MEAHO: Membrane Evolutionary Algorithm for Hyperparameter Optimization of Deep Convolutional Neural Networks

Jie Mu, Jiaxin Hou, and Ying Wan(✉)

School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China
wanying516@qq.com

Abstract. Membrane algorithm has been used to solve many optimization problems since it was put forward. These methods used the membrane algorithm as a container for other algorithms to solve many problems, such as, traveling salesman problem, the knapsack problem and so on. In this paper, from the angle of membrane algorithm, the solution of hyperparameter optimization problem by membrane algorithm is explored, with the hyperparameter space divided by lattice method, which transformed into membrane structure subsequently. For the evolution of membrane, the entropy reduction principle is proposed. Compared with other membrane algorithms, this paper solves the optimization problem from the membrane algorithm itself, designs the corresponding algorithm for the membrane structure, and obtains the experimental results through CNN experiment.

Keywords: Hyperparameter optimization · Deep Convolutional Neural Networks · Membrane Evolutionary Algorithm

1 Introduction

Inspired by the structure and evolutionary function of the biological cells, membrane computing was proposed in 2000 by Paun [1]. Initially, the early work aimed to design basic computing operators and construct membrane computing systems for computing problems at the theoretical level. Various membrane computing systems are also called membrane systems or P systems [1–5]. As a fast-growing branch of natural computing, membrane computing has been applied for a variety of fields, including NP-complete problems, parallel and distributed algorithms, graphics, linguistics and economy. It has been constructed for solving a large range of problems, for example, knapsack problem [25, 26], maximum clique problem [27], Minimum Vertex Cover Problem [28] and so on.

Benefitted from the evolutionary mechanism of biological cells, under the framework of membrane computing, the computing models perform in a parallel, and distributed way to calculate. Because the execution of the rules in membrane computing is random and parallel, which enables it to solve complex problems such as Nphard problems [36], membrane algorithm usually uses the membrane as a container to solve the optimization

M. Zhang et al. (Eds.): CCF NCCA 2023, CCIS 1959, pp. 256–272, 2024.
https://doi.org/10.1007/978-981-99-8764-1_20

problem, and different algorithms are run in each cell to solve the problem, which can be done in a parallel and distributed method. In particular, the membrane system is based on a hierarchy structure in which a single or different algorithm can interact to solve a problem. In [15], an adaptive membrane algorithm combining the hierarchical structure and local search was proposed to solve the travelling salesman problem. The structure of membrane algorithm has a good tolerance to different algorithms, so it can solve the optimization problem well. In [37], Guo et al. combined heuristic algorithm and cell-like P system, proposing a membrane evolutionary algorithm for solving TSP. Membrane algorithm can be used as the basic structure to build the corresponding algorithm to solve the corresponding optimization problems and has certain effectiveness.

As another branch of natural computing, evolutionary computation searches an optimal solution along an evolutionary route. Evolutionary computing can be used to solve optimization problems in multi-dimensional space. In [29], the Genetic algorithm (GA) is used to compute the decentralized control parameters to achieve an optimum operating point in automatic generation control (AGC), this is a problem of finding the optimal value in multi-dimensional space. Delgarm et al. prosed a mono- and multi-objective particle swarm optimization (MOPSO) algorithm to solve multi-objective optimization of building energy performance [30]. Zhu et al. used particle swarm optimization to solve the non-linear constrained portfolio optimization problem with multi-objective functions [31]. These methods use evolutionary algorithm to explore the solution of multi- dimensional space, and prove the effectiveness of evolutionary algorithm for high dimensional space optimization.

With the burst-development of deep learning, hyperparameter optimization of deep models has been becoming a barrier for practical application of deep leaning techniques [33], because deep models are very sensitive to the setting of their hyperparameters [17]. While non-experts have a hard time finding a good setting of hyperparameter parameters, automatic parameter optimization can produce hyperparameter settings that are similar to those of experts [8]. However, the computational cost of fitting large DNNs is very high, and the time overhead of automatic hyperparameter optimization prevents its widespread adoption. Moreover, the advantages and disadvantages of hyperparameter optimization require in-depth investigations through experiments.

In this work, we construct a membrane evolutionary algorithm to optimize hyperparameter setting for deep convolutional neural networks. This is the first time that a membrane system is used to solve the complex problem of hyperparameter optimization of deep neural network models. Unlike other methods that only use membranes as containers to hold other algorithms to solve optimization problems, we use the structure of membrane system itself to solve optimization problems and design corresponding evolution rules. Because the optimization space of hyperparameter optimization for deep neural network is a high dimension issue, so we construct the corresponding structure to solve this problem, and use hierarchical structure to present it. The optimization space is represented in the form of cell membrane, which is divided into a grid, and then the specific hyperparameter value is taken as the median value in each grid. In addition to the fitness evaluation of membrane evolution rules, we also evaluated the expected effect of each super parameter, which is the in-fluence of the hyperparameter on the fitness. Then, for membrane fusion we use the entropy principle for evolution. If the binding of two

dividing cells can produce an entropy increase, then these two cells are worthy of binding, and the definition of entropy increase mainly depends on the expected effect of the hyperparameter. Through constant iteration, a group of better results of hyperparameter optimization are obtained.

The contributions of this paper can be summarized into the following three points:

- The structure of the membrane algorithm as the model of the evolutionary algorithm rather than having the membrane structure include other optimization algorithms.
- The membrane algorithm is used to design the hyperparameter optimization algorithm for the first time.
- The entropy theory is applied to the optimization model to increase the robustness of the model.

The other part of this paper is organized as follows. In Sect. 2, some related works about membrane algorithms and the hyperparameter optimization are introduced. Then, MEA is introduced in Sect. 3, including its operators and algorithm framework. Base on MEA, MEAHO is designed to solve the hyperparameter optimization in Sect. 4. Section 5 gives the experiment results on large real-world data set. As for Sect. 6, some problems in this paper are discussed. Finally, overall conclusions are given in Sect. 7.

2 Membrane Evolutionary Algorithm Framework for Hyperparameter Optimization

In order to handle the Hyperparameter optimization problem for deep convolutional neural network, in this section we propose a cell-like P system with a membrane evolutionary algorithm based on the nested structure of biological cells with its evolutionary rules. Here, we first introduce the hyperparameter optimization problem for deep convolution neural network. Additionally, the pro-posed membrane system is formally defined. Further, a membrane evolutionary algorithm is proposed in the membrane system. Finally, the proposed membrane system and membrane evolutionary algorithm are applied to hyperparameter optimization of deep convolutional neural network.

2.1 Hyperparameter Optimization Problem for Deep Convolutional Neural Networks

A set of hyperparameters should be configured before training, such as number of layers (depth), number of neural cells for each layer (width), learning rate, drop-out rate, size of mini bath. However, the influence of the model structure based on the results is far greater than that of other hyperparameters, so this paper focuses on the optimization of the hyperparameters of the model structure. The main hyperparameters considered in this work are listed as follows:

1) Number of layers (depth): the number of layers of convolution layer and full connection layer.
2) Number of neural cells for each layer (width): W_{ci} is number of neural cells for i th layer of convolution layer, W_{fj} is number of neural cells for j-th layer of convolution layer.

2.2 Definition of P System

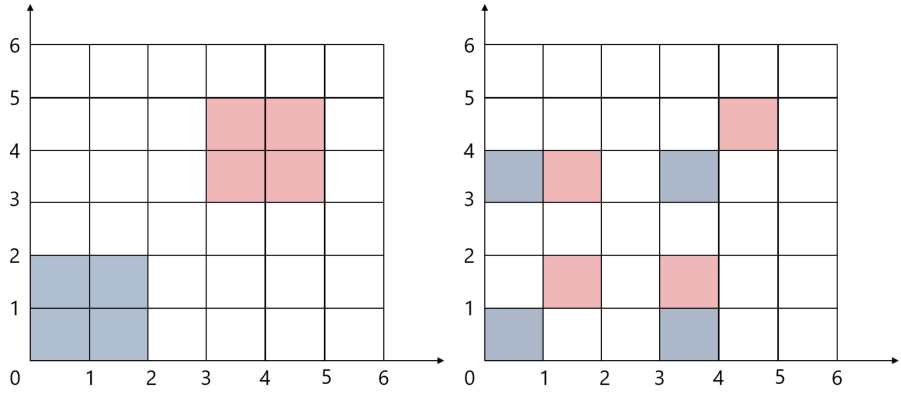
At first, the proposed P system is formally defined as follows:

$$\Pi = (O, H, \mu, \omega_1, \ldots, \omega_n, R_1, \ldots, R_n, i_0) \tag{1}$$

where,

1) O is the limited alphabet. Each symbol represents one kind of objects in the system Π;
2) μ is the membrane structure with n membranes, labeled by $1, 2, \ldots, n$;
3) $\omega_i (1 \leq i \leq n)$ is multisets of objects present in the membrane i;
4) $R_i (1 \leq i \leq n)$ is the finite set of evolution rules in the membrane i;
5) i_0 is the label of the output membrane and it reserves the final result.

2.3 Space Exploration



(a) Initial space (b) Split the merged space

Fig. 1. (a) represents the initial space selected, and (b) represents the space involved after splitting and fusion.

For super parameter optimization, it all takes a point to determine a point. This article through the use of membrane structure for super parameter optimization of the algorithm, and then expands the concept of a super parameter points to the space concept, namely from a select of a point in space to represent the effect of the whole space, which can be strengthened for the exploration ability of whole space, Fig. 1 shows the split algorithm combined the membranes of the space, this paper designed by membrane algorithm is divided evenly divided, namely each dimension evenly divided, and then split up and other cells are combined, the formation of new cells, new cell contains space isn't completely close to, through this merger, can increase the whole space exploration ability. As shown in Fig. 1(b), when cells are fused in two far apart Spaces, the space in the middle part will be explored. By uniformly taking points, the whole space can be explored and a global optimal solution can be obtained.

2.4 Membrane Evolutionary Algorithm Framework (MEAF)

Inspired from the biological membrane structure and evolution process, we first construct an algorithm frame-work for hyperparameter optimization in this section. In this framework, the proposed membrane system has two layer membranes and four evolutionary rules.

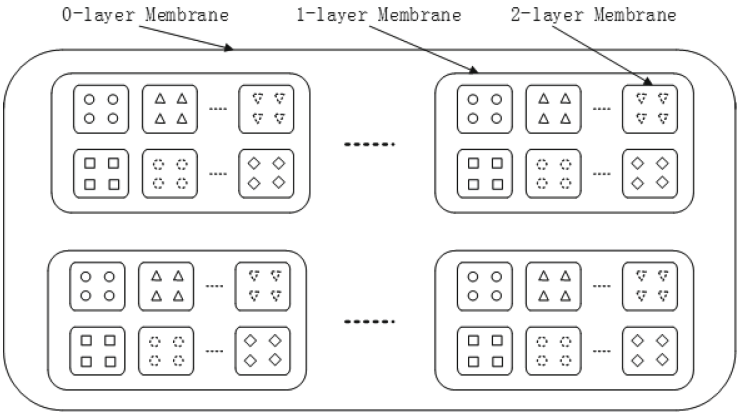


Fig. 2. Membrane structure

Membrane System Structure: In MEAF, the membrane structure is shown in Fig. 1. There are three-layer membranes. The skin membrane, which is also called as 0-layer membrane, embraces m 1-layer membranes in which the n 2-layer membranes are scattered. Each 1-layer membrane $m_i(1 \leq i \leq m)$ corresponds to a candidate model, and each 2-layer membrane $p_j(1 \leq j \leq n)$ corresponds to a hyperparameter. Each object $o_t(1 \leq t \leq v)$ in $p_j(1 \leq j \leq n)$ represents one candidate value for the j-th hyperparameter. So, a 2-layer membrane is related to a searching space of one hyperparameter. The system is evolved under the four evolutionary rules: division rule, apoptosis rule, replication rule and fusion rule, which are described as follows.

Initialization rule: Uniform values from each dimension according to uniform sampling, and then use these values as starting points.

Division rule: A biological system grows by cell division. The proposed membrane system evolves by the di-vision rule from one generation to the next generation. Each inner membrane is divided into two new 1-layer membranes. The objects of each original 1-layer mem-brane are wrapped up by one of the two new membranes in terms of the entropy decrease rule which is explained later.

Apoptosis rule: A biological system is improved and removes the inner garbage by cell apoptosis. In the pro-posed membrane system, 1-layer membranes with the lower fitness values.

Selection rule: A biological system obtains a better adaptability to the environment by selection. The 1-layer membranes with the higher fitness values are selected to evolve further.

Replication rule: self-replication.

Fusion rule: Two new 1-layer membranes are fused into a 1-layer membrane with four objects.

The membrane system is evolved under the four rules and the proposed membrane evolutionary algorithm framework is summarized in Algorithm 1.

Algorithm1
Input: The initial membrane system
Output: The evolved membrane system
Step 1: The initial membrane structure is constructed. The initial 50candidated parameter settings are evaluated. Step 2: The division rule is applied. Step 3: The fusion rule is applied. Step 4: The selection rules is applied. Step 5: The selection rule is applied. Step 6: Go to Step 1.

Figure 2 illustrates the initial membrane structure of MEAF, which includes the three-layer membranes. There is a 0-layer membrane, m 1-layer membranes which are related to models, and n 2-layer membranes which are related to the m hyperparameters considered. The m 1-layer membranes are labeled from 1 to m, which constitute the population of MEAF and are denoted as $\mathbb{M} = \{M_1, \ldots, M_m\}$, where m is the size of the population. $\forall M_i \in \mathbb{M}$ is one of the individuals in the population. The v objects in P_i are denoted as $O_i = \{o_{i1}, \ldots, o_{iv}\}$. One object o_{it_i} from M_i is one value in the feasible range of the i th hyperparameter. And a combination of n objects $\{o_{1t_1}, o_{2t_2}, \ldots, o_{pt_v}\}$ from the m 1-layer membranes $\{M_1, M_2, \ldots, M_m\}$ constitute a feasible solution for the hyperparameter optimization problem. The population is evolved iteratively in parallel under the membrane evolutionary algorithm framework. In the evolution process, the definition of individual's fitness (1-layer membrane fitness) will be various in terms of the considered problems. In this work, we consider the hyperparameter optimization of convolutional neural networks, so an individual's fitness is calculated according to the performance of the convolutional neural network.

The four operators can be used alone or in combination for a specific problem. For example, the selection operator can be combined with the cytolysis operator to form a complex operator. In Sect. 4, we implement a specific membrane evolutionary algorithm for hyperparameter optimization (MEAHO) under the MEAF framework.

3 Hyperparameter Optimization Based on Membrane Evolutionary Algorithm (MEAHO)

In this section, we propose a membrane evolution algorithm to solve hyperparameter optimization problem for deep convolutional neural networks, which is named MEAHO (membrane evolution algorithm for hyperparameter optimization problem).

Definitions and Notation: In MEAHO, the membrane structure is shown in Fig. 2. In this work, we consider the seven hyperparameters in this work, so $n = 7$ here. For each

hyperparameter, its value range is equally divided into v segments. The v objects for the hyperparameter P_j are generated through selecting the medium value of each segment. So for each neural network model, there are v^m hyperparameter configurations, in which the setting with the lowest MSR measure is called the best object combination and denoted as O_{best}. So, each individual (1-layer membrane) generates v^m possible neural network models for each generation. Next, we define the fitness of individual in this work.

Definition 1: Fitness. The fitness $fit(M_i)$ of the individual M_i is defined as follows,

$$fit(M_i) = 1 - MSR(M_i, O_{best}) \tag{2}$$

where $MSR(M_i, O_{best})$ represents the MSR performance of the model M_i with its current best hyperparameter combination O_{best}.

MEAHO Algorithm: In this work, we construct a three-layer convolution neural network and consider seven hyperparameters, such as the width of the first convolutional layer w_1, the width of the second convolutional layer w_2, the width of the third convolutional layer w_3, the width of the fully-connected layer w_4, the learning rate w_5, the size of minibatch w_6 and the ratio for dropout w_7. In this work, $m = 7$ and $v = 4$. Here, for each generation, there are 4^7 candidate hyperparameter configurations in the membrane system. This section introduces the algorithm of MEAHO for solving the HO problem. MEAHO solves the HO problem by iteratively performing division, fusion, cytolysis, and selection operations.

At the start of each iteration, the size of the population maintains to be n, and the size of the hyperparameter division segments is v. In each generation, we first need to evaluate the fitness of the n individuals. To find the globally best model of the individual M_i we need to evaluate the all 4^7 CNN models. However, it is impractical for the time cost. Here, for each model membrane, s hyperparameter configurations are selected from the v^m. The selected configurations are respected to be most representative with the limited size s. In this work, $s = 50$. So for each model membrane, 50 CNN models from the 4^7 model set are evaluated and the performance of the best model is used to compute the fitness of the model membrane according to Eq. (2).

In order to evaluate the quality of each inner membrane, we need to define its fitness. For each hyperparameter combination, the performance of the CNN model is evaluated. Here, we select MSE as the evaluation metric. The lower the MSE value is, the higher the fitness is. The membrane with larger fitness value is more adaptive to the environment.

The natural selection strategy is uniform random sampling on the hyperparameter grid. In this work, we adopt the uniform design method which is constructed for biochemical test strategy design. The uniform design method ensures the selected points distribute on the hyperparameter grid as uniformly as possible. To illustrate the performance of uniform design sampling on the hyperparameter grid. Here, we give an example considering three hyperparameters with four-segment division for each hyperparameter value range. In Fig. 1, 50 random selected points distribute uniformly on the 3-Dim hyperparameter grid. Here, 50 selected points correspond to 50 CNN models with different hyperparameter configurations in each individual.

After the selected CNN models are evaluated for each model membrane, the fitness of individuals can be computed. In each model membrane, the best fitness from those

of the 50 evaluated CNN models are selected as the fitness of the considered model membrane. So the fitness values of all the model membranes can be sorted. Then the population starts to evolve toward the biologically benefited direction. At first, in terms of the selection rule, the top α model membranes are selected and copied, where $0 < \alpha < 1$. The bottom α model membranes are selected and deleted in terms of the cytolysis rule. For the left middle $1 - 2\alpha$ model membranes, each model membrane splits into two new membranes which are called as temporary model membranes in terms of the division rule. With model membranes splitting, each hyperparameter membrane are divided into two temporary hyperparameter membranes. Meanwhile, the four value objects in each hyperparameter are grouped into two clusters according to a specific biological rule. Here, we select the entropy decrease rule to conduct the clustering procedure of the four value objects. In nature systems, a biological system is generally evolving to the entropy decrease direction. The entropy decreasing is the intrinsic rule in biological systems, which is also applied in the node division for the decision tree algorithm. The proposed MEAHO membrane system imitates this rule, and the inner membrane is divided according to the entropy decrease rule.

The entropy is used to describe the degree of distribution. Here, we define as follows.

Definition: Entropy: The degree of chaos of a system itself, in this paper entropy mainly refers to the effect value that can be achieved by cells of the membrane system.

Entropy Decrease Principle: The basic principle of the development of the world is the increase of entropy. As the living body is an open system, it can absorb nutrients from the outside world, so its growth and evolution direction are toward the direction of entropy reduction, which is a more orderly direction of organisms. Meanwhile, keeping various molecular species separated in individual compartments is another entropy reducing process [35].

In this paper, it is believed that the increase of hyperparameter effect represents the decrease of entropy, that is the combination of hyperparameter with good effect is an operation of entropy reduction, which can make the whole membrane system more stable. For the evaluation of the hyperparameter effect, we use the hyperparameter combination under the current state of the membrane system and the corresponding fitness to carry out the fitting.

$$\sigma(\ddagger) = \frac{1}{1+e^{-w^T x}} \tag{3}$$

Logical regression is the main fitting method, as shown in 3. In 3, Z represents the fitness gained which is a value, and the x represents the combination of hyperparametric. For the prediction of hyperparameter effect, the prediction is mainly based on the way of division. A possible fitness for each hyperparameter can then be obtained to achieve a fusion based on entropy decrement.

In the specific algorithm, after the membrane division, Logistic regression is performed based on the current hyperparameter combination of each cell and the corresponding fitness, and then the possible fitness can be predicted for the divided interval of each cell. The cells were then fused based on the predicted values. Based on the principle of entropy reduction, cells that can achieve better fitness can fuse with each other to achieve a more stable state. According to this rule, continuous division and fusion can be carried out to continuously reduce entropy and avoid the entropy increase

caused by ablation of cells. One round of the evolution has been completed and then the next generation evolution is started. The iterative evolution process stops until an end condition to be satisfied, for example, a specific iteration number and a specific fitness. This iterative process follows the principle of entropy reduction to iterate so as to get better results.

Algorithm 2 Membrane Evolutionary Algorithm for Hyperparameter Optimization (MEAHO)
Input: The initial membrane system, population size PS, the maximum runtime MRT, α.
Output: a hyperparameter combination

(1) Initialization
The initial membrane structure is constructed.
The initial 50 candidate parameter settings are evaluated.
repeat
(2) Evolve membranes in parallel: selection(P);
cytolysis(P);
division(P);
fusion(P);
until elapsed time >= MRT;
(3) return best;

4 Experimental Results

To demonstrate the effectiveness of MEAHO, this section compares the experimental results of MEAHO with random search, genetic algorithm and PSO.

4.1 Data Sets

We used three data sets, which are MNIST,CIFFAR10,and a kind of traffic data which from the PEMS(the public accessible Caltrans Performance Measurement System).MNIST data sets is a very classic data sets in the field of machine learning, consisting of 60,000 training samples and 10,000 test samples, each of which is a grayscale handwritten digital image with 28*28 pixels.CIFFAR10 data sets is a kind of RGB color plates include 10 categories: airplanes, cars, birds, cats, deer, dogs, frogs, horses, boats, and trucks. The image size is 32*32, and there are 50,000 training and 10,000 test images in the data set. Traffic data stets is from the PeMS dataset. We select 26 sensors from the data. So we evaluate all the compared methods under these data sets.

4.2 Evaluation Metrics and Baselines

In this paper, the Mean Average Percentage Error (MAPE) is used as evaluation metric of traffic data sets, which is defined as follow:

$$MAPE = \frac{1}{N}\sum_{I=1}^{N}\frac{\left|\widehat{y_i}-y_i\right|}{y_i} \tag{4}$$

where N is the number of all samples, and $\widehat{y_i}$ is the prediction value, and y_i is the true value. The classification accuracy is used as evaluation metric of MNIST data sets and CIFFAR10 data sets. Our proposed model is compared with the following methods:

- Random Search (RS): Within a certain interval, randomly generate points and then compare the objective function, keep the points with better results, and finally get the results through multiple iterations.
- Particle Swarm Optimization (PSO): PSO is an evolutionary computing technique derived from the study of predatory behavior of birds. The basic idea of particle swarm optimization algorithm is to find the optimal solution through the cooperation and information sharing among individuals in the group [10].
- Genetic Algorithm (GA): Genetic algorithm is a computational model that simulates the natural selection and genetic mechanism of Darwinian biological evolution. It is a method to search for the optimal solution by simulating the natural evolution process. [11].

4.3 Setting of Experimental Parameters

This experiment mainly uses three data sets, MNIST, CIFFAR10 and Traffic data set. The number of iterations of MBU for these three data sets is 50. In the experiment of MNIST data set, there are 2500 sets of super parameters in each iteration, and the number of the first layer membrane is 50, and the number of the second layer membrane is 50. The first layer is divided into 4, and the second layer is divided into 10. In the experiment of Transport data set, there are 2500 sets of super parameters in each iteration, and the number of the first layer membrane is 50, and the number of the second layer membrane is 50. The first layer is divided into 3, and the second layer is divided into 6. In the experiment of CIFFAR10 data set, there are 500 sets of super parameters in each iteration, and the number of the first layer membrane is 10, and the number of the second layer membrane is 50. The first layer is divided into 5, and the second layer is divided into 8.

4.4 Results and Analysis

The comparison of the performance of our model with the three comparison methods is shown in Table 1, and the experimental results of Mnist, ciffar10 and traffic data are given respectively. Based on the consideration of training time, training is divided into coarse training and fine training. The number of iterations of coarse training is less, while the number of iterations of fine training is increased on the basis of coarse training. The number of iterations increased by different contrast methods is uniform. Coarse training is the training method used in the model. After the optimization results are obtained, the results of intensive training are obtained. As you can see from I, compared with other methods, our method can get a better result in coarse training, and the effectiveness of the optimization result is proved by the use of fine training.

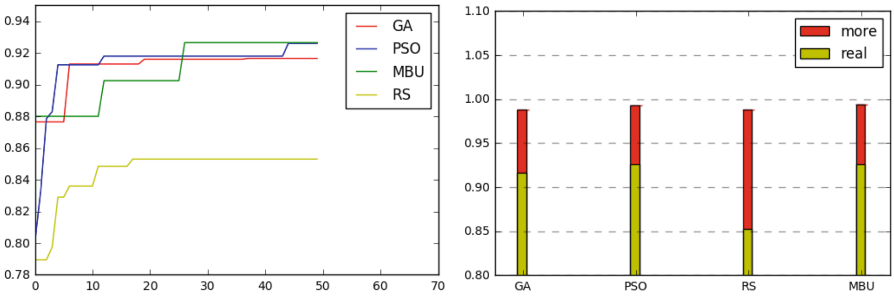
The model using MNIST data set has 5 parameters that need to be trained, and each iteration has 2500 sets of parameters. The model using CIFFAR10 data set has 7 parameters that need to be trained, and each iteration has 500 sets of parameters. The model using Transport data set has 4 parameters that need to be trained and each iteration

Table 1. Result of contrast.

Methods	mnist		ciffar10		traffic data	
	Coarse training	Fine training	Coarse training	Fine training	Coarse training	Fine training
RS	0.853	0.989	0.511	0.785	0.651	0.143
GA	0.917	0.988	0.523	0.811	0.651	0.141
PSO	0.926	0.993	0.522	0.813	0.646	0.137
MBU (ours)	0.927	0.994	0.524	0.813	0.649	0.138

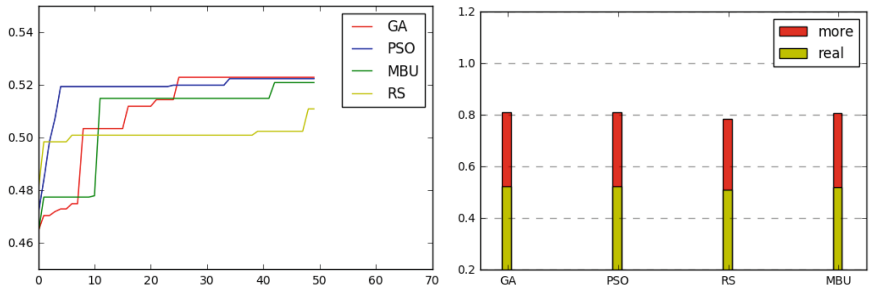
has 2500 sets of parameters. The experimental results are divided into coarse training with less iteration times and intensive training with retraining on the basis of rough training. The results of coarse training and fine training are shown in Figs. 3 a, 4 a and 5 a. We can see coarse training can better reflect the results of intensive training. Therefore, when looking for better parameter values, it is reasonable to use coarse training to calculate the fitness. Coarse training can better reflect the results of fine training, and because the number of iterations is less, the results can be obtained faster and the running time of the algorithm can be reduced. Parameters with week coarse training results have been greatly improved after intensive training, but will not interfere with the overall results.

Random search is superior to grid search, Random search takes less time than grid search, while more parameter values can be tried. So this paper uses random search as the baseline model, The heuristic rules of bio-logical evolution can help to optimize the optimal path and find the optimal solution more quickly in the high-dimensional space. The heuristic method can get a better result than the random method, the results of GA and PSO are better than random search.



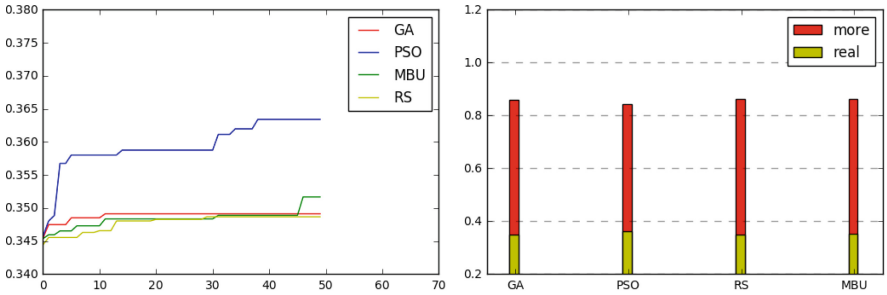
(a) Results of contrast under MNIST dataset (b) Iteration curve under MNIST dataset

Fig. 3. For MNIST data set, coarse training was firstly used for parameter tuning, and then intensive training was used to verify the effect of parameters. (a) represents the results of contrast under MNIST dataset, and (b) represents the iteration curve under MNIST dataset.



(a) Results of contrast under CIFFAR10 dataset (b) Iteration curve under CIFFAR10 dataset

Fig. 4. For CIFFAR10 data set, coarse training was firstly used for parameter tuning, and then intensive training was used to verify the effect of parameters. (a) represents the results of contrast under CIFFAR10 dataset, and (b) represents the iteration curve under CIFFAR10 dataset.



(a) Results of contrast under transport dataset (b) Iteration curve under transport dataset

Fig. 5. For transport data set, coarse training was firstly used for parameter tuning, and then intensive training was used to verify the effect of parameters. (a) represents the results of contrast under transport dataset, and (b) represents the iteration curve under transport dataset.

Table 2. The effect of the number of membranes on the result

	8	16	32	64
8	0.821	0.857	0.824	0.811
16	0.826	0.826	0.861	0.892
32	0.817	0.833	0.916	0.861
64	0.858	0.875	0.894	0.913

Compared with the genetic algorithm, MBU adds the guidance of evolution. The evaluation of each parameter can make the optimization result move forward in a beneficial direction, while the evolutionary rule of genetic algorithm is a kind of weak guidance and contains more random factors. It can be seen from the results of I that the effect of the method in this paper is better than that of genetic algorithm.

Table 3. The effect of the degree of divide on the result

	4	6	8	10
4	0.890	0.891	0.882	0.879
6	0.900	0.901	0.890	0.856
8	0.899	0.887	0.848	0.856
10	0.916	0.891	0.895	0.881

MBU works better on a dataset with a large number of parameters, while PSO works better on a dataset with a small parameter space. The iterative curves of the three data sets are shown in Fig. 3 b, 4 b and 5 b. MBU can be steadily promoted, but the initial speed of promotion is not as good as that of PSO, because MBU's consideration of blocks can increase the search range, but also reduce the search speed. At the same time, the space exploration scope of MBU is larger than that of other methods. In MBU, a point is used to represent a space, and the scope of exploration can be greatly improved. In this way, MBU can get a better result than other methods.

Table 4. Modular validation of the model

Model	Result	Dataset
Non-uniform initialization	0.877	mnist
No entropy minus rule	0.902	mnist
Complete model	0.927	mnist

Therefore, the method proposed in this paper can solve the problem of hyper-parameter optimization well, and the effectiveness of this method is proved by experiments.

4.5 Self-contrast Experiment

In this section, we will compare the effects of the size of fine granularity and the number of membrane of the model on the experimental results. Fine granularity and number of membrane are defined as follows:

Definition 1 Fine granularity: Fine granularity refers to the number of points an interval is divided into, such as the interval [0,128]. When the fine granularity is 4, the interval is evenly divided into 4 parts.

Definition 2 Number of membrane: randomly select an interval for each parameter to form the first film, and then randomly select the second film based on the first membrane. For example, when the fine granularity size of the first layer of membrane is 4, an interval is selected by one parameter to form the membrane, and the second layer of membrane is divided by the fine particle size of 10 on this interval, and then an interval is selected. Then, parameter values are randomly selected from this range.

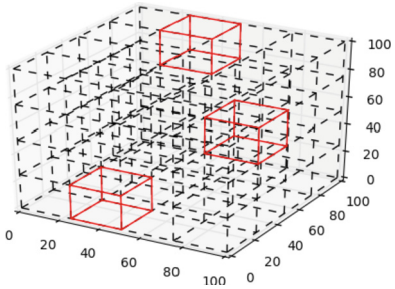


Fig. 6. Membrane space

Table 2 shows the influence of the number of mem-branes on the results. This experiment uses the accuracy of completing the classification task by using CNN as evaluation index, and MNIST is used in the data set. The fine granularity default parameter is [4, 10], which is an empirical value, and 20 iterations. It can be seen that the results do not increase completely according to the in-crease in the number of films. The value on the diagonal line increases first and then decreases. When the number of the first layer of membrane is fixed, with the number of the second layer of membrane increasing, the result shows a trend of increasing first and decreasing. When the number of the second layer of membrane is fixed, with the increase of the number of the second layer, the results also show a trend of increasing at first and then decreasing. When the number of the first layer of membrane is 32 and the number of the second layer of membrane is 32, the optimal value is obtained. As you can see, the number of mem-branes is not completely relevant to the result.

In order to study the influence of fine granularity on the experimental results, fine granularity was used as a parameter to conduct the experiment, and Table 3 was obtained. The accuracy of the classification task realized by CNN was used as the evaluation index, and MNIST was used for the data set. The parameter of membrane number is 32*32 according to the results in Table 2, and the number of iterations is 20. When the size of fine granularity of the first layer is fixed, the result increases gradually with the increase of the fine particle size of the second layer. When the size of fine granularity of the second layer is fixed, the result decreases gradually with the increase of the first layer, and the optimal value is obtained when the first layer is divided into 4 and the second layer is divided into 10. It can be seen that the fine granularity of the first layer is negatively correlated with the result, while the fine granularity of the second layer is positively correlated with the result. The first layer represents the value range of parameters, while the second layer is the refinement of the space of the first layer. The first layer could narrow the search area. The higher the level of refinement of second layer, the better.

4.6 Structural Validation

There are several important parts in our model, and this section discusses the validity of these Settings. The first is the selection of the initial point. This paper uses the uniform sampling method to select the initial point, and thinks that the global search can be carried out by this way. As for the guidance of cell division and merger, entropy minus

rule is designed as a guide according to biological characteristics, and logistic regression is used to predict the effect of each set of parameters.

Experiments were designed for uniform values and entropy subtraction rules, and the results in Table 4 were obtained, according to Table 4, can know when initialized randomly take the model, the model of effect is poor, this is because the selection of initial point is effect model to explore space, random values may lead to a value into a local, from which to state of global search. When the model does not follow the entropy minus rule, the effect is also poor, because at this time, the model adopts the mode of random splitting and merging, which is an unguided state.

5 Discussion

In this section, we will discuss some of the issues in this article.

5.1 Membrane Algorithm for Space Exploration

The space explored by the membrane algorithm is a grid. Figure 6 shows the structure of MEAHO in 3-dimensions space. This algorithm explores the space in the form of a grid. A point in the grid is used to reflect the effect of the whole space, which can expand the scope of exploration and is similar to exploring the entire optimized space. This method can search a larger space than the point search. Although there are important discoveries revealed by these studies, there are also limitations. First, the size of the membrane is an important parameter. The size of the membrane greatly affects the results and is an empirical parameter. Second, the selection of initial points should be able to represent the information of the whole space. In this paper, uniform sampling is used to select the initial points. However, the uniform use of a single dimension is insufficient, and multiple dimensions should be considered simultaneously to conduct a uniform sampling. Meanwhile, how to choose a uniform point that can represent the whole space is also at issue.

5.2 Evaluation of Hyperparameter Effect

In this paper, we evaluated the expected influence of the hyperparameters on the results to combine the hyperparameters that might lead to better results, so as to conduct cell division and fusion. In this paper, the evaluations of the different hyperparameters are independent from each other, which increases independence, but also loses a way to obtain more information. The hyperparameters of different dimensions may be coupled, and therefore the interaction between different hyperparameters should be taken into account.

At the same time, the evaluation method should improve the memory. The past situation can be used as a reference for the current state, and the current approach only evaluates the parameters based on the current results. You can set up an experience pool to save past hyperparameter cases to guide the current situation.

5.3 Mechanism of Membrane Evolution

The evolution mechanism of the membrane proposed in this paper is a form of 50/50 division of the hyperparametric space. When the degree of division is greater than 2, part of the space cannot be changed. In order to solve this problem, it is necessary to refine the splitting size of the membrane, conduct the corresponding splitting process according to the degree of membrane division, and then complete the fusion of the membrane on this basis. This can better increase the ability of space exploration.

References

1. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000)
2. Martín-Vide, C., Păun, G., Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theor. Comput. Sci. **296**(2), 295–326 (2003)
3. Alsalibi, B. et al. A comprehensive survey on the recent variants and applications of membrane-inspired evolutionary algorithms. Archives of Computational Methods in Engineering: 1–17 (2022) https://doi.org/10.1007/s11831-021-09693-5
4. Pan, L., Wu, T., Su, Y., Vasilakos, A.V.: Cell-like spikingneural P systems with request rules. IEEE Trans. Nanobiosci. **16**(6), 513–522 (2017)
5. Peng, H., et al.: Spiking neural P systems with multiple channels. Neural Netw. **95**, 66–71 (2017)
6. Young, G O. Synthetic structure of industrial plastics, in Plastics, 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, pp. 15–64. (1964)
7. Chen, W K. Linear Networks and Systems. Belmont, CA, USA:Wadsworth, pp. 123–135 (1993)
8. Bergstra, J., et al. Algorithms for hyper-parameter optimization .In: 25th Annual Conference on Neural Information Processing Systems (NIPS 2011), vol. 24. Neural Information Processing Systems Foundation (2011)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
10. Kennedy, J., Eberhart, R. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks. Vol. 4. IEEE. (1995)
11. Whitley, D.: A genetic algorithm tutorial. Stat. Comput. **4**(2), 65–85 (1994)
12. Bergstra, J., Bengio, Y. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13.2 (2012)
13. David, Omid E., Greental, I. Genetic algorithms for evolving deep neural networks. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation. (2014)
14. Hutter, F., Hoos, H H., Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning And Intelligent Optimization. Springer, Berlin, Heidelberg. (2011)
15. Juanjuan, H.E., Jianhua, X.I.A.O., Zehui, S.H.A.O.: An adaptive membrane algorithm for solving combinatorial optimization problems. Acta Math. Sci. **34**(5), 1377–1394 (2014)
16. Lorenzo, P R., et al. Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proceedings of The Genetic and Evolutionary Computation Conference. (2017)
17. Montavon, G., Orr, G.B., Müller, K.-R. (eds.): Neural Networks: Tricks of the Trade: Second Edition. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8
18. Nishida, Taishin Y. Membrane algorithms. International Workshop on Membrane Computing. Springer, Berlin, Heidelberg. (2005). https://doi.org/10.1007/11603047_4

19. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: a review of bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016). https://doi.org/10.1109/JPROC.2015.2494218
20. Xiao, J., Huang, Y., Cheng, Z., He, J., Niu, Y.: A hybrid membrane evolutionary algorithm for solving constrained optimization problems. Optik **125**(2), 897–902 (2014). https://doi.org/10.1016/j.ijleo.2013.08.032
21. Zhang, G.-X., Gheorghe, M., Chao-Zhong, W.: A quantum-inspired evolutionary algorithm based on P systems for knapsack problem. Fund. Inform. **87**(1), 93–116 (2008)
22. Zhang, G., et al. A Novel Membrane Algorithm Based on Particle Swarm Optimization for Solving Broadcasting Problems. J. UCS 18.13: 1821–1841. (2012)
23. Zhang, G., Cheng, J., Gheorghe, M., Meng, Q.: A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems. Appl. Soft Comput. **13**(3), 1528–1542 (2013). https://doi.org/10.1016/j.asoc.2012.05.032
24. Zhang, Gexiang, et al. A population-membrane-systeminspired evolutionary algorithm for distribution network reconfiguration. Chinese J. Electron. 23.3: 437–441. (2014)
25. Pan, L., Martin-Vide, C.: Solving multidimensional 0–1 knapsack problem by P systems with input and active membranes. J. Parallel and Distrib. Comput. **65**(12), 1578–1584 (2005)
26. Pérez-Jiménez, Mario J., and Agustin Riscos-Núnez. A lineartime solution to the knapsack problem using P systems with active membranes. In: International workshop on membrane computing. Springer, Berlin, Heidelberg. (2003)
27. Attacking NP-complete problems: Liu, Xiangrong, et al. The power of time-free tissue P systems. Neurocomputing **159**, 151–156 (2015)
28. Guo, P., Quan, C., Chen, H.: MEAMVC: A membrane evolutionary algorithm for solving minimum vertex cover problem. IEEE Access **7**, 60774–60784 (2019)
29. Golpira, H., Bevrani, H.: Application of GA optimization for automatic generation control design in an interconnected power system. Energy Convers. Manage. **52**(5), 2247–2255 (2011)
30. Delgarm, N., et al.: Multi-objective optimization of the building energy performance: a simulation-based approach by means of particle swarm optimization (PSO). Appl. Energy **170**, 293–303 (2016)
31. Zhu, H., Wang, Yi., Wang, K., Chen, Y.: Particle swarm optimization (PSO) for the constrained portfolio optimization problem. Expert Syst. Appl. **38**(8), 10161–10169 (2011). https://doi.org/10.1016/j.eswa.2011.02.075
32. Leporati, Alberto, and Pagani, D. A membrane algorithm for the min storage problem. In: International Workshop on Membrane Computing. Springer, Berlin, Heidelberg. (2006)
33. Bischl, B., Binder, M., Lang, M., et al.: Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges. Wiley Interdisc. Rev.: Data Min. Knowl. Discovery **13**(2), e1484 (2023)
34. Pelikan, Martin, David E. Goldberg, Erick Cantú-Paz. BOA: The bayesian optimization algorithm. In: Proceedings of The Genetic and Evolutionary Computation Conference GECCO-99. Vol. 1. (1999)
35. Davies, P.C.W., Rieper, E., Tuszynski, J.A.: Self-organization and entropy reduction in a living cell. Biosystems **111**(1), 1–10 (2013). https://doi.org/10.1016/j.biosystems.2012.10.005
36. G. Päun, Computing with membranes: attacking NP-complete problems, in Unconventional Models Computer. London, U.K.: Springer, pp. 94–115. (2001)
37. Guo, P., Hou, M., Ye, L.: MEATSP: a membrane evolutionary algorithm for solving TSP. IEEE Access **8**, 199081–199096 (2020). https://doi.org/10.1109/ACCESS.2020.3035058