engineering proceedings

MDPI

Proceeding Paper

Minimizing a Just-In-Time Objective on a Single-Batch-Processing Machine Using a Hybrid Differential Evolution Algorithm †

Chen Wang, Yarong Chen *, Fuh-Der Chou and Shenquan Huang

School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, China
* Correspondence: yarongchen@126.com
† Presented at the 2nd International Conference on Advances in Mechanical Engineering (ICAME-22), Islamabad, Pakistan, 25 August 2022.

Abstract: A hybrid differential evolution (HDE) algorithm that minimizes the total earliness and tardiness time (ET), a just-in-time objective, is studied for a single-batch-processing machine (SBPM) scheduling problem with different processing times, release times, sizes and delivery times. A hybrid differential evolution algorithm with an integrated Tabu search (TS) algorithm is proposed to improve the capacity of the neighborhood search of a differential evolution algorithm (DE). The experimental results show that the proposed HDE can obtain better an objective value than the basic DE can.

Keywords: differential evolution algorithm; Tabu search algorithm; single-batch-processing machine scheduling problem

check for updates

Citation: Wang, C.; Chen, Y.; Chou, F.-D.; Huang, S. Minimizing a Just-In-Time Objective on a Single-Batch-Processing Machine Using a Hybrid Differential Evolution Algorithm. *Eng. Proc.* **2022**, *23*, 4. https://doi.org/10.3390/engproc2022023004

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

Published: 20 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 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

The problem of batch processing machine (BPM) scheduling is a significant branch of the production scheduling problem. The scheduling problem of burn-in operations in semiconductor manufacturing is a BPM scheduling problem. Therefore, it is important to carry out research on BPM scheduling in the context of intelligent manufacturing.

Many research studies have been conducted in the literature. Donbson and Nambimadom studied the SBPM scheduling problem to optimize the mean weighted flow time with different job sizes [1]. Chou et al. designed an improved genetic algorithm to solve the SBPM scheduling problem with different sizes and delivery times of jobs [2]. Zhou improved the DE by selecting the mutation operator adaptively for the same problem that is mentioned in the literature to minimize the makespan [3]. In order to optimize the maximum lateness for the SBPM problem with non-identical job sizes and release dates, Zhou presented a modified particle swarm optimization (MPSO) algorithm [4]. Parsa et al. designed a dynamic programming algorithm to optimize the BPM scheduling for the JIT objective with a common due date, and different processing times and sizes of jobs [5]. Zhang et al. designed SLTS heuristic rules for the SBPM scheduling problem with different processing times and due dates, which can effectively minimize the ET [6]. Zhang et al. designed AABF and ITSLS rules to optimize the total weighted earliness and tardiness [7].

This article researches an SBPM problem to minimize the ET objective, and a hybrid differential evolution algorithm is designed to minimize the ET.

2. Problem Description

The problem can be described like this: n jobs will be processed on a SBPM. First, the jobs will be assigned to a batch and one job can only be assigned to a batch once. The total size of the jobs in a batch cannot be greater than the constraint of the machine S. Then, the processing sequence of the batches is determined to optimize the ET.

Some assumptions need to be considered: (1) p_j, r_j, s_j and d_j, respectively, represent the processing time, release time, size and delivery time of job j. (2) The process cannot be interrupted, even if there are still some jobs in the batch have not been finished. (3) The largest release time of the jobs in the batch determines the release time of a batch. (4) The longest processing time of the jobs in the batch determines the processing time of a batch. Using the three-field notation $\alpha|\beta|\gamma$ which was introduced for describing scheduling problems, we denote our problem by $1\left|p_j, r_j, d_j, s_j, S\right|ET_{min}$.

3. Description of DE and HDE

As a frequently used evolutionary algorithm, DE is widely used. DE includes population initialization, mutation, crossover, selection operations, and it can quickly explore solution space.

3.1. Initialization

The initial population of DE is a P × n set that is constructed by P individuals. P denotes the population size. Each individual has n elements, and n denotes the number of jobs that there are. Therefore, the initial population can be denoted by $x_i^t = (x_{i,1}^t, x_{i,2}^t, \cdots, x_{i,n}^t)$, which represents the ith individual in a generation t(i = 1, … , P). In this article, a job permutation list is obtained by a ranked-order-value (ROV) rule, which is dependent on the random key representation. As Table 1 shows, the random key of the jobs is {0.3, 0.6, 0.1, 0.8, 0.4, 0.9}, and we can get the permutation of job X = {3, 1, 5, 2, 4, 6} by using ROV. Then, a first-fit (FF) rule is employed to group the jobs that are sorted into batches. We assume that the capacity of the machine is $S = 10$, and the two batches.

Table 1. Job permutation obtained by ROV.

Jobs	**1**	**2**	**3**	**4**	**5**	**6**
Random key	0.3	0.6	0.1	0.8	0.4	0.9
Job permutation	3	1	5	2	4	6
Job size	1	8	5	1	3	1

B = $\{b_1, b_2\} = \{\{3, 1, 2\}, \{5, 4, 6\}\}$ will be obtained. The individual X_i^t, i = 1, … , P is randomly generated. Each random key value of the individual is generated by Formula (1)

$$x_{i,j}^0 = x_d + random \times (x_u - x_d), \quad j = 1, \cdots, n \tag{1}$$

$x_d = -1$, $x_u = 1$, *random* is a random value, *random* ∈ [0, 1].

3.2. Mutation

In this step, this article applies *DE/rand/2* to generate the mutant individuals v_i^t. The mutation operator can be described as Equation (2):

$$V_i^t = X_a^t + F \times \left(X_b^t - X_c^t\right) + F \times \left(X_d^t - X_e^t\right) \tag{2}$$

where a, b, c, d, e are not equal to each other and are respectively generated in the set {1, … , P}, and F is randomly chosen in the range [0, 1].

3.3. Crossover

A trial individual U_i^t will be produced in the crossover operator, j_{random} is a random natural number $j_{random} \in \{1, \ldots, n\}$. The CR determines whether an element of V_i^t can be a part of U_i^t, and the CR is randomly generated in the range [0, 1]. The crossover operator can be described as Equation (3):

$$u_{i,j}^t = \begin{cases} v_{i,j}^t & if\ random \le CR\ or\ j = j_{random} \\ x_{i,j}^t & otherwise \end{cases} \quad j = 1, \cdots, n \tag{3}$$

3.4. Selection

In this step, we demonstrate how to compare the fitness of U_i^t with V_i^t and select the better one which will be used as a new individual of the next population. f represents the fitness value.

$$x_i^{t+1} = \begin{cases} U_i^t & if\ f(U_i^t) \le f(X_i^t) \\ X_i^t & otherwise \end{cases} \quad (4)$$

Equation (4) describes how to select an individual to become a new part of the next population.

3.5. Combination of Two Algorithms

The DE has a weak local search capacity. Therefore, a local search strategy is embedded into the DE to improve the local search capacity of DE, as it is proposed. The TS algorithm is a common local search algorithm. The TS sets a Tabu list to forbid some repeated operations and uses aspiration criterion to assoil a good solution status. After the selection operator of DE, the next step is choosing the best individual of this population as a candidate solution to perform the Tabu operation. Introduced next is 2-opt which produces a new candidate solution. As shown in Figure 1, by exchanging two and four, we can obtain a new solution.

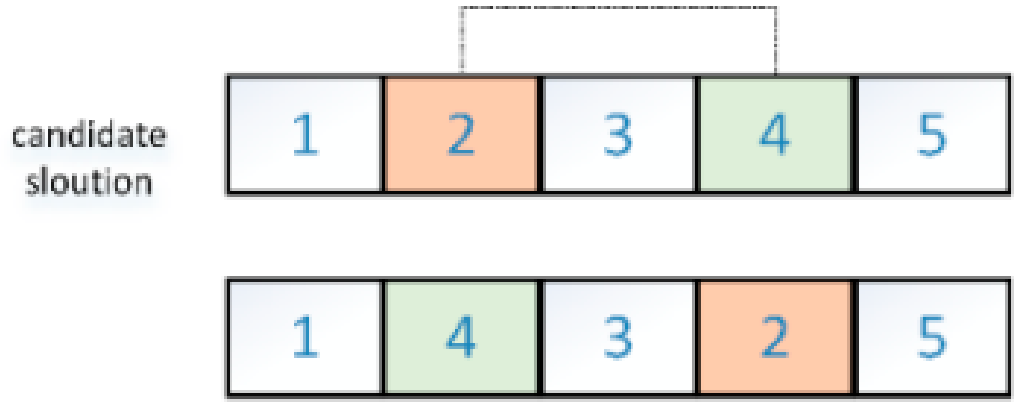


Figure 1. The process of 2-opt.

3.6. The Process to Sort the Batches

The batches are sorted according to the median of the job due date in the batch, and the steps are as follows:

1. They are given a job permutation, and the jobs are assigned to a batch by the FF rule, and b batches are obtained;
2. The median of the jobs due dates $d_{B_i}^{median}$, $(i = 1, \ldots, b)$ in the batch are found, and they are sorted in to an increasing order which decides the order of the batches that are being processed: $B_1, B_2, \ldots, B_b$;
3. The start time of the batch $S_{B_i} = max\{r_{B_i}, C_{B_{i-1}}\}$ is, when $i = 1$, $C_{B_0} = 0$. The completion time of the batch is checked, to see if $C_{B_i} < d_{B_i}^{median}$, B_i is shifted from the left to the right until $C_{B_i} = d_{B_i}^{median}$; if the situation is otherwise, then nothing needs to be changed. The operation is repeated until all the batches are scheduled.

4. Computational Experiments

4.1. Data Generation

Extensive computational experiments are done. Let n = 10, 12, 15, 20, 40, 60, 80, 100 for these. The processing time p_j, the release time r_j and the size s_j of job were, respectively, discrete distributions in the range of [1, 100], [0, 50] and, [1, 10]. The due date of the job d_j satisfies the following formulae $d_j = p_j + U[d_{min}, d_{min} + \beta \times P]/2$ and $d_{min} = \max(0, P \times (\alpha - \beta/2))$, $P = \sum_{j=1}^{n} p_j$, $\alpha = 1$, $\beta = 0.5$. The machine's capacity is $S = 30$.

4.2. Results Analysis

To compare the quality of the DE and HDE, each scale data were randomly produced in 10 instances. Table 2 shows the average values of the completion time (CT) and the ET that were obtained by testing every scale of the jobs.

Table 2. Results that were obtained by different algorithms.

n	DE		HDE		Gap (%)
	CT	ET	CT	ET	
10	2.24	224.3	9.04	224.3	0
12	2.72	309.9	20.96	306.2	−0.01
15	13.39	384.5	27.30	378.2	−0.02
20	52.81	617.8	106.42	585.4	−0.05
40	110.12	2035.5	207.26	1434.9	−0.30
60	168.83	5744.6	570.46	2807.1	−0.51
80	222.60	10966.4	803.57	4304.3	−0.61
100	274.67	22962.7	925.48	6649	−0.71

The sixth Column of Table 2 shows the gap between the DE and HDE, $Gap = (ET_{HDE} - ET_{DE})/ET_{DE} \times 100\%$. According to Table 2, the ET that was obtained by HDE is close to the DE when $n \leq 0$, because two kinds of algorithm can explore the solution space completely. With the increase of job scale, the ET that can be obtained by HDE is much better than the DE when $n \geq 40$, which indicates that the exploitation capacity of HDE is obviously better than DE.

5. Conclusions

In this paper, a SBPM scheduling problem with non-identical processing times, release times, due dates and arbitrary job sizes to minimize the ET was addressed. An HDE was proposed according to the characteristics of the problem. Compared to the basic DE, HDE can achieve a better solution, especially in situations where there are more than 60 jobs. Future research may consider the parallel-batch processing machine problem. At the same time, many objectives will be considered to be closer to the reality.

Author Contributions: Conceptualization, C.W. and Y.C.; methodology, C.W. and S.H.; software, C.W. and F.-D.C.; validation, C.W. and F.-D.C.; formal analysis, C.W. and Y.C.; investigation, F.-D.C. and S.H.; data curation, Y.C.; writing—original draft preparation, C.W.; writing—review and editing, Y.C. and S.H.; supervision, S.H.; project administration, C.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number [No. 51705370].

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 conflict of interest.

References

1. Dobson, G.; Nambimadom, R.S. The Batch Loading and Scheduling Problem. *Oper. Res.* **2001**, *49*, 52–65. [CrossRef]
2. Der Chou, F. A Joint GA+DP Approach for Single Burn-in Oven Scheduling Problems with Makespan Criterion. *Int. J. Adv. Manuf. Technol.* **2007**, *35*, 587–595. [CrossRef]

3. Zhou, S.; Xing, L.; Zheng, X.; Du, N.; Wang, L.; Zhang, Q. A Self-Adaptive Differential Evolution Algorithm for Scheduling a Single Batch-Processing Machine with Arbitrary Job Sizes and Release Times. *IEEE Trans. Cybern.* **2021**, *51*, 1430–1442. [CrossRef] [PubMed]
4. Zhou, H.; Pang, J.; Chen, P.K.; Chou, F. Der A Modified Particle Swarm Optimization Algorithm for a Batch-Processing Machine Scheduling Problem with Arbitrary Release Times and Non-Identical Job Sizes. *Comput. Ind. Eng.* **2018**, *123*, 67–81. [CrossRef]
5. Parsa, N.R.; Karimi, B.; Husseini, S.M.M. Exact and Heuristic Algorithms for the Just-in-Time Scheduling Problem in a Batch Processing System. *Comput. Oper. Res.* **2017**, *80*, 173–183. [CrossRef]
6. Zhang, H.; Wu, F.; Yang, Z. Hybrid Approach for a Single-Batch-Processing Machine Scheduling Problem with a Just-in-Time Objective and Consideration of Non-Identical Due Dates of Jobs. *Comput. Oper. Res.* **2021**, *128*, 105194. [CrossRef]
7. Zhang, H.; Yang, Y.; Wu, F. Just-in-Time Single-Batch-Processing Machine Scheduling. *Comput. Oper. Res.* **2022**, *140*, 105675. [CrossRef]