Genetic Algorithm for Predicting Protein Folding in the 2D HP Model

A Parameter Tuning Case Study

Eyal Halm
Leiden Institute of
Advanced Computer Science,
University of Leiden
Niels Bohrweg 1
2333 CA Leiden,
The Netherlands
ehalm@liacs.nl
(Received 21 December 2007)

ABSTRACT

Given the amino acid sequence of a protein, predicting its tertiary structure is known as the protein folding problem. This problem has been widely studied under the HP model in which each amino acid is classified, based on its hydrophobicity, as an H (hydrophobic or non-polar) or a P (hydrophilic or polar). Conformation of a protein in the HP model is embedded as a self-avoiding walk in either a two-dimensional or three-dimensional lattice. The protein folding problem in the HP model is to find a conformation (a folded sequence) with the lowest energy. In [5], a genetic algorithm is introduced in order to find the lowest energy conformation for some benchmark HP sequences of amino acids. Many parameters in the initial setup of the algorithm can influence the performance of the algorithm. In this paper we will play with the parameters in order to investigate the influence on the success rate of this algorithm.

Keywords

Genetic Algorithm, 2D HP Model, Protein Folding, Parameter Tuning

1. INTRODUCTION

The protein folding problem is known to be NP-Complete in both two-dimensional and three-dimensional square lattices (see [2]). In [5] a genetic algorithm combined with Simulated Annealing [1] is introduced in order to solve the protein folding problem using the 2D HP Model. We now take this algorithm, and let it run multiple times with different parameters in the initial setting. In this case study, we use the benchmark strings given in [5], to be seen here in Table 2. From this table, we can see what is the minimum energy for each of the given strings. We then set the initial parameters differently and look for the results. As a result we will check which of the parameters have the most influence on the percentage of success in finding the minimum energy configuration, how many evaluations and how many generations were needed in order to find this minimum.

2. THE 2D HP MODEL

The 2D HP Model was introduced by [3].
This model is quite simple and captures the essence of the important components of protein folding. Every protein is represented by a linear sequence of "amino-acids" of only two types: H (hydrophobic or Non-polar - we will designate it the color black) or P (hydrophilic or Polar - we will designate it the color white). This sequence is folded on a two-dimensional square lattice on which at each point the chain can turn 90° left, 90° right, or continue ahead. When looking at the folded conformation of the protein, each hydrophobic (black) amino-acid that is found in front of another hydrophobic amino-acid but not bounded with it, contributes -1 to the total energy of this conformation. In Figure 1, the 20 amino-acid molecule B-W-B-W-W-B-B-W-B-W-W-B-W-B-B-W-W-B-W-B is to be seen in a conformation that has energy -4. In Figure 2, the same sequence is folded in a different way, and it has energy -9 (which is also the minimum energy for this 20 amino-acids long protein). During the process of folding the protein on this square-lattice, only one amino acid can be found in each location of the lattice - The folded protein can not "collide" with itself.

3. THE ALGORITHM

The algorithm introduced in [5] combines two search techniques: Monte Carlo and a Genetic Algorithm. The Genetic Algorithm works with a population of conformations, in which 2 parents are being chosen, and one new offspring is introduced, using a crossover operator. This process is repeated every time with 2 different parents to fill the new population with new offsprings. The Monte Carlo step takes care of mutating the parents just before they are being evaluated in order to be chosen by the genetic algorithm. In the coming two subsections, these two steps will be explained more thoroughly.

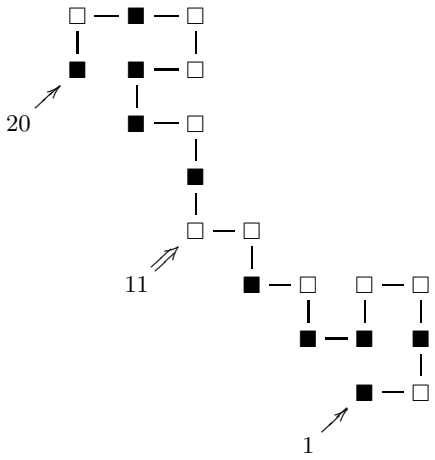

Figure 1: (Adapted from [5]) The 20 amino-acids long sequence B-W-B-W-W-B-B-W-B-W-W-B-W-B-B-W-W-B-W-B found here in a conformation yielding an energy of -4 according to the 2D HP model. 4 pairs of hydrophobic residues (marked black) in the conformation are found in front of each other without being connected with each other. Each such pair contributes -1 to the total energy of the conformation

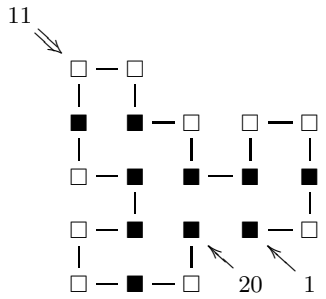

Figure 2: The same sequence found here in a conformation yielding an energy of -9. Here 9 pairs of hydrophobic residues are to be found, each contributing a -1 to the total energy of the conformation. There are in total 83,779,155 ways to fold this 20 amino-acids long sequence. Only 4 conformations yield an energy of -9, which is the lowest energy conformation that can be found for this 20 amino-acids long sequence. Note that this conformation with energy -9 can be obtained by changing the direction the conformation takes after residue 11 from 90° right turn (in Figure 1) to a 90° left turn (in this figure). That is a mutation that can take place in the Monte Carlo step explained later

3.1 Monte Carlo step

A monte carlo step takes one conformation of amino-acids S_1. First, the energy E_1 of this conformation is being evaluated. It then chooses randomly one of the amino-acids in the conformarion and randomly changes there the direction of the chain (90° left, 90° right, or continue ahead). The result, S_2, is a different conformation of this sequence of amino-acids. S_2 is being checked for self-avoidance - the conformation may not collide with itself (two amino-acids on the same space in the square lattice). If S_2 collides with itself, a new monte carlo step is being performed on S_1 until a conformation is found which is not colliding with itself. Once such S_2 is found, the energy E_2 of this conformation is being evaluated. The new (mutated) conformation S_2 is being accepted in two cases:
1) If its energy $E_2 \leq E_1$ (The energy of the not mutated conformation S_1) or
2) If its energy $E_2 > E_1$ but a random number Rnd between 0 and 1 is satisfying the equation $Rnd < exp[\frac{E_1 - E_2}{c_{kmc}}]$.
c_{kmc} is gradually decreased (cooled) during the simulation to achieve convergence (In the beginning, more destructive mutations will be accepted).
If the new S_2 did not comply with these 2 cases, a new monte carlo step will occur on S_1.

3.2 Genetic Algorithm step

The Genetic Algorithm step takes N different conformations of the same sequence of amino-acids. By using a roulette wheel mechanism, it then chooses 2 parents and recombines them into one offspring. The selection of the two parents is made in such a way that the probability $p(S_i)$ of a structure being selected is proportional to its energy E_i:

$$p(S_i) = \frac{E_i}{\sum_{j=1}^{N} E_j}$$

So the lower the energy, the higher the probability that it will be chosen. The crossover of these two parents takes place by choosing a random location in the conformation. The turns the offspring conformation takes are being copied from the first parent until this random location is reached. The rest of the directions are being copied from the second parent. In order to connect these two parts, a random direction is being chosen for the gluing (90° left, 90° right, or continue ahead). The offspring is being checked for self-avoidance (not colliding with itself). If it does collide with itself, another random direction is being chosen. If all 3 random gluing directions lead to a colliding structure, a new pair of parents is being chosen. Once we found a self-avoiding offspring structure, its energy E_k is being evaluated and compared to the average energy of its parents ($\overline{E}_{ij} = (E_i + E_j)/2$). The structure S_k will be accepted if $E_k \leq \overline{E}_{ij}$. If $E_k > \overline{E}_{ij}$, the structure is still accepted if

$$Rnd < exp[\frac{\overline{E}_{ij} - E_k}{c_{kga}}]$$

The crossover operation is repeated on $N-1$ pairs of parents until $N-1$ new offsprings are created. The parent with the lowest energy is copied to the next generation as offspring number N. Figure 3 shows a genetic algorithm step in action. Parent A is a conformation with energy of -5. Parent B is another conformation of the same sequence but this time with energy -2. The crossover between parent A

and parent B on the marked residue, with a specific random gluing direction yields the offspring C, which has energy -9. Figure 4 shows a single run of the genetic algorithm. The distribution of the different energies among the population is to be seen, along the progress of the algorithm while looking for the optimum solution.

4. EXPERIMENTAL RESULTS

In [5] the genetic algorithm is tested with 300 generations of 200 individuals. The cooling scheme of the Monte Carlo step started with $c_{kmc} = 2$ and was cooled by $c_{kmc} = 0.97 \times c_{kmc}$ every 5 generations. The Genetic Algorithm cooling scheme started with $c_{kga} = 0.3$ and was cooled by $c_{kga} = 0.99 \times c_{kga}$ every 5 generations. For each sequence the simulation was run 5 times. They experimented with different sequences with lengths from 20 to 64. Using exactly this same algorithm, I implemented it using C language, and ran some tests with different parameters to check the influence on the performance of the algorithm. The following parameters were changed and tested:

1. Length of the sequence (given benchmark sequences as seen in Table 2)
2. Population size of each generation in the genetic algorithm (N)
3. Maximum number of generations the genetic algorithm will execute
4. Number of mutations each parent will go through before being chosen
5. Number of generations between cooling down the Monte Carlo and Genetic Algorithm cooling factors
6. Start value for MC cooling (c_{kmc})
7. Start value for GA cooling (c_{kga})

Every test outputs the minimum energy found in each generation with the corresponding number of evaluations that were needed to create the new generation. If the test reaches the known minimum value for this sequence, the test stops and the best conformation is being printed.
Every parameter configuration was run 50 times. The tests were run using a computer on the faculty which is deliberately limited with its computation power (multi user), so at some point, with sequences of length 48, I reached this limit and the program terminated abnormally. This is the reason I have no tests with sequence length larger than 36. It usually takes a couple of seconds to find the minimum energy conformation for a sequence with length 20. For the longer sequences, it can sometimes take a couple of minutes. Ofcourse the longer the algorithm runs, the more difficult it will be to find mutations that are not causing collisions with itself. The algorithm tries again and again until such a mutation is found. In general the first generations will need less time to be calculated. A summary of all results of the tests is being shown in Table 1. In each line we see what were the 9 parameters the algorithm was run with, and the following results:

1. Average number of evaluations that were needed to reach the minimum energy conformation (from 50 tests)
2. The minimum number of evaluations that were needed to reach the minimum energy conformation (out of 50 tests)
3. Average number of generations that were needed to reach the minimum energy conformation (from 50 tests)
4. The minimum number of generations that were needed to reach the minimum energy conformation (out of 50 tests)
5. Percent of success - how many of the 50 tests reached the minimum energy conformation within the given number of generations?

5. DISCUSSION

From Table 1 we try to conclude the influence of the parameters on the efficiency of the algorithm. We will now discuss each of the changed parameters:

5.1 Sequence length

It is easy to see that the shorter the sequence is, the higher the percentage of success with finding the minimum energy conformation. This is of course due to the fact that a longer sequence introduces a much larger search space, simply because there are more positions where we can change the direction the conformation takes.
Comparing test1 with test3, we see they both have exactly the same parameters, but test3 runs on a 24 amino-acids sequence, while test1 on a 20 amino-acids sequence. This means we have 4 extra positions we have to find the turn direction for. This makes the search space larger, which drops the success rate from 90% to 58%.
Comparing test1 with test6 shows that if we search a solution for a 25 long sequence, the success rate drops even further, to 16%.

5.2 Population size

The population size dictates how many individuals are found in each generation of the genetic algorithm. If we have more individuals in each generation, the algorithm is more likely to diverge into the ultimate solution within the given number of generations. More individuals means a higher probability of finding the minimum energy among the population.
Comparing test22 with test9 shows that indeed, when we double the number of individuals per generation (from 200 to 400), the success rate grows from 48% to 80%. But the cost we have to pay is of course more evaluations during the run of the algorithm. It is interesting to see that the average number of generations that are needed to find the optimum is lower when the population size is larger. The diversity of the population increases the probability of finding the optimum within the current population.

5.3 Number of generations

In the first generation all individuals have energy 0, since none of them is folded at any point along the sequence. A couple of generations later, when the conformations are folded in some points along the sequence, we see that the conformations have different energies (distributed from energy 0 to the lowest energy found until now). It is a question

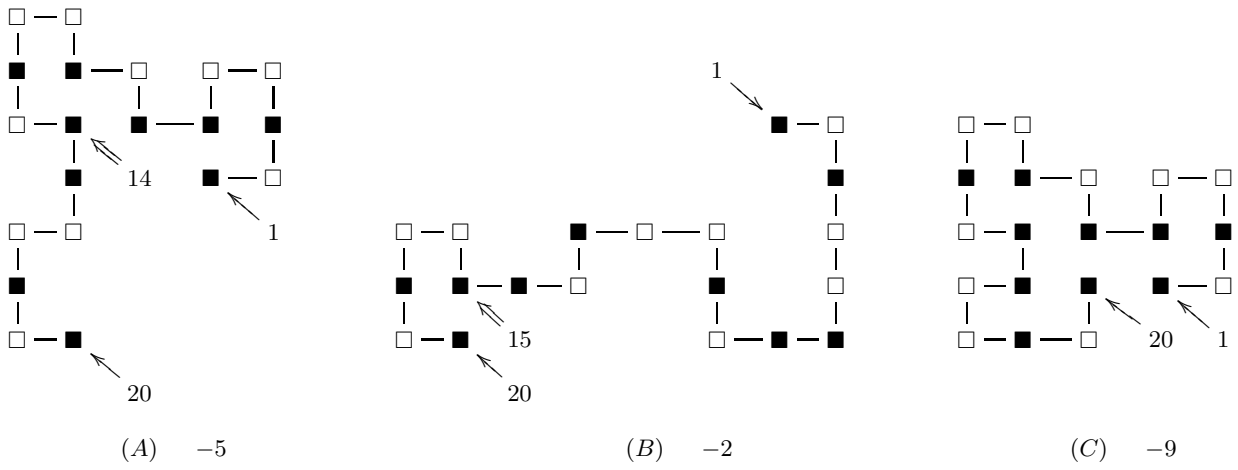

Figure 3: (Adapted from [5]) A Genetic Algorithm step: (A) and (B) are two parents. Parent (A) has energy -5, and parent (B) has energy -2. The randomly chosen residue for the crossover is residue number 14. The first 14 turns are copied from parent (A). The last 5 turns are being copied from parent (B). In between the two parts a random turn is inserted, in this case a 90° right turn. The new born offspring (C) does not collide with itself and the energy can be evaluated. It gets the energy -9, which is better than the average of the two parents (-3.5). The new offspring is accepted. This is also the minimum energy that can be found for this sequence, so the algorithm terminates

Table 1: Results of 50 tests with each parameter setup

Test No.	Length	Pop	Gens	Mutat	Cool Every	Start MC	Start GA	Factor MC	Factor GA	Evals Avg	Min	Gens Avg	Min	Percent Success
2	20	200	300	2	5	2.0	0.3	0.97	0.99	241585	25015	79	11	100
1	20	200	300	1	5	2.0	0.3	0.97	0.99	100012	19380	50	14	90
5	24	200	500	2	5	2.0	0.3	0.97	0.99	525262	52875	116	16	94
4	24	200	300	2	5	2.0	0.3	0.97	0.99	520332	29023	116	10	82
3	24	200	300	1	5	2.0	0.3	0.97	0.99	307475	45839	103	21	58
22	25	400	300	2	10	2.0	0.3	0.97	0.99	965562	161744	112	23	80
9	25	200	300	2	10	2.0	0.3	0.97	0.99	619833	59978	142	17	48
13	25	200	300	2	5	3.0	0.6	0.97	0.99	618601	91232	148	27	46
14	25	200	300	2	10	3.0	0.3	0.97	0.99	626821	68991	147	20	46
15	25	200	300	2	10	3.0	0.6	0.97	0.99	667566	58540	161	18	38
10	25	200	300	2	10	2.0	0.6	0.97	0.99	646931	156553	154	41	34
8	25	200	300	2	5	2.0	0.6	0.97	0.99	659362	155414	155	40	34
12	25	200	300	1	5	3.0	0.6	0.97	0.99	245028	28646	92	16	32
7	25	200	300	1	5	2.0	0.6	0.97	0.99	133645	24865	53	14	30
11	25	200	300	1	5	3.0	0.3	0.97	0.99	184485	19908	65	12	30
6	25	200	300	1	5	2.0	0.3	0.97	0.99	218026	18615	76	12	16
21	25	200	300	3	10	2.0	0.3	0.97	0.99	889281	309355	151	55	6
17	36	200	300	1	5	2.0	0.6	0.97	0.99	2031608	2031608	181	181	3

Table 2: 4 sequences that were used as benchmark tests in [5]. The Optimal Energy is taken from the runs made by Ron Unger and John Moult. When this energy is reached, the algorithm terminates

Length	Optimal Energy	Sequence
20	-9	BWBWWBBWBWWBWBBWWBWB
24	-9	BBWWBWWBWWBWWBWWBWWBWWBB
25	-8	WWBWWBBWWWWBBWWWWBBWWWWBB
36	-14	WWWBBWWBBWWWWWBBBBBBBWWBBWWWWBBWWBWW

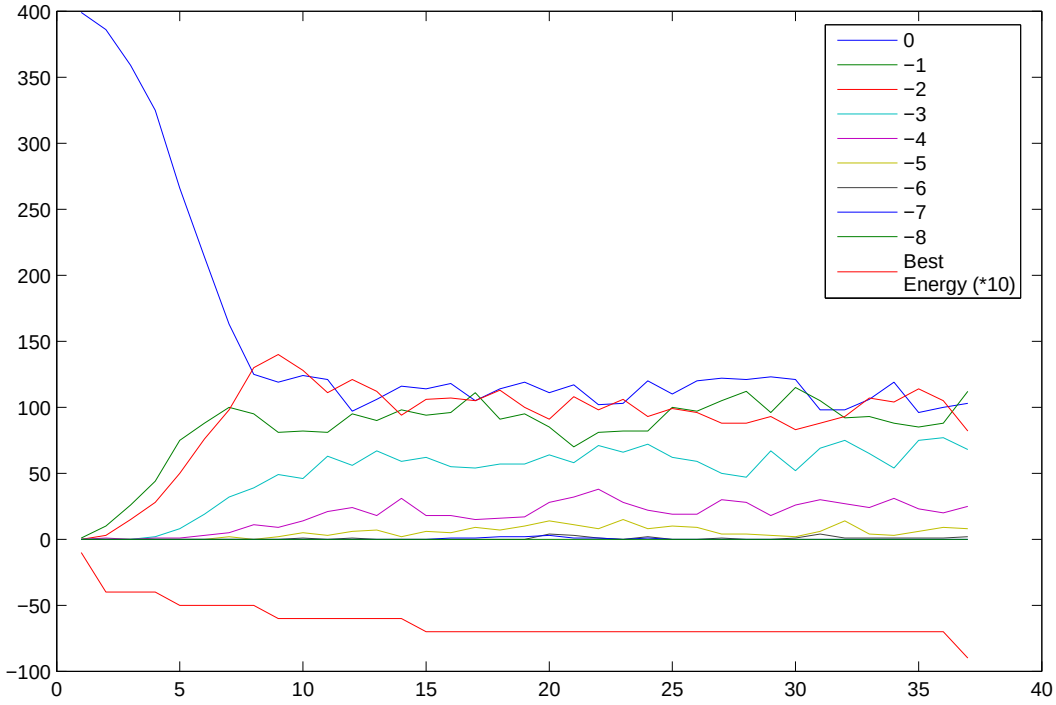

Figure 4: A run of the genetic algorithm. This is one of the outputs of test22. It runs 300 generations with a population of 400 individuals. We see that in the first generation, all 400 conformations have energy 0 (they are all not folded). Generation 5 has only 214 conformations with energy 0, 88 conformations with energy -1, 76 with energy -2, 19 with energy -3, and 3 with energy -4. One of the mutated -4 conformations yielded a conformation of -5 (See the best energy below the zero axis). The 400 conformations keep changing their energies until at some point, somewhere during the 36th generation, a conformation with energy -9 is found. At that point the algorithm terminates and reports its findings

of luck that, at some point in time, we choose 2 parents, a location for a crossover and a good gluing direction that gives us the ultimate solution - the conformation with the minimum energy. At that point the algorithm stops. But having said it is a question of luck, if we try some more generations, maybe we will get lucky... So adding some generations will increase the probability that we get lucky with finding the correct combination.
However, the cooling mechanism of both the Monte Carlo step and the Genetic Algorithm step, makes sure that the further we progress through the generations, the less we accept conformations that are worse than the parents. That means that as time goes by, we will be doing more exploitation (looking deeper - in the current region) than exploration (looking wider - in other regions). Due to that fact, we are most likely to be stuck in a local minimum, which has small chances of bringing us to the global minimum we are actually looking for.
So increasing the number of generations with the same factor of increasing the number of individuals per generation, will probably have a smaller effect on the efficiency of the algorithm in finding the global minimum.
Comparing test4 with test5 shows that increasing the number of generations from 300 to 500, improved the success rate from 82% to 94%. At the same time, doubling the number of individuals (see test9 compared to test22) increased the success rate from 48% to 80%. It is quite interesting to see that the average number of evaluations and average number of generations are quite the same for test4 and test5. However, the minimum number of evaluations and minimum number of generations are lower for test4. This proves that we deal here with a very random algorithm which can get lucky with finding the optimum solution within the huge search space.

5.4 Number of mutations per parent

In [5] in section 4 (page 78 bottom left) it is claimed that for the 20-residue long molecule, they performed a simulation with a population of 200 structures with 20 steps of individual mutations per structure between crossover stages. This does not prove to be so useful. One can see that 2 mutations instead of 1 indeed gives better success rates (compare test2 with test1 - 100% success and 90% success respectively). But, when we apply 3 mutations (in test21), it goes awfully wrong. Comparing with test9, which had 48% success, a test with 3 mutations had only 6% success! This tendency was even worse when applying more mutations. Mutating so many times goes awfully wrong because we have a high probability of mutating in the "wrong direction" - too many times we take a structure with a good energy, and by changing so many turning positions we actually spoil the structure. We then either see a structure with a higher energy (will not be selected) or it will introduce a self-colliding structure. In both cases the progress is hindered.
It is wise to use 2 mutations instead of 1, but using 3 mutations or more leads to very low success rates. I assume this is a misunderstanding in the interpretation of [5] .

5.5 Cooling every # generations

In test10 the cooling constant is cooled every 10 generations of the genetic algorithm. In test8, this is done every 5 generations. The cooling in test10 is thus slower (meaning that it takes more time before we start exploiting instead of exploring). It could have had some influence on the results but as one can see it has minor or no influence at all.

5.6 Cooling start value for the Monte Carlo step

Test11 starts with a Monte Carlo cooling factor of 3.0. Test6 begins with a cooling factor of 2.0. Test11 has in the beginning a lower probability of accepting a mutation on a conformation which yields a conformation with a worse energy evaluation. That means test11 is a bit more greedy and tries to go faster in the direction of a better solution. Indeed we see that test11 performed almost twice as well (30% success) as test6 (only 16% success).

5.7 Cooling start value for the Genetic Algorithm step

Test7 starts with a Genetic Algorithm cooling factor of 0.6. Test6 begins with a cooling factor of 0.3. Test7 has in the beginning a lower probability of accepting an offspring which yields a conformation with a worse energy evaluation than the average of its parents. That means test7 is a bit more greedy and tries to go faster in the direction of a better solution. Indeed we see that test7 performed almost twice as well (30% success) as test6 (only 16% success). However, comparing test14 with test15 shows that it can work also the other way around. There we see that test14, which starts with a lower cooling factor for the Genetic algorithm scores slightly better than test15. This shows the complexity of this problem. Since we are speaking about a genetic algorithm that uses recombination of two parents into one offspring, it can be sometimes the case that a parent (or even two) with a very low score (quite high energy) can be recombined into an extremely good offspring (see Figure 3 - where energies -2 and -5 yield an offspring with energy -9). In this case, exploration can be used as a 'backtracking' mechanism - we make sure we still have some 'bad' individuals in the population and we still have a probability of yielding very good offsprings.

6. CONCLUSIONS

We are dealing here with a very complex problem. If we were searching in the 3D space, it would have been even more complex (there we have 5 turn options at each point on the sequence). A genetic algorithm can indeed search this space in a better way than a totally random algorithm. Longer sequences introduce of course larger search spaces, it takes a longer time to find the optimum and the success rate is doomed to decrease. Fine tuning the genetic algorithm can have major influence on the performance and the success rate. Working with large population sizes improves the success rate more than running the algorithm for more generations. Interesting is the fact that 2 mutations on each parent before creating a new offspring improves the performance tremendously but going any higher than that number, results in a terrible deterioration of the success rate. In 2005, a paper was published [4] which introduces an even more efficient way of using genetic algorithms to solve the protein folding problem. The solutions the Genetic Algorithm finds are locally optimized by using another Genetic Algorithm that looks at the secondary structure of the protein.
It might be a good idea to try and fine tune the genetic algorithm presnted in [5] automatically by using another evolutionary algorithm.

7. REFERENCES

[1] E. Aarts and J. Korst, *Simulated annealing and boltzmann machines*, John Wiley and Sons, New York (1989).

[2] B. Berger and T. Leighton, *Protein folding in the hydrophobic-hydrophilic (hp) model is np-complete*, International Conference on Computational Molecular Biology, New York, NY, USA, ACM Press (March 1998), 30–39.

[3] K.F. Lau and K.A. Dill, *Theory of protein mutability and biogenesis*, Proc. Nat. Acad. Scie. (1990), no. 87, 638–642.

[4] G. Sundarraj and T.N. Bui, *An efficient genetic algorithm for predicting protein tertiary structures in the 2d hp model*, GECCO '05, ACM Press (June 25-29, 2005), 385–392.

[5] R. Unger and J. Moult, *Genetic algorithms for protein folding simulations*, J. Mol. Biol. (1993), no. 231, 75–81.