Brainstorm Optimization Algorithm with *K*-means Clustering for Disassembly Line Balancing Problems

PengKai Xiao
Liaoning Petrochemical University
Fushun, China
2050064275@qq.com

XiWang Guo
Liaoning Petrochemical University
Fushun, China and *Monmouth University*
West Long Branch, USA
x.w.guo@163.com

JiaCun Wang
Monmouth University
West Long Branch, USA
jwang@monmouth.edu

ShuJin Qin
Shangqiu Normal University
Shangqiu, China
sjchin@vip.126.com

Liang Qi
Shandong University of Science and Technology
Qingdao, China
qiliangsdkd@163.com

Yuanyuan Tan
Shenyang University of Technology
Shenyang, China
yytan@sut.edu.cn

***Abstract*—In the Internet era, the continuous innovation and progress of science and technology promote the renewal of electronic and electrical products and tend to shorten their life cycle. As the recycling rate of these waste products is very low, this causes a great waste of resources. How to disassemble and recycle valuable parts is a common problem faced by the world. In essence, the recycling of waste products by enterprises is to obtain most valuable parts and components from obsolete products to gain profits. This paper considers the traditional linear disassembly line, which is widely used in factories at present. By combining the Brainstorming Optimization (BO) algorithm with the *K*-means clustering algorithm, this work proposes a novel Improved Brainstorming Optimization algorithm to obtain the near optimal solution quickly. It is compared with an Artificial Bee Colony algorithm and Gray Wolf Optimization algorithm to verify its superiority in solving disassembly line balancing problems.**

Keywords—Disassembly line, brainstorming optimization algorithm, K-means clustering algorithm.

I. INTRODUCTION

The development of science and technology has brought great convenience to people's life. The new generation of products occupies the market at an incredible speed, which makes many of the older generation products obsolete or abandoned as end-of-life (EOL) ones. Many components in these EOL products are valuable, and throwing them away would not only waste resources but also cause serious environmental problems. Therefore, it is of great importance to recycle these EOL products.

How to separate these components well in a disassembly line so as to obtain the maximum disassembly profit is a Disassembly Line Balancing Problem (DLBP) [1-3]. Gungor *et al*. [4] point out that DLBP is an *NP*-hard problem. Li *et al*. [5] study the U-shaped partial DLBP with the maximal profit. Liang *et al*. [6] design a bilateral disassembly line with consumption rate and energy consumption as the target in consideration of the uncertainty of disassembly time.

Because U-shaped disassembly lines and bilateral ones require high skills of workers or robots, as well as layout planning of such lines, most factories do not adopt the above layout in an actual disassembly process. At present, a serial disassembly line is still the mainstream, which can be used to disassemble large scale EOL products with high disassembly efficiency, clear process division, and less equipment investment. Therefore, its improvement and optimization is of great significance.

Comparing a heuristic algorithm [7] and a mathematical programming method [8], an intelligent optimization algorithm [9] is faster in solving problems than an existing heuristic algorithm [7] and a mathematical programming method [8]. It can achieve better performance in solving large-scale problems. Grey Wolf Optimization (GWO) algorithm [10], Migratory Bird (MB) algorithm [11], Artificial Bee Colony (ABC) algorithm [12], Brainstorming Optimization (BO) algorithm [13] and so on have also been used to solve DLBP. Among them, the BO algorithm is proposed by Shi [14] in 2011. It brainstorms people from different fields with different views on the same issue and evolves towards the optimal view direction, achieving great success in solving task scheduling problems. Therefore, It is selected in this work to deal with DLBP. To make the solution faster, the improved *K*-means clustering algorithm [15-16] is combined with it to find the local optimal solution in each cluster, and then the global optimal solution is obtained by comparing all local optimal solutions. This paper has the following new contributions:

1) A DLBP mathematical model is established for the first time by Considering and weighing the switching cost of tasks within the same workstation and the number of workstations opened turned on. Such that the maximum disassembly profit is achieved.
2) Improved *K*-means clustering algorithm is innovatively combined with a BO. The local optimal solution of each cluster is calculated and the global optimal solution is obtained by comparing all of them and selecting the best one.

The rest of this article is arranged as follows. The research questions are described in Section II. Section III

This work is supported in part by NSFC under Grant Nos. 62003221, 62073069 and 51405075. This work is supported in part by Archival Science and Technology Project of Liaoning Province, No.2021-B- 004; LiaoNing Revitalization Talents Program under Grant No. XLYC1907166; the Natural Science Foundation of Shandong Province under Grant ZR2019BF004 and National Natural Science Foundation of China under Grant 61573089.

introduces the proposed algorithm. Section IV uses small, medium, and large cases to carry out numerical experiments and analyze the experimental results. Section V summarizes the work and states the future research directions.

II. Problem Statement

DLBP aims to optimize the allocation of tasks to the disassembly the workstations in the disassembly lines. To the end, we need to understand the disassembly process of a given product. This paper uses disassembly Petri Nets (*PN*) to model such a process, from which we can observe the precedence and conflict relationships among disassembly tasks, and obtain feasible disassembly.

A. Disassembly Petri Nets

As a graphic and mathematical model, Petri nets and their extensions, e.g., timed Petri nets, have found wide applications in areas such as in automated factories [17], semiconductor manufacturing [18], manufacturing systems [19], autonomous driving systems [20]. A Petri net consists of *places*, *transitions*, and *arcs* that connect places and transitions. Places are passive elements and can contain *tokens*. The distribution of tokens in places constitutes the *marking* of the net. Transitions are active components and model activities or events or actions. Transitions can *fire* if they are *enabled*. When the transition fires, it removes tokens from its input places and adds tokens to its output places.

Taking the washing machine of Fig. 1 as an example. Its *PN* model is shown in Fig. 2, in which each transition represents a disassembly task, and each place represents the components obtained after the disassembly task. There is a conflicted relationship between task t_2 and task t_3; only one of them can be executed. There is a precedence relationship between task t_2 and task t_5, in which t_5 can only be executed after the execution of task t_2. The conflict relation matrix and precedence relation matrix are formed from the conflict relation and precedence relation respectively. It is helpful to adjust the disassembly sequence to the feasible disassembly sequence when initializing the population in the algorithm so that disassembly becomes reasonable.

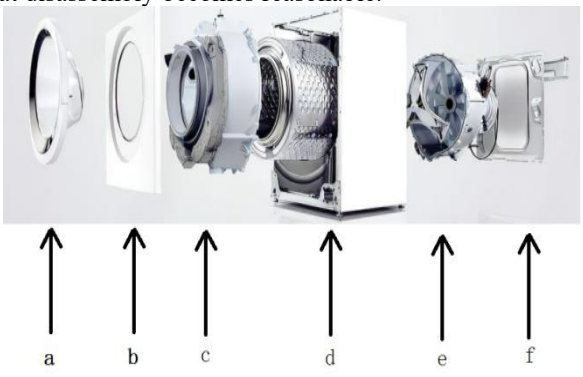

Fig. 1. Schematic diagram of a washing machine

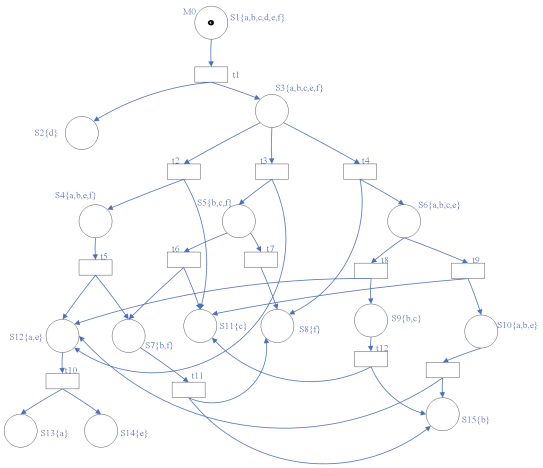

Fig. 2. Petri Net of washing machine

To build up an DLBP model, we assume that:

1) Each workstation has a series of disassembly stations.
2) One disassembly task can only be assigned to one disassembly station of one workstation.
3) At least one disassembly task is assigned to the workstation that is turned on.
4) The switching time between different workstations is short and can be ignored.
5) The switching time between tasks is known.
6) There is a precedence constraint relationship between disassembly tasks.

B Mathematical model

Based on the above description, the disassembly task is reasonably allocated to the disassembly station of the workstation, the disassembly method aiming at profit maximization is determined, and the corresponding linear mathematical model is established:

$$\max f = P - C_j - Z_{w,k} - C_w \quad (1)$$

$$Z_{w,k} \geq S_{j,j^{'}w}(x_{j,w,k-1} + x_{j^{'},w,k} - 1) \quad (2)$$

In the above model, j represents the disassembly task, w represents the number of workstations, and k represents the number of disassembly stations in the workstation. (1) is the objective function of the mathematical model, P represents the component value obtained by disassembly product, C_j represents the cost consumed by disassembly task j, C_w represents the cost consumed by opening the w workstation, $Z_{w,k}$ represents the switching time between adjacent tasks in the same workstation. Since this disassembly line adopts linear modeling, constraint (2) is added, $S_{j,j^{'},w}$represents the switching time between i and $i^{'}$ in workstation w, $x_{j,w,k-1}$ represents that task j is executed at the $k-1$ position of workstation w, $x_{j^{'},w,k}$ represents that the next task $j^{'}$ is executed at the $k-1$ position of workstation w. When the task is executed, x=1, otherwise x=0. The switch time occurs if and only if both tasks are executed.

Table 1. Recycling value of each component

Subassembly index	Resue value	Subassembly index	Resue value
1	253	9	79
2	12	10	179

3	241	11	13
4	228	12	100
5	128	13	50
6	192	14	50
7	115	15	66
8	49		

Table 2. Switching costs between adjacent tasks

Adjacent to the task	Switch costs	Adjacent to the task	Switch costs
1-2	13	4-9	5
1-3	11	5-10	15
1-4	12	5-11	12
2-5	9	6-11	15
3-6	15	8-12	13
3-7	6	9-13	6
4-8	11		

Table 3. Cost of product disassembly

Task index	disassembly cost	Task index	disassembly cost
1	21	8	15
2	22	9	32
3	16	10	32
4	23	11	12
5	19	12	15
6	26	13	24
7	24		

In this paper, the cost caused by switching time is inversely propotional to the cost workstations. That is, if there are a large number of workstations opened, the switching time will be reduced, but the workstation opening cost will increase. Therefore, we must balance these two aspects and allocate the task reasonably. Table 1 is the recovery value of each component, Table 2 is the switching cost between adjacent tasks, And Table 3 is the disassembly cost of the product. Set the start-up cost of each workstation to be 10. According to the mathematical model, we can calculate the objective function values (f=430).

$$\begin{aligned} f = P - C_j - Z_{j,j'} - C_w &= 253 + 12 + 128 + 179 \\ &\quad - 21 - 22 - 19 - 32 \\ &\quad - 13 - 15 \\ &\quad - 10 - 10 \\ &\quad = 430 \end{aligned}$$

B. Proposed Algorithm

BO was proposed by Professor Shi in 2011 [21]. It mimics people in different fields coming up with different ideas and then optimizes those ideas to get the best idea. So far, it has been applied in various fields and has shown good performance. Hao *et al.* [22] use the hybrid BO to solve the distributed mixed-flow shop scheduling problem, and Zhang *et al.* [23] use the improved BO to divide software and hardware. As far as I know, BO is rarely used in the field of disassembly lines. So, we think if the BO is slightly improved, the disassembly efficiency will be greatly improved. So, combining BO and the *K*-means clustering algorithm [24], using 1 minus the similarity between every two individuals as the distance between the individuals, the crossover operator is improved to maintain the diversity of the population, and the mutation operator is used to make the target evolve towards the optimal solution.

A. Encoding and Decoding

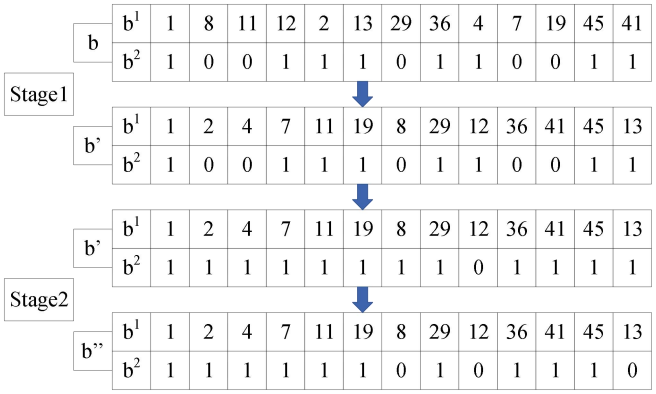

Fig. 3 Encoding and decoding processes

Fig. 3 shows the process of encoding and decoding. A two-stage heuristic method is used in this work. b^1 is the task disassembly sequence, b^2 is a binary vector, b^2 =1 means task is executed, and b^2 =0 means task is not executed. In the first stage, monte carlo random simulation is used to obtain the disassembly sequence b. Then, according to the precedence matrix provided by *PN*, the disassembly sequence is adjusted to match the actual situation and the sequence $b^{'}$ is obtained. In this case, the encoding may contain more than one disassembly sequence. In the second stage, select one sequence $b^{''}$ from the conflict relation set provided by *PN*.

B. Search process

IBO includes initialization population, *K*-means clustering, new individual generation, and selection process. monte carlo random simulation is used to generate a disassembly sequence during population initialization, and then the disassembly sequence is adjusted to a reasonable one according to the precedence relation set and conflict relation set. Clustering, to calculate the similarity between individuals, and then 1 minus the similarity to calculate the distance between the two individuals. For example, if the cluster number is 5, then 5 randomly selected individuals from the population serve as the five cluster centers, and then other individuals according to the nearest way belongs to the corresponding clusters, and then take the average of the data in each class as a clustering center, has been iteration. The clustering is completed when the clustering center no longer changes. At the stage of new individual generation, new individuals are generated by changing the disassembly sequence with improved crossover and mutation operations. In the selection stage, the locally optimal solution is firstly selected from each cluster, and then the global optimal solution is obtained by comparing the local optimal solution of each cluster.

K-means clustering, as shown in Algorithm 1, can quickly divide data into multiple clusters.

***Algorithm 1**: Clustering.*

Input: Initial population, number of centers *K*
Output: clusters.
Begin
while (Iterate through the initialization population, and take out two disassembly sequences *A* and *B* each time.)
Compare the tasks in *A* and *B*.
If $((A_i, A_{i+1}) == (B_j, B_{j+1}))$
The corresponding positions of disassembly sequence *A* and *B* have the same disassembly task, and record the occurrence times of this situation temp++.

endif
Calculate the distance *d* between *A* and *B*.
end
random select *K* individuals as the initial cluster centers.
for (*i*=0 to *N*, where *N* is the number of individuals)
if(*i* is closest to a cluster center)
put *i* in this cluster
end if
Calculate the average value of each cluster as the class center.
Repeat the above procedure until the cluster center no longer changes
End
End

The generation stage of new individuals is shown in Algorithm 2, where p_g is the probability of generating new individuals from old individuals, p_o is the probability of generating new individuals from cluster centers, and p_t is the probability of generating new individuals from two cluster centers.

Algorithm 2: New individual generation.

Input: Individuals in population, and the cluster
Output: New individual
Begin
For (*i*=0 to Number of population)
Randomly generate a value r_g between 0 and 1.
if ($r_g < p_g$) **then**
Randomly select a cluster and generate a random value r_0 in the range [0,1).
if ($r_o < p_o$) **then**
Select a cluster center at random and add a random value to it to generate a new individual
else
Select an individual from the selected cluster and add a random value to the solution for a new individual.
end if
Adjust the obtained solution through *K*-means phase.
else
Randomly generate a value r_t in the range [0,1).
if ($r_t < p_t$) **then**
Combine two clustering centers to generate new individuals.
else
Randomly select two individuals from each selected cluster to perform crossover and mutation operations to generate new individuals.
end if
Adjust the obtained solution through *K*-means phase.
end if
Screen the newly generated individuals and keep the good ones.

End.

C. Crossover and Mutation Operation

At the stage of generating new individuals, this work imitates the characteristics of biological inheritance in nature and selects the way of two-point crossover to carry out partial gene exchange in an interval way to maintain population diversity. The specific process of two-point crossover is :(1) randomly select two crossover points in the coding string of two paired individuals; (2) a portion of the chromosome between the two exchanged individuals is selected intersections. Fig. 4 shows a schematic diagram of crossover operations. The new individual formed after crossover operation has a certain probability of genetic variation. The mutation operation can make the algorithm have local random searchability. When the algorithm has approached the optimal solution neighborhood through the crossover operator, the mutation operator can speed up the convergence of the algorithm. Fig. 5 shows the schematic diagram of mutation operation.

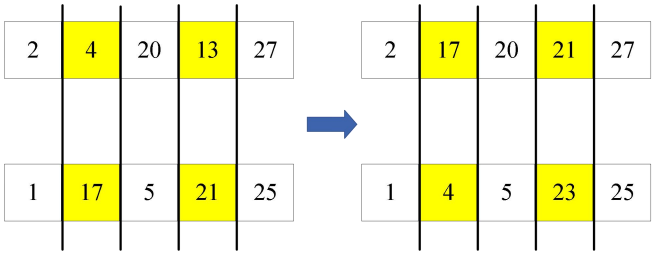

Fig. 4 Two points crossover operation process

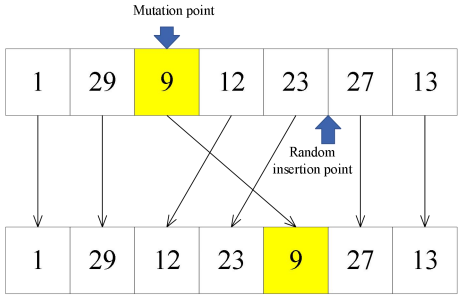

Fig. 5 A single point mutation

D. Experimental Results

a) Operating Environment and Test Case Study

To verify the feasibility and performance of the proposed mathematical model and algorithm, experiments are carried out from large, medium, and small cases, and IBO is compared with the ABC and GWO. All the above solving processes are implemented in IntelliJ IDEA 2020.2.1x64, operating system is Intel(R) Core (TM) I7-11800h @ 2.30 Windows 10.

As for the parameter settings of IBO: the initial number of individuals is 10, the number of clusters is 5, whether a new individual is determined by one or two old individuals, the probability p_g =0.8, the probability that a new individual is determined by the cluster center or other ordinary individuals is p_o =0.4, and the probability p_t =0.5. Determine whether two cluster centers or two other ordinary individuals will create a new individual. The number of iterations is 80.

We tested small, medium, and large cases, and the case description is shown in Table 4. Case 1 and case 2 were small-scale cases, case 3 was medium-scale, and case 4 was relatively large-scale.

Table 4. Case description

Case Num.	Instance	Number of Tasks
Case 1	Washing Machine	13
Case 2	Ballpoint Pen	20
Case 3	Radio	30
Case 4	Hammer Drill	46

b) Analysis of experimental results

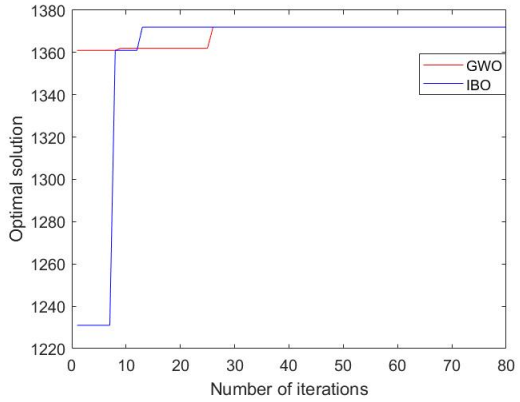

Fig. 6 Comparison of the performance of IBO and GWO in the washing machine case.

A washing machine is taken as an example to plot its iterative process, as shown in Fig. 6. It can be seen that the convergence speed of IBO is faster than that of the GWO, and it can find the near optimal solution in a shorter time. This shows that the performance of IBO is better.

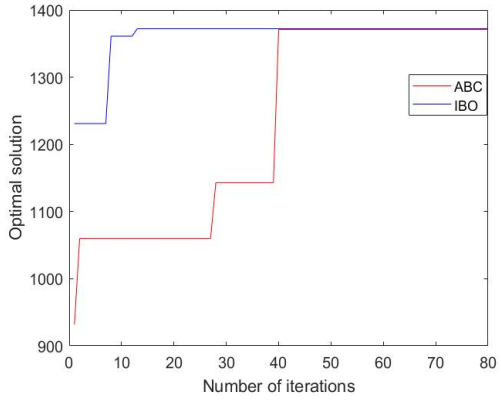

Fig. 7 Comparison of performance of IBO and ABC in the washing machine case.

To further verify the superiority of the performance of the improved algorithm, it is compared with GWO in Fig. 7. Similarly, the convergence speed of IBO is better than GWO.

Next, starting from small, medium and large disassembly cases, IBO, ABC, and GWO are used to solve them respectively, and the results and solving time are compared. The near optimal solution and time obtained by each algorithm are shown in Tables 5 and 6, where Gap= (Maximum Profit1-Maximum Profit2) /Maximum Profit2*100%, △T1 and △T2 are the comparison of solving speed of IBO algorithm, ABC algorithm and GWO respectively. After analyzing the results, the IBO is superior in both the solution time and the near optimal solution obtained, and its performance is better.

Table 5: Comparison between IBO and ABC

Case Num.	IBO		ABC		△T1	Gap/%
	Maximum Profit 1	Computation time	Maximum Profit 2	Computation time		
Case 1	1372.0	0.282s	1371.0	0.456s	-0.174s	0.07%
Case 2	2791.0	0.562s	2044.0	0.633s	-0.071s	36.5%
Case 3	4780.0	0.802s	3635.0	0601s	0.201s	31.5%
Case 4	6663.0	2.001s	6001.0	2.211s	-0.21s	11.03%

Table 6: Comparison between IBO and GWO

Case Num.	IBO		GWO		△T2	Gap/ %
	Maximum Profit 1	Computation time	Maximum Profit 2	Computation time		
Case 1	1372.0	0.282s	1375.0	0.491s	-0.209s	0
Case 2	2791.0	0.562s	743.0	0.357s	0.205s	280%
Case 3	4780.0	0.802s	4780.0	1.045s	-0.243s	0
Case 4	6663.0	2.001s	6661.0	3.189s	-1.188s	0.03%

From what has been discussed above，this paper selects the linear disassembly line selected by most factories, designs a linear mathematical model, and considers the switching time cost between disassembly lines. To solve this problem, IBO is combined with the improved *K*-means clustering algorithm, and 1 is used to subtract the similarity between individuals as the distance between individuals. To study the performance of the proposed IBO, the solution is combined with small, medium and large cases in the work, and the results are compared and analyzed. The results show that the IBO algorithm is better than the ABC and GWO in most cases, especially in large cases, the gap between solving speed and the near optimal solution is more obvious.

c) Algorithm time complexity

This section analyzes the complexity of the algorithms involved in IBO.

1) *K*-means clustering: select two individuals in turn, and then calculate the similarity between them, to calculate their distance. So we need to iterate over all the individuals in order (1+N) times N over 2). All individuals continued to be traversed, and the individuals are classified into different class clusters according to the size of the distance. The time complexity is O(N). In general, the time complexity of clustering is $O(N^2)$, where N is the number of initialized individuals.

2) Crossover operator: in a crossover operation, two individuals are randomly selected as the parent individuals, and a binary integer vector is randomly generated to determine the crossover position of the parent individuals. Therefore, two superclass individuals and binary integer vectors need to be traversed in time complexity O(J), where J is the total number of operations.

3) Mutation operator: the time complexity is O(1) for an operation at a specified location.

4) Local search: according to the local search process, the cycle includes K-means clustering, crossover operators, and mutation operators, where the time complexity is $O(\text{iteration times} *(J+N^2))$, where the iteration times are the iteration times of local search.

D. CONCLUSION

In this paper, we propose a class of DLBP that takes the switching time within the same workstation into account for the first time. Its goal is to achieve the maximal profit. Both *K*-means clustering algorithm and IBO are improved and then used to construct a new IBO. The algorithm changes the traditional clustering mode, obtains the disassembly sequence through monte carlo random simulation, and adjusts the sequence by combining the precedence relationship matrix and conflict matrix provided by the *PN*

model to make it conform to real disassembly restrictions. In the clustering process, similarity is used to replace the distance between individuals. The higher the similarity is, the closer the distance is. In this way, population clustering is realized. The clustered population is proved to be able to accelerate the IBO, and its advantages can be seen in its comparison with both ABC and GWO.

Our future work should focus on two directions: 1) Studying different layout of disassembly lines, such as a circular disassembly line; 2) Combining IBO with other optimization strategies to better solve practical problems [25-34].

REFERENCES

[1] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin and L. Qi. "Multi-objective U-shaped Disassembly Line Balancing Problem Considering Human Fatigue Index and An Efficient Solution," *IEEE Transactions on Computational Social Systems*. 2022.

[2] J. Wang, "Patient flow modeling and optimal staffing for emergency departments: A Petri net approach," *IEEE Transactions on Computational Social Systems*. 2022.

[3] X. W. Guo, M. C. Zhou, F. Alsokhiry, and K. Sedraoui, "Disassembly sequence planning: a survey," *IEEE/CAA J. Autom. Sinica,* vol.8, no.7, pp.1308-1324, 2021.

[4] A. Gungor and S. M. Gupta. "Disassembly line in product recovery," *Int. J. Prod. Res.*, vol. 40, pp. 2569-2589, 2002.

[5] Z. X. Li and M. N. Janardhanan, "Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem," *Expert Syst. Appl.*, vol. 183, pp. 115431, 2021.

[6] J. Y. Liang, S. S. Guo, Y. F. Zhang, W. F. Liu and S. W. Zhou, "Energy-Efficient Optimization of Two-Sided Disassembly Line Balance Considering Parallel Operation and Uncertain Using Multi-objective Flatworm Algorithm," *Sustainability*, vol. 13, no. 6, pp.3358,2021.

[7] W. Z. Wang, B. Y. Jia, S. P. Simonovic, S. Q. Wu, Z. W. Fan and L. Ren, "Comparison of Representative Heuristic Algorithms for Multi-Objective Reservoir Optimal Operation," *Water Resour. Manag.*, vol. 35, pp. 2741-2762, 2021.

[8] Andreas, April K, Smith and J Cole, "Mathematical Programming Algorithms for Two-Path Routing Problems with Reliability Considerations," *INFORMS J Comput*, vol. 20, no. 4, pp. 553-564, 2008.

[9] J. Y. Liang, S. S. Guo, B. G. Du, Y. B. Li, J. Guo, Z. J. Yang and S. B. Pang, "Minimizing energy consumption in multi-objective two-sided disassembly line balancing problem with complex execution constraints using dual-individual simulated annealing algorithm," *J. Clean. Prod.*, vol. 284, no. Feba15, pp. 125418.1-125418.20, 2021.

[10] X. W. Guo, Z. W. Zhang, L Qi, S. X. Liu, Y. Tang and Z. Y. Zhao. "Stochastic Hybrid Discrete Grey Wolf Optimizer for Multi-objective Disassembly Sequencing and Line Balancing Planning in Disassembling Multiple Products", *IEEE Trans. Autom. Sci. Eng.*, vol.19, no.3, pp. 1744-1756, July 2022.

[11] Q. X. Xiao, X. P. Guo and D. Li, "Partial disassembly line balancing under uncertainty: robust optimisation models and an improved migrating birds optimisation algorithm," *Int. J. Prod. Res.*, vol. 59, no. 10, pp. 2977-2995, 2021.

[12] S. W. Wang, X. P. Guo and J. Liu, "An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times," *Eng. Optim.*, vol. 51, no.11, pp. 1920-1937, 2019.

[13] Y. H. Shi, J. Q. Xue and Y. L. Wu. "Multi-Objective Optimization Based on Brain Storm Optimization Algorithm," *Int. J. Swarm. Intell. Res.*, vol. 4, pp. 1-21, 2013.

[14] Y. H. Shi, "An optimization algorithm based on brainstorming process," *Int. J. Swarm. Intell. Res.*, vol. 2, no. 4, pp. 35-62, 2011.

[15] S. Na, L. Xumin and G. Yong, "Research on *K*-means Clustering Algorithm: An Improved *K*-means Clustering Algorithm," *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 63-67, 2010.

[16] R. M. Li, Y. F. Huang and J. Wang, "Long-term traffic volume prediction based on K-means Gaussian interval type-2 fuzzy sets," *IEEE/CAA Journal of Automatica Sinica*, pp. 1-8, 2019.

[17] B. KARLIK, M. UZAM, M. CINSDIKICI and A. H. JONES, "Neurovision-based logic control of an experimental manufacturing plant using neural net le-net5 and automation Petri nets," *J Intell Manuf.*, vol. 16, pp. 527-548, 2005.

[18] M. C. Zhou and F. DiCesare, Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Norwell, MA, USA: Kluwer, 1993.

[19] J. Wang, M. Zhou, and Y Deng. "Throughput analysis of discrete event systems based on stochastic Petri nets," *International Journal of Intelligent Control and Systems,* vol. 3 (3), 343-358, 1999

[20] Z. Zhang, J. Liu, G. Liu, J. Wang, and Y. Zhang, " Robustness verification of swish neural networks embedded in autonomous driving systems " , *IEEE Transactions on Computational Social Systems*. 2022. Accepted 05-27-2022.

[21] Y. Shi. "Brain storm optimization algorithm," in *International Conference in Swarm Intelligence*, *pp*. 303-309, 2011.

[22] J. Hao, J. Li, Y. Du, M. Song, P. Duan and Y. Zhang, "Solving Distributed Hybrid Flowshop Scheduling Problems by a Hybrid Brain Storm Optimization Algorithm," *IEEE Access*, vol. 7, pp. 66879-66894, 2019.

[23] T. Zhang, C. F. Yang and X. Zhao, "Using Improved Brainstorm Optimization Algorithm for Hardware/Software Partitioning," *Appl. Sci.*, vol. 9, pp. 866, 2019.

[24] E. Tuba, I. Strumberger, N. Bacanin, D. Zivkovic and M. Tuba, "Cooperative clustering algorithm based on brain storm optimization and *K*-means," *International Conference on Radioelektronika.*, pp. 1-5, 2018.

[25] H. Han, W. Li, J. Wang, G. Qin, and X. Qin, "Enhance explainability of manifold learning," *Neurocomputing. Accepted* 05-28-2022.

[26] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, 110-120, 2021.

[27] B. Hu and J. Wang, "Deep learning based hand gesture recognition and UAV flight controls," *International Journal of Automation and Computing*, vol. 17, no. 1, 17-29, 2020.

[28] X. W. Guo, M. C. Zhou, S. X. Liu, L. Qi, " Lexicographic Multiobjective Scatter Search for the Optimization of Sequence-Dependent Selective Disassembly Subject to Multiresource Constraints " *IEEE Transactions on Cybernetics*., vol.50, no.7, pp.3307-3317.July 2020.

[29] Y. J. Ji, S. X. Liu, M. C. Zhou, Z.Y. Zhao, X.W. Guo, L Qi. " A machine learning and genetic algorithm-based method for predicting width deviation of hot-rolled strip in steel production systems,"*Information Sciences*, vol. 589. pp. 360-375.

[30] Z. Zhao, S. Liu, M. Zhou and A. Abusorrah, "Dual-Objective Mixed Integer Linear Program and Memetic Algorithm for an Industrial Group Scheduling Problem," *in IEEE/CAA Journal of Automatica Sinic*a, vol. 8, no. 6, pp. 1199-1209, June 2021.

[31] Z. Zhao, S. Liu, M. Zhou, X. Guo and L. Qi, "Decomposition Method for New Single-Machine Scheduling Problems From Steel Production Systems," *in IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1376-1387, July 2020.

[32] Z. Zhao, M. Zhou, and S. Liu, "Iterated Greedy Algorithms for Flow-shop Scheduling Problems: A Tutorial," *in IEEE Transactions on Automation Science and Engineering*, 2021, doi: 10.1109/TASE.2021.3062994.

[33] A. S. Asan, Q. Kang, O. Oralkan, and M. Sahin, "Entrainment of cerebellar Purkinje cell spiking activity using pulsed ultrasound stimulation," Brain Stimul, vol. 14, no. 3, pp. 598-606, May-Jun 2021, doi: 10.1016/j.brs.2021.03.004.

[34] Y. Wang, Q. Kang, Y. Zhang, and X. Liu, "Optically computed optical coherence tomography for volumetric imaging," Opt Lett, vol. 45, no. 7, pp. 1675-1678, Apr 1 2020, doi: 10.1364/OL.382045.