A New Genetic Algorithm for Simplified Protein Structure Prediction

Mahmood A Rashid[1,2], Md Tamjidul Hoque[3],
M.A.Hakim Newton[1,2], Duc Nghia Pham[1,2], Abdul Sattar[1,2]

[1] Queensland Research Lab, National ICT Australia
[2] Institute for Integrated & Intelligent Systems, Griffith University
[3] Computer Science, University of New Orleans, USA

Abstract. In this paper, we present a new genetic algorithm for protein structure prediction problem using face-centred cubic lattice and hydrophobic-polar energy model. Our algorithm uses *i*) an exhaustive generation approach to diversify the search; *ii*) a novel hydrophobic core-directed macro move to intensify the search; and *iii*) a random-walk strategy to recover from stagnation. On a set of standard benchmark proteins, our algorithm significantly outperforms the state-of-the-art algorithms for the same models.

Keywords: Protein Structure Prediction, Genetic Algorithms, Local Search, Lattice Models, Energy Models, Random-walk

1 Introduction

Protein Structure Prediction (PSP) is computationally a very hard problem [24]. Given a protein's amino acid sequence, the problem is to find a three dimensional structure of the protein such that the total interaction energy amongst the amino acids in the sequence is minimised. The protein folding process that leads to such structures involves very complex molecular dynamics [4] and unknown energy factors. In the pursuit of addressing this difficulties in a hierarchical way, researchers have considered simplified models [21,17,26] for PSP. However, the complexity of the simplified problem still remains challenging.

There are a large number of existing search algorithms that attempt to solve the PSP problem by exploring feasible lattice-based structures called *conformations.* The state-of-the-art results on face-centred cubic (FCC) lattice based hydrophobic-polar (HP) energy model have been achieved by local search (LS) methods [5,9]. On the other hand, genetic algorithms (GA) [14], and tabu search [3] found promising results on 2D and 3D hexagonal lattice based HP models. In general, the success of GA and LS methods crucially depends on the balance of diversification and intensification of the search. Moreover, these algorithms often get stuck in local minima. As a result, they perform poorly on large proteins. Any further progress to these algorithms require addressing the above issues appropriately.

In this paper, we introduce a population based algorithm (GA$^+$) under the GA framework for simplified PSP. We use HP based energy model on 3D FCC lattice to simplify the problem. In GA$^+$, we use *i*) an exhaustive generation approach to diversify the search; *ii*) a novel hydrophobic core-directed macro move to intensify the search; and *iii*) a random-walk based approach to recover from stagnation. On a set of benchmark proteins, GA$^+$ significantly outperforms the state-of-the-art PSP algorithms in the same models.

The rest of the paper is organised as follows: Sect. 2 reviews background knowledge, Sect. 3 discusses related work on PSP; Sect. 4 describes our new GA for simplified PSP; Sect. 5 presents the experimental results and analyses; and finally, Sect. 6 draws the conclusion and outlines our future research.

2 Preliminaries

Proteins are essentially sequences of amino acids. They adopt specific folded three-dimensional structures to perform specific tasks. The function of a given protein is determined by its *native* structure, which has the lowest possible free energy level. Nevertheless, misfolded proteins cause many critical diseases such as Alzheimer's disease, Parkinson's disease, and Cancer [8]. Protein structures are important in drug design and biotechnology.

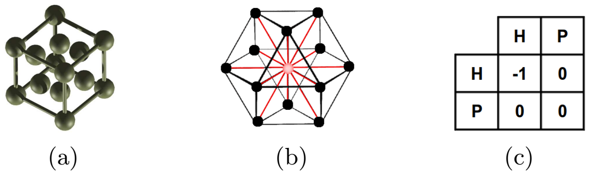
Homology modeling, *protein threading*, and *ab initio* are three computational approaches used in PSP. Prediction quality of *homology modeling* and *protein threading* depend on previously known structures of sequentially similar proteins. Our work is based on the *ab initio* approach that depends only on the amino acid sequence of the target protein. In our simplified PSP model, we use FCC lattice for mapping conformations that satisfy a self-avoiding walk. We also use HP energy model for conformation evaluation, and an enhanced genetic algorithm for conformation search. The self-avoiding walk constraint, FCC lattice, HP energy model, and genetic algorithms are described below.

2.1 Self-Avoiding Walk

In lattice based protein representation, the amino acids of a given sequence are mapped on lattice points satisfying a self-avoiding-walk constraint. A self-avoiding walk constraint ensures no revisitation of any lattice point during the sequence mapping.

2.2 FCC Lattice

The FCC lattice has the highest packing density compared to the other existing lattices [10]. In FCC, each lattice point has 12 neighbours (Fig. 1a) with 12 *basis vectors* $(1,1,0)$, $(-1,-1,0)$, $(-1,1,0)$, $(1,-1,0)$, $(0,1,1)$, $(0,1,-1)$, $(1,0,1)$, $(1,0,-1)$, $(0,-1,1)$, $(-1,0,1)$, $(0,-1,-1)$, and $(-1,0,-1)$. The hexagonal closed pack (HCP) lattice, also known as cuboctahedron (Fig. 1b), was used in [14]. In HCP, each



	H	P
H	-1	0
P	0	0

(a) (b) (c)

Fig. 1. a) FCC lattice, b) HCP lattice, c) HP energy model [16] lattice point has 12 neighours that correspond to 12 basis vertices with real-numbered coordinates, which causes the loss of structural precision for PSP. In simplified PSP, conformations are mapped on the lattice by a sequence of basis vectors, or by the *relative vectors* that are relative to the previous basis vectors in the sequence.

2.3 HP Energy Model

The HP energy model is based on the hydrophobicity of the amino acids. In the HP model [7,16], when two non-consecutive hydrophobic amino acids become topologically neighbours, they release a certain amount of energy, which for simplicity is shown as -1 in Fig. 1c. The total free-energy E (Shown in Equation 1) of a conformation, based on the HP model, becomes the sum of the energy released by all pairs of non-consecutive hydrophobic amino acids.

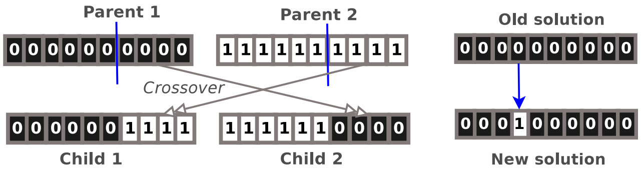
$$E = \sum_{i<j-1} c_{ij}.e_{ij} \tag{1}$$

Here, $c_{ij} = 1$ if ith and jth amino acids are non-consecutive in the sequence but are neighbours on the lattice, otherwise 0; and $e_{ij} = -1$ if ith and jth amino acids are both hydrophobic, otherwise 0.

2.4 Genetic Algorithms

GAs are a population-based search for optimisation problems. A genetic algorithm maintains a set of solutions known as population. In each *generation*, it generates a new population from the current population using a given set of genetic operators known as *crossover* and *mutation*. It then replaces inferior solutions by superior newly generated solutions to get a better current population. A typical crossover operator randomly splits two solutions at a randomly selected crossover point and exchanges parts between them (Fig. 2a). A typical mutation operator alters a solution at a random point (Fig. 2b). In the case of PSP, conformations are regarded as solutions of a GA. Below we describe genetic operators used in PSP.

Crossover Operators: The crossover operators are applied on two selected parent conformations to exchange their parts to generate child conformations. In a *single-point crossover*, both parents are splitted at a single point (Fig. 3 a) while in a *multi-point crossover* they are splitted at more than one point. Nevertheless, the crossover operations succeed if they produce conformations that satisfy the self-avoiding walk constraint.



(a) Crossover (b) Mutation

Fig. 2. Typical (a) crossover and (b) mutation operators

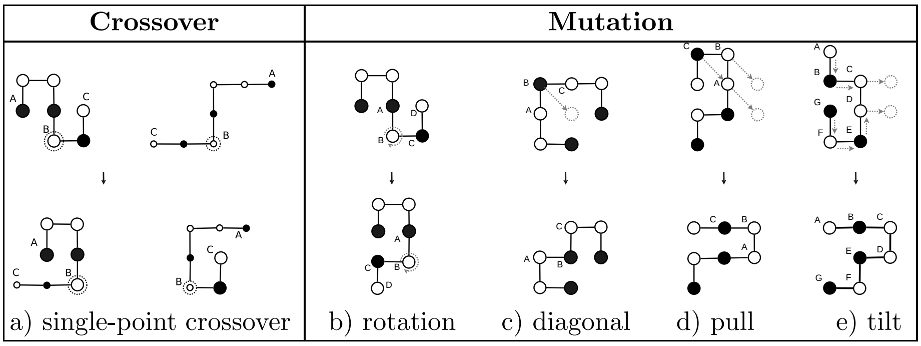


Fig. 3. The primitive operators that are used in our GAs on 3D FCC lattice space. For easy comprehension, the figures are presented in 2D space.

Mutation Operators: The mutation operators are applied on a single conformation. The operators can perform single-point change or multi-point changes. The mutation operations succeed if the resultant conformation remains a self-avoiding walk on the lattice. The primitive mutation operators (as shown in Fig 3 (b-e)) are described below:

1. **Rotation:** One part of a given conformation is rotated around a selected point (Fig 3 b). This move is mostly effective at the begining of the search.
2. **Diagonal Move:** Given three consecutive amino acids at lattice points A, B, and C, a diagonal move at position B takes the corresponding amino acid diagonally to a free position (Fig 3 c). Diagonal moves are very effective on FCC lattice [5,9] points.
3. **Pull Moves:** The amino acids at points A and B are pulled to the free points (Fig 3 d) and the connected amino acids are pulled as well to get a valid conformation. Pull moves [18] are local, complete and reversible. Pull moves are very effective especially when the conformation is compact.
4. **Tilt Moves:** Two or more consecutive amino acids connected in a straight line are moved by a tilt move to immediately parallel lattice positions [12]. Tilt-moves pull the conformation from both sides until a valid conformation is found. In Fig. 3 e), the amino acids at points C and D are moved and subsequently other amino acids from both sides are moved as well.

3 Related Work

Different types of metaheuristic have been used in solving the simplified PSP problem. These include Monte Carlo Simulation [23], Simulated Annealing [22], Genetic Algorithms [25,12], Tabu Search with GA [3], Tabu Search with Hill Climbing [15], Ant Colony Optimisation [2], Immune Algorithms [6], Tabu based Stochastic Local Search [5], and Constraint Programming [9]. Below we describe PSP methods that are based on local search and genetic algorithms.

Local Search: Starting from an initial solution, local search algorithms move from one solution to another to find a better solution. Local search algorithms are well known for efficiently producing high quality solutions, which are difficult for systematic search approaches. However, they are incomplete [1], and suffer from revisitation and stagnation. Restarting the whole or parts of a solution remains the typical approach to deal with such situations. In PSP problem, Cebrian *et al.* [5] used a local search algorithm combined with tabu heuristic. They implemented their method for the 3D FCC lattice and the HP model, and tested its effectiveness on Harvard instances [28]. Later, Dotu *et al.* [9] extended the work in [5] by using a hybrid method that combines local search and constraint programming together. Overall, these two methods have produced the current state-of-the art results for PSP on FCC lattice and HP energy model.

Genetic Algorithms: Unger and Moult [25] first applied GA to PSP and found their method to be more promising than the Monte Carlo based methods. They used absolute encodings on the square and cubic lattices, and the HP energy model. They only applied single point crossovers, and discarded infeasible solutions. Later, Patton [20] used relative encodings to represent conformations and a penalty method to enforce the self-avoiding walk constraint. In [14], GAs have been used by Hoque *et al.* for cubic, and 3D HCP lattices. They also introduced a twin-removal operator [13] to remove duplicates from the population and to prevent premature convergence.

4 Our Algorithms

Fig. 4 presents our GA$^+$. The algorithm initialises the current population and evaluates them. At each generation, it selects a genetic operator based on a given probability distribution to use through the generation. This operator is used in an exhaustive way to obtain all conformations in the new population. We ensure that no duplicate conformation is added to the new population. For a given number of generations, if the best conformation in the new population is not better than the best in the current population, our algorithm then resorts to a random-walk procedure to diversify the new population. Nevertheless, after each generation, the new population becomes the current population; and the search continues. Finally, the best conformation found so far is returned.

Note that our GA is different from a typical GA in a number of ways. A typical GA *i*) randomly selects an operator every time before generating a new solution; *ii*) selects parent solutions randomly; *iii*) applies the operators on randomly selected points; *iv*) generates only one (for mutation) or two (for crossover) solutions; *v*) does not use any macro-move; *vi*) does not use a random-walk in stagnant situation; and *vii*) does not remove duplicate conformations.

4.1 Exhaustive Generation

For mutation operators, our algorithm adds one resultant conformation to the new population for *each* conformation in the current population. In Fig. 4 Procedure mutConf, notice that the child conformations are generated by applying the genetic operator at *each* position of the parent conformation. The resultant conformation of a mutation operation is either the parent conformation itself or a child depending on the quality of the conformations.

For crossover operators, two resultant conformations are added to the new population from each application of the operator on two randomly selected conformations in the current population. Crossover operators (Procedure crsConfs)

```
Procedure gaPlus(opR,rwT)
op: Operators, $c, c'$: Conformations
opR: Operator selection probabilities
curP,newP: Current and new populations
rwT: Number of non-improving
    generations before random walk.
//======================
initPopulation(curP)
foreach Generation until timeout do
    selectOperator(op, opR)
    if mutation(op) then
        foreach $c \in$ curP do
            newP.add(mutConf($c$))
    else //crossover(op)
        while $\neg$full(newP) do
            $c, c' \leftarrow$ randomConfs(curP)
            newP.add(crsConfs($c, c'$))
    if $\neg$improved(newP, rwT) then
        rndWalk(newP)
    curP $\leftarrow$ newP
return bestConformation(curP)
```

```
Procedure mutConf(conf)
mutants.add(conf)
foreach $1 \leq$ pos $\leq$ conf.length() do
    $c \leftarrow$ applyOperator(conf, pos)
    mutants.add($c$)
return bestConformation(mutants)
```

```
Procedure crsConfs(conf,conf')
N: Number of iteration
// typically $N =$ conf.length()/10
crossbred.add(conf, conf')
for $i = 1$ to N do
    pos $\leftarrow$ random(1, conf.length())
    $c, c' \leftarrow$ applyOperator(conf, conf', pos)
    crossbred.add($c, c'$)
return best2Conformations(crossbred)
```

Fig. 4. Our new genetic algorithms for PSP

generate child conformations by randomly selecting the crossover points on the parent conformations. Note that the child generation method is not strictly exhaustive in crossovers. However, unlike typical GAs, a number of child conformations are generated by our algorithm. The best two conformations from the parents and the children then become the resultant conformations.

4.2 Macro-Move Operator

Protein structures have hydrophobic cores (H-core) that hide the hydrophobic amino acids from water and expose the polar amino acids to the surface to be in contact with the surrounding water molecules [27]. H-core formation is the main objective of HP based PSP. To achieve this, the total distance of all H-H pairs is minimised in [5]. A predefined motif based segment replacement strategy is applied in [14]. In this paper, we present a macro-move operator to aid forming the H-core. Our macro move performs a series of diagonal-moves on a given conformation to build the H-core around the hydrophobic core centre (HCC). See Fig. 5 for an example. The macro-move squeezes the conformation and quickly forms the H-core. In our implementation, the macro-move is used like any other mutation operators.

In the macro-move (Fig. 6 Procedure macroMove), the HCC is calculated by finding arithmetic means of x, y, and z coordinates of all hydrophobic amino acids. The macro-move for a given number of iterations repeatedly applies the

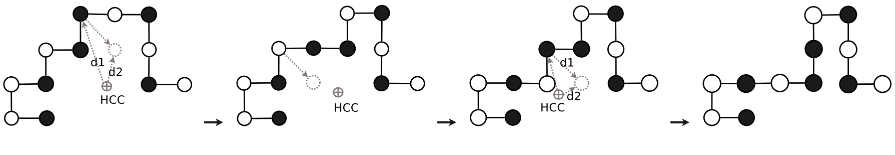


Fig. 5. A macro move operator comprising a series of diagonal moves. For easy understanding, the figures are presented in 2D space. The solid-black circles represent the hydrophobic amino acids and the hollow ones the hydrophilic.

```
Procedure macroMove(conf)
for $i = 1$ to Repeat do
    $T$ = P if bernoulii($p$), else H
    $A[j]$ : $j$th amino acid in conf.
    foreach $j$ : type($A[j]$) = $T$ do
        apply diagonal move at $j$, if $T$ = P
        or dist($A[j]$,hcc) is non-increaseing.
        break on first success.
return conf
```

```
Procedure rndWalk(pop)
foreach conf ∈ pop do
    for $i = 1$ to Repeat do
        $A[j]$ : $j$th amino acid in conf.
        foreach $A[j]$ do
            apply pull-move move at $j$.
            break on first success.
return pop
```

Fig. 6. Our macro move and random walk algorithms

diagonal move either at each P- or at each H-type amino acid positions. Whether to apply the diagonal move on P- or H-type amino acids is determined by using a Bernoulii distribution with probability p (typically $p = 20\%$ for P-type amino acids). For a P-type amino acid, we consider the first successful diagonal move. For a H-type amino acid, we take the first successful diagonal move that does not increase the distance of the amino acids from the HCC. Note that a large number of iterations would prematurely squeeze the conformation to a great extent while a small number of iterations could allow other genetic operators to play their roles in the search. We typically use 10 iterations for the first few hundreds of generations, later we use 5 iterations as the search progresses.

4.3 Stagnation Recovery by Random Walk

When the best conformation found so far remains the same for a number of generations (Fig. 4 Procedure gaPlus Line 14), we term this as a stagnation. In a stagnation, a random-walk algorithm (Fig. 6 Procedure rndWalk) applies unconditional pull moves on each conformation of the new population. We repeat the process for a number of iterations. A large number of iterations would greatly diversify the population. We typically use a number between 5 to 10 for this.

4.4 Further Implementation Details

Below we describe the other implementation choices of our algorithm in details.

Conformation Representation: We represent conformations by 3D coordinates and relative encodings. While coordinates help us determine whether a point on the lattice is free, relative encodings help apply genetic operators in generating conformations and then eliminate duplicate conformations.

Conformation Generation: For conformation generation, we use the genetic operators listed in Sect. 2 as well as the macro-move operator.

Conformation Evaluation: The fitness function we use in our algorithm to evaluate conformations is the exact energy function for the HP energy model (see Sec. 2). We do not use any other fitness functions such as sum of all H-H pair distances as is used in [5].

Operator Selection: The probability distribution to select operators is chosen intuitively. The single- and multi-point crossovers are selected with 15% and 5% probabilities giving 20% chance to crossovers. The rotation, diagonal-move, pull-move, tilt-move, and macro-move are selected respectively with probabilities 20%, 10%, 30%, 10%, and 10%. For experiments, when macro-moves are not used, diagonal moves are alone given 20% chance. Pull moves are given more chance than tilt moves as the latter tends to make more changes (in both sides) to the conformation than the former (in one side).

Population Size: The number of conformations explored in each generation should be more for a large protein than for a small one. In our algorithm, the number of such conformations are $O(n \times l)$ where n is the population size and l is the protein length. This is because we apply mutation operators at each amino acid position of each conformation in the population. For crossovers, the case is slightly different, but they are selected only with 20% probability. For the time being, we use $n = 100,\ 80,\ 60,\ 50$ for $l >= 50,\ 100,\ 200,\ 400$ respectively.

Population Initialisation: We generate the initial population by randomly selecting the basis vectors between each consecutive pair of amino acids. The generated conformations are all valid and satisfy the self-avoiding walk constraint.

5 Experimental Results

Among the protein instances (Table 1) used in our experiment, the H instances are taken from Harvard benchmarks [28]; F, S, and R instances are taken from Peter Clote laboratory website[1]. Cebrian et al. [5] and Dotu et al. [9] used these instances to test their algorithms. We also use three more large sequences, which are taken from the CASP[2] competition having CASP target IDs: *T0516*, *T0570*, and *T0563*. The CASP targets are converted to HP sequences based on the hydrophobic and polar properties of the constituent amino acids. The lower bounds of free energy (in Column LB-FE of Table 1) are obtained from [5] and also by using the CPSP tool [19]; however, there are some unknown values because CPSP tool cannot find lower bounds of free energy for large sequences.

Protein Info			**Energy Values (-ve) Achieved by Different Algorithms**												**T**
			BGA		**RGA**		**WGA**		**MGA**		**GA$^+$**		**LS [5]**		
Seq	Size	LB-FE	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	Best	Avg	mins
H1	48	69	65	58	=	67	=	68	=	67	**=**	**69**	68	66	30
H2		69	63	57	=	67	=	68	=	68	**=**	**69**	=	65	
H3		72	64	57	71	69	=	70	=	71	**=**	**72**	69	66	
F90_1	91	168	133	119.5	159	144	165	161	165	159	**=**	**166**	164	160	120
F90_2		168	132	123.5	155	142	166	161	164	158	**=**	**165**	165	158	
F90_3		167	138	124.4	158	146	165	161	163	158	**=**	**164**	165	159	
S1	135	357	300	279.0	332	313	352	342	344	336	**355**	**348**	351	341	120
S2	151	360	298	258.2	332	301	351	339	346	335	**356**	**349**	355	343	
S3	162	367	290	250.4	322	297	347	336	347	334	**361**	**349**	355	340	
R1	200	384	249	221.6	295	274	353	331	345	327	**355**	**346**	332	318	300
R2		383	262	219.4	302	277	351	334	345	327	**360**	**346**	337	324	
R3		385	250	220.8	299	274	344	331	340	323	**363**	**344**	339	323	
T0516	229	455	274	251.2	340	310	399	384	395	373	**423**	**402**	390	373	480
T0570	258	494	288	250.9	359	317	406	388	394	376	**421**	**404**	388	359	
T0563	279	?	359	315.5	428	390	494	474	482	459	**519**	**490**	491	461	

= denotes the lower bound of free energy is found. ? denotes unknown.

Table 1. Experimental results of GA$^+$, different GA variants, and the local search algorithm in [5]. Column *LB-FE* presents the lower bound of free energies.

[1] Peter Clote Lab: http://bioinformatics.bc.edu/clotelab/FCCproteinStructure/

[2] CASP website: http://predictioncenter.org/casp9/targetlist.cgi

The main goal of the experiment is to compare the result of our final algorithm GA$^+$ with the state-of-the-art result obtained by LS [5,9]. However, to prove the effectiveness of our new enhancement techniques, we implemented a baseline genetic algorithm denoted by BGA. In BGA, we select one operator for each generation based on the given probability distribution. However, like other typical genetic algorithms, we select parents in BGA by using a *Roulette Wheel* based on the quality of conformations in the current population. Also, we generate only one (for mutations) or two (for crossovers) child conformations from each application of the genetic operators. The points on the conformations, to apply the operators to, are selected randomly as well. Further, inspired by the results of twin-removal in [13], we discard duplicate solutions from new generations. Notice that BGA does not use any of the macro-move, random-walk, and exhaustive generation approaches.

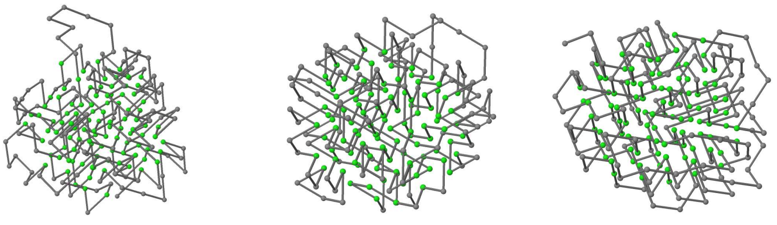
We ran experiments with our algorithm denoted by GA$^+$ and three of its other variants. These variants and BGA all are implemented in Java2 programming language. Nevertheless, these variants allow us to investigate the effect of each new aspect of GA$^+$ individually and in a combined way. RGA adds exhaustive generation in the genetic operators (as described in Sect. 4) to BGA. Variants WGA and MGA respectively add random-walk and macro-move methods to RGA. Thus, variants WGA and MGA respectively exclude macro-move and random-walk methods from GA$^+$ while RGA excludes both methods. For the time being, we do not consider other possible combinations that include adding macro-move or random-walk, or their combinations to the BGA.

We also ran the local search algorithm in [5], which is developed in COMET [11]. This algorithm [5] helps us compare our results with the state-of-the-art results of PSP on FCC lattice and HP energy model. We tried to run the algorithm in [9], but unfortunately for most of the proteins, the program aborted on exhausting the memory available. Any effective comparison in this case is therefore not possible.

We ran the experiments on the NICTA[3] cluster. The cluster consists of a number of identical Dell PowerEdge R415 computers, each equipped with 2 x AMD 6-Core Opteron 4184 processors, 2.8GHz clock speed, 3M L2/6M L3 Cache, 64 GB memory and running Rocks OS (a Linux variant for cluster). For each protein, we ran each algorithm 50 times with a time limit specified in Table 1 Column T. In the same table Columns Best and Avg, we report the best and average energy values obtained over 50 runs. Due to space limits, only the magnitudes of the energy values (ignoring the minus signs in all) are shown. Therefore, the larger the number in the table, the better the performance.

From the results in Table 1, we see that the energy values obtained in small proteins by all algorithms are close to the lower bounds. For better comparison, we therefore consider the large proteins, where our RGA significantly outperforms BGA. The differences between RGA and BGA are in the application of the genetic operators. In BGA, genetic operators generate one (for mutations) or two (for crossovers) random conformations. In RGA, an exhaustive genera-

[3] NICTA website: www.nicta.com.au



a) by LS, E=-333 b) by GA$^+$, E=-355 c) LB-FE=-384

Fig. 7. 3D structures of Protein R1 obtained by a) LS and b) GA$^+$, and c) the structure with the lower bound free energy.

tion is used and the best children are returned. These results clearly show the effectiveness of the exhaustive generation.

The results in Table 1 also show that WGA and MGA clearly outperform RGA. These results indicate the importance of our random-walk based stagnation recovery approach and the macro-move operator. Notice that when WGA is compared with MGA, the former significantly outperforms the latter; which means the random walk alone is more effective than the macro-move. This further suggests that given an exhaustive generation approach accompanied by a greedy best child selection method as in RGA, recovery from stagnations is more crucial than intensifying the search.

As noted before, GA$^+$ is the final version of our algorithm. GA$^+$ combines both macro-move and random-walk with RGA. Notice that GA$^+$, benefited from our all three techniques, clearly outperforms BGA, RGA, MGA and WGA. Nevertheless, we observe that the results obtained by GA$^+$ are better than that of LS with wide margins. LS is also outperformed by WGA, but it outperforms RGA and BGA; the results of LS and MGA are very close.

Relative Improvement: In Table 2, we present a comparison of (%) improvements in average conformation quality. We compare GA$^+$ (target) with BGA and LS (references). For each protein, the relative improvement of the target t w.r.t. reference r is $(E_t - E_r)/(E_{lb} - E_r) \times 100$; where E_t and E_r denote the average energy value achieved by t and r respectively, and E_{lb} is the lower

Relative improvements of *GA*$^+$ w.r.t. *BGA* and *LS*							
Protein info			**GA$^+$**	**BGA**		**LS**	
Seq	Size	LB-FE	Avg	Avg	RI	Avg	RI
H1	48	-69	-69	-58	100%	-66	100%
H2		-69	-69	-57	100%	-65	100%
H3		-72	-72	-57	100%	-66	100%
F90_1	91	-168	-166	-120	96%	-160	75%
F90_2		-168	-165	-124	93%	-158	70%
F90_3		-167	-164	-125	93%	-159	63%
S1	135	-357	-348	-279	88%	-341	44%
S2	151	-360	-349	-268	88%	-343	35%
S3	162	-367	-349	-250	85%	-340	33%
R1	200	-384	-346	-223	76%	-318	42%
R2		-383	-346	-219	77%	-324	37%
R3		-385	-344	-221	75%	-323	34%
T0516	229	-258	-402	-251	74%	-373	35%
T0570	258	-494	-404	-251	63%	-359	33%

Table 2. Relative improvements (RI columns) of GA$^+$ over *BGA* and *LS*. The values are calculated using the formula explained in *Relative Improvement* subsection. Column *LB-FE* presents the lower bound of free energies.

bounds of free energy for the protein in the HP model. We present the relative improvements only for the proteins having known lower bounds of free energy. Further, we show the best structures found by GA$^+$ and LS for protein R1 in Fig. 7; the figure also shows the structure of R1 with the lower bound free energy.

Search Progress: We compare the search progresses of different variants of GA and LS over time. Fig. 8 shows the average energy values obtained with times by the algorithms for Protein R1. We observe that MGA achieves very good progress initially, but almost becomes flat later on. WGA and LS perform equally initially but later WGA makes more progress than LS. GA$^+$ combines the positive aspects of MGA and WGA. Initially, it achieves the same progress as MGA does and later it is mostly benefited by random-walk as WGA is. The difference between performances of WGA and GA$^+$ roughly remains in the initial boosted progress made by the macro-move i.e. MGA.

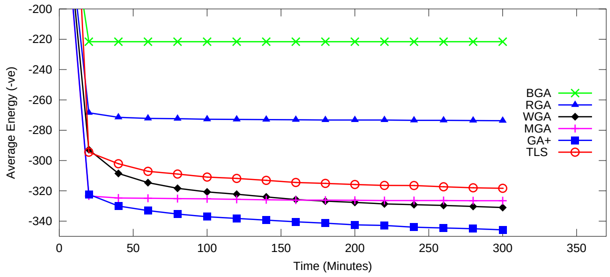


Fig. 8. Search progress of different approaches for Protein R1.

6 Conclusion and Future Work

In this paper, we presented five variants of genetic algorithms that individually and in a combined way use three different enhancement techniques: *i*) an exhaustive conformation generation approach; *ii*) a novel hydrophobic-core directed macro-move; and *iii*) a random-walk based stagnanation recovery technique. We compared our results with the state-of-the-art local search algorithm for simplified PSP. We found that our final algorithm GA$^+$ that use a combination of all the three enhancements significantly outperforms all current approaches of simplified PSP. In future, we intend to apply GA$^+$ in high resolution PSP.

Acknowledgments

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Berger, B., Leightont, T.: Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. Journal of Computational Biology 5(1), 27–40 (1998)

2. Blum, C.: Ant colony optimization: introduction and recent trends. Physics of Life reviews 2(4), 353–373 (2005)
3. Böckenhauer, H.J., Ullah, A.Z.M.D., Kapsokalivas, L., Steinhöfel, K.: A local move set for protein folding in triangular lattice models. In: WABI. Lecture Notes in Computer Science, vol. 5251, pp. 369–381. Springer (2008)
4. Bonneau, R., Baker, D.: *Ab initio* protein structure prediction: progress and prospects. Annual Review of Biophysics and Biomolecular Structure 30(1), 173–89 (2001)
5. Cebrián, M., Dotú, I., Van Hentenryck, P., Clote, P.: Protein structure prediction on the face centered cubic lattice by local search. In: Proceedings of the 23rd national conference on Artificial intelligence - Volume 1. pp. 241–246 (2008)
6. Cutello, V., Nicosia, G., Pavone, M., Timmis, J.: An immune algorithm for protein structure prediction on lattice models. IEEE Transaction on Evolutionary Computing 11(1), 101–117 (2007)
7. Dill, K.A.: Theory for the folding and stability of globular proteins. Biochemistry 24(6), 1501–1509 (1985)
8. Dobso, C.M.: Protein folding and misfolding. Nature 426(6968), 884–890 (2003)
9. Dotu, I., Cebrián, M., Van Hentenryck, P., Clote, P.: On lattice protein structure prediction revisited. IEEE Transactions on Comp. Bio. and Bioinformatics (2011)
10. Hales, T.: A proof of the kepler conjecture. The Annals of Mathematics 162(3), 1065–1185 (2005)
11. Hentenryck, P., Michel, L.: Constraint-based local search. The MIT Press (2009)
12. Hoque, M.T.: Genetic algorithm for *ab initio* protein structure prediction based on low resolution models. Ph.D. thesis, Gippsland School of Information Technology, Monash University, Australia (Sep 2007)
13. Hoque, M.T., Chetty, M., Lewis, A., Sattar, A.: Twin removal in genetic algorithms for protein structure prediction using low-resolution model. Transactions on Computational Biology and Bioinformatics 8(1), 234–245 (2011)
14. Hoque, M.T., Chetty, M., Sattar, A.: Protein folding prediction in 3D FCC HP lattice model using genetic algorithm. vol. 2007, pp. 4138–4145. IEEE Congress on Evolutionary Computation (2007)
15. Klau, G.W., Lesh, N., Marks, J., Mitzenmacher, M.: Human-guided tabu search. In: The Eighteenth National Conference on Artificial Intelligence (AAAI-02) (2002)
16. Lau, K.F., Dill, K.A.: A lattice statistical mechanics model of the conformational and sequence spaces of proteins. Macromolecules 22(10), 3986–3997 (1989)
17. Lee, J., Wu, S., Zhang, Y.: Ab initio protein structure prediction. From protein structure to function with bioinformatics pp. 3–25 (2009)
18. Lesh, N., Mitzenmacher, M., Whitesides, S.: A complete and effective move set for simplified protein folding. In: Research in comp. mol. biology (RECOMB) (2003)
19. Mann, M., Will, S., Backofen, R.: CPSP-tools – exact and complete algorithms for high-throughput 3D lattice protein studies. BMC Bioinformatics 9(1), 230 (2008)
20. Patton, A.L., Punch III, W.F., Goodman, E.D.: A standard GA approach to native protein conformation prediction. Int. Conf. on Genetic Algorithms (1995)
21. Rohl, C., Strauss, C., Misura, K., Baker, D.: Protein structure prediction using Rosetta. Methods in enzymology 383, 66–93 (2004)
22. Tantar, A.A., Melab, N., Talbi, E.G.: A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. Soft Computing-A Fusion of Foundations, Methodologies and Applications (2008)
23. Thachuk, C., Shmygelska, A., Hoos, H.H.: A replica exchange monte carlo algorithm for protein folding in the HP model. BMC bioinformatics 8(1), 342 (2007)
24. The Science Editorial: So much more to know. The Science 309(5731), 78–102 (July 2005)
25. Unger, R., Moult, J.: A genetic algorithm for 3D protein folding simulations. p. 581. Morgan Kaufmann Publishers, The 5th International Conference on Genetic Algorithms (1993)
26. Xia, Y., Huang, E.S., Levitt, M., Samudrala, R.: *Ab initio* construction of protein tertiary structures using a hierarchical approach. Journal of Mol. Biology (2008)
27. Yue, K., Dill, K.A.: Sequence-structure relationships in proteins and copolymers. Physical Review E 48(3), 2267 (1993)
28. Yue, K., Fiebig, K.M., Thomas, P.D., Chan, H.S., Shakhnovich, E.I., Dill, K.A.: A test of lattice protein folding algorithms. Proceedings of the National Academy of Sciences of the United States of America 92(1), 325 (1995)