ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.172

Deep Evolutionary Algorithm for Large-Scale Sequence Optimization

Detian Zeng [1,2], Gaofeng Zhu [1], Qiucheng Miao [2], and Hao Liu [1 +]

[1] Information Institute, Hunan University of Humanities, Science and Technology, Loudi, China
[2] School of Computer, National University of Defense Technology, Changsha, China

Abstract. Aiming at the problem of slow convergence and poor effect of GA in solving large-scale sequence optimization problems, a deep evolutionary algorithm is proposed in this work. The algorithm uses the trained network to quickly find the initial solution of the problem and injects the initial solution into the GA population for further optimization. Finally, the optimal solution in the final population will be further optimized by the 2-OPT. The proposed algorithm is compared with other algorithms on the tsplib95 instances and industrial sorting sequence optimization dataset. Experimental results show that the proposed algorithm achieves the best optimization performance compared with other algorithms, especially for large-scale sequence optimization problems.

Keywords: genetic algorithm; deep reinforcement learning; 2-OPT; large-scale sequence optimization.

1. Introduction

The Travelling Salesman Problem (TSP) is a typical combinatorial optimization problem and seeking efficient solutions is a key task. TSP and its approximation problems are widely distributed in industrial optimization and scheduling, which has broad application scenarios [1]. At present, the method of solving the TSP problem is mainly divided into approximate method and exact method. As the scale of problem-solving increases, the time cost of the exact method increases exponentially, which cannot meet the real-time requirements. While the approximate method achieved a certain balance between the solution's accuracy and speed, but no theoretical guarantee can be given on solution's accuracy [2].

Meta-heuristic is the mainstream method in approximate method, such as Simulated Annealing (SA) [3], Particle Swarm Optimization (PSO) [4], Genetic Algorithm (GA) [5], Ant Colony Optimization (ACO) [6]. Note that there is still room for Meta-heuristic to be improved. Combining Meta-heuristic with other methods is a feasible research direction. For example, Zhu et al. [7] combined GA and reinforcement learning (RL) to optimize the FPGA circuit structure.

Since 2015, deep learning has brought new ideas to combinatorial optimization problem [8]. Compared with the online search of traditional meta-heuristic, the trained neural network only performs forward calculation once, so its time cost is smaller. For example, Dai et al [9] used a graph attention mechanism to represent city nodes and DQN for decision-making, achieving the best optimization performance below 50 nodes. However, when the scale of node further increases, the optimization performance of the above network will degenerate. Kool et al [10] performed RL on the improved Transformer network, and combined the 2-OPT algorithm for local optimization. The best optimization performance is achieved below 100 nodes.

To efficiently and reliably solve large-scale TSP and its approximation problems, this work proposes a deep evolutionary algorithm, which combines deep reinforcement learning and evolutionary algorithms to solve the better path, and combines the 2-OPT to further optimize the path. The proposed algorithm is compared with other algorithms on 6 tsplib95 instances of different scales. Meanwhile, the deep evolutionary algorithm is applied to the optimization of the sorting sequence of large-scale real industrial parts with constraints, and the problem is carried out on different scales. The experimental results show that the proposed algorithm gives the best optimization improvement compared to other algorithms.

2. Problem Description

+ Corresponding author. Tel.: +86 151 15897178.
E-mail address: lhkd0407@126.com.

2.1. Traveling Salesman Problem

As a classic NP-Hard problem, TSP can be described as a businessman who has to traverse a series of cities. Each city can only be visited once. After traversing all the cities, he will return to the first city. The goal of optimization requires the shortest total path. The problem can be expressed as an undirected complete graph G=(V, E), where V represents a collection of N city nodes and $N \in N^+$, E represents a collection of connecting edges between city nodes. The connecting edge $e_{ij} \in E$ of city i and j (positive integer i, $j \in V$ and i is not equal to j), the distance of e_{ij} is d_{ij}. The decision variables for this problem are:

$$x_{ij} = \begin{cases} 1 & if\ x_{ij}\ in\ the\ current\ solution \\ 0 & otherwise \end{cases} \tag{1}$$

The optimization goal for this problem is:

$$min: \sum_i^N \sum_j^N x_{ij} e_{ij} \tag{2}$$

2.2. Constrained Large-Scale Sorting Sequence Optimization

Constrained large-scale sorting sequence optimization can be approximated as an unclosed large-scale TSP problem. A typical example is the optimization of the sorting sequence of parts on a mechanical engineering manufacturing production line. Assuming that several cut steel plates need to be sorted, each plate contains several cut parts and the parts need to be placed in the material frame by a robotic arm. Due to the requirements of the industrial standardization process, the frame has set palletizing rules (the frame is divided into T areas, and each area is stacked with the same parts which are no more than L). Once the palletizing rules are not met, the frame clearing process is required, and frequent frame clearing requires repeated scheduling of AGV transportation. Generally, when planning a smart factory, it is necessary to minimize ineffective material handling. Therefore, it is possible to reduce the number of frame clearing by optimizing the sorting sequence of steel plates and parts in the plates. The meaning of the symbols related to this problem is defined in Table I.

The optimization objective and constraints are shown in Equation (3), which is to minimize f(S, P), where two constraints correspond to the palletizing rule (T=4, L=10 in this work).

$$\begin{aligned} min: f = & \begin{cases} 0 & if\ \theta = 0\ and\ V_\theta = 0 \\ f(S, P; V_\theta, \theta) & if\ \theta > T\ or\ V_\theta > L \end{cases} \\ s.t. & \begin{cases} \theta \le T \\ V_\theta \le L \end{cases} \end{aligned} \tag{3}$$

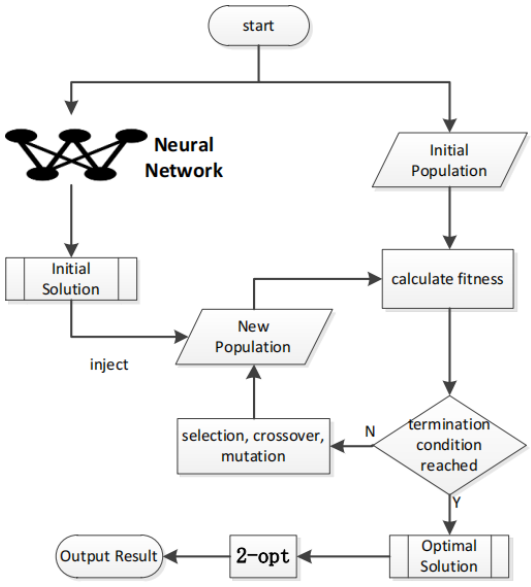


Fig. 1. The structure of the deep evolutionary algorithm.

TABLE I. The Definition of Related Symbols

Symbol	Definition
S	The data of N ∈ N+ steel plates with sorting sequence which is recorded as [... → Pi−1 → Pi →... → Pj−1 → Pj →...]. P represents the data of steel plate, where the positive integer subscripts i, j ∈ [1, N], and i != j.
P	The data of M ∈ N+ parts with sorting sequence which is denoted as [... → ki−1 → ki →... → kj−1 → kj →...]. where k stands for parts, where a positive integer subscript i, j ∈ [1, M], and i!=j. P represents the data of steel plate, where i, j <= N and i != j.
V_θ	The number of stacking layers of parts in any area of the current material frame, V_θ ∈ N.
θ	The number of types of all parts in the current material frame, θ ∈ N.
f	The objective function. The input is S and P, the steel plate set of a given sorting sequence is sorted according to the palletizing rule, and the output is the total number of frame clearing. Its mathematical definition is shown in Equation (1), which is a recursive form. The value of f(S, P) is 0 during initialization, and f(S, P) is incremented by 1 each time the frame clearing is triggered.

3. The structure of the deep evolutionary algorithm

The deep evolutionary algorithm gives the initial solution through the trained deep network and uses the initial solution as one of the evolutionary algorithm individuals, which makes the subsequent evolutionary algorithm has a higher quality solution in the initial stage. Meanwhile, the evolutionary algorithm Combining the elite retention strategy to enhance the diversity and reliability of the population and avoid falling into the local optimum. The following introduces the basic framework of the deep evolutionary algorithm.

3.1. The Framework of Deep Evolutionary Algorithm

The evolutionary algorithm is a calculation method that replaces the problem parameter space with the coding space. It needs to establish a one-to-one mapping between the actual representation of the target problem and the coding bit string. The path notation is used in the coding of the above problems. For example, the path [1→...→i→...→j→...→N→1] in the TSP problem represents starting from the first city, traversing each city in turn, and then returning to the original city. The coding of constrained sorting sequence optimization is shown in S and P in Table I.

Selection operator: The roulette operation is used. the reciprocal of an individual's fitness value is taken as the probability of being selected, and individuals with a lower fitness value are selected as the evolutionary father and mother.

Crossover operator: Firstly, two different subscripts start and end are chosen and the sequence segment of the father[start:end] is retained to the next generation. Then all the father[start:end] genes contained in the mother are deleted, and the father[start:end] sequence segment is inserted at the mother's start position to form a new offspring.

Mutation operator: The single-point cross mutation is used. If the mutation operator is executed, two different positions in the individual are randomly selected for exchange. The elite retention strategy is used in the evolution of the population. In other word, the parent population and the offspring population are merged and sorted according to the individual's fitness value. The first d ∈ N+ individuals are selected as the next evolutionary population, which is conducive to maintaining excellent individuals. After randomly initializing the population individuals, the selection, crossover, and mutation operators are performed in sequence to generate new individuals, so as to find individuals with lower fitness values.

From the above description, it can be found that since the initial solution of the evolutionary algorithm is randomly generated, the quality is not high and it is very likely to fall into a local optimum. To solve this problem, as shown in the algorithm framework in Figure 1, the attention network is used to inject high-quality initial solutions into the evolutionary algorithm.

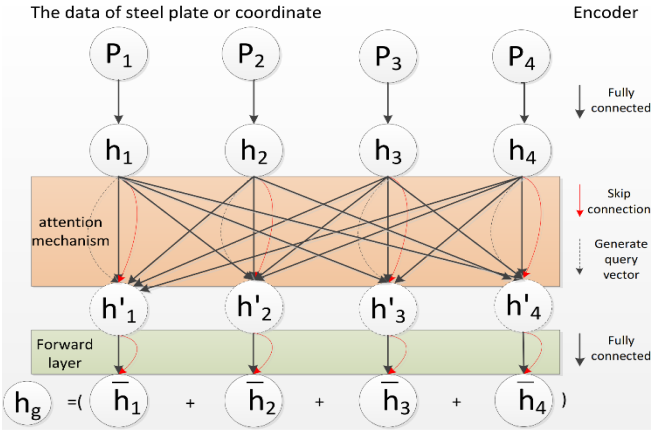


Fig. 2. The structure of the encoder.

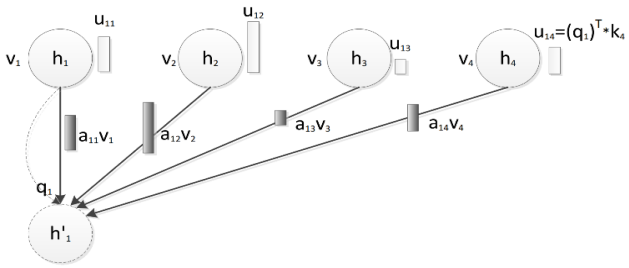


Fig. 3. The calculation of attention mechanism.

3.2. Initial Solution Generation Based on Attention Network

The attention network is divided into two parts: 1)Encoder-decoder. Which is mainly responsible for establishing the relationship between problem input and output. 2)Attention mechanism. Which integrates the relevant input and output in the encoder-decoder and calculates the degrees of attention of the node to be visited at the current moment. The calculation is repeated until all the moments are executed. The specific structure of the model is divided into three parts:

1) Encoder. The encoder uses a single-layer attention structure, and the input is the original data. For the TSP problem, it is the original city coordinate data. For the constrained sorting sequence optimization, the bag-of-words model is referenced to abstract a single steel plate as a fixed dimension vector about the part (the vector's dimensions is the total types of the part. When traversing the parts in the steel plate, the subscript position of the corresponding part in the vector is added 1) which will be the input. As shown in Figure 2, the input is passed through a shared fully connected layer to obtain the embedding representation. Then the multi-head attention mechanism is used to obtain the updated embedding representation injected with other node information. Finally, the final output representation of the original node in the encoder is obtained by a forward layer, and the output representation is summed and averaged as the overall representation of the problem. This overall representation will form part of the query vector in the attention mechanism of the decoder.

2) Attention mechanism. As shown in Figure 3, assuming that there are embedding representations h_1, h_2, h_3, and h_4, the query vector q_i, key vector k_i, and value vector v_i of each embedding representation hi will be calculated. Please see equation (4), where W^Q, W^K, and W^V are all learnable parameter matrices. Among them, the attention score u_{ij} of node j to node i is q_i points multiplied by k_j. If node j is unreachable to node i, the attention score is negative infinity. After normalizing u_{ij} and multiplying it by v_j , it is the black bar in Figure 3. Adding the four vectors is the output of the attention mechanism to h_i , see Eequations (5)-(7).

$$q_i = W^Q * h_i \quad k_i = W^K * h_i \quad v_i = W^V * h_i \tag{4}$$

$$u_{ij} = \begin{cases} q_i^T k_j & if\ i\ adjacent\ to\ j \\ -\infty & otherwise \end{cases} \tag{5}$$

$$a_{ij} = softmax(u_{ij}) = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}} \tag{6}$$

$$h_i' = \sum a_{ij} v_j \tag{7}$$

3)Decoder. The decoder mainly combines the overall representation output by the encoder and the information of the current solution, and outputs the evaluation of the node that may be selected at the next moment in a targeted manner. As shown in Figure 4, the attention mechanism is still used in the decoder for decoding. The overall representation, the first node's encoding representation of the current solution, and the

encoding representation of the selected node at the previous moment are aggregated (where v1 and vf are learnable logo vectors). The aggregated vector is used as the query vector to decode the node representation output by the encoder. In addition, the masking mechanism is used to shield the city or steel plate that has been visited, so that it will not be selected in the final decision.

3.3. Continued Optimization of 2-OPT Algorithm

The 2-OPT algorithm is an efficient local path improvement algorithm. Its main idea is to optimize the current solution by continuously exchanging two edges for a given initialization path, and the solution with the lowest fitness value is used as the final output. This work uses the 2-OPT algorithm to further optimize the solution given by the deep evolutionary algorithm.

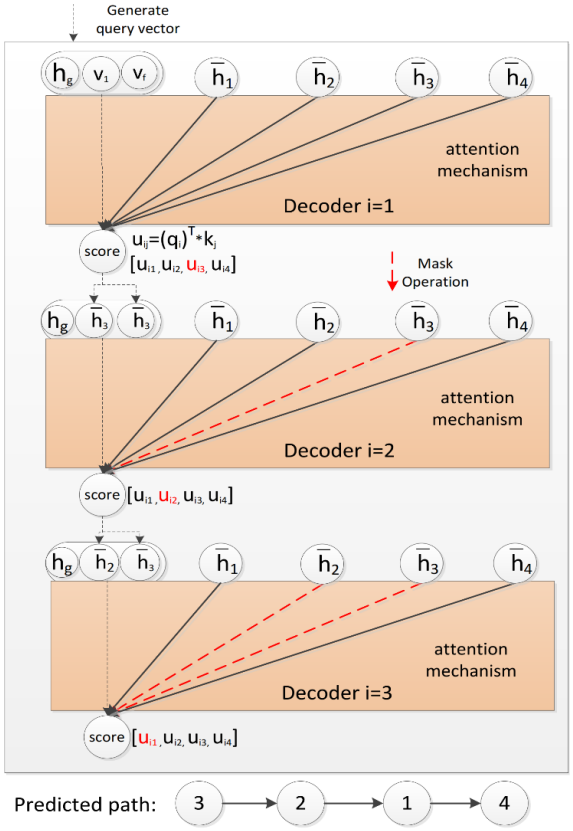


Fig. 4. The structure of the decoder.

4. Network Training

The TSP or the sequence optimization process contains the markov property. The node to be visited at t+1 is only related to the node selected at t and the information of overall node. Note that the parameter of the attention network is w, and the set of input nodes as V, X_{out} is the complete solution (node set with sequence), and x_t is the selected node at time t. The city to be visited (determined by the attention network and the current state) is gradually added to the current solution through chain rules until all nodes are traversed, as shown in equation (8):

$$P(X_{out}|V;w) = \prod_{t=1}^{N} P(x_{t+1}|x_t;w)\, P(x_1;w) \tag{8}$$

To train the attention network, the Actor-Critic framework is used in training.

5. Experiments and Results

5.1. Experimental Design

1) Dataset. For the TSP, this work uses the tsplib95 instances with different scales. For the sorting sequence optimization, this work collects the production data of a certain period on the production line of a machinery manufacturing company, with a total of 350 real steel plates data. The TSP problem and the sequential optimization are used to verify the effectiveness of the deep evolutionary algorithm.

2) Experiment environment. The experiment environment is: CPU processor is Intel(R) Xeon(R) Gold 5218 CPU@2.30GHz, 64G memory under Ubuntu operating system. The programming language is Python3.8 and the deep learning framework is Pytorch1.7.0 without GPU.

3) Comparison algorithms. To verify the performance of the deep evolutionary algorithm, a series of comparison algorithms are used, including GA, PSO, SA, ACO, Pointer Network, Transformer, and the deep evolutionary algorithm without 2-OPT.

4) Parameter settings. The relevant parameters of the Transformer network in the Actor-Critic training are set as follows: the embedding layer dimension of the encoder is 128, and the dimensions of the query vector, key vector, and value vector are all 128, Head=8, Layer=1, Inner=512. The population size of GA is 260, the mutation probability is 0.1, and the maximum number of iterations is 200. The initial temperature of SA is 20, the minimum temperature is 0.1, the maximum number of iterations is 200, and the number of repetitions at the same temperature is 5. The number of particles of PSO is 260, and the maximum number of iterations is 200. The maximum number of ACO iterations is 200, the maximum number of ants is 260, ρ =0.9, α =1, β =1. Pointer Network has one Layer, and the dimensions of the embedded layer and hidden layer are both 128. The maximum number of iterations of 2-OPT is 500.

The optimizer in training is Adam and the initial learning rate is 1e-4, while the batch size is 512. The training scale of TSP and sorting sequence optimization are both 10. In the TSP, the training data is uniformly randomly selected 10 coordinate data between 0 and 1 (10 steel plates are randomly selected for the sorting sequence optimization), and 2048 evaluation scenarios are generated.

5.2. Comparison with Other Algorithms

Table II shows the performance of different methods on TSP instances, where the time cost is in seconds and the node scale of test data ranges from tens to thousands. It can be seen that GA has achieved the four best performances on six test instances in the traditional meta-heuristics. While the deep network algorithm has a great advantage in time cost. Therefore, this work combines GA and deep network. Compared with other algorithms, our algorithm achieves the best performance. Moreover, further optimization by the 2-OPT algorithm ensures that the deep evolutionary algorithm can achieve better performance. But it also further increases in time cost, which is at the cost of increased time complexity.

TABLE II. THE PERFORMANCE OF DIFFERENT ALGORITHMS ON TSPLIB95 EXAMPLES.

case	*GA*		*SA*		*PSO*		*ACO*	
	Avg	*time*	*Avg*	*time*	*Avg*	*time*	*Avg*	*time*
berlin52	7607	7	7807	3	7900	20	7664	46
kroA150	28518	35	28753	5	31159	49	29708	397
a280	3011	53	3027	9	3088	111	2815	1359
fl417	13176	220	13511	14	13360	472	16569	2611
nrw1379	68763	611	68862	62	68743	1224	153010	30701
pr2392	457527	3299	459985	148	458674	3481	-	-

a. Among them, the matrix that ACO needs to maintain in the pr2392 case is too large, and the current test platform cannot calculate it, so the results are no longer listed.

case	*PN*		*TRANS*		*TRANS+GA*		*OURS*	
	Avg	*time*	*Avg*	*time*	*Avg*	*time*	*Avg*	*time*
berlin52	7984	0.03	7634	0.01	7591	7	**7542**	15
kroA150	31008	0.24	28761	0.15	27358	35	**26949**	47
a280	3214	0.39	2798	0.3	2763	53	**2643**	63
fl417	13251	0.76	13219	0.6	12016	220	**12008**	232
nrw1379	179517	2.2	93825	1.7	65274	611	**60278**	637
pr2392	1129395	3.6	909312	3	429366	3299	**413628**	3318

TABLE III. THE PERFORMANCE ON VALIDATION SETS OF DIFFERENT SCALE SORTING SEQUENCE OPTIMIZATION.

scale	*N=10*		*N=70*		*N=140*	
	Avg.f	***time***	***Avg.f***	***time***	***Avg.f***	***time***
RANDOM	32.22	-	223.38	-	443.75	-
FIXED	23.20	-	159.51	-	317.89	-
Transformer	22.85	0.01	157.61	0.02	314.88	0.14
GA	21.53	0.54	156.36	0.84	315.23	1.9
OUR-2-opt	21.53	0.54	156.21	0.84	314.15	1.9
OUR-TRANS	**20.26**	4.14	155.84	16.83	315.20	22.21
OURS	**20.26**	4.14	**155.57**	16.83	**313.76**	22.21

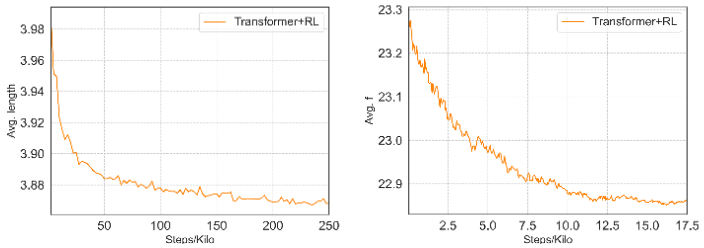
a. '-' means that the time cost is too short.

b. 'OUR-TRANS' means Transformer network is removed in our algorithm.

To further verify the effect of our algorithm on large-scale data, Table III presents the performance of our algorithm on the validation set of constrained large-scale sorting sequence optimization. RANDOM is the current method in the production line, which means a random sorting sequence of steel plates and a random sorting sequence of parts in the plate. FIXED represents the random sorting sequence of steel plates, but the sorting sequence of the parts in the plate is in the order of fixed dictionary id. Transformer, GA, Transformer+GA are all optimizations to the sorting sequence of steel plates, and the sorting sequence of parts in the plate is in the order of fixed dictionary id. GA+2-OPT means that GA optimizes the sorting sequence of steel plates, and 2-OPT optimizes the sorting sequence of parts in the plate. Our algorithm uses Transformer+GA to optimize the sorting sequence of steel plates, and 2-OPT optimizes the sorting sequence of parts in the plate. It can be observed from the Table III that the our method still achieved the least average number of frame clearing on different steel plate scales (10, 70, 140), which is the largest optimization range compared to the RANDOM method.

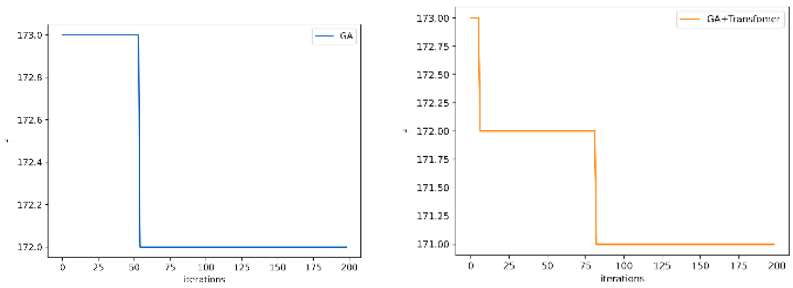
5.3. Network Effectiveness Analysis

Figure 5(a) shows the training curve of the attention network on the TSP training sample. Figure 5(b) shows the training curve of the attention network on the optimization of the sorting sequence. It can be learnt from Figure 5 that the attention network used in this work can achieve convergence in both of the above problems.



(a) TSP, the task scale is 10. (b) The sorting sequence optimization, and the task scale is 10.

Fig. 5. The training process of the transformer.



(a) GA alone. (b) GA+Transformer.

Fig. 6. The performance of algorithm in sorting sequence optimization.

5.4. Case Study

This section studies the optimization example of steel plate sorting sequence and parts sorting sequence. Figure 6(a) shows the optimization example (N=70) of the single GA algorithm in the steel plate sorting sequence while the parts sorting sequence is fixed. In the 51st generation, f dropped from 173 times to 172 times, and didn’ t change after that. Figure 6(b) shows the optimization example of the GA+Transformer algorithm in the steel plate sorting sequence while the parts sorting sequence is fixed. It can be seen that f reaches 172 times around the 5th generation. On this basis, the subsequent f drops to 171 times. The 2-OPT algorithm was used to further optimize the parts sorting sequence while the steel plate sorting sequence was fixed at this time, and the f dropped to 169 times, which is further optimized compared to the previous one.

It can be learnt from the above experiments that each module in our algorithm plays an important role and is closely integrated with each other, which has a good solution performance, especially for large-scale sequential optimization.

6. Conclusion

This work proposes a deep evolutionary path optimization algorithm. The algorithm uses the trained attention network to generate the initial solution, and injects the solution into the GA population to speed up the search, and then uses 2-OPT to further optimize the optimal solution in the population. The results show that the deep evolutionary algorithm has excellent performance on both problems, especially when the problem scale is large.

7. Acknowledgements

This work was supported in part by the National University of Defense Technology pre-research project under Grant ZK21-41.

8. References

[1] N. Yi, J. Xu, L. Yan, and L. Huang, “Task optimization and scheduling of distributed cyber–physical system based on improved ant colony algorithm,” Future Generation Computer Systems, vol. 109, pp. 134–148, 2020.

[2] O. Cheikhrouhou and I. Khoufi, “A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy,” Computer Science Review, vol. 40, p. 100369, 2021.

[3] W. Qingrong, Y. Zhanting, and Z. Qiuyu, “Study on transit scheduling optimization based on improved genetic-simulated annealing algorithm,” Application Research of Computers, vol. 29, no. 7, pp. 2461–2463, 2012.

[4] H. Li, D. Yang, W. Su, J. L¨u, and X. Yu, “An overall distribution particle swarm optimization mppt algorithm for photovoltaic system under partial shading,” IEEE Transactions on Industrial Electronics, vol. 66, no. 1, pp. 265–275, 2018.

[5] M. Al-Shalabi, M. Anbar, T.-C. Wan, and Z. Alqattan, “Energy efficient multi-hop path in wireless sensor networks using an enhanced genetic algorithm,” Information Sciences, vol. 500, pp. 259–273, 2019.

[6] W. Hongyun and Y. Shu, “Dynamic multipath load balancing based on ant colony algorithm in dcn,” Application Research of Computers, vol. 37, no. 07, pp. 2148–2156, 2020.

[7] M. Zhu, S. Yi, C. Yang, and R. Feng, “Research on rlga-based hardware evolution optimization technology,” in 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA). IEEE, 2020, pp. 188–193.

[8] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” arXiv preprint arXiv:1506.03134, 2015.

[9] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” arXiv preprint arXiv:1704.01665, 2017.

[10] W. Kool, H. Van Hoof, and M. Welling, “Attention, learn to solve routing problems!” arXiv preprint arXiv:1803.08475, 2018.