Comparitive Study of Nature Inspired Algorithms on Protein Folding Problem

Anunay Sinha
CSE Dept.
AIACT&R
Delhi, India
anunay.sinha212@gmail.com

Sankalp Singh Gaharwar
CSE Dept.
AIACT&R
Delhi, India
sankalpsan007@gmail.com

Faraz Peerbaksh
CSE Dept.
AIACT&R
Delhi, India
fazkratos@gmail.com

***Abstract*—The protein folding problem involves predicting the shape of a protein based on the sequence of amino acids present in it. The shape of the protein is essential to predict the nature of the protein and the way it acts. This problem is a highly complex NP-Hard problem and hence meta-heuristics can prove vital in solving them. This paper compares the efficiency of two meta-heuristic algorithms: Particle Swarm Optimization and Bat Algorithm by implementing them on the Toy Model for protein folding to predict the shape of artificial sequences as well as real proteins. The bend angles and the protein energies are shown in a graphical manner for the two algorithms and compared. This study proves that Bat Algorithm is far more superior when compared to Particle Swarm Optimization for solving protein folding and further research can create a breakthrough in the protein folding problem.**

Keywords—Particle Swarm Optimization; Bat Algorithm; Protein Folding Problem; Toy Model; Meta-Heuristics

I. INTRODUCTION

A protein's function depends on its shape. And the shape of that protein is known as a fold. A protein chain comprises of amino acids and there are 20 such possible amino acids that can make up a protein. Experimentally the shape of a protein can be determined using X-Ray Crystallography or NMR-spectroscopy, but these methods are highly expensive and time consuming. Further, we also see that some proteins are hard to crystallize while spectroscopy only works on small proteins. Hence it becomes highly necessary to solve the protein folding problem (PFP) using computational methods. This has hence become one of the most essential problems in modern molecular biology.

But solving such problems to generate a 3D structure is not an easy feat and is a very complicated process. PFP still remains an open ended question in molecular biology though various methods have emerged through the years. Researchers have thus found ways to predict the shapes using simpler models. Models such as diffusion collision model and hydrophobic collapse model had been proposed in 1976 and 1984 respectively but they only supported single pathways for protein folding while it was experimentally established that different proteins took different pathways to form their shapes [1]. Hence PFP had largely been an experimental endeavor until the proposition of the energy landscape given by Joseph Bryngelson in 1995 [1]. This approach supported the "principle of minimum frustration" and is shown in Fig. 1.

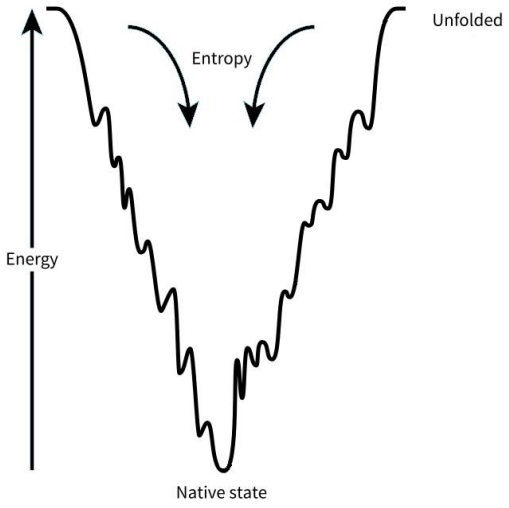

Fig.1 Energy Diagram

This paved the way for new models that were based around the energy of the proteins. Two of the most famous and highly simplified models are the AB Model and the Toy Model.

The AB Lattice Model grouped the 20 amino acids into two categories namely hydrophobic (H) i.e. non polar and hydrophilic (P) i.e. polar. The angle between two bonds was kept constant and the problem was to solve for the lattice conformation with the least energy for a reasonable definition of energy.

The Toy Model was invented by Stillinger in 1993 and it also considered only two types of amino acids hydrophobic (A) and hydrophilic (B). In the toy model however the angle was not fixed and the molecules could freely move around. This made the toy model resemble a real protein more prominently than other models and unlike the lattice model it had free angles which made it more realistic. [2]

Nature Inspired Algorithms (NIAs) are meta-heuristic methods which in the recent years have proved to be of use in a

wide variety of applications for optimization problems such as electric power systems.

Conventional methods which have been practice for a long time like linear and dynamic programming generally do not work in NP-Hard optimization cases with many variables as they often get stuck at local optima. Hence NIAs are essential in solving the protein folding problem.

Particle Swarm Optimization (PSO) is a meta-heuristic algorithm hat relies on the migratory nature of birds that follow the best bird along with the use of past memory to find the best solution. It was created by Kennedy and Eberhart in 1995 and relies on swarm behavior. It has been implemented in many fields in optimization. [3]

Bat Algorithm (BA) is a very recent meta-heuristic algorithm that relies on the echolocation behavior of bats to find obstacles and catch their prey. It was created by Xin-She Yang and published in 2010. [4]

In this paper we compare the implementation and effectiveness of PSO and BA on the Toy Model of protein folding. Section II describes the Toy Model in detail. Section III gives a brief description of PSO. Section IV details BA. Section V shows the experiments and results. The last section concludes the paper.

II. THE TOY MODEL

The toy model was created by Stillinger in 1993 [2]. It supports multiple pathways based energy funneling as it is based around minimizing the energy of the proteins. This model classifies the 20 Amino Acids as being one of the two possible residues namely Hydrophobic (A) and Hydrophilic (B). Fig. 2 shows the basic structure for the peptide chain. As seen from the figure we can deduce that for a chain of length 'n' we have 'n-2' bond angles between them. Ranging from θ_2 to θ_{n-1}, where $-\pi < \theta_i < \pi$. Positive angle implies a counter clockwise rotation and negative angles imply a clockwise rotation.

In this model only Intra-Molecular forces are considered and no forces caused by two separate proteins are taken into account. A protein sequence is represented by the array notation $[E_1 \ldots\ldots E_n]$ such that $E_i = 1$ implies A and $E_i = -1$ implies B. Then we have the energy given by equation (1): -

$$\varphi = \sum_{i=2}^{n-1} P_i(\theta_i) + \sum_{i=1}^{n-2} \sum_{j=i+2}^{2} P_2(l_{ij}, E_i, E_j) \qquad (1)$$

Where, the first term represents the backbone bend potential while the second term represents the non bonded interaction potential. The distance l_{ij} is given by equation (2):

$$l_{ij} = \left\{ \left[1 + \sum_{k=i+1}^{j-1} \cos\left[\sum_{l=i+1}^{k} \theta_i \right] \right]^2 + \left[\sum_{k=i+1}^{j-1} \sin\left[\sum_{l=i+1}^{k} \theta_i \right] \right]^2 \right\}^{1/2} \qquad (2)$$

While the backbone bend potential and non-bonded potential are calculated using the equation (3) and equation (4):

$$P_i(\theta_i) = \frac{1}{4}(1 - cos\theta_i) \qquad (3)$$

$$V_2(l_{ij}, E_i, E_j) = 4\left[l_{ij}^{-12} - C(E_i, E_j) l_{ij}^{-6}\right] \qquad (4)$$

Where,

$$C(E_i, E_j) = \frac{1}{8}\left(1 + E_i + E_j + 5E_iE_j\right) \qquad (5)$$

Hence, we strive to minimize the energy function by PSO and BA over different sequences. Appendix A. depicts the MATLAB code for the energy function.

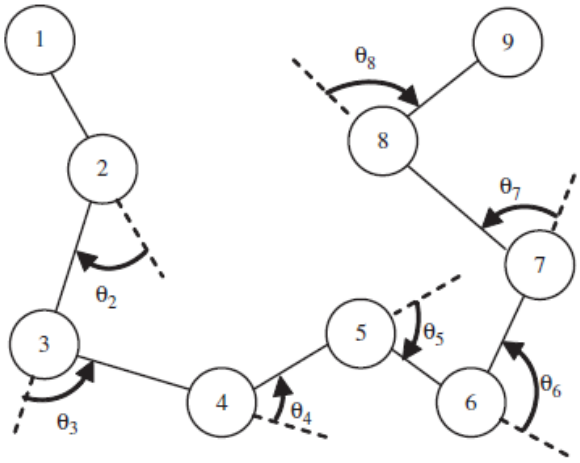

Fig. 2 A Schematic diagram of a 9-mer

III. PARTICLE SWARM OPTIMIZATION

PSO was developed by Eberhart and Kennedy in 1995. It is a meta-heuristic algorithm that tries to use the nature of migrating animals such as a group of birds that are trying to find their destination. This algorithm itself is simple yet very powerful, it has been applied various fields and many researchers have created variations of it. [3]

A. Behavior of Swarm & Standard PSO

In PSO each individual solution is a 'bird' among an entire flock of the bird population and is called a 'particle'. The steps involved in finding the optimized solution are as follows:

- The bird population is first initialized and each bird or particle proceeds to move in a random direction with a particular velocity.
- The fitness function is run on each particle to find the global best bird in the flock.
- The personal best from every particle's individual past and the global bestfrom the entire flock are taken into consideration to find the new velocity of each particle.
- The particles are moved to their new position and the steps are repeated till solution is reached.

Thus PSO uses both the local optima along with global optima to reach the solution. It combines private thinking by considering the personal past best and social collaboration by following the best particle to reach the solution.

This was a revolutionary approach since it avoids getting stuck in a local optima. The following sections show the formulation of velocity and discusses the algorithm in detail to help students in implementing this algorithm.

B. Formulation for the Algorithm

The following diagram shows how an individual particle moves by taking into consideration its past best along with the global best:

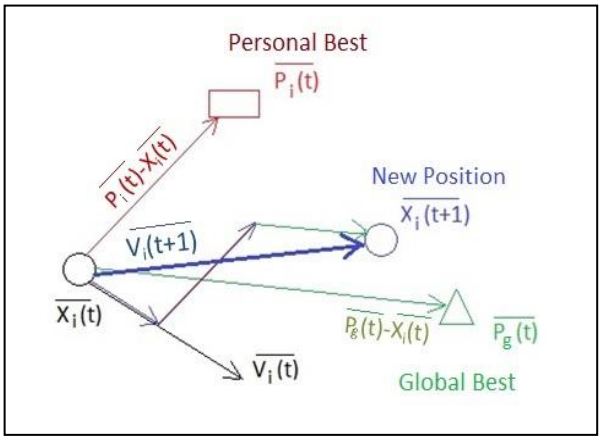

Fig. 3 Vector Diagram of a particle for PSO

Fig. 3 shows the movement of a single particle in a swarm of birds. Let, N be the number of random particles initialized. X_i is the current position of the i^{th} particle, P_i is calculated by finding personal best position reached in previous iterations, V_i is the bird's velocity and t represents the t^{th} cycle.

The i^{th} particle is represented in N-Dimensional space by a point where N is the number of variables. Hence we have:

$$X_i(t) = (x_{i1}, x_{i2}, x_{i3} \dots \dots \dots \dots \dots \dots x_{in})$$
$$P_i(t) = (p_{i1}, p_{i2}, p_{i3} \dots \dots \dots \dots \dots \dots p_{in})$$
$$V_i(t) = (v_{i1}, v_{i2}, v_{i3} \dots \dots \dots \dots \dots \dots v_{in})$$

Let the g^{th} particle be the best fitness particle in each cycle and hence its position is given by P_g. Consequently each particle updates its position using a newly calculated velocity. From Fig.2 we can easily deduce the following [3]:

$$V_i(t) = \mu \times V_i(t) + k_1 \times r() \times \left(P_i(t) - X_i(t)\right) + k_2 \times R() \times \left(P_g(t) - X_i(t)\right) \quad (6)$$

where, $-V_{max} <= V_i <= V_{max}$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (7)$$

Where, the constants k_1 & k_2 are known as learning factors such that $k_1, k_2 > 0$, rand ( ) and Rand ( ) are two methods that produce random number between [0, 1], V_{max} is the maximum possible velocity [3], μ is called the inertia weight which was proposed as an improvement to the PSO by Yang and Eberhart [5] to limit the effect of the previous velocities on the current velocity. Such that ω decreases linearly which balances out global search and local search components. Hence the global search begins with a larger weight and gradually reduces with time to increase the weight of local search. The second term in equation (1) denotes the cognitive part i.e. private thinking and the third term in equation (1) specifies the social collaboration globally.

The main parameters that affect the optimization are:

- Population Size (N)
- Number of Cycles (t)
- Max change of velocity (v)
- Inertia weight (ω)
- Dimensions (n)

For Protein folding we have 'n' equal to the size of the protein. The population size is fixed at 20 while maximum cycles were kept at 1000. Appendix B. Gives the MATLAB code for PFP using PSO.

IV. BAT ALGORITHM

Bat algorithm (BA) is an algorithm that tries to imitate the way bats find their prey using echolocation. It was created by Yang in 2010 [4].

A. Behavior of Bats & Standard BA

This algorithm tries to use the behavior of bats trying to find their prey using echolocation which acts sonar. Bats emit sound waves which get reflected or bounced off the prey and bats then have to estimate the distance and speed at which the prey is moving. This concept is further expanded to fit into optimization problems in a way similar to PSO except here the parameters are not fixed and in fact keep on changing. The parameters involved are r_i the pulse rate which lies between

[0,1] and A_i which is the loudness of the wave sent by the bat to catch the prey and has max and min constraints on it.

B. Key Features

- Frequency Tuning: In BA frequencies are slowly varied like in other swarm based algorithms. Hence BA has same advantages of other similar swarm based methods.
- Automatic Zooming: This involves zooming into a particular area where a promising solution is present. Hence it automatically switches a explorative mode to a more local mode and hence has a really fast convergance rate.
- Parameter Control: Unlike other NIAs, here the parameters are not fixed i.e. they are varied as the iteration moves forward so we can easily switch from a more global approach to a more selective local approach

C. Formulation for Algorithm

The equation (8), equation (9) and equation (10) depict the formulae that govern the frequency which the bat emits and the velocity & position of the bat in a fashion similar to PSO.

$$f_i = f_{min} + (f_{max} - f_{min}) \times k \quad (8)$$

$$v_i(t+1) = v_i(t) + \left(x_i(t) - x_g(t)\right) \times f_i \quad (9)$$

$$x_i(t+1) = x_i(t) + v_i(t) \quad (10)$$

where, f_i is the frequency being emitted by the i^{th} particle, f_{max} and f_{min} are the limits, k is a random vector between [0,1], v_i is velocity of the i^{th} particle, x_i is position and x_g is the globally best particle. The algorithm is given in appendix B and the flowchart for the algorithm is shown in Fig. 4. And the value for Amplitude was taken constant [4].

V. EXPERIMENTS AND RESULTS

In this section we first fold Artificial Sequences of Protein and compare the energies to see if the algorithm is capable of stabilizing the artificial sequences. After which we try to find the shapes of 4 real protein sequences.

A. Protein Folding on Artificial Sequences

The parameters we used involved 1000 generations/ iterations for a population size of L=30. Each Sequence of Protein is conformed individually and we compare the energies to see if the algorithm is capable of stabilizing the artificial sequences [6] [7] [8]. We found out that the alpha helix structure and the beta sheet structure were formed for the cases AABABB and AAABAA respectively as shown in Fig. 4(a) and Fig. 4(b). Table 1 shows the energy values and the time consumed using PSO and BA on the Toy Model.

TABLE I. ARTIFICIAL SEQUENCES

S. n.	Sequence	Energy Value Φ		Time (s)	
		PSO	*BA*	PSO	BA
1	AAA	-0.6582	−0.6582	2.440	1.531
2	AAB	0.0322	0.0322	3.098	1.264
3	ABA	-0.6582	-0.6582	2.592	1.142
4	ABB	0.0322	0.032227	3.604	1.172
5	BAB	-0.0303	-0.0302	3.339	1.257
6	BBB	-0.0302	-0.0302	2.563	1.200
7	AAAA	-1.6762	-1.6763	4.963	2.226
8	AAAB	-0.58527	-0.5852	3.868	2.201
9	AABA	-1.4510	-1.4510	4.104	2.838
10	AABB	0.0672	0.0672	3.829	2.269
11	ABAB	-0.6493	-0.6493	5.884	2.676
12	ABBA	-0.0361	-0.0361	4.197	2.623
13	ABBB	0.0047	0.0047	4.021	2.176
14	BAAB	0.0617	0.0617	3.973	2.242
15	BABB	-0.0007	-0.0007	4.286	2.230
16	BBBB	-0.1395	-0.1397	5.639	2.996
17	AAAAA	-2.8410	-2.4664	6.773	4.468
18	AAAAB	-1.5894	-1.5894	5.995	5.555
19	AAABA	-2.4449	-2.4449	7.488	4.254
20	AABAA	-2.5317	-2.5317	7.357	5.250
21	AABAB	-1.3431	-1.3477	6.996	4.694
22	AABBA	-0.9160	-0.9266	7.827	4.631
23	AABBB	0.0401	0.0401	8.276	3.838
24	ABAAB	-2.4449	-2.4449	8.248	5.430
25	ABABA	-2.2106	-2.2146	7.921	4.286
26	ABABB	-0.6168	-0.6168	7.009	4.862
27	ABBAB	0.0264	0.0264	8.174	3.682
28	ABBBA	-0.3949	-0.3980	6.404	3.737
29	ABBBB	-0.0638	-0.0278	7.370	5.336
30	BAAAB	-0.5210	-0.5210	5.572	3.749
31	BAABB	0.0962	0.0962	5.671	3.804
32	BABAB	-0.6480	-0.6480	7.603	4.501
33	BABBB	-0.1826	-0.1826	5.575	3.756
34	BBABB	-0.2245	-0.2402	5.531	3.774
35	BBBBB	-0.4483	-0.4526	5.755	3.832

We can deduce from Table 1 that although both the algorithms produce almost identically stable peptides they still take different amounts of time to do the same task [9]. Bat Algorithm takes almost half the time it takes for PSO in most cases. Fig. 4(a) & 4(b) show the structural representation of two artificial peptide chains and the angles at which we have the bends as predicted by our implementation.

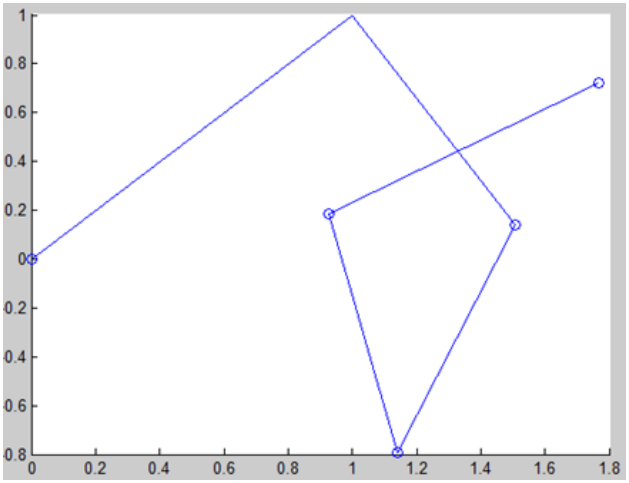

Fig. 4 (a) Alpha Helix Structure shape for AABABB
Angles: [1.0449 1.9496 -1.7886 -8.4053]

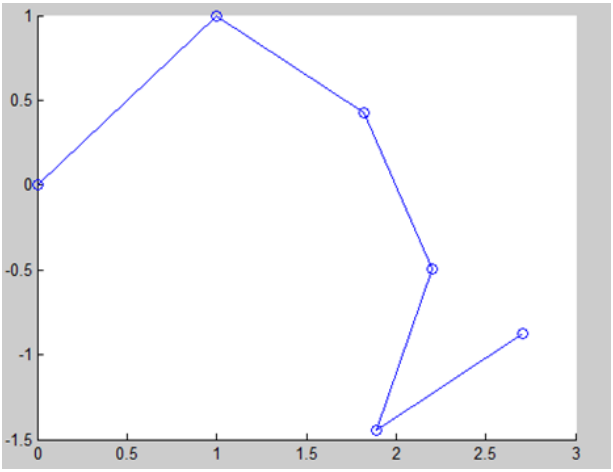

Fig. 4 (b) Beta Sheet Structure shape for AAABAA
Angles: [162.7518 30.2403 -14.4576 0.6102]

B. Protein Folding on Real Protein Sequences

The real protein taken into account is 1AGT. In this experiment we use the KD Method to distinguish the proteins into hydrophobic and hydrophilic.

Briefly speaking, amino acids I, V, L, P, C, M, A, G are hydrophobic and D, E, F, H, K, N, Q, R, S, T, W, Y are polar.

1AGT: As per the information from PDB Portal we have 38 amino acids in 1AGT.

GVPINVSCTG SPQCIKPCKD QGMRFGKCMN RKCHCTPK

TABLE II. REAL PROTEIN 1 AGT

S.No.	Sequence	Energy Value Φ		Time (s)	
		PSO	*BA*	PSO	BA
1	1AGT	-19.050	-19.457	406.347	280.234

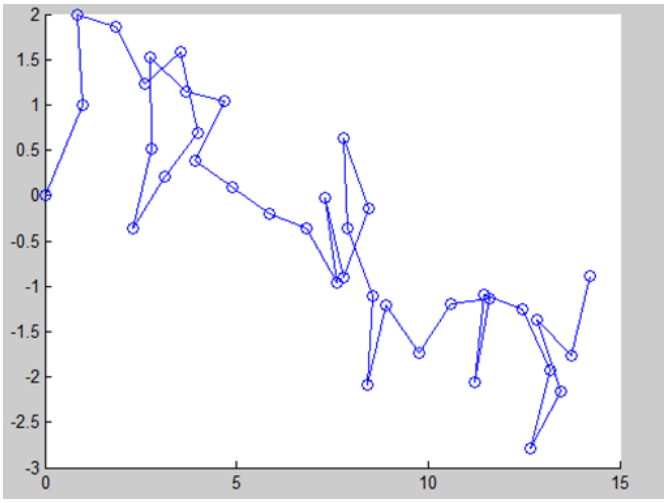

Fig. 5 1 AGT Structure Prediction

Angles: [-4.55178526963153 -0.123526045777103 -25.8199534387078
6.63920975367892 -1.09293535796865 -2.63555371581972
3.74878863837581 57.6400525371456 32.9924193188726
-6.66763047268972 -0.106795394319625 -197.196926870800
-75.6952126453075 -0.284339759593891 -0.172792665390074
5.64269275758630 -42.0692023023288 -1.06207637972407
-5.42386024192861 153.060932177181 -1.47567465595182
-0.831207533329612 551.206178300402 19.9222429826848
-0.542638835104338 19.4191739529587 0.0524139208320072
35.7342178236110 560.532248849488 251.165887391045
24.3935426840814 10.4612385657602 0.684007707130321
109.058749615307 3348.53644463614 -42.9139370357563]

Fig. 5 shows the structure of real protein 1 AGT along with the various angles for the bends as predicted by our implementation. The above results show that bat algorithm is superior to PSO even for conforming real proteins. As on comparing with PDB resource we find the angles and shapes resemble. Appendix C. depicts the plotting function we built to plot the shapes.

CONCLUSION

In this paper we were able to prove that Bat Algorithm is much faster when compared to PSO while conforming artificial sequences as well as real proteins. It was found that BA was almost twice as fast as PSO. Hence further research needs to take place in improving the bat algorithm for the protein folding problem.

Further it should also be noted though that even though the Toy Model was able to produce almost identical structures and energy levels to real proteins it is still a simplified 2D version and there is a future scope in improving this algorithm and the protein folding problem as it can provide fast solutions to complex NP-Hard Problems.

Appendix A. Energy Function

```
function phi=energy(x)
n=5;
E = [1 1 1 1 -1];
y=zeros(1,n);
y(1)=0;
y(n)=0;
for i=2:n-1
y(i)=x(i-1);
end
a=0;b=0;
function c=C(i,j)
c=(1/8)*(1+E(i)+E(j)+(5*E(i)*E(j)));
end
function f0 = d(i,j)
t=0;
for k=i+1:j-1
t=t+cos(Q);
end
t=(t+1)^2;
s=0;
for k=i+1:j-1
s=s+sin(Q);
end
s=(s)^2;
f0=(t+s)^(1/2);
end
function f1 = V1(y)
f1=(1/4)*(1-cos(y));
end
function f2 = V2(i,j)
f2=4*((d(i,j)^(-12))-C(i,j)*(d(i,j)^(-6)));
end
for i = 2:n-1
a=a+V1(y(i));
end
for i = 1:n-2
for j = i+2:n
b=b+V2(i,j);
end
end
phi=a+b;
end
```

Appendix B. PSO Code

```
CostFunction = @(x) energy(x);
nVar = <Number of bend angles>;
VarSize = [1 nVar];
VarMin = -pi;
VarMax =pi;
MaxIt = 1000;
nPop = 80;
c1=2.8;
c2=1.3;
w=1;
wdamp=0.99;
empty_particle.Position = [];
empty_particle.Velocity = [];
empty_particle.Cost = [];
empty_particle.Best.Position = [];
empty_particle.Best.Cost = [];
particle = repmat (empty_particle, nPop, 1);
GlobalBest.Cost = inf;
fori=1:nPop
particle(i).Position=unifrnd(VarMin,VarMax,VarSize);
particle(i).Best.Cost = particle(i).Cost;
if particle(i).Best.Cost<GlobalBest.Cost
GlobalBest = particle(i).Best;
end
end
BestCosts = zeros(MaxIt,1)
for it=1:MaxIt
fori=1:nPop
particle(i).Velocity=w*particle(i).Velocity+c1*rand(
VarSize).*(particle(i).Best.Position-
particle(i).Position)+c2*rand(VarSize).*(GlobalBest.
Position-particle(i).Position);
particle(i).Position=particle(i).Position          +
particle(i).Velocity;
particle(i).Cost=CostFunction(particle(i).Position);
if particle(i).Cost<particle(i).Best.Cost
particle(i).Best.Position=particle(i).Position;
particle(i).Best.Cost=particle(i).Cost;
if particle(i).Best.Cost<GlobalBest.Cost
GlobalBest = particle(i).Best;
end
end
end
BestCosts(it) = GlobalBest.Cost;
w = w * wdamp;
end
```

Appendix C. Plotting Function

```
px=zeros(1,nVar+2);
py=zeros(1,nVar+2);
px(1)=0;
py(1)=0;
px(2)=1;
py(2)=1;
hold on;
line([px(1),px(2)],[py(1) py(2)]);
plot(px(1),py(1),'O');
plot(px(2),py(2),'O');
for i=3:nVar+2
    px(i)=px(i-1)+cos(GlobalBest.Position(i-2));
    py(i)=py(i-1)+sin(GlobalBest.Position(i-2));
    line([px(i-1),px(i)],[py(i-1),py(i)]);
    plot(px(i),py(i),'O');
end
```

REFERENCES

[1] Ahluwalia, U., Katyal, N., & Deep, S. (2013). Models of protein folding. *Journal of Proteins & Proteomics*, *3*(2).

[2] Stillinger, F. H., Head-Gordon, T., & Hirshfeld, C. L. (1993). Toy model for protein folding. *Physical review E*, *48*(2), 1469.

[3] Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on* (pp. 39-43). IEEE.

[4] Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. *Nature inspired cooperative strategies for optimization (NICSO 2010)*, 65-74.

[5] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on* (pp. 69-73). IEEE.

[6] Liu, J., Wang, L., He, L., & Shi, F. (2005, August). Analysis of toy model for protein folding based on particle swarm optimization algorithm. In *International Conference on Natural Computation* (pp. 636-645). Springer Berlin Heidelberg.

[7] Cai, X., Kang, Q., Wang, L., & Wu, Q. (2014). Bat Algorithm for Toy Model of Protein Folding. *Journal of Computational and Theoretical Nanoscience*, *11*(6), 1569-1572.

[8] Malhotra, R., & Khari, M. (2014). Test suite optimization using mutated artificial bee colony. In *Proc. of Int. Conf. on Advances in Communication, Network, and Computing, CNC, Elsevier* (pp. 45-54).

[9] Khari, M., & Kumar, P. (2016, March). A Novel Approach for Software Test Data Generation using Cuckoo Algorithm. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies* (p. 98). ACM.