engineering proceedings

MDPI

Proceeding Paper

Actor–Critic Algorithm for the Dynamic Scheduling Problem of Unrelated Parallel Batch Machines †

Xue Zhao *, Yarong Chen and Mudassar Rauf *

School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China; yarongchen@126.com

* Correspondence: zx2710750937@163.com (X.Z.); mudassarrauf16@gmail.com (M.R.)

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

Abstract: With the continuous development of the information industry, semiconductor manufacturing has become a key basic industry in the information age. Due to the demands of the process, there are more batch processes in the semiconductor manufacturing process, such as the aging test session of chips. In this paper, in the context of semiconductor manufacturing, we consider the unrelated parallel batch processing machine (UPBPM) scheduling problem in which jobs have different processing times, arrival times, sizes, and processing eligibility constraints, where the machines have different capacity constraints and the objective of minimizing the makespan. We propose the actor–critic algorithm, incorporating the Rolling Time Window (R-AC algorithm) to solve the UPBPM scheduling problem. Through simulation experiments, the R-AC algorithm outperforms the separate heuristic scheduling rules.

Keywords: unrelated parallel batch machines; rolling time window; actor–critic algorithm

check for updates

Citation: Zhao, X.; Chen, Y.; Rauf, M. Actor–Critic Algorithm for the Dynamic Scheduling Problem of Unrelated Parallel Batch Machines. *Eng. Proc.* **2024**, *75*, 12. https://doi.org/10.3390/engproc2024075012

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

Published: 23 September 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the continuous development of the information industry, such as the Internet, terminal equipment, and other areas of increasing demand for chips, the semiconductor manufacturing industry has become a key basic industry in the information age. In this context, the optimization problem of semiconductor production scheduling has received more and more attention from both academia and industry. Due to the demand of the process, there are more batch processes in the semiconductor manufacturing process; for example, in the aging test link of the chip, the process belongs to the final test stage of the production, which means that the jobs enter into the high-temperature furnace in batches to eliminate the unqualified products. This is time-consuming, often several times more than that of other processes; therefore, the scheduling result of the stage has a significant impact on the productivity of the whole production line.

Batch scheduling problems are typically non-deterministic polynomial (NP)-hard, and many scholars and researchers have proposed numerous methods to solve various types of batch scheduling problems. The most commonly used methods are heuristic methods and the swarm intelligence algorithm.

When the scale of the batch processing problem is small, the system is not complex, and a relatively optimal feasible solution can be obtained in a short time using certain heuristic algorithms. For example, Li et al. [1] designed some heuristic algorithms to solve single-machine batch scheduling problems with incompatible job families. Zhou et al. [2] aimed to minimize job makespan on a single batch processing machine and developed a series of efficient constructive heuristic algorithms (FRS, MDS, and UD). Computational experiments demonstrated the superiority of the proposed heuristic algorithms in terms of solution quality, especially for small-scale job problems. Furthermore, the computational cost of the proposed heuristic algorithms is very low. Li et al. [1] studied the batch processor

scheduling problem with the objective of minimizing the maximum latency (L-max) and designed heuristics to solve the studied problem, improving these heuristics by optimizing the makespan or arrival date of critical batches.

However, with the rapid development of various manufacturing industries, batch scheduling problems have become increasingly complex and larger in scale. At this point, using simple heuristic rules to solve the problem yields poor feasible solutions; thus, swarm intelligence algorithms have gained widespread application due to their excellent optimization capabilities. For example, Wang et al. [3] devised a cooperative coevolution algorithm tailored to the parallel batch processing machine (PBPM) scheduling problem, which hinges on the search and coordination among three ant colonies. They incorporated an adaptive search strategy to preserve the diversity of solutions. In their study, Li et al. [4] aimed to minimize both the maximum delay and the total pollution emission cost for uniform parallel batch processing machines. They innovatively designed an angle-based genetic algorithm to ensure population diversity. Zhou et al. [5] tackled the issue of dynamically arriving jobs on PBPMs by proposing an efficient multi-objective differential evolution algorithm. Arroyo et al. [6] introduced an iterative greedy algorithm with the goal of minimizing total processing time to solve the scheduling problem for uncorrelated parallel batch processing machines (UPBPM) with varying part sizes and non-zero part preparation times, demonstrating the algorithm's superiority over existing meta-heuristic approaches. Schorn et al. [7] developed a genetic programming (GP) approach for PBPM scheduling in semiconductor wafer fabrication facilities, addressing a hybrid objective of total weighted tardiness and total energy consumption, with computational experiments validating the high-quality performance of their program. Jiang et al. [8] explored the UPBPM scheduling problem, utilizing an iterative greedy algorithm combined with a batch local search method to minimize total processing time, and their experimental results confirmed the excellent performance of the proposed algorithm.

In summary, our literature review indicates that current research on batch scheduling problems predominantly focuses on heuristic algorithms and swarm intelligence algorithms, often addressing single and equivalent parallel machine scenarios. This paper, however, leverages deep reinforcement learning (DRL) algorithms with autonomous learning capabilities to tackle the scheduling problem of uncorrelated parallel batch processing machines (UPBPMs) in the context of semiconductor manufacturing. This approach not only provides a novel application but also offers significant reference value for future studies on batch scheduling problems.

2. Problem and Methodology

2.1. Problem Description

This paper investigates the scheduling problem of minimizing the makespan C_{max} for n dynamically arriving jobs on unrelated parallel batch processing machines (UPBPMs) M_i $(i = 1, 2, \cdots, m)$. Each job J_j $(j = 1, 2, \cdots, n)$ arrives according to a Poisson distribution and has different processing times p_{ij}, arrival times r_j, and sizes s_j. The batch processing capacity Q_i varies across different machines. Some jobs have processing qualification constraints, meaning they can only be processed on certain batch-processing machines. This scheduling problem is denoted using the three-field notation as $R_m|Q_i, p_{ij}, s_j, r_j|C_{max}$. The main decisions in this problem include assigning n jobs to m machines, determining the batching method for the jobs assigned to each machine, and the batch processing order.

The basic assumptions satisfied by the problem are:

(1) Jobs can be combined into a batch as long as the total size of all jobs in the batch does not exceed the capacity limits of the assigned batch processor and the machining eligibility requirements are met.
(2) Once a batch starts processing in a batch processor, the process must continue uninterrupted until all jobs within the batch are completed.

(3) The arrival time of a batch is set by the latest arrival time among the jobs in the batch; likewise, the processing time of a batch is determined by the longest processing time among the jobs in the batch.

2.2. Actor–Critic Algorithm Based on Rolling Time Window

The UPBPM scheduling system studied in this paper has Markovianity due to the arrival process of jobs obeying Poisson distribution. Therefore, the deep reinforcement learning algorithm actor–critic (AC) algorithm can be applied to solve the UPBPM dynamic scheduling problem. In order to perform batch processing more rationally, a trade-off is made between reducing the total number of batches and avoiding excessive waiting time for jobs. In this section, the AC algorithm incorporating the rolling time window method (R-AC) is designed. The flowchart of the R-AC algorithm is shown in Figure 1

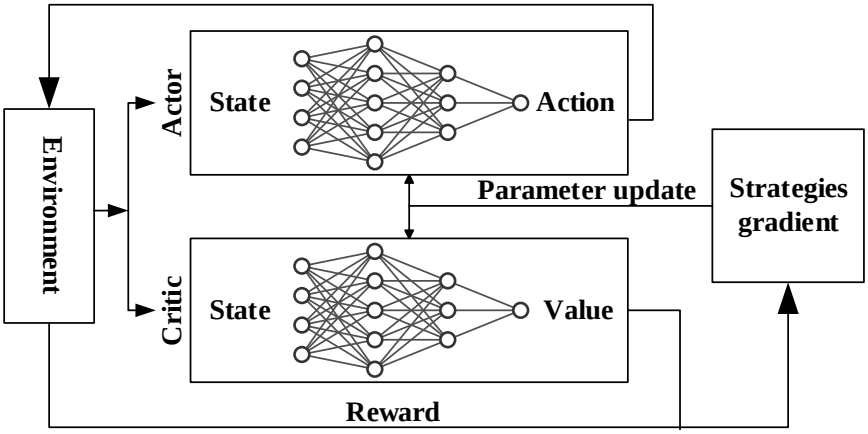


Figure 1. R-AC algorithm flow chart.

The time window interval T^w is defined as the time interval (T, T') between the last completion of processing by the machine and the completion of processing by the current machine. If the number of jobs L_i arriving within (T, T') does not satisfy the group batch requirement, the intelligent body does not execute the action to select jobs for processing and waits for the end of the next time window to execute the action for processing.

2.2.1. Reward Design

In this paper, we address the dynamic scheduling problem of uncorrelated parallel batch processors, aiming to minimize the makespan C_{max}. In the group batch process, it is necessary to achieve the smaller job processing time allocated on batch processors, the minimization of batch processor residual capacity, as well as the similarity of the total processing time on different batch processors. Therefore, the reward function designed in this paper is shown in Equation (1).

$$r = \frac{1}{\left[\sum_{x=1}^{y}(P_b - p_x)\right]\cdot\left[\sum_{i=1}^{m}\{\max(C_i) - C_i\}\right]\cdot Q_{ir} + 1}, i = 1, 2, \cdots, m \quad (1)$$

where P_b is the processing time of the batch that is being grouped into batches at the decision moment, p_x is the processing time of the jobs in the batch, and y is the number of jobs in the batch; C_i is the total processing time of the batch processor M_i at the decision moment; and Q_{ir} is the residual capacity of the upcoming allocated batch on the batch processor M_i at the decision moment.

2.2.2. Action Design

In the UPBPM scheduling system designed in this paper, the decision-making moment intelligences select machinable jobs to group batches according to heuristic actions, i.e., the

jobs that best meet the optimization objective of the heuristic actions are directly grouped in a batch for processing in a buffer under the premise of satisfying the qualification constraints of the jobs to be processed and the batch capacity constraints of the machine.

The group batch flowchart is shown in Figure 2.

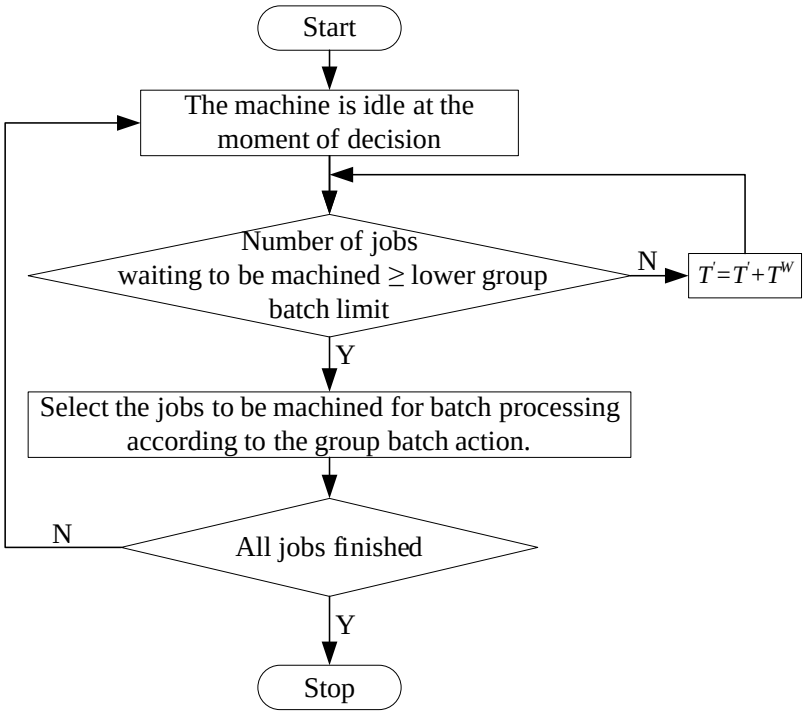


Figure 2. Group batch flow chart.

The heuristic actions are described as follows:

Action 1: First In First Out (FIFO)

Step 1: Choose the job that arrives earliest to add to the batch. If multiple jobs arrive at the same time, select one randomly.

Step 2: Determine the remaining capacity of the batch. Add the earliest arriving job whose size is less than the remaining capacity to the batch. Repeat this step until no more jobs can be added to the batch.

Action 2: Shortest Processing Time (SPT)

Step 1: Add as many jobs with the shortest processing time as possible to the batch. If there are several jobs with the same shortest processing time, pick one at random.

Step 2: Determine the remaining capacity of the batch. Add the job with the shortest processing time whose size is less than the remaining capacity to the batch. Repeat this step until no more jobs can be added to the batch.

Action 3: Minimum Size (MS)

Step 1: Add as many jobs with the smallest size as possible to the batch. If there are several jobs with the same smallest size, select one at random.

Step 2: Determine the remaining capacity of the batch. Add the job with the smallest size whose size is less than the remaining capacity to the batch. Repeat this step until no more jobs can be added to the batch.

2.2.3. State Design

The intelligent agent selects actions according to the current state of the system; therefore, the state characteristics need to be able to effectively reflect the current system state. The state characteristics of the system designed in this paper are shown in Table 1.

Table 1. State characteristic.

State	Meaning
$ft_1 = n_{BF}$	The quantity of jobs awaiting processing in the buffer.
$ft_2 = p_{ij},\ i = 1,2,\cdots,m; j \in BF$	The processing duration of the jobs in the buffer.
$ft_3 = r_j, j \in BF$	Arrival time of all jobs in the buffer.
$ft_4 = s_j,\ j \in BF$	The sizes of all jobs in the buffer.
$ft_5 = C_i, i = 1,2,\cdots,m$	Machining time of the batch processor at the decision moment.
$ft_6 = Q_{ir}, i = 1,2,\cdots,m$	The residual capacity of the machine after batch processing.

2.3. Data Generation

In this section, Pycharm software is used for programming to build a simulation environment for the dynamic scheduling problem of uncorrelated parallel batch processors.

In conjunction with the actual production conditions of the enterprise, combined with the data given in the references, the job arrival time r_j, the job processing time p_{ji} (where the job exists the processing eligibility constraints), and the job size s_j. Three kinds of scale test data are generated. The ranges and generation formulas for the batch processor capacity Q_i constraints and other parameters are shown in Table 2.

Table 2. The scale and parameter range of the experimental problem.

Instance	n	m	r_j	p_{ji}	s_j	Q_i
Small	10,20	2	Poisson's arrival $\lambda_i = 1$	U[1,100]	U[1,10]	U[1,10]
Medium	40,60	2				
Large	80,100	3				

For different combinations of jobs and machines, 10 sets of test data were generated for each group for experimentation, for a total of 60 sets of experimental data for all scales.

3. Results and Analysis

This section compares the proposed actor–critic algorithm with rolling time windows (R-ACs) with three heuristic rule methods, First In First Out (FIFO), Shortest Processing Time (SPT), and Minimum Size (MS), which are consistent with the action design. The aim is to validate the effectiveness of the R-AC algorithm. The results solved using each method are shown in Table 3.

Table 3. R-AC algorithm with FIFO rule, SPT rule, and MS rule method solution results.

Instance	n	m	R-AC	FIFO	SPT	MS
Small	10	2	110	140	111	140
	20	2	199	218	192	235
Medium	40	2	361	391	394	465
	60	2	480	565	565	622
Large	80	3	493	645	635	589
	100	3	573	761	706	762

From Table 3, it can be seen that the R-AC-based method shows a clear advantage over the FIFO, SPT, and MS rules in terms of the objective for overall problem-solving. Specifically, for small-scale instances with $n = 10$ and $n = 20$, the R-AC algorithm does not perform better than the SPT rule, and the results for $n = 20$ are even worse than those of the SPT rule. This is because the optimization direction of the SPT rule is closely aligned with the objective of minimizing the makespan C_{max} in small-scale problems. However, the overall results demonstrate the superiority of the R-AC algorithm.

4. Conclusions

In this paper, an actor–critic algorithm based on rolling time windows is proposed for the dynamic scheduling problem of UPBPM in the context of semiconductor manufacturing with the objective of minimizing the makespan. The rewards, actions, and states of the algorithm are established and validation experiments are designed. The experimental results show that the R-AC algorithm is able to obtain better objective values than simple heuristic rules. In future research, the experimental scale will be expanded, and other swarm intelligence algorithms (DE, ABC, etc.) and deep reinforcement learning algorithms (DQN, PPO, etc.) will also be introduced as comparison algorithms for the proposed R-AC algorithm, so as to further validate the effectiveness of the proposed method.

Author Contributions: Conceptualization, X.Z. and Y.C.; methodology, X.Z. and M.R.; software, X.Z. and Y.C.; validation, X.Z. and M.R.; formal analysis, X.Z. and Y.C.; writing—review and editing, Y.C. and M.R.; visualization, M.R.; supervision, X.Z.; funding acquisition, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: Basic scientific research project of Wenzhou City (G2023036 & G20240020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data are available from the corresponding author upon request.

Acknowledgments: I am very grateful to Y.C. and M.R. for their assistance in writing this article.

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

References

1. Li, X.L.; Li, Y.P.; Huang, Y.L. Heuristics and lower bound for minimizing maximum lateness on a batch processing machine with incompatible job families. *Comput. Oper. Res.* **2019**, *106*, 91–101. [CrossRef]
2. Zhou, S.; Chen, H.; Xu, R.; Li, X. Minimising Makespan on a single batch processing machine with dynamic job arrivals and non-identical job sizes. *Int. J. Prod. Res.* **2014**, *52*, 2258–2274. [CrossRef]
3. Wang, Y.; Jia, Z.; Li, K. A multi-objective co-evolutionary algorithm of scheduling on parallel non-identical batch machines. *Expert Syst. Appl.* **2021**, *167*, 114145. [CrossRef]
4. Li, K.; Zhang, H.; Chu, C.; Jia, Z.H.; Chen, J. A bi-objective evolutionary algorithm scheduled on uniform parallel batch processing machines. *Expert Syst. Appl.* **2022**, *204*, 117487. [CrossRef]
5. Zhou, S.; Li, X.; Du, N.; Pang, Y.; Chen, H. A multi-objective differential evolution algorithm for parallel batch processing machine scheduling considering electricity consumption cost. *Comput. Oper. Res.* **2018**, *96*, 55–68. [CrossRef]
6. Arroyo, J.E.C.; Leung, J.Y.T. An effective iterated greedy algorithm for scheduling unrelated parallel batch machines with non-identical capacities and unequal ready times. *Comput. Ind. Eng.* **2017**, *105*, 84–100. [CrossRef]
7. Schorn, D.S.; Mönch, L. Learning Priority Indices for Energy-Aware Scheduling of Jobs on Batch Processing Machines. *Trans. Semicond. Manuf.* **2024**, *37*, 3–15. [CrossRef]
8. Jiang, W.; Shen, Y.L.; Liu, L.X.; Zhao, X.; Shi, L. A new method for a class of parallel batch machine scheduling problem. *Flex. Serv. Manuf. J.* **2021**, *17*, 19–24. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.