Detecting Secondary Peptide Structures by Scaling a Genetic Algorithm

S. Michaud*, J. Zydallis*, G. Lamont* and R. Pachter**

* Dept of Electrical and Computer Engineering
Air Force Institute of Technology, Wright-Patterson AFB, OH 45433, USA
Steven.Michaud, Jesse.Zydallis, Gary.Lamont@afit.af.mil
** Materials and Manufacturing Directorate, Air Force Research Laboratory,
Wright-Patterson AFB, OH 45433, USA
Ruth.Pachter@wpafb.af.mil

ABSTRACT

The ability to accurately predict a polypeptide's molecular structure given its amino acid sequence is important to numerous scientific, medical, and engineering applications. Studies have been conducted in the application of Genetic Algorithms (GAs) to this problem with initial results shown to be promising. In this paper we use the fast messy Genetic Algorithm (fmGA) to attempt to find the minimization of an empirical CHARMM energy model and generation of the associated conformation. Previous work has shown that the fmGA provided favorable results, at least when applied to the pentapeptide [Met]-Enkephelin. We extend these results to a much larger Polyalanine peptide by utilizing secondary structure information. This information is utilized to conduct localized searches on the energy landscape. Results indicate that on average this localized search always produces a better final solution.

Keywords: Alanine Peptide, Energy Minimization, Protein Structure Prediction Problem, Genetic Algorithm, Fast Messy Genetic Algorithm.

1 INTRODUCTION

Many scientists and engineers are working hard to solve the Protein Structure Prediction Problem (PSP) [1]. The problem of developing a generalized technique for predicting a polypeptide's molecular structure given its amino acid sequence is commonly referred to as the PSP problem. This problem requires the minimization of an energy function in conformational space [2]. In this paper we present a stochastic, population based search approach to solving the protein structure prediction problem, the fast messy genetic algorithm (fmGA).

The fmGA was developed in 1991 by Goldberg, Deb and Kargupta [3] and later applied to the PSP problem by Merkle, Gates, Lamont and Pachter [4]. The fmGA is a Genetic Algorithm (GA) that takes advantage of building blocks to solve optimization problems. Previous work has shown the fmGA to be an effective and efficient method in solving the PSP problem using the CHARMM energy model and the pentapeptide [Met]-Enkephelin [5], [6]. This fact has led us to believe that favorable results will be obtained when applying the fmGA to larger proteins. This paper focuses on solving a much larger Polyalanine peptide model. The Polyalanine peptide we used consists of 14 residues and 56 independent variables as compared to the 5 residues and 24 independent variables of [Met]-Enkephelin [5]. Each of the dihedral angles is represented by a binary string of 10 bits yielding a landscape size of 56^{1024} compared with the 24^{1024} of [Met]-Enkephelin. In order to effectively scale the fmGA search algorithm to handle larger proteins and continue to obtain "good" solutions, additional domain information is used in the form of secondary structure information.

In Section 2 we present background information on the fast messy genetic algorithm as applied to the PSP problem. Sections 3 and 4 present the secondary structure analysis, testing and results. Section 5 finishes with our conclusions and comments on future work.

2 DESCRIPTION OF THE FAST MESSY GENETIC SEARCH ALGORITHM

The fmGA is an approach that explicitly exploits "good" Building Blocks (BBs) in solving optimization problems. These BBs represent "good" information in the form of partial strings that can be utilized to obtain even better solutions. The BB approach is used in the fmGA to increase the number of "good" building blocks that are present in each subsequent generation of the algorithm. The fmGA algorithm executes in three phases, the *Initialization Phase*, the *Building Block Filtering Phase*, and the *Juxtapostitional Phase* [4]. Each of these phases is now described in more detail.

The algorithm begins with the Probabilistically Complete Initialization Phase. This phase randomly generates a number of population members. These population members are of a specified length and are evaluated to determine each member's corresponding fitness value. Our implementation utilizes a binary scheme in which each bit is represented with either a 1 or a 0 and the CHARMM energy model is used to calculate each string's fitness value.

The Building Block Filtering (BBF) Phase follows and randomly deletes allele values in each of the population member's strings until the strings reach a predeter-

mined BB size. This process is alternated with a selection mechanism to keep only the strings with the "best" building blocks found. An input schedule is used to specify the number of generations to execute each phase and the generations upon which BBF and selection will occur. Through the BBF phase the string lengths decrease but must continue to be evaluated for selection. These strings are referred to as "underspecified" since each locus does not have an associated allele value. In order to evaluate "underspecified" strings, a competitive template is used [3]. This template contains the allele values of the best found string in the population from the previous block size phase. To evaluate an underspecified population member, the member is overlayed upon the template to fully specify the member. This process essentially takes the missing allele values from the template and places them into the population member to allow the fitness evaluation to take place. This overlay function process is repeated any time an underspecified population member needs to be evaluated.

The juxtapositional phase then takes the building blocks found through BBF and uses recombination operators to create strings that are fully specified. The recombination operation may result in overspecified strings which are strings containing multiple allele values for the same locus position. In this case a left to right method is used. This method takes the first allele value encountered for any locus and uses that value even if subsequent values for the same locus appear later in the string. Recombination is alternated with a binary thresholding tournament selection operator to keep the best solutions found. Upon completing of the juxtapositional phase, the best population member becomes the new competitive template. The algorithm restarts with a new BB size and repeats the three phases. If there are no more BB sizes to execute, the best population member is presented as the solution.

3 SECONDARY STRUCTURE ANALYSIS

The ability to predict a protein's three-dimensional structure or conformation from its one-dimensional amino-acid sequence is a significant problem. The prediction of the secondary structure is an important step towards predicting the three-dimension structure of the protein [7]. This secondary structure may consist of helices, beta-strands, turns, etc. The prediction of the secondary structure has been used as a precurser to finding a "good" tertiary structure [7], [8]. We present a modification to our fmGA algorithm that incorporates localized secondary structure searches.

We are dealing with the Polyalanine peptide which consists of only an α-helix secondary structure. Since the focus of this paper is to show the effectiveness of the fmGA on larger peptides, our modifications to the fmGA are restricted to an α-helix secondary structure but can be expanded to any secondary structure in the future. Upon completion of the three phases of the fmGA, each population member's backbone dihedral angles are analyzed. This analysis records the total number of each specific dihedral angle that falls within the user specified dihedral angular constraints. This constraint is a +/- percentage of the known secondary structure's angle values. For example, if the angular constraint is +/-10% and the ω dihedral angle of the secondary structure has a value of 100°, any ω angle between 90° and 110° is added to the total number of angles meeting the constraint. If the algorithm is successful in partially predicting the secondary structure of the protein, a localized search on the competitive template is conducted to try and find the "optimal" conformation. The competitive template is now modified since it contains the current best solution found and directly affects subsequent population members and the final solution obtained.

The algorithm's success at predicting the secondary structure is measured by determining if a user specified percentage of the total number of residue angles in the population is met. This is done by analyzing the total number of angles that are within a percentage of the values in the known secondary structure. In our preceding example, if the percentage constraint is 10% and 15 of a total of 100 population members, 15%, had an angle between 90° and 110°, then we state that the algorithm was successful at predicting the respective secondary structure angle and subsequently a localized search is conducted based on that determination. On the other hand if at least 10% of the angles did not meet the angular constraint, a localized search operation is not conducted, since the tested secondary structure does not appear to exist in this protein. In all tests conducted the percentage constraint and angular constraint were set to the same value.

The Secondary Structure Analysis (SSA) supports the localized search operation. A left to right sweep of the competitive template is completed. This Sweep Operator (SO) compares the value of each angle in the template to the corresponding angle of the known secondary structure. If the value is different, the template's angle is changed to reflect the secondary structure. The template is then evaluated to determine if this change has resulted in an improved fitness value. If the fitness value is improved, the change is kept and the next angle is compared to that of the secondary structure, otherwise the original angle and its corresponding fitness are kept. The SO continues to sweep through each angle of the template until a sweep of each angle in the template does not result in an improved fitness value. Essentially this means that the algorithm repeats the sweeping operation until no better solution is found. The given secondary structure angles can be modified by the user via

Computational Nanoscience 2001, (www.cr.org), ISBN 0-9708275-3-9.

an input parameter. This allows the user to focus the localized search on specific angular combinations that are expected to yield good results, while not restricting the algorithm to only produce results similar to the inputs.

Following the SO, a second localized search operation is conducted. The Backbone Residue Sweep Operator (BRSO) analyzes n-1 of the n groups of backbone residue angles ϕ, ψ, and ω. The analysis focuses on only n-1 of the angular groups due to the data structure being utilized and the reduced versatility of the algorithm that would occur if all n angular groups were analyzed. The BRSO compares the template's angles of each group to those of the secondary structure under examination. If at least one of the three angles in the group has a percentage rating greater than the percentage indicated by the input parameter, then the remaining two angles of that group are modified to reflect the accepted secondary structure. The resulting string is evaluated and as before the change is only kept if the resulting string has an improved fitness value. The BRSO operator continues to sweep through the groups of angles of the template until a sweep of each group in the template does not result in an improved fitness value.

The resulting changes to the template from this SSA are only kept if they produce an improved fitness values in the template. The algorithm then continues to execute until all BB sizes are completed. Only the backbone angles were looked at in this study due to the effect that optimizing those angles has on the results of the algorithm. The authors are aware that optimization of the side chain positions are important, but this was not the focus of the paper. The main motivation for the SSA was to utilize the building block information that is obtained from each execution of the algorithm for different BB sizes. In the original algorithm, once the algorithm completes, all of the BB information is thrown away and the next BB iteration is executed. The modifications that we have presented here allow us to utilize this BB information in terms of predicting secondary structure. In the next section we present the results obtained from the addition of this SSA.

4 TESTING AND RESULTS

All of the tests were conducted in parallel on our Cluster of Workstations. A total of seven Intel Pentium III machines consisting of five 933 MHz machines and two 1 GHz machines were chosen from the available pool of machines. These machines are interconnected via a 100 Mbps fast Ethernet switch. The operating system that the fmGA code was executed on was Red Hat Linux 6.2. The code was written in ANSI C with Message Passing Interface (MPI) constructs to execute in parallel. This algorithm was previously parallelized to allow for interoperability on different parallel platforms [2], [6]. In particular, the parallelization centered on all three phases of the fmGA. Synchronous MPI communication calls were utilized to conduct communications between machines.

Table 1: Constraint Analysis

fmGA with SSA (model Polyalinine)					
SSA %	Max Fitness	Median Fitness	Best Fitness	Average Fitness	Stand Dev
0%	-100.160	-125.449	-136.433	-123.573	10.881
5%	-110.490	-130.933	-133.811	-125.449	9.635
10%	-101.837	-128.188	-138.137	-123.588	13.425
15%	-110.286	-130.105	-140.560	-129.495	8.943
20%	-104.369	-134.745	-143.786	-131.767	11.054
25%	-120.941	-132.295	-139.384	-131.469	6.233
30%	-116.572	-136.028	-139.445	-132.374	8.725
35%	-107.275	-135.722	-145.900	-133.244	10.406
40%	-106.880	-131.993	-137.386	-128.940	9.262
45%	- 98.273	-133.537	-145.450	-131.819	13.072
50%	-104.501	-132.501	-143.801	-130.846	10.594
60%	-123.386	-137.333	-145.439	-136.655	6.564
70%	-122.441	-133.848	-141.089	-133.201	5.813
80%	-109.138	-136.016	-145.695	-133.499	10.880
90%	-125.925	-137.140	-145.923	-136.323	6.777
100%	- 93.407	-127.440	-131.671	-122.959	12.089

The fmGA algorithm was executed 10 times for all experiments in order to provide statistical results. All of the results presented in tables 1 and 2 are thus averaged over 10 runs. The population size is the total population over all machines. Over all runs and population sizes the following fmGA parameters were kept constant; cut probability = 0.02, splice probability = 1.00, primordial generations = 200, juxtapositional generations = 200, total generations = 400. An input schedule was used to specify at what generations BBF would occur, and the sizes of the building blocks the algorithm would utilize. Tests were conducted using both the [Met]-Enkephelin and model Polyalanine peptides. The results are presented here as a comparison.

The results from the first set of experiments is presented in Table 1. These experiments were run to determine what effects modifying the input constraint parameter would have on the overall effectiveness of the algorithm on the peptide model Polyalanine. Therefore, the algorithm was executed 10 times for different percentage values on a single processor with a constant population size of 30. In each of these runs, both the dihedral angle constraint and the percentage constraint described earlier were kept the same as well as the fmGA parameters previously specified. The results from Table 1 indicate that the use of the SSA produced the best results, in terms of the best result found, with the SSA percentage set between 15-90%. What can be concluded from this is that the secondary structure analysis is integral to finding improved solutions. Additionally the low and high ends did not produce very good results since both extremes require the population to have the exact angles of the secondary structure or a large percentage of those angles present in the population. There

Table 2: Protein Energy Fitness Values

fmGA without SSA ([Met]-Enkephelin)					
Pop Size	Max Fitness	Median Fitness	Best Fitness	Average Fitness	Stand Dev
100	-22.189	-26.133	-29.598	-25.976	2.045
200	-22.721	-26.114	-28.075	-26.167	1.606
400	-26.608	-27.582	-30.315	-27.865	1.240
800	-23.979	-27.061	-30.141	-26.899	1.991
fmGA with SSA ([Met]-Enkephelin)					
Pop Size	Max Fitness	Median Fitness	Best Fitness	Average Fitness	Stand Dev
100	-23.860	-25.181	-29.615	-25.675	1.579
200	-23.356	-26.355	-29.389	-26.347	1.739
400	-25.349	-27.122	-30.054	-27.288	1.548
800	-24.973	-27.304	-30.041	-27.593	1.642
fmGA without SSA (model Polyalinine)					
Pop Size	Max Fitness	Median Fitness	Best Fitness	Average Fitness	Stand Dev
100	-107.970	-125.792	-137.711	-126.745	9.766
200	-114.521	-136.491	-140.097	-131.804	8.961
400	-127.158	-136.440	-143.126	-135.559	5.183
800	-137.429	-139.203	-150.731	-140.893	4.377
fmGA with SSA (model Polyalinine)					
Pop Size	Max Fitness	Median Fitness	Best Fitness	Average Fitness	Stand Dev
100	-120.727	-131.821	-146.498	-134.587	7.948
200	-132.302	-138.315	-148.532	-138.840	4.458
400	-134.452	-139.478	-149.222	-140.618	4.432
800	-133.226	-140.827	-152.053	-140.920	4.743

is some minor fluctuation present in the results obtained between 15-90% as would be expected with a stochastic algorithm.

The results obtained from the second set of experiments is presented in Table 2. Table 2 illustrates that our modified fmGA algorithm is more effective at finding a lower energy value when utilizing the SSA. The data obtained from the SSA modified fmGA when applied to the Polyalanine peptide indicates that in all cases of population sizes tested, the standard deviation, median, best, and average fitness values were lower i.e. improved over the results obtained without the SSA. The results obtained from the fmGA as applied to the [Met]-Enkephelin pentapeptide showed no considerable difference with or without the SSA. This was expected since the [Met]-Enkephelin pentapeptide does not have a secondary structure. Additionally the use of the SSA has very little overhead in terms of the computational cost. In all experiments run, the number of additional fitness calls conducted with the SSA were less than 0.1% of the number of fitness calls conducted without the SSA. Therefore the SSA essentially does not increase the specific computational requirements of the fmGA algorithm.

It is clear that the Polyalinine peptide which is nearly 300% larger than the Met-Enkephalin, in terms of the number of residue angles and almost 250% larger in terms of the data structure takes a considerable amount of time to analyze; however, the goal of this paper was not to find a solution to larger proteins in the same time as smaller ones but instead to find "good" solutions to the larger proteins. The results presented here validate our work as an improvement in the effectiveness of the fmGA.

5 CONCLUSIONS

This paper has shown that it is possible to increase the effectiveness of the fmGA through the addition of secondary structure information. We have shown that the secondary structure analysis conducted improved the final energy obtained in all cases over the original algorithm for the Polyalanine peptide.

Future work will investigate increasing the number of secondary structures that the algorithm utilizes. Also, we will study varying fmGA parameter values as well as the increased utilization of BB information.

REFERENCES

[1] R. Pachter and Z. Wang, "Prediction of Polypeptide Conformation by the Adaptive Simulated Annealing Approach", Journal of Computational Chemistry, 18, 323, 1997.

[2] G. H. Gates, Jr., R. Pachter, L. D. Merkle, and G. B. Lamont, "Parallel Simple GAs vs Parallel Fast Messy GAs for Protein Structure Prediction", Proceedings of the Intel Supercomputer Users' Group Users Conference, 1995.

[3] D. E. Goldberg, K. Deb, H. Kargupta, G. Harik, "Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms", University of Illinois at Urbana-Champaign, IlliGAL Report 93004, 1993.

[4] L. D. Merkle, G. H. Gates, Jr., G. B. Lamont, and R. Pachter, "Application of the Parallel Fast Messy Genetic Algorithm to the Protein Structure Prediction Problem", Proceedings of the Intel Supercomputer Users' Group Users Conference, 189-195, 1994.

[5] S. R. Michaud, "Solving the Protein Structure Prediction Problem with Parallel Messy Genetic Algorithms", MS Thesis, AFIT/GCS/ENG/01M, Air Force Institute of Technolgy, Wright Patterson AFB, OH.

[6] S. R. Michaud, J. B. Zydallis, D. M. Strong, and G. B. Lamont, "Load Balancing Search Algorithms on a Heterogeneous Cluster of PCs", SIAM, 2001.

[7] D. R. Westhead and J. M. Thornton, "Protein Structure Prediction", Current Opinion in Biotechnology, 9, 383-389, 1998.

[8] D. Frishman and P. Argos, "Seventy Five Percent Accuracy in Protein Secondary Structure Prediction", Proteins, 29, 443-460, 1997.