engineering proceedings

MDPI

Proceeding Paper

FlexSim-Simulated PCB Assembly Line Optimization Using Deep Q-Network †

Jinhao Du, Jabir Mumtaz *, Wenxi Zhao and Jian Huang

School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China; jinhaoduwenda@163.com (J.D.); wenxizhao2000@163.com (W.Z.); hpersona@163.com (J.H.)

* Correspondence: jabirmumtaz@live.com

† Presented at the 4th International Conference on Advances in Mechanical Engineering (ICAME-24), Islamabad, Pakistan, 8 August 2024.

Abstract: The balance scheduling of Printed Circuit Board (PCB) assembly lines plays a crucial role in enhancing production efficiency. Traditional scheduling methods rely on fixed heuristic rules, which lack flexibility and adaptability to changing production demands. To address this issue, this paper proposes a PCB assembly line scheduling method based on Deep Q-Network (DQN). The PCB assembly line model is constructed using the FlexSim simulation tool, and the optimal scheduling strategy is learned through the DQN algorithm. Comparative analysis is conducted against traditional heuristic rules. Experimental results indicate that the DQN-based scheduling method achieves substantial improvements in balance and production efficiency. For instance 1, the DQN approach achieved a total completion time (S) of 2.521×10^5, compared to the best heuristic rule result of 2.541×10^5. Similarly, for instance 2 and instance 3, the DQN method achieved total completion times of 2.549×10^5 and 2.522×10^5, respectively, outperforming all heuristic rules evaluated. This study provides a novel approach and method for intelligent scheduling of PCB assembly lines.

Keywords: printed circuit board; FlexSim simulation; PCB assembly line; deep reinforcement learning algorithm

Citation: Du, J.; Mumtaz, J.; Zhao, W.; Huang, J. FlexSim-Simulated PCB Assembly Line Optimization Using Deep Q-Network. *Eng. Proc.* **2024**, *75*, 34. https://doi.org/10.3390/engproc2024075034

Academic Editors: Muhammad Mahabat Khan, Muhammad Irfan, Mohammad Javed Hyder and Manzar Masud

Published: 9 October 2024

1. Introduction

Optimizing the scheduling of Printed Circuit Board (PCB) assembly lines is crucial in the electronics production sector, as it directly impacts production efficiency. Traditional scheduling methods mainly rely on fixed heuristic rules or swarm intelligence methods, which lack the flexibility and adaptability to respond to changing production demands. In theoretical research, current studies on PCB assembly scheduling primarily focus on its bottleneck process, Surface Mount Technology (SMT). Early research mainly addressed individual problems such as the Component Assignment Problem (CAP) and the Component Placement Sequencing Problem (CPSP). However, recent studies have concentrated on the integrated optimization of CAP and CPSP. Classic studies on CAP include Ho et al.'s work, which explored the CAP problem aiming to minimize total distance and introduced a hybrid genetic algorithm [1]. Zhu et al. proposed an improved leapfrog algorithm to solve the CPSP problem and used three-factor variance analysis to set the parameters of the improved leapfrog algorithm [2]. Additionally, Guo et al. sought to reduce PCB assembly cycle time and proposed an enhanced genetic algorithm, analogous to the multi-depot vehicle routing problem, to address both the CAP and CPSP problems concurrently [3]. Gao et al. proposed a layered multi-objective heuristic algorithm to optimize PCB assembly [4]. Heuristic rule methods, such as the Earliest Completion Time (ECT) rule and the Nearest Neighbor (NNH) rule, are simple, easy to implement, and computationally fast, making them suitable for small-scale problems [5]. However, these rules are typically designed

based on experience and intuition, lacking global optimization capability and struggling to handle complex and dynamic production environments. Metaheuristic methods, such as the Ant Colony Algorithm, Particle Swarm Optimization, and Spider Monkey Optimization, can discover optimal solutions by mimicking collective behaviors found in nature, making them effective for solving complex optimization problems. Nonetheless, these methods often require substantial computational resources and time, and the settings of algorithm parameters significantly affect the results, making it challenging to ensure solution stability and consistency.

Therefore, exploring more intelligent and adaptive scheduling strategies has become a research hotspot. Recently, Deep Reinforcement Learning (DRL) has made notable strides in addressing complex issues. Jianxiong Zhang et al. proposed an adaptive multi-task multi-objective scheduling model and AMDQN to optimize manufacturing time and cost, which experiments have shown to be effective [6]. L. Wan et al. presented a deep reinforcement learning approach integrated with a heterogeneous graph neural network (MHGNN) to efficiently address the flexible job shop scheduling problem (FJSP) using dual policy networks and a soft double-actor critic algorithm [7]. Wu X.Q. et al. utilized the PPO algorithm to tackle the job shop scheduling problem. Although progress has been made in various areas, further research is required to optimize scheduling for PCB assembly lines [8].

Therefore, this paper proposes a static scheduling method for PCB assembly lines based on Deep Q-Network (DQN). Using the FlexSim simulation tool, a PCB assembly line model is constructed, and the optimal scheduling strategy is learned through the DQN algorithm [9]. The main contributions are summarized in the following four points:

1. A visual environment for PCB assembly lines was constructed using the FlexSim simulation tool, which also provides an interactive learning environment for reinforcement learning.
2. A new DQN-based scheduling method for PCB assembly lines is proposed, significantly improving the balance and production efficiency of assembly lines.
3. A detailed modeling and DQN algorithm design scheme for the scheduling problem is provided, offering a reference for subsequent research.
4. The performance of the DQN method is compared with traditional heuristic rules through experiments, with results showing that the DQN method outperforms traditional methods in multiple instances.

This paper is organized as follows: Section 2 provides a detailed overview of the problem formulation and assumptions; Section 3 details the design and implementation of the DQN algorithm, covering state feature design, action space, and reward mechanism; Section 4 evaluates the algorithm's effectiveness through numerical experiments; and Section 5 summarizes the research findings and suggests directions for future work. This study aims to present a novel methodology and approach for intelligent PCB assembly line scheduling, addressing the needs of modern manufacturing for efficiency, flexibility, and intelligent production.

2. Problem Formulation

This paper addresses two sub-problems in balancing scheduling within the Surface Mount Technology (SMT) process of PCB assembly lines [5]. The SMT process is the bottleneck in PCB assembly production, where PCB components are picked and placed on multiple Surface Mount Machines (SMM). The Component Assignment Problem (CAP) involves distributing the workload by efficiently assigning components, while the Component Placement Sequence Problem (CPSP) focuses on determining the optimal sequence for placing components on SMMs to reduce the total completion time C_{max}. The processing time on an SMM consists of two parts: the moving time of the SMM head and the time for picking and placing components.

The scheduling problem can be described as follows: there are n_p identical PCB orders, each requiring m SMMs to place n_t types of n_c components. The task is to assign and

sequence components on machines to minimize total completion time C_{max}. The PCB scheduling problem in this paper assumes the following: (1) All SMMs can handle any type of component, although different machines have varied efficiencies and speeds. (2) An SMM can only process one component at a time. (3) It is assumed that the supply of all components is stable, meaning there are no quality issues or shortages of components during the production process. (4) It is assumed that there are no equipment failures during the production process. Figure 1 shows the problem description of PCB assembly line scheduling.

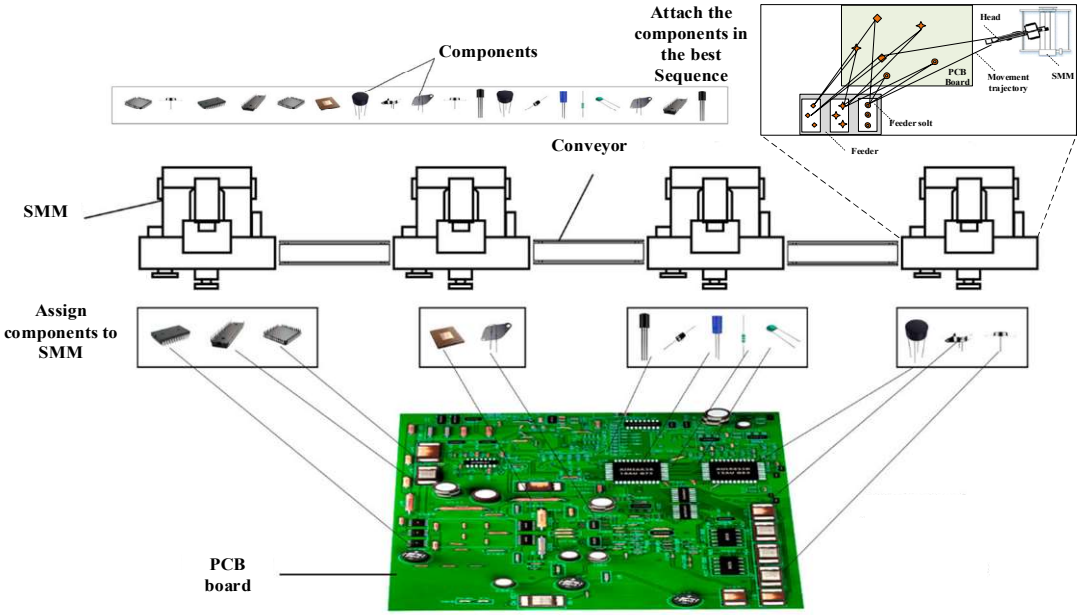

Figure 1. PCB assembly line scheduling problem description diagram.

3. Design of Deep Q-Network Algorithm Based on FlexSim

3.1. Algorithmic Framework

This study introduces an approach to address the scheduling problem in PCB assembly lines, where the balance scheduling problem (CAP & CPSP) is treated as a sequential decision problem triggered by machine requests. A scheduling environment was built using FlexSim simulation software (Figure 2). First, the deep neural network, including the main and target networks, is initialized. An experience replay buffer is set up to store data from the agent's interactions within the FlexSim environment.

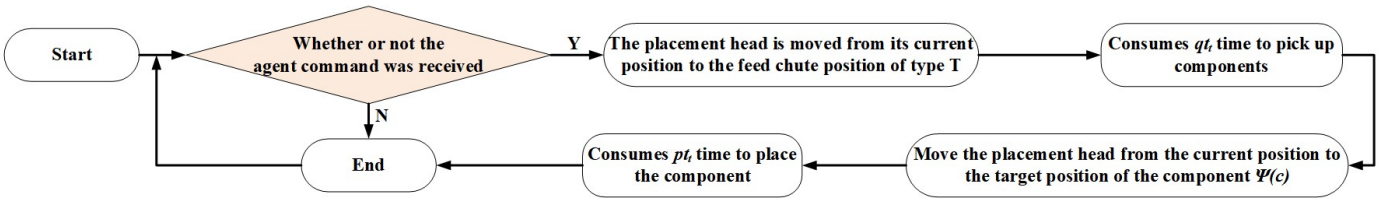

Figure 2. The operational logic of the SMM equipment.

The agent interacts with the environment by executing actions and observing results, storing experiences in the replay buffer. An ϵ-greedy policy selects actions, and valuable samples from the replay buffer train the neural network. The main network's parameters are periodically copied to the target network. The agent continues interacting with the environment, and the DQN algorithm optimizes the neural network parameters until a predetermined number of iterations is reached or a stopping condition is met. Through

this iterative process, the DQN algorithm enables the agent to learn the optimal strategy to maximize cumulative rewards. The overall framework is shown in Figure 3.

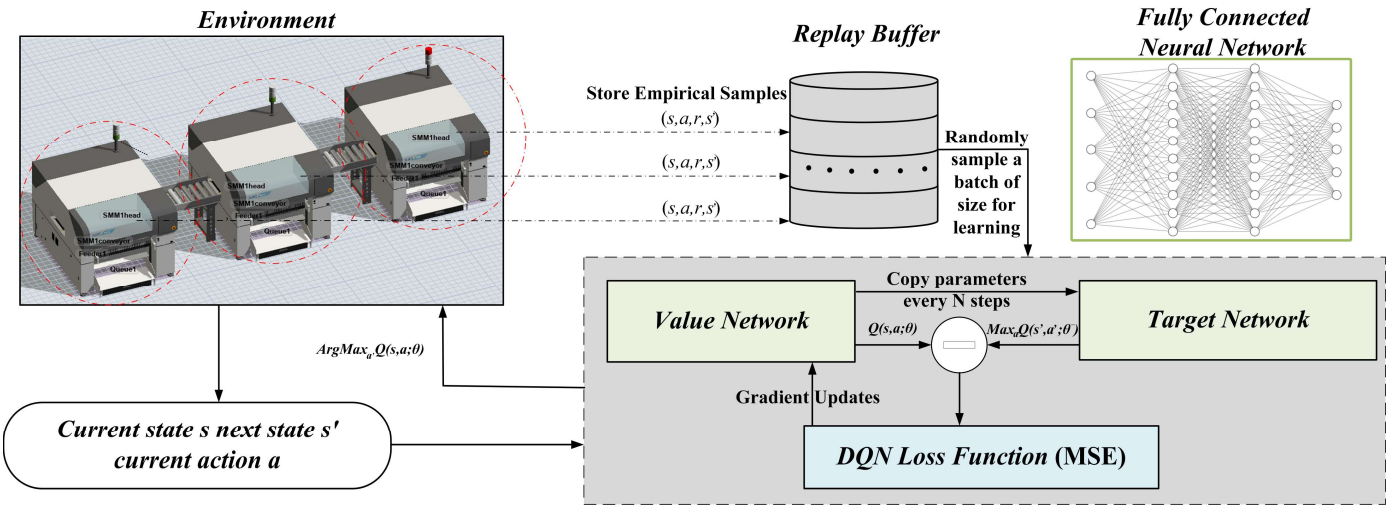

Figure 3. DQN algorithm framework diagram.

3.2. State Features

In this section, five state features are introduced to characterize the scheduling environment: the position where the placement head is about to arrive L_h, the horizontal locations of various feeders L_f, the movement speed of each placement head V_m, the position of components on the PCB L_c, and the completion rate of component allocation C_c^t.

3.3. Actions

This section presents five dispatching rules designed to implement actions in RL. Each rule is represented by an output node in the DQN [10]. The specifics of these five actions are as follows:

(1) Earliest Completion Time (ECT) Rule: Calculate the completion time of unassigned components allocated to idle machines and select the component with the earliest completion time [5].
(2) Nearest Neighbor Heuristic (NNH) Rule: Select the component closest to the current placement head position for placement [4].
(3) Nearest Lateral Distance (NLD) Rule: Select the component with the shortest lateral distance from the current placement head position for placement.
(4) Leftmost Unallocated Placement (LUP) Rule: Select the unallocated component with the smallest horizontal coordinate for placement.
(5) Latest Completion Time on Non-Current Machines (LCT-NCM) Rule: Select the component with the largest difference between the longest and second longest completion times on other machines for processing.

3.4. Rewards

The reward design primarily focuses on balancing the workload across devices and optimizing the production takt time. The specific process involves calculating the difference between the completion time of the current device and the maximum completion time among all devices at two consecutive decision points, thereby maximizing the expected cumulative reward. The calculation formula is shown in Equation (1), where $SDP_i^p(t)$ represents the estimated completion time of agent/device i, and $Max(SDP_{i\in M}^p(\mathrm{t}-1))$ represents the maximum estimated completion time among all devices at time $t-1$. The purpose of this reward design is to encourage the system to balance the workload across all devices and minimize the time differences between devices during production. When a

device's completion time is close to the maximum completion time, it indicates that the device has a heavier load and may become a bottleneck. By calculating the difference between the current device's completion time and the maximum completion time, we can measure the device's relative progress compared to the slowest device. The reward is negative because a larger negative difference indicates that the current device is lagging behind, necessitating adjustments. This negative reward motivates the optimization algorithm to adjust the scheduling strategy, reducing time differences between devices and thus optimizing production takt time and balancing device workloads.

$$R_t(S_t, a) = Max\left(Max(SDP^p_{i\in M}(\mathrm{t}-1)\right) - SDP^p_i(t) \quad (1)$$

4. Numerical Example and Analysis

4.1. Parameter Settings

To assess the efficiency and quality of the DQN algorithm, simulation experiments were conducted. The experimental parameters included the number of machines n_i, placement head movement speed v_i, component placement time pt_t, time taken to pick up components qt_t, number of PCB orders n_p, number of components n_c, number of component types n_t, and PCB size PCB_{size}, which indicates the dimensions of the bare PCB board in millimeters. These parameters were derived from production data provided by a specific company and corroborated with relevant literature to ensure the experiments' authenticity and reliability. The parameter ranges for different problem instances are presented in Table 1.

Table 1. Scale of experimental problems and parameter.

Parameter	Value		
	Small	**Medium**	**Large**
n_p	1000	2000	3000
n_c	50	100	150
n_i	2	3	4
n_t	10	20	30
PCB_{size}	100 × 100	150 × 150	200 × 200
v_i		[100, 150]	
qt_t		U [0, 0.1]	
pt_t		U [0, 0.1]	

To further optimize the DQN algorithm's performance, we also set several algorithm parameters, which are crucial for the training and effectiveness of the DQN model. Replay buffer size (N) is used to store experience tuples from the agent's interactions with the environment; larger buffer sizes allow for more extensive learning but require more memory. Episode number (L) represents the total number of training episodes; more episodes typically lead to better training but also require more computational time. The learning rate is used to update the network weights, set at 0.0001 to ensure stable and gradual learning. Exploration rate (ϵ) indicates the probability of choosing a random action versus the action recommended by the policy, starting at 1 and decaying to 0.1, allowing for exploration during initial training and exploitation of learned policies later. The discount factor (λ) is used to balance immediate and future rewards, set at 0.98. Batch size represents the number of experience tuples sampled from the replay buffer to update the network at each step, set at 32 to ensure efficient learning without overloading memory. These algorithm parameters are summarized in Table 2 below and were determined through preliminary experiments.

Table 2. Parameter values for algorithm.

Parameter	Value		
	Small	**Medium**	**Large**
Replay buffer size N	1000	2000	3000
Episode number L	200	400	500
Learning rate of training		0.0001	
ϵ in the action implementation		1–0.1	
Discount factor λ		0.98	
batch size		32	

4.2. Computational Experiments and Discussion

The computational results for various algorithms are shown in Table 3. Each problem instance was run independently 10 times. The symbols are explained as follows: ECT, NNH, NLD, LUP, and LCT-NCM represent different heuristic rules, while DQN represents the Deep Q-Network algorithm proposed in this paper. The performance of the DQN algorithm is measured by the total completion time C_{max}, which is compared across different problem instances (small, medium, large) to evaluate the effectiveness of the DQN algorithm relative to traditional heuristic methods.

Table 3. Performance results of different algorithms.

Instance Type	Algorithm	Instance 1 (s)	Instance 2 (s)	Instance 3 (s)
Small	ECT	2.564	2.575	2.547
	NNH	2.569	2.563	2.541
	NLD	2.628	2.554	2.531
	LUP	2.541	2.557	2.541
	LCT-NCM	2.562	2.607	2.558
	DQN	2.521	2.549	2.522
Medium	ECT	7.196	7.113	7.231
	NNH	7.117	7.113	7.133
	NLD	7.120	7.111	7.178
	LUP	7.188	7.131	7.087
	LCT-NCM	7.050	7.097	7.104
	DQN	7.016	7.020	7.051
Large	ECT	1.317	1.323	1.332
	NNH	1.307	1.316	1.325
	NLD	1.324	1.321	1.340
	LUP	1.310	1.332	1.338
	LCT-NCM	1.316	1.325	1.335
	DQN	1.302	1.308	1.323

To illustrate the convergence behavior of the DQN algorithm, Figure 4 shows the average episode reward over the training episodes. The plot indicates how the reward evolves as the training progresses, demonstrating the learning capability and stability of the algorithm. Specifically, the convergence curve is based on a medium-scale example, iterating over a total of 400 episodes, which effectively showcases the convergence trend of the algorithm.

According to the experimental results, the DQN algorithm significantly optimized the scheduling efficiency of PCB assembly lines, demonstrating superior performance compared to traditional heuristic rules across all instances. The DQN approach achieved notable improvements. In small instance 1, the total completion time for the DQN method was 2.521×10^5, compared to the best heuristic rule (ECT, 2.541×10^5), a reduction of approximately 0.79%. In small instances 2 and 3, the DQN method achieved total completion times of 2.549×10^5 and 2.522×10^5, respectively, outperforming all heuristic rules (NNH, NLD, LUP, LCT-NCM). The superiority of the DQN algorithm lies in its ability

to learn and adapt dynamically to changing conditions. Unlike fixed heuristic rules, the DQN algorithm can explore different scheduling strategies and adjust its approach based on the feedback received during training. This adaptive nature enables the DQN algorithm to find optimal solutions that are tailored to the specific characteristics and complexities of each problem instance. The results for medium and large instances also show that the DQN algorithm continues to excel in these more complex scenarios. For example, in medium instance 1, the total completion time for DQN was 7.016×10^5, compared to the best heuristic rule (LCT-NCM, 7.050×10^5). Similarly, in large instance 1, the total completion time for DQN was 1.302×10^6, while the best heuristic rule (NNH) was 1.307×10^6.

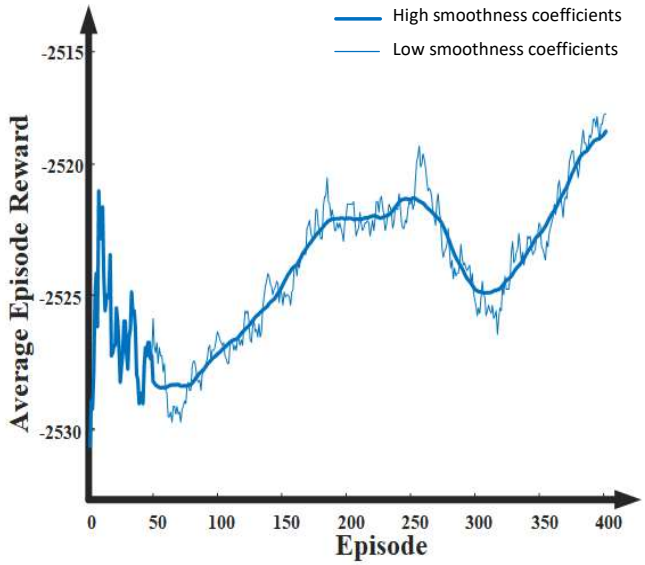

Figure 4. Convergence curve of the DQN algorithm showing the average episode reward over training episodes.

These results clearly demonstrate that the DQN-based scheduling method achieves significant improvements in balance and production efficiency compared to traditional heuristic rules. By leveraging the DQN algorithm, the scheduling process becomes more adaptive and efficient, capable of handling the complexities and dynamic nature of PCB assembly line scheduling.

5. Conclusions

This study focuses on the CAP and CPSP issues in PCB assembly line scheduling. The experimental results show that the proposed DQN algorithm surpasses traditional algorithms, delivering effective solutions. The DQN algorithm significantly improves total completion time, demonstrating its robustness and effectiveness in managing the complexities and dynamic nature of PCB assembly line scheduling. By leveraging the DQN algorithm, the scheduling process becomes more adaptive and efficient, capable of handling the intricacies of modern manufacturing. Future research should further explore and consider applying the proposed DQN algorithm to solve dynamic scheduling problems. Additionally, exploring the integration of other advanced reinforcement learning techniques could provide further improvements and insights into optimizing PCB assembly line scheduling, ensuring that the methods remain effective in increasingly complex and variable production environments.

Author Contributions: Conceptualization, J.M.; methodology, J.M., J.H., W.Z and J.D.; software, J.M.; validation, J.M.; formal analysis, J.D.; investigation, J.D.; writing—original draft preparation, J.D.; writing—review and editing, J.M.; funding acquisition, J.M.; visualization, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the Basic Scientific Research Project of Wenzhou City (G20210024), (G2023036) and (G20240020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: All individuals included in this section have consented to the acknowledgement.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mumtaz, J.; Guan, Z.; Yue, L.; Zhang, L.; He, C. Hybrid spider monkey optimisation algorithm for multi-level planning and scheduling problems of assembly lines. *Int. J. Prod. Res.* **2020**, *58*, 6252–6267. [CrossRef]
2. Zhu, G.-Y.; Zhang, W.-B. An improved Shuffled Frog-leaping Algorithm to optimize component pick-and-place sequencing optimization problem. *Expert Syst. Appl.* **2014**, *41*, 6818–6829. [CrossRef]
3. Guo, S.; Geng, F.; Takahashi, K.; Wang, X.; Jin, Z. A MCVRP-based model for PCB assembly optimisation on the beam-type placement machine. *Int. J. Prod. Res.* **2018**, *57*, 5874–5891. [CrossRef]
4. Gao, H.; Li, Z.; Yu, X.; Qiu, J. Hierarchical Multiobjective Heuristic for PCB Assembly Optimization in a Beam-Head Surface Mounter. *IEEE Trans. Cybern.* **2022**, *52*, 6911–6924. [CrossRef] [PubMed]
5. Chen, Y.; Zhong, J.; Mumtaz, J.; Zhou, S.; Zhu, L. An improved spider monkey optimization algorithm for multi-objective planning and scheduling problems of PCB assembly line. *Expert Syst. Appl.* **2023**, *229*, 120600. [CrossRef]
6. Zhang, J.; Guo, B.; Ding, X.; Hu, D.; Tang, J.; Du, K.; Tang, C.; Jiang, Y. An adaptive multi-objective multi-task scheduling method by hierarchical deep reinforcement learning. *Appl. Soft Comput.* **2024**, *154*, 111342. [CrossRef]
7. Wan, L.; Fu, L.; Li, C.; Li, K. Flexible job shop scheduling via deep reinforcement learning with meta-path-based heterogeneous graph neural network. *Knowl. -Based Syst.* **2024**, *296*, 111940. [CrossRef]
8. Wu, X.; Yan, X.; Guan, D.; Wei, M. A deep reinforcement learning model for dynamic job-shop scheduling problem with uncertain processing time. *Eng. Appl. Artif. Intell.* **2024**, *131*, 107790. [CrossRef]
9. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
10. Velyka, O.T.; Martyn, E.V.; Liaskovska, S.E. Simulation of the Production and Transport Problem in the FlexSim Environment. In *IOP Conference Series: Materials Science and Engineering*; IOP Publishing: Bristol, UK, 2023; p. 1277.