ISBN: 978-981-18-5852-9
WCSE 2022 Spring Event: 2022 9th International Conference on Industrial Engineering and Applications
doi: 10.18178/wcse.2022.04.113

Reconfigurable Flexible Assembly Line Balancing Problem Based on Reinforcement Learning

Jingzhao Gan[1], Long Zeng[1+], Fengyuan Shi[1]

[1] Intelligent Manufacturing and Machine Vision Center, Tsinghua University, Shenzhen, China

Abstract. Because of the variability of models, constraints and objectives of reconfigurable flexible assembly line, there are few methods specially applied to reconfigurable flexible assembly line. The universality and learning ability of reinforcement learning give it the potential to solve such problems. This paper focuses on the number of assembly workstations required to form an assembly line and the cycle time of the assembly line to complete the task, and takes the average assembly workstation rate as the optimization goal. The paper attempts to solve this problem through reinforcement learning, which is rare in this field, and experiments are carried out with some cases. The results show that reinforcement learning is feasible to improve the work efficiency of reconfigurable flexible assembly line.

Keywords: reconfigurable flexible assembly line, maximize average workstation rate, reinforcement learning

1. Introduction

The assembly line was first introduced into Ford manufacturing company by Henry Ford, and the assembly line balancing problem (ALBP) was proposed by Salveson in 1955 and has been widely studied in recent decades. The balance problem of assembly line is tightly related to the production efficiency, cost and quality of products [1].

In the production process, due to the differences of site, equipment and other factors, the more common classification method used in the application is to classify according to the optimization objectives. (a) SALBP-1 reduces the number of workstations over a given period of time. (b) SALBP-2 reduces cycle time on a given number of workstations. (c) SALBP-E maximizes production line efficiency with variable cycle times and number of workstations [2]. With the development of research in this field and the complexity of modern production, more optimization goals have been focused on, such as minimizing production costs, minimizing carbon footprint, minimizing energy consumption, minimizing the cost of robot installation, and maximizing production efficiency [3].

In the past, the methods of solving such problems have also been deeply concerned, including exact, heuristic, and meta-heuristic methods. For instance, Bautista and Pereira [4] utilized dynamic programming, whereas Thangavelu and Shetty [5] used mathematical programming. Ogan and Azizoglu [6] drew on the branch and bound method to solve ALBPs. Furthermore, the studies in the literature relied on various meta-heuristic algorithms, out of which genetic algorithms, Particle Swarm Optimization, and ant colony optimization had been utilized more than others [3]. In addition to these three algorithms, some researchers also use other meta heuristics, such as Memes algorithm, Simulated annealing, Petri net, Tabu search et al. Some studies even combine several meta heuristics. Lin, Che, Chiang, Che, and Chiang [7] created a mixed model of genetic algorithm. Dong, Zhang, and Xiao [8] relied on Particle Swarm Optimization and Simulated annealing to solve the problems.

In general, the assembly line balancing problem has been studied by many scholars, but the methods used are mainly heuristic and metaheuristic algorithms. In recent years, artificial intelligence algorithm has been greatly developed. There is no doubt that it is a hot player in the field of computer algorithm. In 2016, Google deepmind's computer Go program alphago beat multiple world champion Lee sedol with four wins

+ Corresponding author.
E-mail address: zenglong@sz.tsinghua.edu.cn.

and one loss. In the spring of 2017, alphago beat the world's No. 1 player Ke Jie with 3-0. The excellent performance of learning algorithm in Go makes people notice that reinforcement learning method also has more application possibilities in other fields. Reinforcement learning algorithm is a kind of decision-making that allows agents to continuously interact with the environment to obtain rewards, and agents use past experience to automatically learn how to correctly select the decisions that can obtain the maximum long-term benefits in different states. The most commonly used reinforcement learning methods are Q-learning and Markov decision forms. It is mentioned in the research of Fangxing Li and Yan Du [9] that reinforcement learning can be applied to power system to solve power market bidding, distribution market transaction and distribution problems. According to the literature review completed by eisha akanksha [10], we can know that reinforcement learning algorithm has achieved success in many fields, including game AI, investment decision-making problem, medical management problem and transportation scheduling problem.

However, there is not much research on reinforcement learning method in assembly line optimization. In theory, reinforcement learning is very suitable for assembly line optimization. RL has been shown to have the ability to learn many different tasks using the same algorithm, which may mean great potential in engineering [11]. In addition, the assembly line balance problem can be regarded as a continuous optimization problem, which is the strength of reinforcement learning. Moreover, different assembly line balancing problems may have different objectives and constraints. When general algorithms want to solve personalized problems, they may need to modify many parameters or models, but reinforcement learning can control the optimization direction only by modifying the reward function.

Based on the above situation, this study will rely on the reconfigurable flexible assembly line developed by the intelligent manufacturing and machine vision center of Tsinghua University to build a multi constraint and single objective assembly line balance mathematical model, and use reinforcement learning to further study the assembly line optimization problem. The biggest feature of reinforcement learning is its general nature, which allows learning almost anything through general algorithms. The balance problem of diversified assembly lines is like tailor-made for it.

2. Problem Description and Mathematical Modeling

2.1. Reconfigurable Flexible Assembly Line

This study will rely on the reconfigurable flexible assembly line developed by the intelligent manufacturing and machine vision center of Tsinghua University. The equipment is composed of many assembly workstations with manipulator as the core, including various tool libraries, fixture libraries, handling manipulator and handling AGV trolley. It is described in more detail in the articles of Shi F [12] and Xiao L [13].

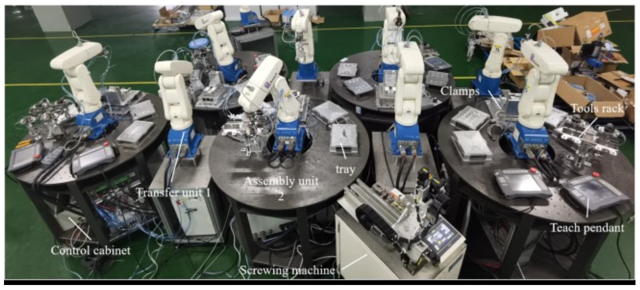

Fig. 1. An overview of the reconfigurable flexible assembly line.

2.2. Problem Description

The reconfigurable flexible assembly line balancing problem studied in this paper can be defined as: On the basis of having multiple (at least one assembly line) all-round workstations, assign work tasks to the workstations and form these all-round workstations into multiple assembly lines. In this study, there is no strong constraint on the assembly workstation and assembly cycle time, hoping to achieve the highest efficiency of overall resources and the balance between each workstation.

Before establishing the mathematical model of this study, the following assumptions need to be made to avoid unnecessary misunderstanding, eliminate irrelevant variables and simplify the mathematical model. The assumptions are as follows:

- All assembly equipment is based on the assembly equipment developed by the intelligent manufacturing and machine vision center of Tsinghua University. The equipment is composed of many assembly workstations with manipulator as the core, including various tool libraries, fixture libraries, handling manipulator and handling AGV trolley. All assembly workstations can complete each task freely according to the needs of assembly tasks by replacing tools, fixtures and processes.
- All assembly equipment here are sufficient to meet at least one assembly line constituting any assembly scheme.
- Assembly station and assembly workstation are different concepts. Assembly station is an abstract concept, which may consist of zero to several assembly workstations. All assembly workstations in the same assembly station are mechanical equipment that undertake the same task.
- In the process of calculation, it is assumed that the replacement time of tools and fixtures can be ignored.
- In the production process, all resources are sufficient, and the shutdown caused by insufficient raw materials does not need to be considered.
- During the production process, the storage and transfer time of raw materials, intermediate products and finished products will not affect the calculation of assembly scheme.

2.3. Mathematical Modeling

In order to adapt to the input requirements of reinforcement learning, the mathematical modeling of this study will be completed in the form of formulas and matrix. For the convenience of description, the relevant notations used to build the model are given below.

TABLE I. DEFINITION OF NOTATIONS

Notations	**Definition**
n	Sum of assembly tasks
N	Sum of stations
i,j	Serial number of task or station, i,j∈(1,2,3,······,n)or(1,2,3,······,N)
AS(i)	Assembly station with serial number "i"
task(i,j)	Task relationship function, which represents the relationship between task i and task j, i,j∈(1,2,3,······,n)or(1,2,3,······,N) task (i, j) = 1, i is the preceding operation of J; task (i, j) = 9, i and j are independent; task (i, j) = 0, i and j are the same operation
task_m	Task relation matrix, which records the matrix of all task relationship functions
S(i,j)	Station relationship function, which represents the relationship between stations i and j, i,j∈(1,2,3,······,n)or(1,2,3,······,N) If S(i, j) = 1, it means that i is the front station of j; if S(i, j) = 9, it means that stations i and j are independent; if S(i, j) = 0, it means that i and j are the same station
S_m	Station relation matrix, which records the matrix of all station relations functions
Inf_s(i,j)	Station information function, representing the relationship between station i and task j i∈(1,2,3,······,N)，j∈(1,2,3,······,n) Inf_ s (i, j) = 1, process j is executed by station i, inf_ s (i, j) = 9, process j is not executed by station i
Inf_m	Station information matrix，which records the matrix of all station information functions
Loc_n(i)	Workstation number function, which represents the number of workstations contained in station i, i∈(1,2,3,······,N)

Notations	Definition
Loc_m	Workstation number matrix, which records the matrix of all workstation number functions
t(i)	Task time function, the time required to complete task i, i∈(1,2,3,······,n)
tlist	Task time matrix, which records the matrix of all task time functions
T(i)	Station time function, the time required to complete all tasks at station i , i∈(1,2,3,······,N)
Tlist	Station time matrix, which records the matrix of all station time functions
C	Cycle time
S.N	The number of all workstations in the assembly scheme
Sar	Average workstation assembly rate
A_m	The characteristic matrix of Agent is composed of Inf_m , Loc_m, S_m and Tlist

In this paper, the mathematical model is as follows:

Assembly station and assembly workstation are different concepts. A station may contain multiple assembly workstations that complete the same task. When the station does not contain assembly workstation, although it does not undertake any task and has nothing to do with the scheme temporarily, the station still exists in the abstract concept.

$$S(i) \equiv 1 \quad i = 1,2\cdots\cdots N \tag{1}$$

The value of task relation matrix.

$$\text{ta}sk\_m = \begin{pmatrix} task(1,1) & \dots & 1,n) \\ \vdots & \ddots & \vdots \\ task(n,1) & \cdots & n,n) \end{pmatrix} \tag{2}$$

The value of station relation matrix.

$$S\_m = \begin{pmatrix} S(1,1) & \dots & N) \\ \vdots & \ddots & \vdots \\ S(N,1) & \cdots & N) \end{pmatrix} \tag{3}$$

The value of station information matrix.

$$Inf\_m = \begin{pmatrix} Inf\_s(1,1) & \dots & \_s(1,n) \\ \vdots & \ddots & \vdots \\ Inf\_s(N,1) & \cdots & s(N,n) \end{pmatrix} \tag{4}$$

The value range of workstation number function.

$$\text{Loc}\_n(i) \geq 0 \quad i = 1,2\cdots\cdots N \tag{5}$$

The value of workstation number matrix.

$$Loc\_m = (Loc\_n(1), Loc\_n(2)\cdots\cdots Loc\_n(N)) \tag{6}$$

The value of station time matrix.

$$tlist = (t(1), t(2)\cdots\cdots t(n)) \tag{7}$$

The value of station time function.

$$T(i) = \frac{\sum_{Inf\_m(i,j)=1} t(i)}{Loc\_n(i)} \quad j = 1,2\cdots\cdots N \tag{8}$$

The value of station time matrix.

$$Tlist = (T(1), T(2)\cdots\cdots T(N)) \tag{9}$$

The value of cycle time.

$$C = \max(Tlist) \tag{10}$$

The value of S.N.

$$S.N = \sum_{1}^{N} Loc\_m \tag{11}$$

The method in this paper is mainly applied to the balance of reconfigurable flexible assembly line. The assembly line is characterized by multiple all-round assembly workstations, and each workstation can complete each task independently. Therefore, this study focuses on the overall assembly efficiency. This requires that not only the cycle time be minimized, but also the number of assembly stations needed to make up the assembly line be minimized. Overall, pursuing the maximum average workstation assembly rate is a good choice.

Maximize average workstation assembly rate.

$$\max : Sar = \frac{1}{C \times S.N} = \frac{1}{\max(Tlist) \times \sum Loc\_m} \tag{12}$$

3. Method

Reinforcement learning is an exploratory solution method. It has a long-term vision and always has a good performance in finding the optimal solution of the problem. If a problem can be described or transformed into a sequential decision problem, and the state, action and reward can be defined, reinforcement learning is likely to find its optimal solution. As we know, in the optimization of assembly line balance scheme, for the same assembly problem, any two assembly schemes can be transformed into each other through several times of station exchange, increase or decrease and task exchange, and this transformation process is very similar to Markov decision process. Theoretically, given an initial solution to the assembly problem, it is possible to find the optimal solution through continuous transformation, which is why the assembly line balance optimization problem can be solved by reinforcement learning method.

3.1. Agent Feature Design

According to the requirements of reinforcement learning, there is an agent as the training object. The agent designed in this study will consist of the most basic assembly scheme. For any assembly task with n processes, the initial characteristics of agent can be designed in the following ways.

Assume that each assembly task is completed by a station alone.

$$S(i,j) = task(i,j) \tag{13}$$

The station matrix S_m is obtained from *S(i,j)*.

Station time function T(i) is equal to task time function *t(i)*.

$$\mathrm{T(i) = t(i)} \tag{14}$$

The station time matrix Tlist is obtained from the station time function *T(i)*.

Initialize station information function *Inf__ s(i,j)*.

$$Inf\_s(i,j) = \begin{cases} 1 & i = j \\ 9 & i \neq j \end{cases} \quad i,j = 1,2\cdots\cdots n \tag{15}$$

The station information matrix Inf_m is obtained from the station information function .

Initialize workstation number function *Loc_n(i).*

$$Loc\_n(i) = 1 \quad i = 1,2\cdots\cdots n \tag{16}$$

The workstation number matrix Loc_m is obtained from the workstation number function.

Calculate the average workstation assembly rate, *Sar*.

Station matrix *S_m*, Station time matrix Tlist, station information matrix *Inf_ m*, workstation number matrix *Loc_ m* and average workstation assembly rate Sar jointly form the basic assembly scheme. The basic assembly scheme is the basic feature of agent. The characteristic matrix *A_m* of agent is composed of *S_m, Tlist, Inf_m* and *Loc_m*.

3.2. Reward Function Design

The design of reward function is very important in reinforcement learning. Reward is the feedback that the agent gets from the environment after performing actions, and it is the basis of reinforcement learning for optimization. The agent will use the previous rewards as experience to guide the strategy of choosing actions in the future. In the assembly line balance optimization problem in this paper, the reward value design will be based on the average workstation assembly rate, in the following form:

$$reward = new\_Sar - Sar - u \tag{17}$$

The new_Sar is the average workstation assembly rate after each update. The Sar is the average workstation assembly rate before the action is performed. The u is a very small value, usually the ratio of the initial SAR to the set number of iterations. The function of the u is to punish when an agent stops optimization, so as to reduce the possibility of falling into local optimal solution. When constraints need to be added, the reward value can be modified. When the agent exceeds the constraint, give it a penalty.

3.3. Network Design Based on Deep Reinforcement Learning

The deep reinforcement learning method in this paper refers to the papers of deep reinforcement learning tutorial and nature dqn in ICML 2016 [12].Therefore, it will include two networks, Q network and target Q network. The structure and initial parameters of the two networks are completely consistent. In this study, a five layer Q network is used. The neuron parameters of each layer are $2n^2 + 2n$, n^2,50,50,5. In the course of the experiment, the five layer neural network has been able to calculate the required results. How to better design the parameters of the network is not the core problem of this paper, so it will not be carried out too much.

3.4. Action Design

In reinforcement learning, the effectiveness of optimization action is directly related to the calculation results of optimization algorithm. In this paper, the optimization methods commonly used by industrial engineers in engineering are referred to as the basic action unit. At the same time, considering that not every action is correct and needs a certain regret space, the reverse merge and reverse split actions are further designed.

3.4.1. Merge: The function of merge is to transfer all tasks in a station to its front station or rear station. The original station becomes invalid, that is, the number of workstations contained in this station is zero, but it is still reflected in the matrix and mathematical formula, which has no practical significance. This action can reduce the number of assembly workstations. The use method is as follows:

a) Input the characteristic matrix of agent

b)Calculate the selection matrix choice_Tlist, according to the value in choice_Tlist is weighted to randomly select the station i to be merged. The significance is that the shorter the time, the higher the probability that the stations will be merged

$$choice\_Tlist = -[Tlist - \max(Tlist)] \tag{18}$$

c)Find the set LR_S of front and rear stations of station I, and randomly select station j from LR_S

d)Find the front station L_S of station i other than station j or the rear station R_S other than station j

e)Process the station matrix S_m. It needs to be executed in order

$$S[k][j] = 1, S[k][j] = 9 \quad k \in L\_S \tag{19}$$

$$S[j][l] = 1, S[i][l] = 9 \quad l \in R\_S \tag{20}$$

$$S[i][j] = 9, S[j][i] = 9 \tag{21}$$

f)Process the Station time matrix TList

$$Tlist[j] = \frac{(Old\_Tlist[j] \times Loc\_m[j] + Tlist[i] \times Loc\_m[i])}{Loc\_m[j]} \tag{22}$$

g)Process the workstation number matrix Loc_m

$$Loc\_m[i] = 0 \tag{23}$$

h)Process the Station information matrix Inf_m

$$\begin{matrix} Inf\_m[j][k] = 1 \\ Inf\_m[i][k] = 9 \end{matrix} \quad k \in \forall Inf\_m[i][k] = 1 \tag{24}$$

i)Calculate the average workstation rate Sar after executing the action, and update all characteristics of the agent

3.4.2. Splitting: The function of splitting is to select the station with the longest time, add a new workstation to complete the same work, and shorten the time for the station to complete a task. The use method is as follows:

a)Find the longest station j in TList

$$j = \arg\max(Tlist) \tag{25}$$

b)Add a workstation to station j

c)Update the function T (J) and matrix TList

$$T(j) = Tlist[j] = \frac{Old\_Tlist[j] \times Loc\_m[j]}{Loc\_m[j]+1} \tag{26}$$

d)Update the function T (J) and matrix TList

e)Update the matrix Loc_m

$$Loc\_m[j] = Old\_Loc\_m[j]+1 \tag{27}$$

f)Update the function T (J) and matrix TList

g)Calculate the average workstation rate Sar after executing the action, and update all characteristics of the agent

3.4.3. Rearrangement: The function of rearrangement is to properly exchange the tasks undertaken by the station within the scope of constraint permission. This method is helpful to increase the exploration scope of the solution space of the algorithm. The application methods are as follows:

a)Eliminate the stations that are not allowed to change the operation position and the stations with multiple front or rear stations according to the constraints

b)Eliminate stations without exchange objects

c)Randomly select station i from the remaining stations

d)Find the interchangeable front and rear station set To_rea of station i, and randomly select station j in this set

e)Exchange the tasks contained in station i and station j, and modify the information matrix Inf_ m. The following five steps need to complete the exchange in strict order

$$\begin{gathered} Inf\_m[i][k], Inf\_m[j][m] = Inf\_m[j][k].Inf\_m[i][m] \\ k = \{k \mid Inf\_m[j][k]=1\}, m = \{m \mid Inf\_m[i][m]=1\} \end{gathered} \tag{28}$$

f)Use the Station relation matrix S_m to find the front station set IL_P and the rear station set IR_P corresponding to station i, and the front station set JL_P and the rear station set RL_P of j

$$S\_m[i][k]=1, S\_m[k][i]=9 \quad k \in JR\_P \tag{29}$$

$$S\_m[k][i]=1, S\_m[i][k]=9 \quad k \in JL\_P \tag{30}$$

$$S\_m[j][k]=1, S\_m[k][j]=9 \quad k \in IR\_P \tag{31}$$

$$S\_m[k][j]=1, S\_m[j][k]=9 \quad k \in IL\_P \tag{32}$$

g)Exchange matrix Tlist

$$Tlist[i], Tlist[j] = Tlist[j], Tlist[i] \tag{33}$$

h)Exchange matrix Loc_m

$$Loc\_m[i].Loc\_m[j] = Loc\_m[j], Loc\_m[i] \tag{34}$$

i)Update all the characteristics of the agent

3.4.4. Reverse merge: This action is the reverse of merge. Its function is to restore the multi tasking station to the position before merging. In addition to the characteristics of the agent, the input of this action also needs the station matrix Old_S_m and time matrix Old_Tlist of the original scheme.

a)Use Inf_m to find the station set To_rm that undertakes two or more tasks

b)Randomly select station i from To_rm

c)Restore matrix Inf_m, Loc_m,Tlist and S_m

d)Calculate the average workstation rate Sar after executing the action, and update all characteristics of the agent

3.4.5. Reverse splitting*:* Reverse splitting is the opposite action of splitting, which is to reduce one station with multiple workstations.

a)Use matrix Loc_m to find the station set RS with multiple workstations

b)Randomly select station i from RS

c)Restore matrix Loc_m and Tlist

d)Calculate the average workstation rate Sar after executing the action, and update all characteristics of the agent

3.5. The Core of the Algorithm

In this study, the reconfigurable flexible assembly line balance optimization algorithm based on deep reinforcement learning is designed according to the following methods .For ease of understanding, the pseudo code is given as follows.

```
Begin:
  input A_m , C_m = [[S_m] , [Tlist] , [Inf_m] , [Loc_m]]
  Approach the target through preprocessing
  action = [Merge() , Splitting() , Rearrangement() , Reverse merge() , Reverse splitting()]
  Q network , target Q = net(w) , new(w') and w' = w
  experience playback set D = []
  while True:
    Q = net(w , C_m)
    choice i by $\epsilon$ − greedy method according to Q
    A = action[i] , C_m' = A(C_m) and R = new_Sar – Sar – u
    D.append([C_m , C_m' , A , R])
    y1,y2 = []
    for j in range(m):  // M is the number of optimized samples taken
      sample = random.choice(D)
      Q = net(w , sample[C_m,A]) and y1.append(Q)
      target Q = net(w', sample[C_m']) and y2.append(target Q)
    Loss = $\Sigma$(y1-y2)**2/m
    Update Q network parameters through loss
    w' = w for every k cycle
    C_m = C_m'
    If Completion conditions reached: break
  output=max(sar , A_m)
```

4. Example Analysis

Otto et al [14] defined a new challenging data set based on the main problem parameters in 2013, which can be applied and extended to any generalized ALBP. At the same time, Morrison [15] applied the branch, boundary and memory algorithm to the case in 2014 for comparison. Although their goal is not to pay full attention to the efficiency of the assembly line, the large number of calculation data is very rare. SALBP-1 goal has a very high correlation with the efficiency of assembly line, so it has great reference value compared with their calculation results.

This paper randomly selects some small and medium-sized cases from the data set for testing. In all cases, the cycle time of Morrison's calculation results must be less than 1000, but reinforcement learning has no time limit.

TABLE II. COMPARISON OF RL AND HEURISTIC ALGORITHMS

Name	Morrison et al.		RL without constraints		
	S. N	*Sar1*	*S.N*	*Sar2*	*Gap with Sar1 (%)*
instance_n=50_1	8	0.000125	***161***	0.000130	4
instance_n=50_7	7	0.000143	***13***	0.000159	11.19
instance_n=50_51	12	0.0000833	***12***	0.0000920	11.04
instance_n=50_326	33	0.0000303	***162***	0.0000351	15.84
instance_n=50_198	28	0.0000357	***518***	0.0000392	9.80
instance_n=20_247	11	0.0000909	***174***	0.000107	17.71
instance_n=20_98	13	0.0000769	***29***	0.0000892	15.99
instance_n=20_469	14	0.0000714	***151***	0.0000919	28.71
instance_n=20_396	13	0.0000769	***234***	0.0000997	29.65

As can be seen from the Table II, the optimization effect of RL is still very obvious. However, the number of workstations calculated by the optimal result of RL will be much larger than expected. Although this situation is acceptable, this paper also calculates when the number of workstations is small.

TABLE III. RL AND MINIMIZE S.N

Name	RL and Minimize S.N				
	C	*S.N*	*Sar*	*Gap with Sar1(%)*	*Gap with Sar2(%)*
instance_n=50_1	766	9	0.000145	16	11.54
instance_n=50_7	485	13	0.000159	11.19	0
instance_n=50_51	906	12	0.0000920	10.4	0
instance_n=50_326	592	54	0.0000313	3.30	-10.81
instance_n=50_198	811	37	0.0000333	-6.72	-15.05
instance_n=20_247	1021	10	0.0000979	7.70	-8.50
instance_n=20_98	832	14	0.0000859	11.70	-3.68
instance_n=20_469	891	13	0.0000863	20.87	-6.09
instance_n=20_396	1520	7	0.0000940	22.24	-5.70

When the number of workstations becomes smaller, the optimization efficiency of RL is still higher in most cases, but there is a big gap compared with the results of RL without constraints. In addition, it is temporarily impossible to distinguish the real gap with Sar1, because Morrison does not give the real cycle time. From the results, we can at least confirm that RL is feasible in the balance problem of reconfigurable flexible assembly line.

However, there is still some room for optimization in the RL exploration process. The following figure records the change of average workstation assembly rate during training in one case.

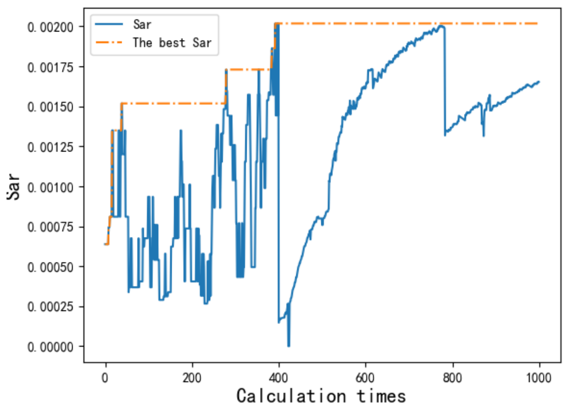

Fig. 2. The first result of training.

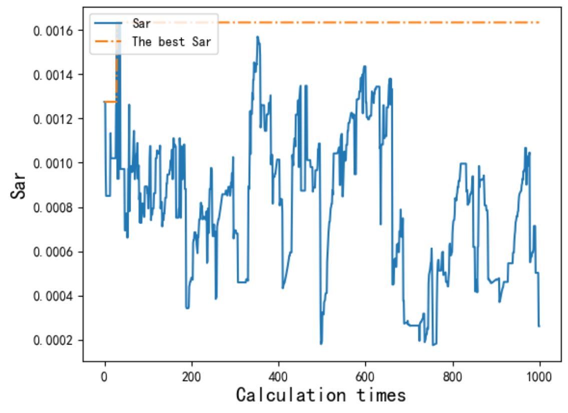

Fig. 3. The second result of training

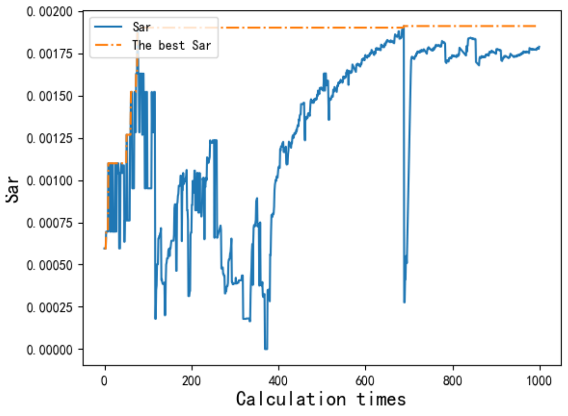

Fig. 4. The third result of training.

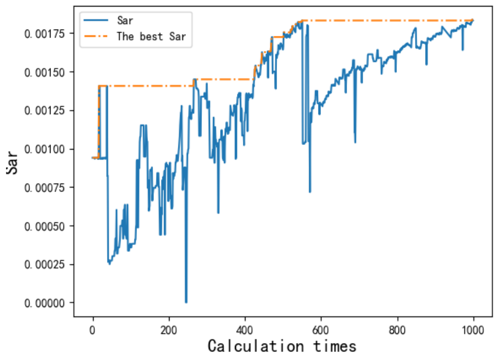

Fig. 5. The fourth result of training

Figures 2, 3, 4 and 5 show the changes of the average assembly workstation rate and the best assembly workstation rate during RL calculation. During the experiment, these four situations are the most common. In other words, when reinforcement learning is applied to assembly line balancing, the time to calculate the optimal result is uncertain, and the convergence of the calculation result is also uncertain. But at the same time, it can also be seen that the exploratory nature of reinforcement learning method is very bold and dare to let agents make special behavior, which may also be the advantage of reinforcement learning compared with other heuristic algorithms.

In a word, reinforcement learning can effectively solve the balance problem of reconfigurable flexible assembly line. In addition, this method has great potential, which is worthy of further exploration by scholars.

5. Conclusions

The demand of manufacturing industry for reconfigurable flexible assembly line is increasing, so the research on the balance of reconfigurable flexible assembly line is of practical significance. Focusing on the overall assembly efficiency, this paper focuses on the number of assembly workstations and cycle time required for each assembly line, with the goal of maximizing the average assembly workstation rate. Reinforcement learning is applied to the optimization of the mathematical model based on this problem. After the application test of several cases, it is proved that the application of reinforcement learning in assembly line balance problem is feasible, and it is also clear that reinforcement learning has in-depth mining value in this field.

6. Acknowledgement

This work was supported by Stabilization Key Project of Shenzhen Municipal Government (WDZC20200821140447001).

7. References

[1] DUAN Y, SU P, ZHENG C. A two-stage method for solving the stochastic two-sided mixed-model assembly line balancing problem[J]. Industrial Engineering Journal, 2016, 19(2): 134.

[2] Rashid M F F, Hutabarat W, Tiwari A. A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches[J]. The International Journal of Advanced Manufacturing Technology, 2012, 59(1): 335-349.

[3] Bakar N A, Ramli M F, Sin T C, et al. A review on robotic assembly line balancing and metaheuristic in manufacturing industry[C]//AIP Conference Proceedings. AIP Publishing LLC, 2019, 2138(1): 040005.

[4] Bautista J, Pereira J. A dynamic programming based heuristic for the assembly line balancing problem[J]. European Journal of Operational Research, 2009, 194(3): 787-794.

[5] Thangavelu S R, Shetty C M. Assembly line balancing by zero-one integer programming[J]. AIIE Transactions, 1971, 3(1): 61-68.

[6] Ogan D, Azizoglu M. A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements[J]. Journal of Manufacturing Systems, 2015, 36: 46-54.

[7] Lin Y C, Liang T H, Chen W S, et al. Differences between juvenile-onset ankylosing spondylitis and adult-onset ankylosing spondylitis[J]. Journal of the Chinese Medical Association, 2009, 72(11): 573-580.

[8] Dong J, Zhang L, Xiao T. A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints[J]. Journal of Intelligent Manufacturing, 2018, 29(4): 737-751.

[9] Li F, Du Y. From AlphaGo to power system AI: What engineers can learn from solving the most complex board game[J]. IEEE Power and Energy Magazine, 2018, 16(2): 76-84.

[10] Akanksha E, Sharma N, Gulati K. Review on Reinforcement Learning, Research Evolution and Scope of Application[C]//2021 5th International Conference on Computing Methodologies and Communication (ICCMC). IEEE, 2021: 1416-1423.

[11] Zhao X, Xia L, Tang J, et al. " Deep reinforcement learning for search, recommendation, and online advertising: a survey" by Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin with Martin Vesely as coordinator[J]. ACM SIGWEB Newsletter, 2019 (Spring): 1-15.

[12] Shi F, Wang S, Zeng L. The balancing problem for a reconfigurable flexible assemble line[C]//2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE). IEEE, 2021: 392-397.

[13] Xiao L, Jing X, Zeng L, et al. A flexible control system for reconfigurable assembly lines[C]//Journal of Physics: Conference Series. IOP Publishing, 2021, 2029(1): 012001.

[14] Otto A, Otto C, Scholl A. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing[J]. European Journal of Operational Research, 2013, 228(1): 33-45.

[15] Morrison D R, Sewell E C, Jacobson S H. An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset[J]. European Journal of Operational Research, 2014, 236(2): 403-409.