{runs}	$t_{avg}, CPUsec$	sq	n_{opt}/n_{runs}	$t_{avg}, CPUsec$
1	20	-9	-9	3473/3473	18.70	-9	711/711	23.90
2	20	-10	-10	1295/1295	19.23	-10	596/596	26.44
3	25	-8	-8	634/634	34.92	-8	120/120	35.32
4	36	-14	-14	25/151	543.20	-14	21/128	4746.12
5	48	-22	-21	3/8	1555.07	-22	1/151	1920.93
6	50	-21	-20	3/81	6220.28	-21	18/43	3000.28
7	60	-34	-33	1/1	2900.39	-34	1/119	4898.77
8	64	-42	-33	1/1	4982.79	-32	1/22	4736.98
9	24	-9	-9	741/741	38.44	-9	247/247	43.48

Table 4: Comparison of the Hybrid GA and the ACO.

step number	mean	stddev	stddev/mean	min	max	median	Q_{25}	Q_{75}	Q_{10}	Q_{90}
10	7.716	0.483	0.063	6	9	8.000	7	8	7	8
20	7.869	0.377	0.048	7	9	8.000	8	8	7	8
30	7.942	0.289	0.036	7	9	8.000	8	8	8	8
40	7.986	0.220	0.028	7	9	8.000	8	8	8	8
50	8.000	0.183	0.023	7	9	8.000	8	8	8	8
60	8.021	0.202	0.025	7	9	8.000	8	8	8	8
70	8.038	0.192	0.024	8	9	8.000	8	8	8	8
80	8.000	0.000	0.000	8	8	8.000	8	8	8	8
90	8.024	0.154	0.019	8	9	8.000	8	8	8	8
100	8.000	0.000	0.000	8	8	8.000	8	8	8	8

Table 5: Basic SQD statistics for different run-lengths, Sequence 1, ACO

step number	mean	stddev	stddev/mean	min	max	median	Q_{25}	Q_{75}	Q_{10}	Q_{90}
1	7.883	0.447	0.057	7	9	8.000	8	8	7	8
2	8.041	0.348	0.043	7	9	8.000	8	8	8	8
3	8.080	0.344	0.043	7	9	8.000	8	8	8	8
4	8.074	0.262	0.032	8	9	8.000	8	8	8	8
5	8.093	0.291	0.036	8	9	8.000	8	8	8	8
6	8.060	0.237	0.029	8	9	8.000	8	8	8	8
7	8.100	0.300	0.037	8	9	8.000	8	8	8	8
8	8.161	0.367	0.045	8	9	8.000	8	8	8	9
9	8.062	0.242	0.030	8	9	8.000	8	8	8	8
10	8.156	0.362	0.044	8	9	8.000	8	8	8	9

Table 6: Basic SQD statistics for different run-lengths, Sequence 1, Hybrid GA

step number	mean	stddev	stddev/mean	min	max	median	Q_{25}	Q_{75}	Q_{10}	Q_{90}
10	8.116	0.549	0.068	7	10	8.000	8	8	8	9
20	8.365	0.506	0.060	8	10	8.000	8	9	8	9
30	8.456	0.521	0.062	8	10	8.000	8	9	8	9
40	8.497	0.521	0.061	8	10	8.000	8	9	8	9
50	8.568	0.513	0.060	8	10	9.000	8	9	8	9
60	8.603	0.510	0.059	8	10	9.000	8	9	8	9
70	8.615	0.502	0.058	8	10	9.000	8	9	8	9
80	8.646	0.504	0.058	8	10	9.000	8	9	8	9
90	8.623	0.494	0.057	8	10	9.000	8	9	8	9
100	8.650	0.521	0.060	8	10	9.000	8	9	8	9

Table 7: Basic SQD statistics for different run-lengths, Sequence 3, ACO

step number	mean	stddev	stddev/mean	min	max	median	Q_{25}	Q_{75}	Q_{10}	Q_{90}
1	8.138	0.684	0.084	7	10	8.000	8	8	7	9
2	8.467	0.648	0.077	7	10	8.000	8	9	8	9
3	8.585	0.609	0.071	8	10	8.000	8	9	8	9
4	8.714	0.598	0.069	8	10	9.000	8	9	8	9
5	8.702	0.573	0.066	8	10	9.000	8	9	8	9
6	8.842	0.608	0.069	8	10	9.000	8	9	8	10
7	8.843	0.601	0.068	8	10	9.000	8	9	8	10
8	8.814	0.536	0.061	8	10	9.000	8	9	8	9
9	8.891	0.528	0.059	8	10	9.000	9	9	8	9
10	8.922	0.436	0.049	8	10	9.000	9	9	8	9

Table 8: Basic SQD statistics for different run-lengths, Sequence 3, GA

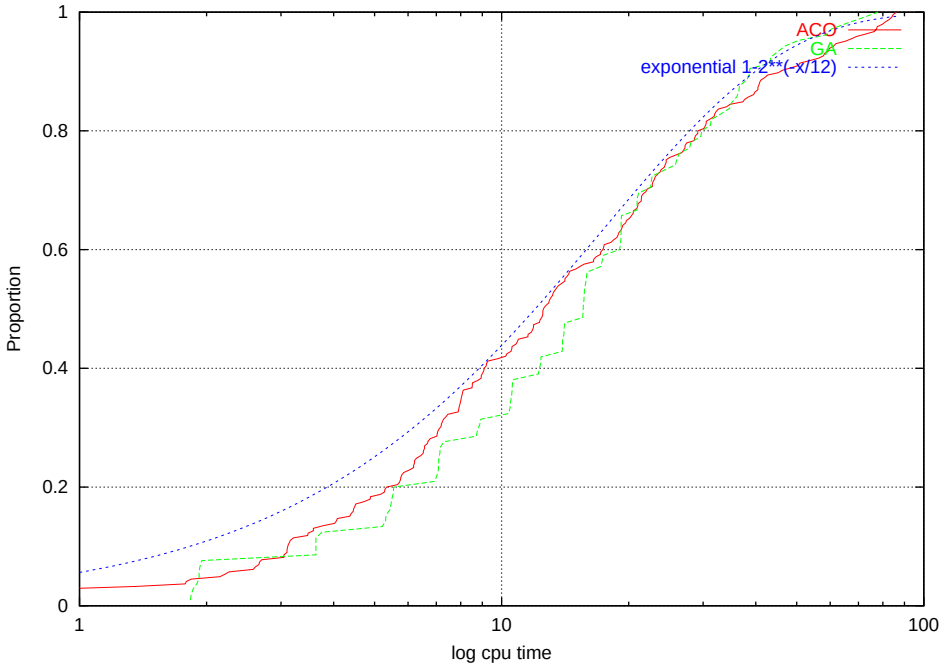

Figure 13: Marginal run time distribution for Sequence 1, Hybrid GA and ACO.

5 Conclusions

In this paper we developed new Hybrid GA algorithm and experimentally studied its behavior on the 2D HP protein folding problem. We then collected some information about how our GA behaves on various instances of the problem based on various strategies and parameter settings.

The Hybrid GA algorithm described in this work is a genetic local search algorithm that performs better on small to medium sequences than the simple GA because it improves the intensification of the search. Another advantage of the genetic local search is that it is possible to work with a small population because the population consists of local optima. The size of the population is an important factor in our implementation because local search is the time crucial step and appropriately chosen population size reduces time needed to perform local search on all of the individuals of the population while keeping enough diversity.

The local search considered was Iterative First Improvement with macro-mutation and point mutation neighbourhoods. We found that an Iterative First Improvement that includes also the macro-mutation neighborhood performs better than the simple Iterative First Improvement that only uses the point mutation neighborhood.

According to our experimental results, we found that it is important to implement a

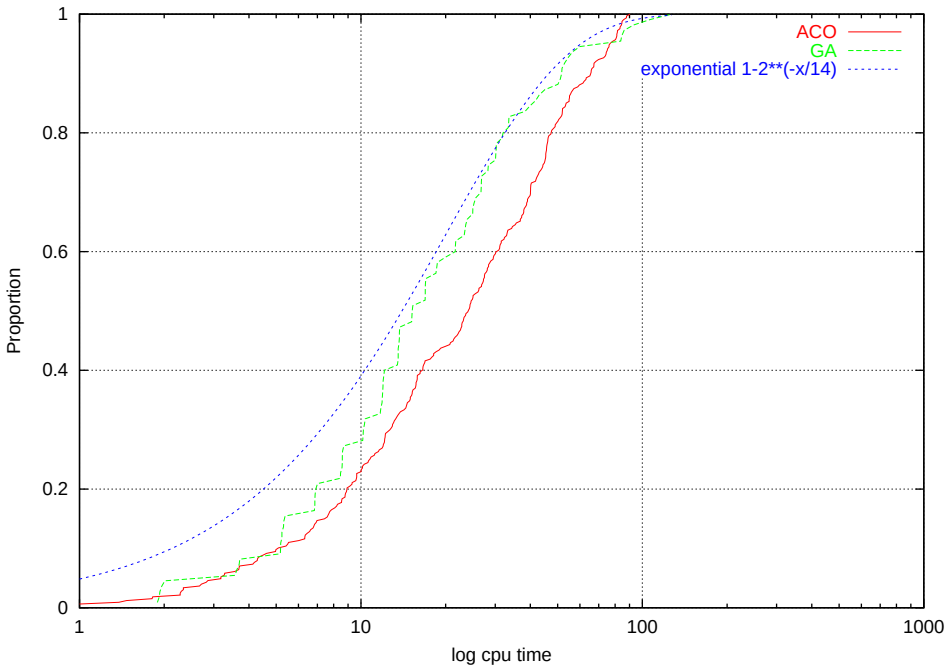

Figure 14: Marginal run time distribution for Sequence 3, Hybrid GA and ACO.

mechanism to generate random feasible conformations since for long proteins there is a problem of entanglement. The backtrack procedure that reconstructs infeasible conformations showed good performance. We also considered different starting folding points for protein conformations. According to the results, the optimal folding point that leads to a better performance of the algorithm depends on the given protein.

We observed that the mutation operator plays an important role in the performance of our Hybrid GA. A probability that a conformation is mutated, P_m, about 0.1-0.3 was optimal. This value is higher than the typical value of 0.01 used in standard GA algorithms, since we are working with the population of the local optima. We also observed that one point mutation that is usually implemented in standard GA algorithms is not enough in terms of the diversification of the search. In our algorithm we perform more than one point mutation on a given conformation. The number of mutations on a given conformation depends on its energy: conformations with low H-H interactions are mutated more often.

We compared a non-random crossover (parents with low Hamming distance) with the traditional random crossover and observed that the last one performs better for the Hybrid GA. We also tested the Hybrid GA algorithm without crossover and observed that the crossover operator is useful in our implementation of the GA.

The selection process was implemented as a roulette selection. We studied the effect of having a selection process between children and parents or just between children as a standard GA algorithm. The selection operator that considers parents was shown to be

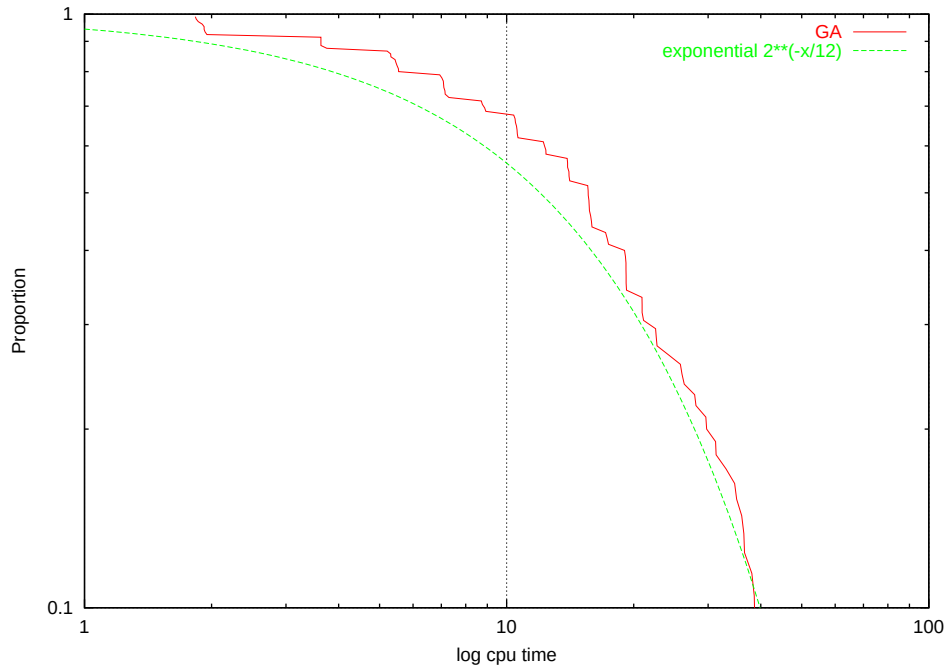

Figure 15: Right tail of the marginal run time distribution for Sequence 1, ACO.

advantageous for all sequences because it helps to intensify the search.

Our GA found the global optima of Sequences 1-4 and 9 (20-25 amino acids long) listed in Table 1. The ACO, on the other hand, found the optimal solution for all sequences except the largest one of length 64. We observed that the GA algorithm was faster on small sequences (length 20 - 36). For sequences larger than 25, both algorithms take long time to find optimum or suboptimum conformations and the success rate (number of runs that find the optimum or suboptimum conformation) is low.

The GA algorithms described by Unger and Moult and by Krasnogor *et al.* outperforms our Hybrid GA. Unger and Moult did not find the optimum conformation for Sequence 8 and Krasnogor *et al.* for Sequences 7 and 8. However, it is difficult to compare our GA algorithm with their algorithm because they did not include an extensive evaluation.

During our study we found that the protein folding problem is hard combinatorial problem even in 2D for sequences larger than 36 residues. By watching and studying the performance of our Hybrid GA on various instances, we found that the search space of the 2D HP protein folding problem is very likely to include a large number of local optima in which algorithms can be easily trapped.

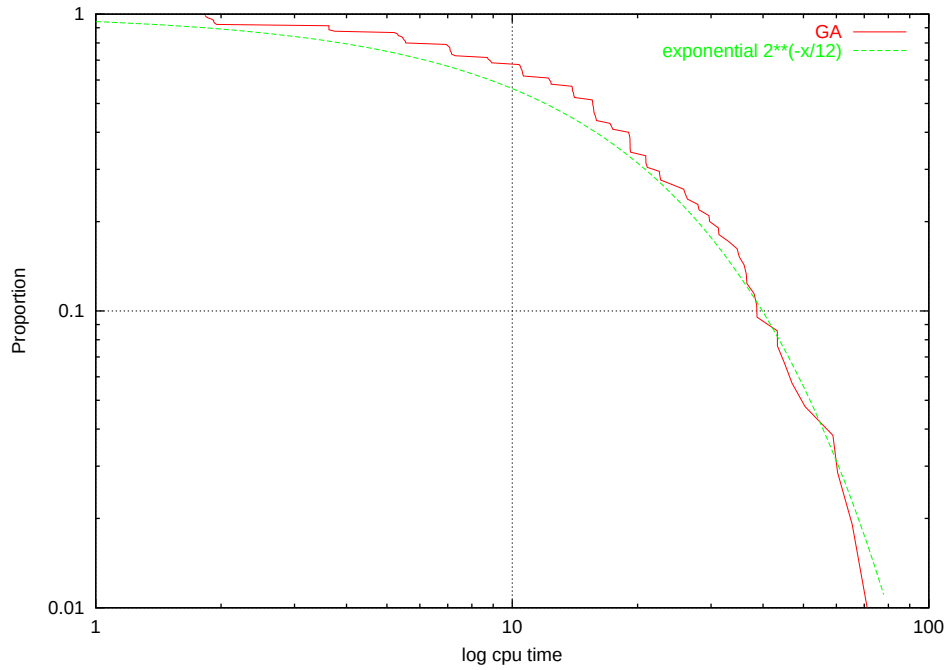

Figure 16: Right tail of the marginal run time distribution for Sequence 1,Hybrid GA.

6 Future Work

There are several issues for future research. Search space analysis of the 2D HP folding problem, using fitness distance correlation analysis, is one of the possible directions for the research. The distance between two sub-optimal solution can be defined in terms of the Hamming distance (the number of directions of residues that vary between two solutions). The study of the correlation of how close the sub-optimal solutions are to each other, and the global optimum, and how close the energy values are to the optimal can provide necessary insight for the future successful development of the algorithms for this problem.

The Hybrid GA should also benefit from additional diversification, such as other kinds of macro-mutations. It would be interesting to limit the rank of the neighborhood of the macro-mutation and consider this value as an additional parameter. Some of the intensification strategies can be considered, for example use more than one elitist individual in the population.

In general, our results indicate that the performance of both new algorithms: ACO and Hybrid GA can be significantly improved.

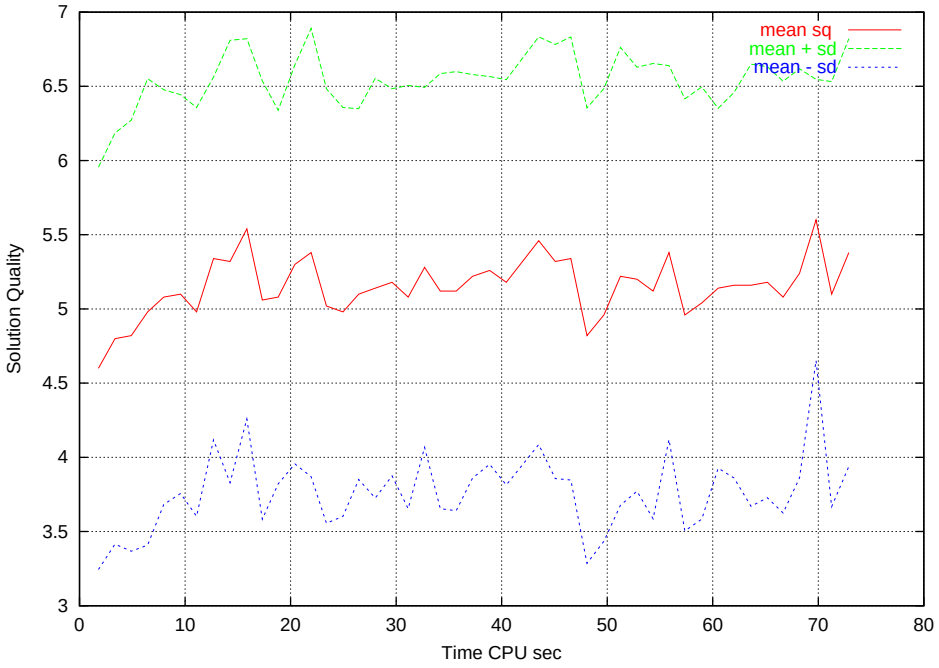

Figure 17: Mean solution quality over time, Sequence 1, Hybrid GA

7 Contributions

Rosalía: project proposal 50 percent, coding 10 percent, correcting project proposal 100 percent, project update 40 percent, testing 20 percent, writing final report 50 percent.

Alena: project proposal 50 percent, coding 90 percent, correcting proposal 0 percent , project update 60 percent, testing 80 percent, writing final report 50 percent.

References

[1] Creighton, T. E., *Proteins. Structure and Molecular Principles.* Freeman, New York 1984.

[2] Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Pub. Co., 1989.

[3] Hart W., Istrail S., "HP Benchmarks", `http://www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html`

[4] Hoos H., Stützle T., *On the empirical evaluation of Las Vegas algorithms.*Proceeding of the IJCAI'99 Workshop on Empirical AI,1999.

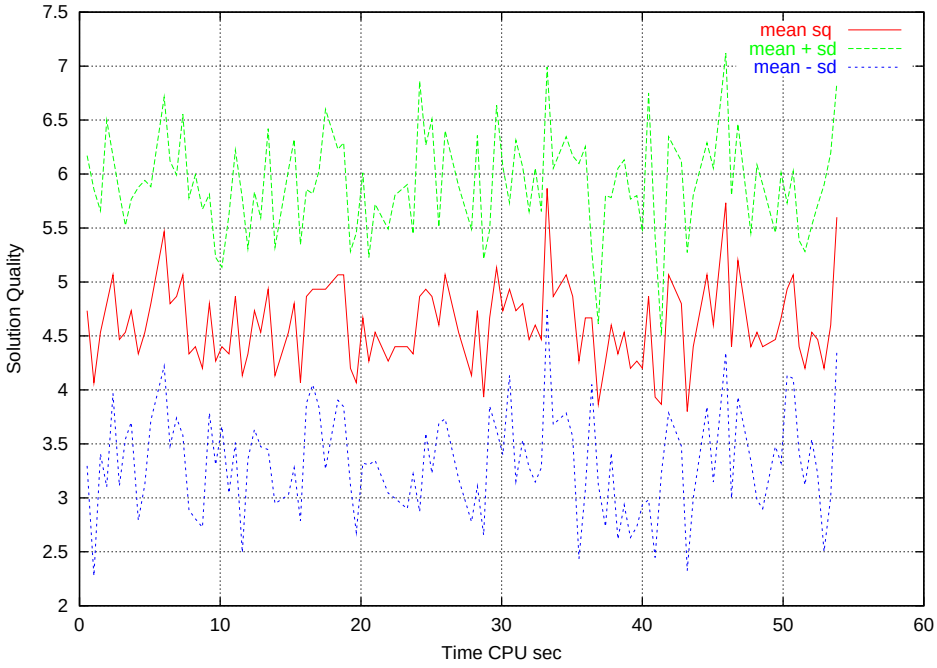

Figure 18: Mean solution quality over time, Sequence 1, ACO.

[5] Hoos H., *Stochastic local search-methods, models, applications.* PhD thesis, TU Darmstadt, 1998.

[6] Krasnogor N., Hart W. E., Smith J., Pelta D. A. *Protein structure prediction with evolutionary algorithms.* Proceedings of the Genetic & Evolutionary Computation Conference, 1999.

[7] Krasnogor, N., Pelta D. A., Lopez P. M., Mocciola P., and E. de la Canal. *Genetic algorithms for the protein folding problem: a critical view.* In C. F. E. Alpaydin, editor, Proceedings of Engineering of Intelligent Systems. ICSC Academic Press, 1998.

[8] Lau K.F., Dill K.A. *A lattice statistical mechanics model of the conformation and sequence space of proteins.* Macromolecules 22:3986 - 3997, 1989.

[9] Patton A., W. P. III, and E. Goldman. *A standard GA approach to native protein conformation prediction.* In Proc 6^{th} Intl. Conf. Genetic Algorithms, pages 574-581. Morgan Kauffman, 1995.

[10] Richards, F. M. *Areas, volumes, packing, and protein structures.* Annu. Rev. Biophys. Bioeng. 6:151-176, 1977.

[11] Rose, G. D. *Hierarchic organization of domains in globular proteins.* J. Mol. Biol. 134:447-470, 1979.

[12] Sali, A., E. Shakhnovich and M. Karplus, *How Does a Protein Fold*, Nature, 369, 248-251, May 1994.

[13] Unger, R., and J. Moult. *Genetic algorithms for protein folding simulations.* Journal of Molecular Biology, 231 (1): 75-81, 1993.

[14] Shmygelska A., Aguirre-Hernández R. *Ant Colony Optimization for Protein Folding, 2D HP model.* CPSC 532D project, to appear.

[15] Unger, R., and J. Moult. *A genetic algorithm for three dimensional protein folding simulations.*In Proc 5^{th} Intl Conf on Genetic Algorithms, pages 581-588. Morgan Kaufmann, 1993.

[16] Wolfgang B., P. Nordin, R. Keller, F. Francone, *Genetic Programming: An Introduction.* Morgan Kaufmann Publishers, Inc. San Francisco, California, 1998.