Project 2: A Generalized Thomson Problem Solved by Genetic Algorithm

Gengpu Li[1] [*] **and Yuanye Lin**[1] [*]

[1]Zhiyuan College, Physics, Shanghai Jiaotong University, Shanghai, CN
[*]These authors contributed equally.

,

Abstract

In this project, we determine the minimum electrostatic potential energy configuration of N electrons confined on the surface of a unit sphere with the electrostatic interaction energy given by the special potential $U = -\log(r)$. The hardest part is how to obtain the global minima. The classical methods including the Newton's method, the conjugate gradient method have a common problem that they may lead to a local minima. We overcome it successfully by employing the Genetic Algorithm(GA) and implementing the algorithm by MATLAB. Further, our result shows that there is no distinction between Coulomb's potential.

Thomson Problem| Genetic Algorithm| Thomson Problem
Correspondence: ***ligengpu_lim@sjtu.edu.cn ythelyy@sjtu.edu.cn***

Main

2.1 Introduction.

The Thomson problem wants to determine the best configuration of N electrons confined on the surface of a unit sphere with the Coulomb potential which has been determined by the Coulomb's law: $U = \frac{1}{r}$. The problem can be simplified as a research of N body behavior and find the minimum energy configuration.

In Coulomb's case, the solution of Thomson problem have been learned. Conclusion comes that the solution has a high symmetry, which is satisfied with our physical intuition.

In our report we focused on the logarithmic potential: $U = -\log(r)$. And we suppose to get the geometrical configurations and optimized energy at situation $N \leq 30$ and compare our results with traditional Thomson problem.

The paper is structured as follows. We will give an introduction of our genetic inheritance arithmetic in Section 2.2. In Section 2.3, we will show our results at $N \leq 30$ and give the final morphology and minimum energy. In section 2.4, we will compare our results with Coulomb's case which has the very accurate results. Finally, we will give our gene simulation code at Supplement I.

2.2 Arithmetic.

In our simulation, we want to find the minimum energy and electrons distributions from genetic inheritance simulation.

First we have to get the total energy of the N body system. The distance between two points located at $\mathbf{r}_i$ and $\mathbf{r}_j$:

$$r_{ij} = |\mathbf{r}_i - \mathbf{r}_j|$$

In the spherical coordination, when r=1:

$$\begin{cases} x = \sin\theta\sin\phi \\ y = \sin\theta\cos\phi \\ z = \cos\theta \end{cases}$$

So that the distance between points i and j

$$r_{ij} = \sqrt{2 - 2(\sin\theta_i \sin\theta_j \cos(\phi_i - \phi_j) + \cos\theta_i \cos\theta_j)}$$

We set the potential coefficient as $k_e = 1$

$$U_{ij}(N) = -\log(r_{ij})$$

And the total energy

$$U(N) = \sum_{i<j} -\log(r_{ij}) \tag{1}$$

The equation Eq. (1) includes 2N-2 parameters and the traditional algorithm including Newton's method and conjugate gradient may lead to a local minima in such a high dimensional optimization problem. To overcome this problem, we introduce the Genetic Algorithm(GA). GA is a metaheuristic inspired by the process of natural selection(1). Compared to another optimization method like simulated annealing, the GA can record the best solution and more interpretable.

The GA consider an ensemble of "possible solutions" which is analogous to the population in biological.A typical genetic algorithm requires(1):

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

In our problem, we choose the configuration of electrons as the genotype of an individual. To simulate the natural selection, during the evolution of population, we find a subtle fitness function to test the "adaptability". The solutions which performs badly are eliminated, on the contrary, the best ones are left in the next generation. Meanwhile, the solutions may "copulate" with each other and changed incidentally to enrich the "genomic library" of population which is a subtle simulation of gene exchange and mutation.

In detail, we random initialized the population. The size of ensemble is chosen as 100 (this is a hyperparameter).

$$population = c_1, c_2, \ldots, c_{100} \tag{2}$$

The first generation begin evolution with calculating the energy of each individual and we sort the ensemble according to the energy of individuals. The best five are introduced into the next generation.

$$ranked\ population = c_{i_1}, c_{i_2}, \ldots, c_{i_{100}} \tag{3}$$

$$ranked\ energy = e_{i_1}, e_{i_2}, \ldots, e_{i_{100}} \tag{4}$$

The fitness function is chosen as

$$f_{i_j} = \exp(-e_{i_j})/(1+\exp(-e_{i_{|101-j|}})) \tag{5}$$

which crosses the smaller and bigger energy. This function has three advantages.

1. This function is positive definite and decreasing which correspond to our conceptions–the fitness is a positive number and a lower energy is what we expect.
2. The range of this function is [0, 1]. This range is friendly for computers to represent the float numbers.
3. The fitness only depends on the relative size of energy since the better one and the worse one cross in the function. So a relative small energy may have a bigger fitness even though it may not be the best.

Next, we use roulette to choose the father and mother of next generation and the copulation probability is proportional to the fitness. And we define the son as the average number of parents.

The mutation is the core progress in GA, without which the algorithm can never converge. We choose the mutation rate as 15%. Each time we just change a site (id. est. moving an electron).

The next generation repeat the before until we reach the max generation. Here we use a simple flow chart Figure 1 to illustrate our algorithm.

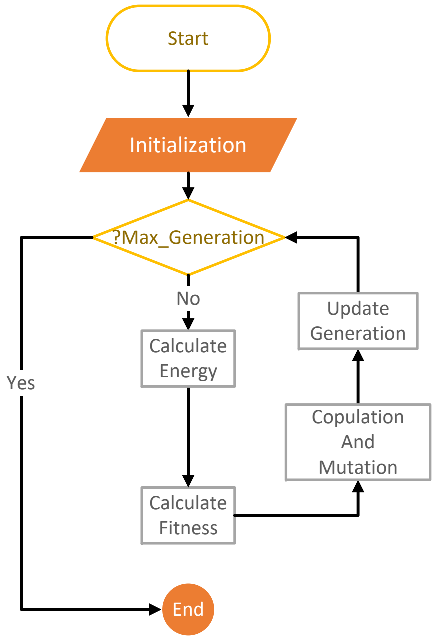


Fig. 1. The GA algorithm

The code is attached in Appendix A. Meanwhile, you can download the programme on https://github.com/joeyli99/project2_Thomson_Problem.

2.3 Result.

First, we want to check whether our algorithm available, that is, we want test the convergence. We illustrate the evolution progress in N=30

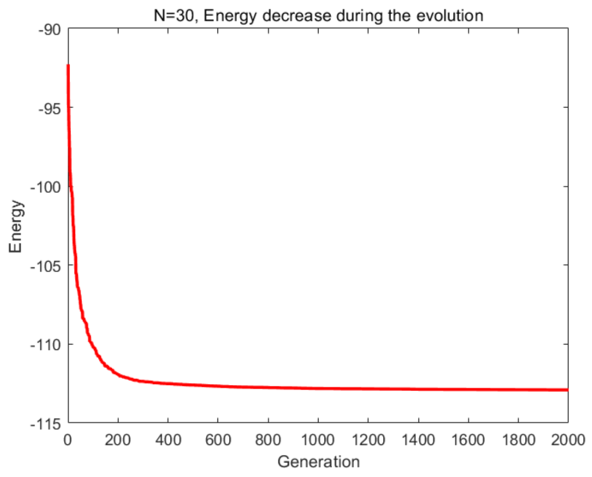


Fig. 2. The Energy decrease during the evolution

Figure 2 shows that the energy decreases rapidly before 200 generation but approaches to the real global minima slowly after 200 generation.

Then we set our number of electrons N from 2 to 30, and we can get the final minimum energy of the system.

N	1	2	3	4	5
E	None	-0.693	-1.648	-2.942	-4.421
N	6	7	8	9	10
E	-6.238	-8.182	-10.428	-12.888	-15.563
N	11	12	13	14	15
E	-18.42	-21.606	-24.866	-28.405	-32.147
N	16	17	18	19	20
E	-36.106	-40.272	-44.648	-49.195	-54.011
N	21	22	23	24	25
E	-58.999	-64.205	-69.568	-75.213	-80.996
N	26	27	28	29	30
E	-87.007	-93.242	-99.655	-106.251	-113.088

Table 1. Minimum Energy at different N

Table 2 shows the minimum energy of the system at different N, We get the minimum energy function of logarithm potential.

Also we can get the configuration of every N, as we exhibit:

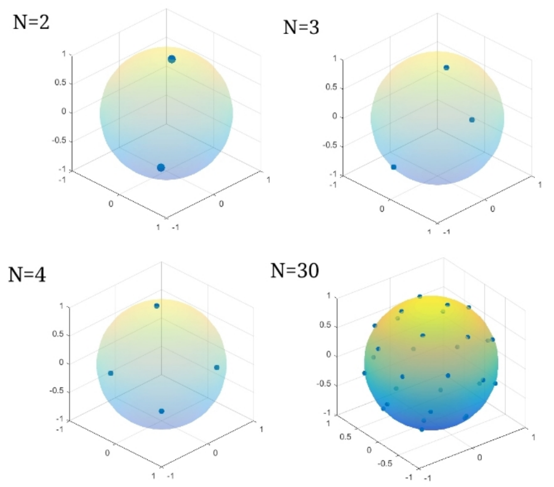


Fig. 3. The configurations of electron number 2, 3, 4 and 30

From these four special situation, we can see the final stable configuration is in high symmetry. And we can calculate the distance matrix D, whose element d_{ij} represent the distance between i^{th} and j^{th} points.

1. N=2: the distance matrix is
$$D_2 = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix} \tag{6}$$
From D_2 we see the two points stand as a digon
2. N=3: the distance matrix is
$$D_3 = \begin{pmatrix} 0 & 1.7321 & 1.7320 \\ 1.7321 & 0 & 1.7320 \\ 1.7320 & 1.7320 & 0 \end{pmatrix} \tag{7}$$
From D_3 we see these three points stand as a triangle
3. N=4: the distance matrix is
$$D_4 = \begin{pmatrix} 0 & 1.6328 & 1.6330 & 1.6333 \\ 1.6328 & 0 & 1.6330 & 1.6330 \\ 1.6330 & 1.6330 & 0 & 1.6328 \\ 1.6333 & 1.6330 & 1.6328 & 0 \end{pmatrix} \tag{8}$$
From D_3 we see these four points form a tetrahedron

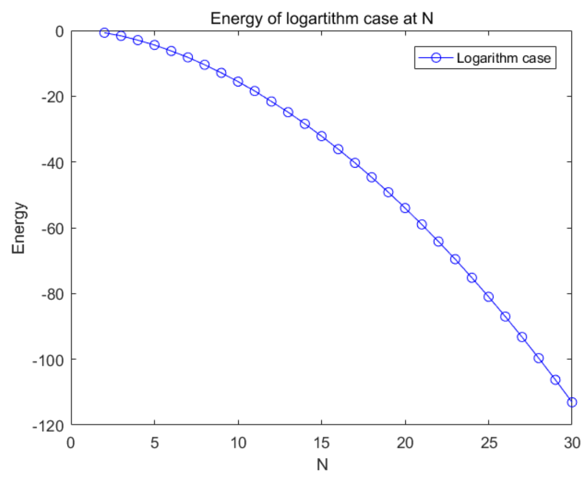


Fig. 4. The Minimum Energy function

Next we want to show how the minimum energy changes at different N, here we have the minimum energy function:

Figure 4 shows that the minimum energy decrease with the number of electrons increase. The result is not a surprise, because as N increase, the area density will increase, which means that distance between the electrons become smaller. So both the E and first derivative are decreasing function.

2.4 Comparison.

From Figure 3 and the analysis from D_2, D_3 and D_4, we see the configuration in logarithm situation is the same with Coulomb's case when N=2,3,4; So we get the assumption that: the configuration in these two different potential case is the same.

And the next step, we want to confirm our assumption:

1. At the logarithm case, record the configuration of the electron in every N, including N θ and ϕ
2. Calculate the distance between every pair of electron
3. Assume these particles are in Coulomb's law and calculate potential from the distance we get, so that we can get the total Energy E_N, the data are cited from Wikipedia(3).
4. Compare our energy we get E_N with the solution of Thomson Problem in Coulomb's case of N, U_N; If E_N=U_N, we can say that our assumption is right!

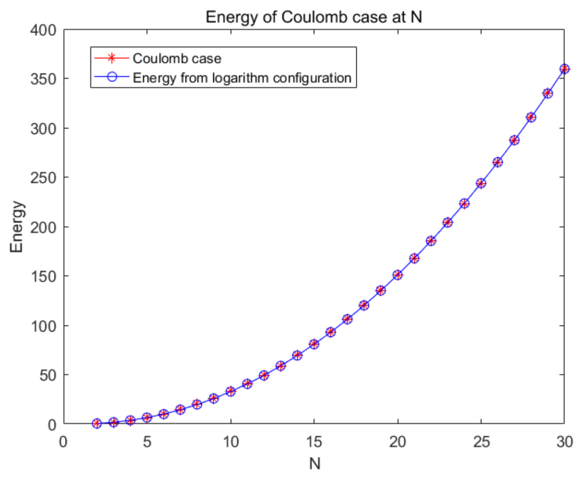


Fig. 5. Comparison between E_N and U_N

From Figure 5, we can see clearly that the Coulomb Potential we get form logarithm's configuration(Circle Points) fit well with the Potential from Coulomb's configuration(Star Points), which suggest these two configuration is actually the same!

2.5 Evaluation.

In our model, there is two important hyperparameters: the size of population and the mutation rate. In this section we will discuss the effect of it. The conclusion is: the small population can converge quicker than the bigger, but it may lead to a local minima.

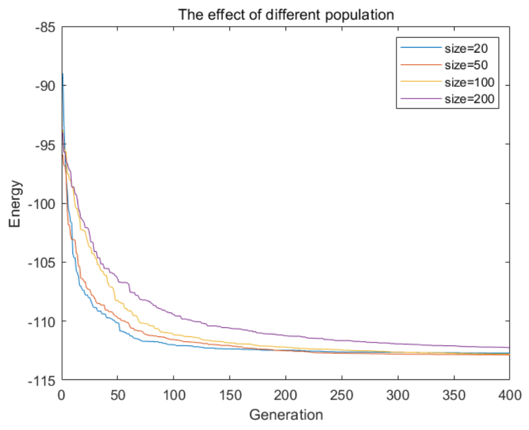


Fig. 6. The effect of population.

We can clearly see that the small size quickly converge. Meanwhile, the speed of computation is also optimistic in small size. So, why do not us choose the small size? The small size may lead to a local minima. A observation suggests that after some generation all individuals in small size are nearly identical, which imply the risk of drop into a local minima. This phenomenon called premature convergence is a common problem found in genetic algorithms(2). So, weighing the advantages and disadvantages, we choose the 100 as the size of population.

We also test the mutation rate.

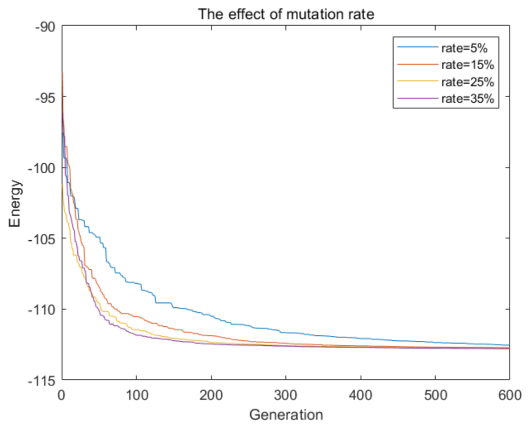


Fig. 7. The effect of mutation rate

We can observe that the higher mutation rate leads to a better convergence. However, in the real world, the 35% mutation rate is too large. So the choice of 15% can lead to a reasonable and high speed solution.

Another sensitive part is the fitness function, which is also active function or probability mapping function in another part of this paper. This function determine the natural section strengthen. If the low energy individuals are set a larger fitness, the solution will quickly converge.

Here we choose another form of fitness function(no-cross):

$$f_{ij} = \exp(-e_{ij})/(1+\exp(-e_{ij})) \quad \textbf{(9)}$$

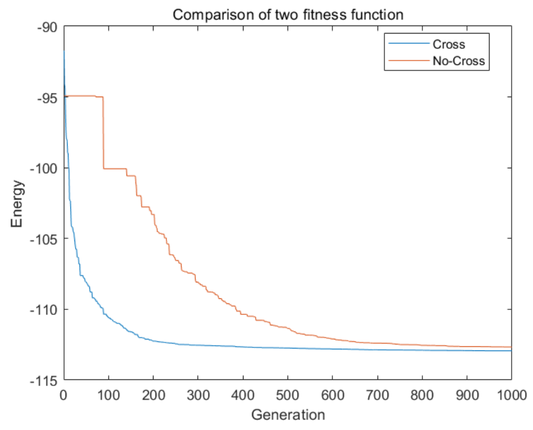


Fig. 8. The effect of fitness function

So, the original fitness function is better.

Conclusion

In this paper, we solve the generalized Thomson Problem using the gene simulation, and we compare the result with traditional Thomson Problem. And here is our conclusions:

1. We show the converge of our gene simulation, and point out the high speed and accuracy of this algorithm.
2. We raise up the particle number form 2 to 30 and get the minimum energy and configuration of N-electron system.
3. Minimum Energy function shows that the final stable energy decrease as the number of electron increase, which is as expected.
4. The configuration figure and distance matrix shows high symmetry of the final system, and we using the inversion method to show up that the configuration in logarithm and coulomb's case is actually the same.

It's a pity that we don't give out the mathematical explanation that why the stable configuration is the same at different Potential's situation. Meanwhile, we suppose that all forms of center force may have a same stable configuration. And another work can be done is to improve the algorithm form two aspect: first, we can find a better function to give a map from energy to probability; second, we can adjust the parameter, for example, the size or the mutation of the population to have a better converge speed. Meanwhile, our algorithm exists the "bad biodiversity" problem, which means at the end of the evolution, nearly all individuals have same or similar configurations(genetype). There are several methods to solve the premature convergence problem but we think our model is good enough for a physical problem.

ACKNOWLEDGEMENTS

We would like to express our thanks to the people who have helped us most throughout our project. We are grateful to our teacher Weidong Luo for support for the project. A special thank of ours goes to our classmates Tianli Liu and Yinan Chen who helped us out in completing the project, where they all exchanged their own ideas, thoughts and results made this possible to complete our project with all accurate information.

Reference

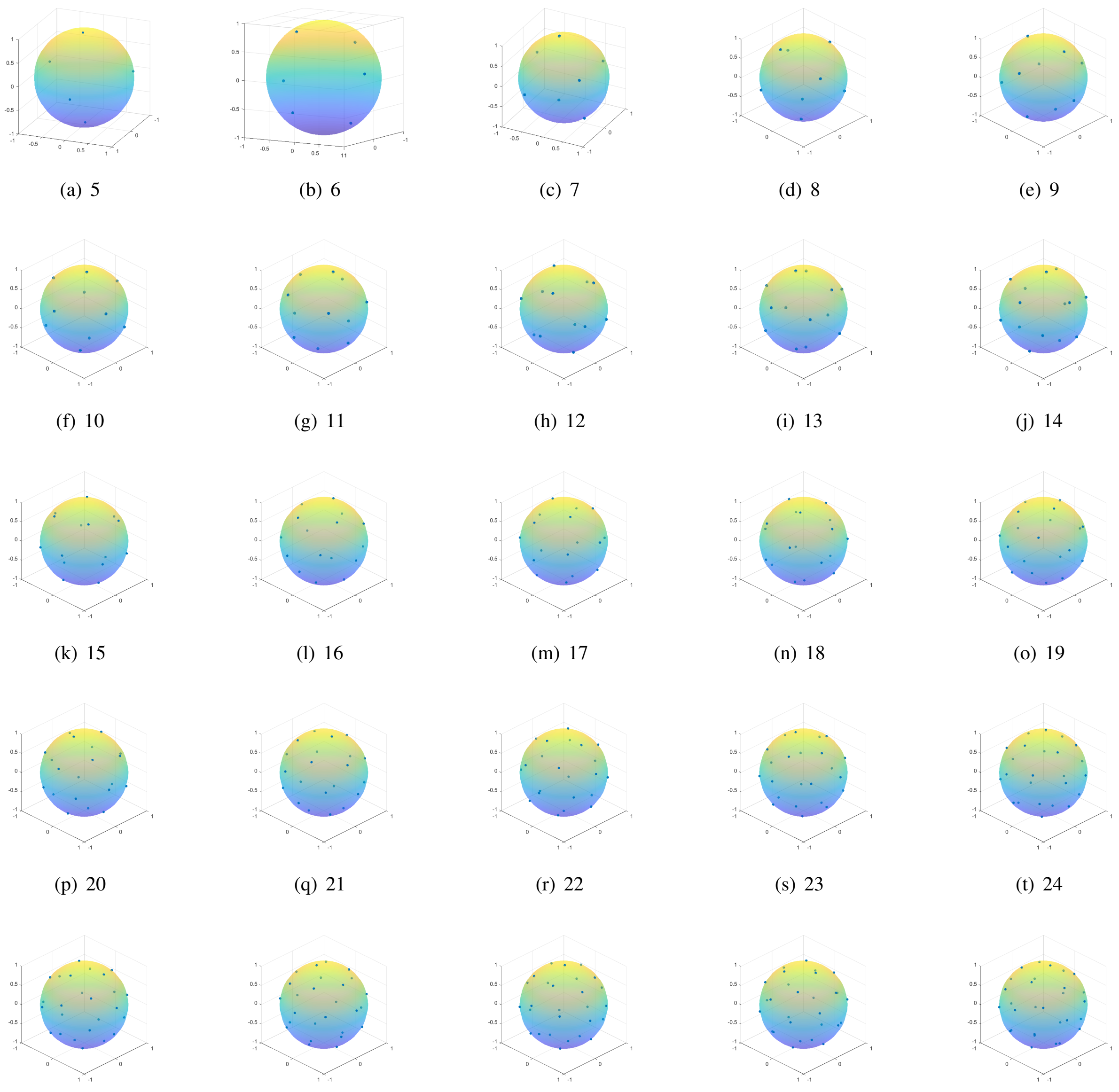
1. Wikipedia contributors. Genetic algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=896518366, 2019. [Online; accessed 11-May-2019].
2. Wikipedia contributors. Premature convergence — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Premature_convergence&oldid=889272836, 2019. [Online; accessed 11-May-2019].
3. Wikipedia contributors. Thomson problem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Thomson_problem&oldid=895951648, 2019. [Online; accessed 11-May-2019].

Supplementary Note 1: The MATLAB implement of Genetic Algorithm

The codes list below and data can also be download in Github https://github.com/joeyli99/project2_Thomson_Problem.

```
clc;
clear;
N = 3; %number of charges.
ensemble_size = 100; %The members in population.
max_generation = 10000; %The max generation
mutation = 15; %mutation members.

energy_gene=zeros(1,max_generation);
map_ensemble = zeros(N, 2, ensemble_size);
for i = 1:ensemble_size
  map_ensemble(:, :, i) = rand_map(N);
end
generation = 0;
while (generation < max_generation)
  energy_ensem = zeros(ensemble_size, 1);
  for i = 1 : ensemble_size
    energy_ensem(i) = energy(map_ensemble(:, 1, i), map_ensemble(:, 2, i));
  end
  [s, index] = sort(energy_ensem);
  energy_gene(generation+1)=s(1);
  disp([num2str(s(1)),' ', num2str(generation)])
  next_generation = zeros(N, 2, ensemble_size);
  next_generation(:, :, 1:5) = map_ensemble(:, :, index(1:5));
  %The best 5 join to the next generation a superparameter
  %index(end-4 : end) = []; %The worst 5 die.
  %s(end-4 : end)= [];

  s = exp(-s)/(1+exp(-s)); %An active function which is obtained by incident.
  s = s/sum(s); % we try to interpret the energy into the probability.
  for i = 1:ensemble_size
    f_prob = rand();
    m_prob = rand();
    prob_sum = 0;
    father = 0;
    mother = 0;
    for j = 1:ensemble_size
      prob_sum = prob_sum+s(j); %The Roulette method.
      if (prob_sum>f_prob)
        father = j;
        break;
      end
    end
    prob_sum = 0;
    for j = 1:ensemble_size
      prob_sum = prob_sum+s(j);
      if (prob_sum>m_prob)
        mother = j;
        break;
      end
    end
    next_generation(:, :, i+5) = (map_ensemble(:, :, index(father)) +...
    map_ensemble(:, :, index(mother)))./2; %The son is the average of the parents.
  end
  %next_generation(:,:,ensemble_size) = rand_map(N); %move in

  for i = 1:mutation
    choose = randi(ensemble_size -5)+5; %The one who is choosed as mutation.
    site = randi(N); %The mutation site.
    next_generation(site, :, choose) = next_generation(site, :, choose)+...
    (rand(1, 2)-0.5)*pi/3;
  end

  map_ensemble = next_generation; %update the ensemble.
  generation = generation + 1; %update the generation.
end
%plot(1:max_generation, energy_gene)
%disp(energy_ensem(1))
```

Supplementary Note 2: Another configurations



(a) 5 (b) 6 (c) 7 (d) 8 (e) 9

(f) 10 (g) 11 (h) 12 (i) 13 (j) 14

(k) 15 (l) 16 (m) 17 (n) 18 (o) 19

(p) 20 (q) 21 (r) 22 (s) 23 (t) 24

(u) 25 (v) 26 (w) 27 (x) 28 (y) 29

Fig. S1. Configurations