Genetic Algorithm Feature-Based Resampling for Protein Structure Prediction

Trent Higgs, Bela Stantic, Md Tamjidul Hoque and Abdul Sattar

Abstract— **Proteins carry out the majority of functionality on a cellular level. Computational *protein structure prediction* (PSP) methods have been introduced to speed up the PSP process due to manual methods, like nuclear magnetic resonance (NMR) and x-ray crystallography (XC) taking numerous months even years to produce a predicted structure for a target protein. A lot of work in this area is focused on the type of search strategy to employ. Two popular methods in the literature are: Monte Carlo based algorithms and Genetic Algorithms. Genetic Algorithms (GA) have proven to be quite useful in the PSP field, as they allow for a generic search approach, which alleviates the need to redefine the search strategies for separate sequences. They also lend themselves well to feature-based resampling techniques. Feature-based resampling works by taking previously computed local minima and combining features from them to create new structures that are more uniformly low in free energy. In this work we present a feature-based resampling genetic algorithm to refine structures that are outputted by PSP software. Our results indicate that our approach performs well, and produced an average 9.5% root mean square deviation (RMSD) improvement and a 17.36% template modeling score (TM-Score) improvement.**

I. Introduction

PROTEINS carry out the majority of functionality within an organism on a cellular level. A protein is formed by a string of amino acids folding into a specific three-dimensional shape, which determines the *biological task* it will perform. An example of this would be the hemoglobin, which performs the task of carrying oxygen to the blood stream. These *biological tasks* are important in the biology domain, but when it comes to computer science the more exciting aspect in terms of proteins is the numerous three-dimensional shapes and sizes they adopt to perform these *biological tasks*. To elicit these three-dimensional shapes, a process known as *protein structure prediction* (PSP) is carried out.

To speed up the PSP process, due to manual methods like nuclear magnetic resonance (NMR) and x-ray crystallography (XC) taking numerous months or years to produce a predicted structure, several computational methods have been proposed. These methods can be grouped into three main categories, comparative or homology modeling, threading or fold recognition, and *ab initio* or *de nova*. Comparative modeling and threading work by aligning a protein target sequence with one or more template sequences. *Ab initio*, which is considerably different from the other two methods, is based on Afinsen's 'Thermodynamic Hypothesis' [1]. This states that a protein's native structure is at its lowest free energy minimum. Both comparative modeling and threading use protein sequences stored in databases to find suitable template sequence matches, however they produce very inaccurate models if the template sequence-similarity to the protein to be folded is $< 30\%$. The *ab initio* approach (which means 'from the origin') tries to produce a protein's three-dimensional structure like nature does - from its sequence alone. Our research predominately uses the *ab initio* approach.

The *ab initio* method in PSP has been looked at from many different perspectives, and therefore has a large amount of work utilising different search approaches. These include numerous versions of Monte Carlo (MC) [2], [3], Ant Colony Optimisation [4], and Simulated Annealing (SA) [5]. Statistical approaches such as Chain Growth (CG) [6], and Contact Interaction (CI) [7] have also been developed. However, most of these search approaches have the problem that as the sequence length increases the accuracy of the predicted structure decreases. Despite this, further research into bio-inspired algorithms [8], [9], such as Genetic Algorithms, have proven to be quite promising in solving the PSP problem [10], [11], [12].

GAs allow for a generic search strategy that can be applied to various cases, which alleviates the need to redefine the search strategies for separate sequences. It can also generate more successful descendants than random search. The most obvious problem with GAs is that after numerous generations the diversity within the population decreases, causing the GA to get stuck in local minima, and therefore producing sub-optimal solutions.

A technique that can be applied to the GA PSP search strategy is feature-based resampling. Feature-based resampling lends itself well to GAs as they can take features from various protein structures and recombine them to produce more *native-like* conformations. Feature-based resampling can be broken down into two concepts: (1) resampling, and (2) feature-based. Resampling by nature is taking an already searched search space, and using the output from that search as input into another search. Feature-based refers to taking various features from the initial search that are more *native-like* in nature and creating new structures with those features.

In this paper we present a feature-based resampling GA to refine structures from an initial PSP run. This is done by using the output (i.e. decoys) of a PSP software as the initial population for the search. Our GA utilises operators

Trent Higgs, Bela Stantic, and Abdul Sattar are with the Institute for Integrated and Intelligent Systems (IIIS), Griffith University, Queensland, Australia; emails: {T.Higgs, B.Stantic, A.Sattar}@griffith.edu.au.

Md Tamjidul Hoque is with Discovery Biology, Eskitis Institute for Cell & Molecular Therapies Griffith University, Queensland, Australia; email: Tamjidul.Hoque@gmail.com.

978-1-4244-8126-2/10/$26.00 ©2010 Crown

that have been used in numerous low-resolution approaches (e.g. Hydrophobic-Hydrophilic model), and we will be using Rosetta's energy function for fitness calculations. Our results indicate that this approach performs well and produced, on average, predictions that were closer to the native conformation when compared to the structures it started with.

The remainder of this paper is organised as follows. In Section II state-of-the-art PSP approaches will be discussed, Section III will look at PSP resampling techniques, Section IV will outline the methodology that we have applied, Section V explains the experimental setup, Section VI will present and discuss the results we gained from our experimentation, and in Section VII we will draw our conclusions.

II. PSP Approaches

PSP, on a computational level, has produced numerous search strategies to tackle this hard optimisation problem - to predict a protein's three-dimensional conformation. There are a number of achievements in this area and in this section we will describe some of the typical approaches: RAMP, TASSER, and Rosetta. Rosetta will be explained in more detail as it is used heavily in our experiments.

RAMP [5], [13] is a PSP software that utilises the hierarchial approach, which is solely based on the *ab initio* method. This approach uses a simple tetrahedral lattice that exhaustively enumerates through all possible compact conformations. From this large pool of conformations it will select the best 10,000 conformations using a lattice-based scoring function. These 10,000 conformations are then used as templates for constructing all-atom models by fitting a four-state off-lattice to the lattice conformations to add proper secondary structure refinement (due to the tetrahedral lattice only capturing overall protein chain connectivity not the finer details of secondary structure).

TASSER (Threading ASSEmbly Refinement) [3], [14], [15] uses a threading modeling approach and was designed to be able to recognise the majority of non-evolutionary related protein folds within the Protein Data Bank (PDB). This was so it could refine the structures it generated in regards to the initial template, and to be able to have better predictions for the loops. TASSER consists of three main processes, these are: template identification, structure assembly, and model selection.

Rosetta [2], [16] uses a Monte Carlo search to minimize an energy function based on 'Afinsen's Thermodynamic Hypothesis' that a protein's structure in nature is often the conformation that has the lowest free energy [1]. Each structure that is being predicted by Rosetta's search strategy goes through a two-stage process: (1) an initial search using a low-resolution representation of the protein structure where the side chains are depicted as centroids, and (2) a high-resolution refinement stage where all atoms are placed and a fitness function that is closer to the true physical energy is utilised [17]. The low-resolution fitness function is limited, and is usually unable to determine the native conformation between *native-like* structures and local minima. Despite this, the main global conformation of the protein structure comes together in the low-resolution search stage where the high-resolution search stage will alter this global conformation in small ways to mainly allow the placement of side chains.

The low-resolution search stage in Rosetta uses fragment replacement moves. This is done by taking a sequence of continuous resides of three or nine in length and replacing their backbone torsion angles with the torsion angles of a fragment obtained from the PDB. This is one of the main innovations that enables a lot of Rosetta's success in PSP. Instead of spending large amounts of computation time searching individual torsion angles, Rosetta can easily shift between structures that are locally viable.

For each target protein that Rosetta is to be run on a fragment pool is created ahead of time. This fragment pool contains for every *frame* of three or nine residues in the target protein a set of 200 fragments obtained from the PDB. Fragments are selected based on two main criteria. The first being the sequence-similarity between the target protein sequence within the frame and the sequence of the fragment in the PDB. The second criteria for fragment selection is matching secondary structure. This is done by comparing the predicted secondary structure for the target protein's fragment with the actual secondary structure of the fragment contained within the PDB. This pool is used for every predicted structure created for a certain target protein, and fragment replacement moves are pulled out of it at random. Once a fragment replacement has occurred local minimisation is applied and the move is either accepted or rejected.

All of the methods mentioned above are state-of-the-art in the PSP field. However, they all still struggle with producing accurate predicted models. This is even more apparent as the length of the protein increases. Another aspect PSP methods struggle with is finding the true global minimum of the energy function. It is very difficult to find the global minimum of the energy function in PSP, compared to any other known critical search space, as it is very high-dimensional and contains a large amount of local minima. The main strategy a lot of PSP software (e.g. Rosetta) use to combat this issue is to perform a large amount of random restarts. The inherent problem with this kind of strategy is that information about previous local minima is thrown away. Samples from previous conformation space may suggest regions of lower energy, which may be beneficial to pursue further sampling around. This leads to the intuition of resampling techniques, which are explained in the next section.

III. PSP Resampling Techniques

The main purpose of resampling techniques is to take an already sampled search space and then refine it. In regards to PSP, this can be looked at as taking already computed local minima and finding samples of conformation space within it that indicate regions which contain consistently lower energy, and focusing further sampling or searching around those regions. There are two main types of resampling used in PSP: structure-based resampling, and feature-based resampling [17].

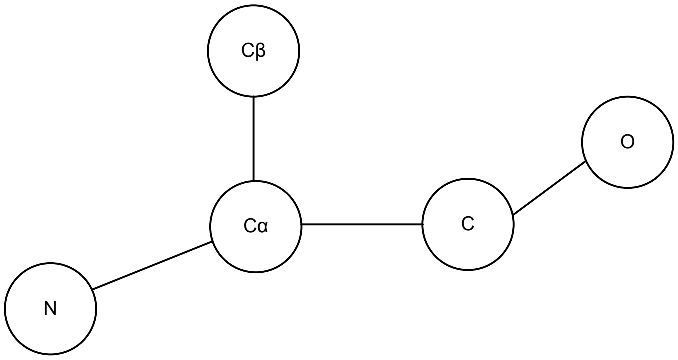

Fig. 1. Each amino acid in our protein structure representation will contain N, C_α, C_β, C, and O atoms.

Structure-based resampling techniques either selects regions of conformation space or individual protein structures and focuses further sampling around them. Feature-based resampling on the other hand is more concerned with *native-like* features from the previous sampling round. If no models from the previous round of sampling produces a structure close enough with the native structure, they still may contain various *native-like* features, which can be recombined to create new structures that are closer to the native conformation. A basic example of feature-based resampling can be seen as having a predicted protein structure with one domain wrong, but by intermixing this protein with another protein, which has the other correct domain, a structure is formed that is closer to the native conformation than either of the two structures alone.

A large amount of the literature in feature-based resampling is focused on using Genetic Algorithms (GA) [18], [19]. This is due to GAs lending themselves well to recombination of structures. For example, GAs can combine the best structures from the previous generation to produce more *native-like* structures in their crossover operator.

IV. Methodology

The main motivation for our work is to create a GA utilising feature-based resampling. This incorporates taking the initial predicted structures from a complete run of a *protein structure prediction* (PSP) software using an arbitrary target protein. These initial structures will then be used as input into our GA for refinement.

The PSP software we have selected to use in our experiments is Rosetta. The reason we have chosen Rosetta is two fold: (1) in Critical Assessment of Techniques for Protein Structure Prediction (CASP) Rosetta has outperformed numerous other PSP approaches [20], [21], [22], and (2) Rosetta is open source, making it easy to use and integrate into our software. For the same reasons we have used Rosetta's energy function in our GA for fitness calculations. In this section we will look at the model we will be using throughout our experiments and the setup for our GA will be explained in detail.

A. Protein Model

To do any manipulation of a protein structure a representation of the protein conformation needs to be decided on. For our experiments we will use a high-resolution centroid model, instead of a low-resolution model. In the low-resolution model primarily one monomer is used to depict the backbone of a protein's conformation, whereas in the centroid model the complete backbone is represented and the sidechain is replaced by a single large centroid. Cartesian coordinates will be utilised for our model (i.e. x, y, and z), rather than torsion angels. This is due to torsion angels being unable to represent the precise spatial interrelationships between residues that are not next/close to each other in the protein's sequence [17].

The atoms that our high-resolution model will consist of are the same as the structures outputted by popular PSP software (e.g. N, C_α, C_β, C, and O). Note this is not a complete high-resolution model as only C_β is used to represent the sidechain, however PSP software can easily be used to add in the finer details if required. An example of this model can be seen in Figure 1.

B. Genetic Algorithms

GAs belong to a specific class of evolutionary algorithms that are bio-inspired. It starts off with a large pool of genetic traits, which by use of genetic operators are reproduced, sometimes with random mutations, and are subjected to natural selection (i.e. the fittest survives). By doing this, there is no guarantee that the absolute best solution will be found, however it is apparent that over time it creates solutions that contain a combination of genetic traits, which function better in its defined environment. In PSP there has been a large amount of work in GAs using low-resolution models [10], [11], [12], [23], and very limited work using high-resolution models [24], [25].

Algorithm 1 PSP Genetic Algorithm

```
$i \leftarrow 1$;
$OF \leftarrow$ optimal $f$ (if known);
$F \leftarrow 0$;
Initialise population ($pop$);
COMPUTE $f$ of all chromosome$_{pop}$;
while i $<= targeted\_generations$ OR $F \neq OF$ do
    SELECTION;
    CROSSOVER;
    COMPUTE $f$ of all chromosome$_{new\_pop}$;
    REPLACE chromosome$_{pop}$ where
    ($f$(chromosome$_{new\_pop}$)) $<$ ($f$(chromosome$_{pop}$));
    MUTATION;
    $F \leftarrow$ min($f$($\forall$chromosome$_{pop}$));
    $i \leftarrow i + 1$;
end while
```

A general algorithm for a GA developed to solve the PSP problem can be found in algorithm 1. In this algorithm you can see that the main idea of a GA is to maintain a population

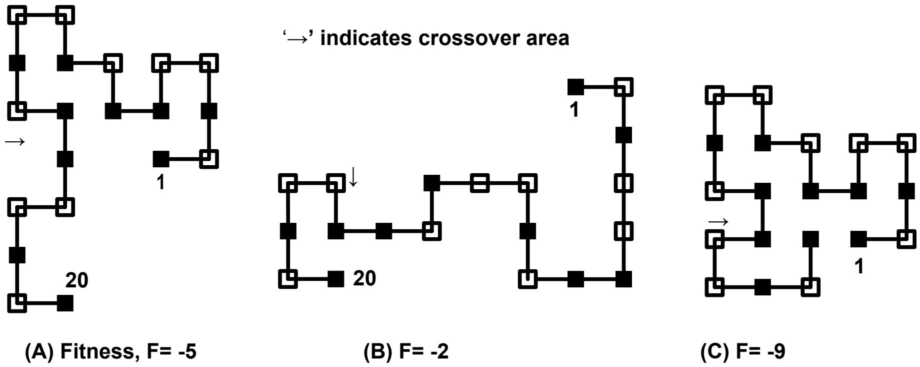

Fig. 2. An example of the crossover operator on a 2D HP Lattice. The first 14 residues of (A) are joined with the last 6 residues of (B). A random rotation of 270° was applied to achieve the compact structure in (C) [10].

of solutions. The population is continually evolving by use of genetic operators, and the size of the population is being maintained by replacing chromosomes in the population with fitter solutions created by the genetic operators. From this it can be seen that we must have a fitness function f, which can express the fitness of each solution as a numerical value.

In algorithm 1 i represents the amount of generations, $targeted\_generations$ refers to the amount of iterations it will complete if the optimal fitness is never reached, OF is the optimal fitness the GA is looking for in its search, and F is the best fitness found in the search so far. Note that OF is often not defined, as the optimal fitness is hard to identify, and therefore $targeted\_generations$ is sufficient to use as the stopping criteria for the search. For each generation selection and crossover operations are computed. The fitness of the new population is then calculated and a replacement function is carried out, which replaces any chromosome from the current population with a fitter chromosome from the new population (if it exists). Finally the population is mutated randomly.

GAs created for PSP applications can be defined by: given an amino acid sequence $s = s_1, s_2, s_3, ...s_m$ (m = the end of the amino acid sequence) a conformation (c) needs to have the form of $c^* \in C(s)$ and fitness where $E^* = E(C) = min\{E(c)|c \in C\}$ [26]. $C(s)$ refers to the set of all the possible conformations formed by sequence s which are valid (i.e. collision free). From this it can be seen that if a free energy calculation of c is q then $E(c)$ can be defined as $E(c) = -q$. Therefore, in PSP the smaller or more negative f is the fitter the conformation (c) is.

For all of our GA genetic operators we have used well-defined processes that are utilised on the Hydrophobic-Hydrophilic (HP) Lattice model by Unger [10], [27]. The next five sections will outline how our genetic operators and scoring methods were developed and integrated into our GA.

1) Selection: For selection a roulette wheel approach is taken. Solutions that contain medium to good fitness scores are more likely to be chosen over chromosomes that have less than average scores. In our GA we have a 80% chance that medium to good solutions are selected, leaving a 20% chance that less than average chromosomes will be selected.

2) Crossover: In Unger's GA the crossover operations are performed by swapping parts of one protein structure, with that of another (i.e. one point crossover). For example, two proteins ($p1$ and $p2$) are selected to breed with each other. A crossover point (n) is randomly selected, and then everything from n onwards in $p1$ is replaced with everything from n onwards in $p2$, and vice versa. This process will produce two offsprings. Figure 2 depicts the crossover procedure on a 2D HP lattice model.

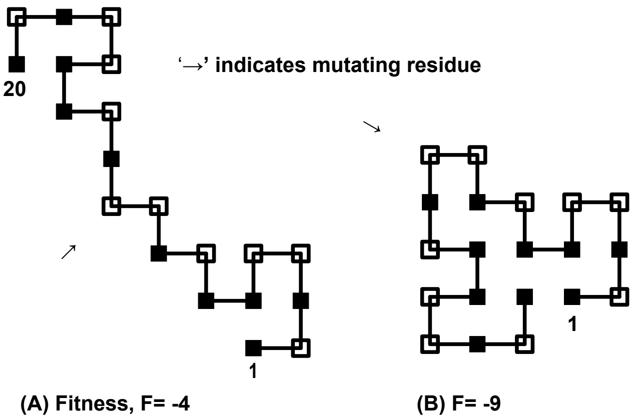

Fig. 3. An example of pivot rotation on a 2D HP Lattice. Residue 11 is randomly selected to be the pivot (n). By rotating the substructure ($n+1-m$) by 180° the fitness in (A) goes from being -4 to -9, which is depicted in (B) [10].

In our approach we took this initial concept that Unger applied in his GA using the HP model and extended it to the high-resolution model. To do this the crossover point (i.e. n) needed to be defined. In the low-resolution model there usually is only one possible monomer to represent a particular amino acid in the protein chain, whereas in the high-resolution model there are numerous atoms that could be used for n (e.g. see Figure 1). For our work we kept it simple and only allowed C_α atoms as our crossover points. In this sense it is very similar to how Unger's algorithm works, but it has been designed to consider the other atoms around the C_α atom when crossover is taking place.

Our crossover operator also contains a simplistic collision detection algorithm. This algorithm gauges how close atoms are to one another in Euclidian distance, and if they are too close it rejects the proposed protein structure. It is important to not let infeasible protein structures enter the gene pool, as this could cause the rest of the population to become diluted, therefore producing sub-optimal solutions. By incorporating this collision detection algorithm a contingency plan needs to be put into action if a protein structure is rejected. Otherwise the crossover acceptance rate may become too low, which can lead to stalling.

The contingency plan we have put in place is to try three times with the same parents ($p1$ and $p2$) using a new randomly chosen n each time. If after three times and the crossover is still not successful we change one of the parents ($p2$ becomes the next parent in the breeding list) and try again. This process is continued until a success has been reached or no more possible parents are left to try with $p1$. If that is the case then $p1$ is marked as incompatible and is either replaced by a randomly selected chromosome that was not originally selected for breeding, or deleted from the breeding list if there is no chromosomes left to replace it with.

3) Mutation: Unger's mutation operator was in the form of a one point pivot rotation move. Pivot rotations work by translating all points to a chosen pivot (n) and rotating the sub-structure around that pivot point ($n + 1$ to m). The sub-structure, in this case, refers to all the points in a protein structure from $n+1$ to the end of the structure (m). Rotation can be done around 3 axises: *x*, *y*, *z*. An example of a pivot rotation, using a 2D HP lattice model, can be found in Figure 3.

Just like the crossover operator we have extended this to a high-resolution model, and only used C_α atoms for n. As for degrees/angles of rotation we have 8 possibilities that are randomly chosen from. These are: 30°, 60°, 90°, 120°...240°. This operator also uses our collision detection algorithm, and will randomly select a new chromosome to mutate if the previous one had a collision.

4) Twin Removal: Twin's are a serious problem in any GA search, and this is also the case for *protein structure prediction* (PSP) [28]. In this version of our GA we have incorporated a simplistic version of twin removal, which will be updated over time. For now, our twin removal algorithm does not allow identical chromosomes into the population. It does this at the evaluation stage of the GA search between the new population and current population. If a chromosome in the new population has a better fitness than a chromosome in the current population it will first check to see if that chromosome already exists in the current population. If this is the case it will ignore it and move on to the next chromosome, otherwise it will replace the current population's chromosome with the new population's chromosome.

5) Scoring Methods: As mentioned before, we are using Rosetta's energy function for fitness calculations. We take the overall score that Rosetta's energy function produces for each structure we pass to it. In regards to scoring the final structures outputted from our GA we use two methods to gauge their structural similarity with the native structure. These two methods are: (1) root mean square deviation (RMSD) [29], and (2) template modeling score (TM-Score) [30].

$RMSD$ works by summing the Euclidian distances (e.g. $RMSD(v,w) = \sqrt{\frac{1}{n}\sum_{i=1} || v_i - w_i ||^2})$ between every residue from the two structures it is trying to compare. This is normally done by using the C_α atom, but it can be done using other atoms such as C, O, etc. TM-Score, in comparison, is more sensitive to close matches than distant matches (unlike RMSD). This means if one part of the protein is completely wrong, but the other topology is quite similar the TM-Score will not be as drastically effected.

V. Experimental Setup

All experiments used a population of 200 structures created by a random Rosetta run, 70% crossover rate and a 10% mutation rate. Each run went for 100 generations with results saved in 10 generation increments, from these results the decoy that had the best RMSD improvement was chosen as a representative for a particular protein. Protein structures were picked at random, with the following constraints: (1) they were between 40 and 110 residues in length, and (2) all structures contained only one chain. This was to keep the results as accurate as possible and eliminate any bias the GA algorithm might have to certain structures (except for the length).

The sample size we have used in our experiments is ±10 randomly selected proteins, which were each run through our algorithm for 100 generations. Even though, our testing was done on a relatively small number of proteins, uniform results obtained from the tested proteins depict a good indication of the capabilities of our approach.

VI. Empirical Results

Table I contains the results obtained from our simulations. It compares the results our GA produced to the original decoys created by Rosetta. It depicts for each protein used in our experiments: the PDB identifier, the protein's length, the fitness, RMSD and TM-score for the highest ranked Rosetta decoy, and the fitness, RMSD, TM-Score, RMSD improvement in %, and TM-Score improvement in % for the highest ranked GA decoy. The highest ranked decoy refers to the predicted protein structure, which had the best RMSD result (i.e. closest to 0).

Figure 5 depicts the number of generations each protein took to reach its optimal result. Figure 4 graphically compares the protein 2ptl native conformation with Rosetta's and our GA's best predicted structure for 2ptl. Figure 6 shows a comparison between the fitness values obtained for each

TABLE I

GA RESULTS COMPARED TO ROSETTA

Protein	**Length**	**Rosetta**			**GA**				
		f	RMSD	TM-Score	*f*	RMSD	TM-Score	RMSD Imp	TM Imp
2ptl	78	-124.10	8.764Å	0.4084	-129.46	3.888Å	0.5442	55.64%	33.25%
1pgx	83	-120.12	4.385Å	0.6570	-88.82	3.980Å	0.6304	9.24%	-4.05%
1bds	43	-21.97	6.151Å	0.2138	6.88	5.954Å	0.2406	3.20%	12.54%
1bm8	99	-82.09	7.814Å	0.2751	-67.86	7.688Å	0.2973	1.61%	8.07%
1emw	88	-48.36	8.415Å	0.2856	-51.69	8.591Å	0.3097	-2.09%	8.44%
1aoy	78	-57.80	6.028Å	0.3854	-62.71	5.350Å	0.5425	11.25%	40.76%
1csp	67	-84.95	2.576Å	0.7156	-79.62	2.527Å	0.7305	1.90%	2.08%
2ppp	107	-27.43	9.632Å	0.2780	-48.82	9.681Å	0.4483	-0.51%	61.26%
1kjs	74	-42.35	4.541Å	0.4706	-48.51	4.431Å	0.4999	2.42%	6.23%
1vcc	77	-70.08	2.950Å	0.6800	-60.89	2.586Å	0.7140	12.34%	5.00%

protein depicted in Table I. And finally Figure 7 demonstrates the improvement comparison between RMSD and TM-score.

A. Analysis and Discussion

The first observation that can be made from our results is that each protein we tested only took, on average, between 10 and 20 generations to reach their optimal result (see Figure 5). If more generations were applied the fitness would continue to improve, but the likeness to the native would degrade. This could be due to the fitness function we employed. Rosetta's fitness function is not perfect and can find structures that have a low energy score, but are not close to the native. If this is the case then it can quite easily let structures like this dominate the population, making the overall population less like the native. A comparison using different fitness functions and their effectiveness could give us more insight into this.

In Table I, it can be seen that all proteins used in our experiments had an improvement to some degree (i.e. RMSD, TM-Score, or both). Improvement is measured by calculating the percentage improvement between the previous (v_1) and new value (v_2) (e.g. $((|v_1 - v_2|)/v_1) * 100$). A negative improvement is represented with a $-$ in front of it, and is used for two cases, these are: (1) $RMSDv_1 < RMSDv_2$, and (2) $TMv_2 < TMv_1$. The proteins we used within our experiments ranged in length from 43-107 residues. The best result we obtained was from protein 2ptl, which is 78 residues in length. It had a 55.64% improvement in RMSD and a 33.25% improvement in its TM-Score (see Figure 4 for a graphical representation). On average we had a 9.5% improvement in RMSD, and a 17.36% in TM-Score using our method when compared to the original decoys outputted by Rosetta.

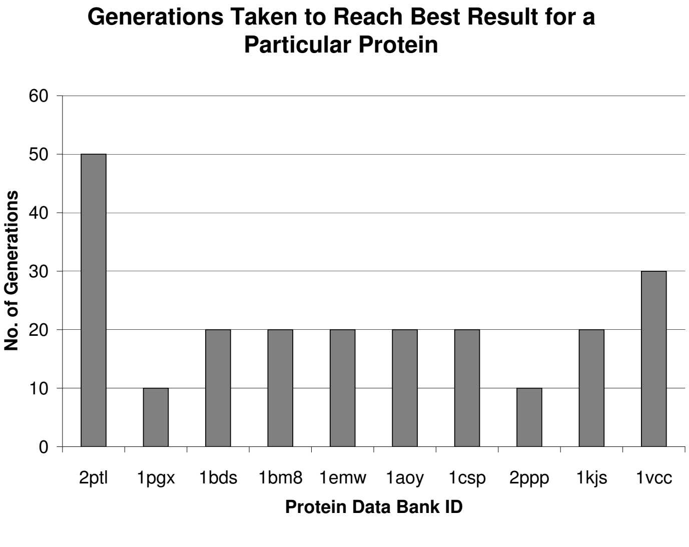

Fig. 5. Number of generations each protein took to obtain its best result.

In Figure 6 we have shown the fitness value comparison

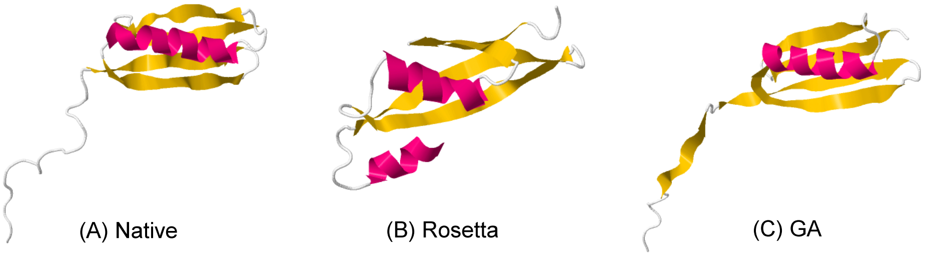

Fig. 4. (A) contains model 1 of the native conformation for 2ptl from the PDB, (B) contains Rosetta's best predicted structure for 2ptl, and (C) contains our GA's best predicted structure for 2ptl. Note that both predicted structures have been rotated to match as close as possible with the native conformation. All protein images were generated with Jmol [31].

between Rosetta's best predicted structures to ours. These results show that the fitness value has a limited effect on how good a structure is. This can be inferred by there being a 50% split between the GA structures having and not having better fitness values when compared to Rosetta's original fitness values. It can also be seen that on average there is not too much difference between the fitness values for the same protein between the two approaches (i.e. Rosetta and GA), which can be seen in Figure 6. This, like the number of generations, could also be further investigated by applying different fitness functions and evaluating their effectiveness using our GA feature-based resampling approach.

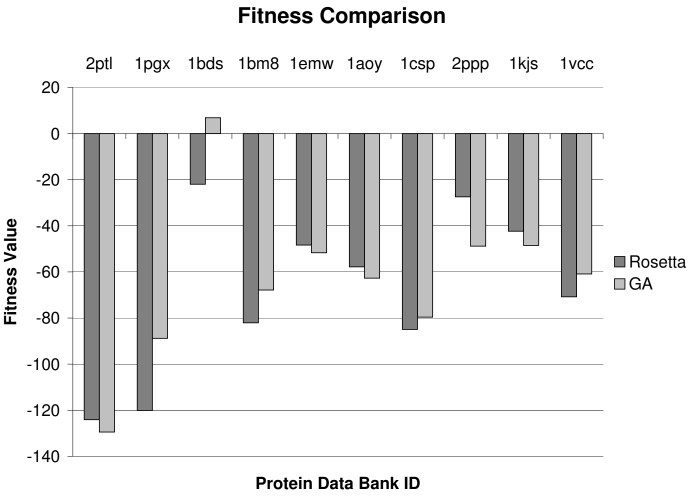

Fig. 6. Fitness comparison between Rosetta and GA for each protein depicted in Table I.

The last aspect we looked at was the RMSD improvement compared with that of the TM-Score improvement. As mentioned before, we had an overall average 9.5% improvement for RMSD and a 17.36% improvement for TM-score. In Figure 7 we have depicted the improvement values for RMSD and TM-Score. RMSD values are classified as an improvement if the new value produced is closer to 0 than the previous value. TM-Score on the other hand is based on whether or not the new value produced is closer to 1 than the previous value. As our averages show, Figure 7 depicts that we achieved better improvement values for TM-Score than RMSD. TM-Score calculates the similarity of two proteins based on topology and puts more emphasis on close matches rather than distant matches. This makes it a lot more sensitive than the RMSD measure, and in some cases more accurate.

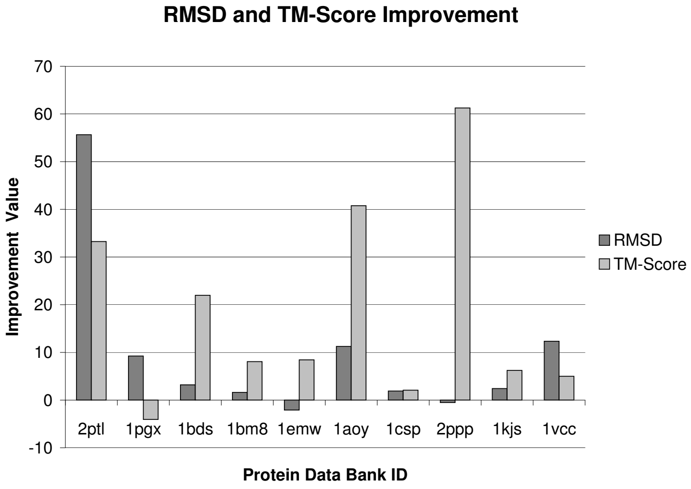

Fig. 7. RMSD and TM-Score improvement comparison.

VII. Conclusion

In this paper we have developed a feature-based resampling approach for *protein structure prediction* (PSP) utilising Genetic Algorithms (GA). GAs lend themselves well to PSP as they can be created as a generalised search that does not require its main search operators to be modified for particular cases or domains. It also naturally incorporates the ability to perform feature-based resampling in its search procedure. Feature-based resampling works by taking previously computed local minima and combing features from them to create new structures that are more uniformly low in free energy.

Our GA was created by utilising similar crossover, and mutation operators that were defined in Unger's GA approach. This incorporated a one point crossover technique that spliced together two protein structures with the same sequence, and a one point mutation operator that used rotational move sets. Our algorithm also included a simplistic twin removal scheme, which removed twins based off of 100% similarity with any other chromosome in the population. For scoring the final structures we used two structural measures: RMSD and TM-Score. Results obtained from the randomly selected proteins indicate that our proposed feature-based resampling GA performed well, and produced an average 9.5% RMSD improvement and a 17.36% TM-Score improvement.

In regards to future work, it would be interesting to evaluate different fitness functions other than just Rosetta's to see if the results could be improved further by using a different energy function. Other improvements could be to incorporate a more sophisticated twin removal scheme, biological move sets and constraints, and samples or decoys from different PSP software, rather than just using Rosetta.

Acknowledgment

This research is partly sponsored by ARC (Australian Research Council) grant no DP0557303.

References

[1] C. Anfinsen, "Principles that govern folding of protein chains," *Science*, vol. 181, pp. 223–230, 1973.

[2] C. Rohl, C. Strauss, and D. Baker, "Protein structure prediction using rosetta," *Methods Enzymol*, vol. 383, pp. 66–93, 2004.

[3] S. Wu, J. Skolnick, and Y. Zhang, "Ab initio modeling of small proteins by iterative TASSER simulations," *BMC Biology*, vol. 5, no. 17, 2007, doi:10.1186/1741-7007-5-17.

[4] A. Shmygelska and H. Hoos, "An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem," *BMC Bioinformatics*, vol. 6, no. 30, 2005.

[5] L.-H. Hung, S.-C. Ngan, T. Liu, and R. Samudrala, "PROTINFO: new algorithms for enhanced protein structure predictions," *Nucleic Acids Research*, vol. 33, pp. 77–80, 2005.

[6] E. Bornberg-Bauer, "Chain growth algorithms for HP-type lattice proteins," in *Research in Computational Molecular Biology RECOMB*, 1997, pp. 47–55.

[7] L. Toma and S. Toma, "Contact interactions methods: A new algorithm for protein folding simulations," *Protein Science*, vol. 5, no. 1, pp. 147–153, 1996.

[8] V. Cutello, G. Nicosia, P. Pavone, and J. Timmis, "An immune algorithm for protein structure prediction on lattice models," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 101–117, 2007.

[9] M. Judy, K. Ravichandran, and K. Murugesan, "A multi-objective evolutionary algorithm for protein structure prediction with immune operators," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 12, no. 4, pp. 407–413, 2009.

[10] R. Unger and J. Moult, "Genetic algorithms for 3D protein folding simulations," *Journal of Molecular Biology*, vol. 231, pp. 75–81, 1993.

[11] T. Hoque, M. Chetty, and A. Sattar, "Protein folding prediction in 3D FCC HP lattice model using genetic algorithm," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 4138–4145.

[12] ——, "Extended HP model for protein structure prediction," *Journal of Computational Biology*, vol. 16, pp. 85–103, 2009.

[13] Y. Xia, E. Huang, M. Levitt, and R. Samudrala, "Ab initio construction of protein tertiary structures using a hierarchical approach," *Journal of Molecular Biology*, vol. 300, pp. 171–185, 2000.

[14] Y. Zhang and J. Skolnick, "Tertiary structure predictions on a comprehensive benchmark of medium to large size proteins," *Biophysical Journal*, vol. 87, pp. 2647–2655, Oct. 2004.

[15] ——, "Automated structure prediction of weakly homologous proteins on a genomic scale," *PNAS*, vol. 101, no. 20, pp. 7594–7599, May 2004.

[16] K. Simons, C. Kooperberg, E. Huang, and D. Baker, "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions," *Journal of Molecular Biology*, vol. 268, pp. 209–225, 1997.

[17] B. Blum, "Resampling methods for protein structure prediction," Ph.D. dissertation, Electrical Engineering and Computer Sciences University of California at Berkeley, Dec. 2008.

[18] J. Pedersen and J. Moult, "Ab initio structure prediction for small polypeptides and protein fragments using genetic algorithms," *PROTEINS: Structure, Function, and Bioinformatics*, vol. 23, pp. 454–460, 1995.

[19] Y. Cui, R. Chen, and W. Wong, "Protein folding simulation with genetic algorithm and supersecondary structure constraints," *PROTEINS: Structure, Function, and Genetics*, vol. 31, pp. 247–257, 1998.

[20] K. Simons, R. Bonneau, I. Ruczinski, and D. Baker, "Ab initio protein structure prediction of CASP III targets using ROSETTA," *PROTEINS: Structure, Function, and Bioinformatics*, vol. 3, pp. 171–176, 1999.

[21] R. Bonneau, J. Tsai, I. Ruczinski, D. Chivian, C. Rohl, C. Strauss, and D. Baker, "Rosetta in CASP4: Progress in ab initio protein structure prediction," *PROTEINS: Structure, Function, and Genetics*, vol. 5, pp. 119–126, 2001.

[22] P. Bradley, D. Chivian, J. Meiler, K. Misura, C. Rohl, W. Schief, W. Wedemeyer, O. Scueler-Furman, P. Murphy, J. Schonbrun, C. Strauss, and D. Baker, "Rosetta predictions in CASP5: Success, failure, and prospects for complete automation," *PROTEINS: Structure, Function, and Genetics*, vol. 53, pp. 457–468, 2003.

[23] T. Jiang, Q. Cui, G. Shi, and S. Ma, "Protein folding simulations of hydrophobic-hydrophilic model by combining tabu search with genetic algorithms," *Journal of Chemical Physics*, vol. 119, no. 8, pp. 4592–4596, 2003.

[24] J. Pedersen and J. Moult, "Protein folding simulations with genetic algorithms and a detailed molecular description," *Journal of Molecular Biology*, vol. 269, pp. 240–259, 1997.

[25] J. Arunachalam, V. Kanagasabai, and N. Gautham, "Protein structure prediction using mutually orthogonal latin squares and a genetic algorithm," *Biochemical and Biophysical Research Communications*, vol. 342, pp. 424–433, 2006.

[26] T. Hoque, M. Chetty, and L. Dooley, "A guided genetic algorithm for protein folding prediction using 3D hydrophobic-hydrophilic model," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 8103–8110.

[27] R. Unger and J. Moult, "On the applicability of genetic algorithms to protein folding," in *The 26th Hawaii International Conference on System Sciences*, 1993, pp. 715–725.

[28] T. Hoque, M. Chetty, A. Lewis, and A. Sattar, "Twin removal in genetic algorithms for protein structure prediction using low resolution model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2009, doi:10.11.09/TCBB.2009.34.

[29] V. Maiorov and G. Crippen, "Significance of root-mean-square deviation in comparing three-dimensional structures of globular proteins," *Journal of Molecular Biology*, vol. 235, pp. 625–634, 1994.

[30] Y. Zhang and J. Skolnick, "Scoring function for automated assessment of protein structure template quality," *PROTEINS: Structure, Function, and Bioinformatics*, vol. 57, pp. 702–710, 2004.

[31] "Jmol: an open-source java viewer for chemical structures in 3D," http://www.jmol.org/.