Parallel Machine Scheduling Problem with Machine Rental Cost and Shared Service Cost

Rongteng Zhi *, Yinfeng Xu, Feifeng Zheng, Fei Xu

Posted Date: 25 February 2025

doi: 10.20944/preprints202502.1852.v1

Keywords: shared manufacturing; scheduling optimisation; rental cost; shared service cost; heuristic algorithm

 doi:10.20944/preprints202502.1852.v1

Article

Parallel Machine Scheduling Problem with Machine Rental Cost and Shared Service Cost

Rongteng Zhi [1,*], Yinfeng Xu [2], Feifeng Zheng [3] and Fei Xu [4]

[1] School of Economics & Management, Qilu Normal University, Jinan 250200, China; 15949979656@163.com
[2] School of Management, Xi'an Jiaotong University, Xi'an 710049, China
[3] Glorious Sun School of Business & Management, Donghua University, Shanghai 200051, China
[4] College of Preschool Education, Qilu Normal University, Jinan 250200, China
* Correspondence: 15949979656@163.com

Abstract: An identical parallel machine offline scheduling problem with rental costs and shared service costs of shared machines is studied. In machine renting, manufacturers with a certain number of identical parallel machines will incur fixed rental costs, unit variable rental costs, and shared service costs when renting the shared machines. The objective is to minimize the sum of the makespan and total sharing costs. To address this problem, an integer linear programming model is established, and several properties of the optimal solution are provided. A heuristic algorithm based on the number of rented machines is designed. Finally, numerical simulation experiments are conducted to compare the proposed heuristic algorithm with a genetic algorithm and the longest processing time (LPT) rule. The results demonstrate the effectiveness of the proposed heuristic algorithm in terms of calculation accuracy and efficiency. Additionally, the experimental findings reveal that the renting and scheduling results of the machines are influenced by various factors, such as the manufacturer's production conditions, the characteristics of the jobs to be processed, production objectives, rental costs, and shared service costs.

Keywords: shared manufacturing; scheduling optimisation; rental cost; shared service cost; heuristic algorithm

1. Introduction

Shared manufacturing, relying on new infrastructures such as 5G, artificial intelligence, the industrial Internet, and the Internet of Things, has quietly emerged and shown a booming development [1,2]. Shared manufacturing is the reuse of idle production capacity, which can optimize resource allocation, break information asymmetry, reduce manufacturing costs, expand effective supply, and enhance manufacturing service capabilities and levels [3]. As a result, shared manufacturing has become one of the key themes driving the high-quality and sustainable development of the manufacturing industry, aligning with the trend towards intelligent, efficient, and service-oriented economic development.

Under the impetus of development demands and policy support, many Chinese enterprises have achieved fruitful practices in implementing sharing systems. For example, the "Oscar COSMOPlat" has connected 900,000 enterprises, the INDICS platform has integrated over 1.7 million pieces of industrial equipment, and Tao Factory has enabled online collaboration for millions of factories and traders. In addition, there are also Mould Lao crowd creative space, Shugen Internet, China Aerospace Cloud Platform, Haichuanghui, and the Shenyang Machine Tool Sharing Platform, among others. However, the supply and demand configuration of enterprise resources under shared manufacturing still faces many new challenges. These are primarily reflected in the lack of effective guidance for enterprises in sharing both internal and external manufacturing resources. Additionally, there are ongoing issues with the full utilization of resources and the optimization of manufacturing costs [1]. China's small and medium-sized manufacturing enterprises are characterized by their relatively small scale and limited investment. They are often at a disadvantage in terms of human and material resources.

 doi:10.20944/preprints202502.1852.v1

However, these enterprises are numerous and densely distributed. If external shared resources can be reasonably scheduled, it will effectively compensate for the shortage of manufacturing resources, thereby improving production efficiency, profitability, and customer satisfaction.

Production scheduling, as a crucial component of real-world production management, significantly impacts the production efficiency, cost, and competitiveness of enterprises [4,5]. However, most existing research on production scheduling focuses solely on the scheduling of enterprises' own machines. In the context of shared manufacturing, optimizing the rental costs of shared machines while enhancing the production efficiency of small and medium-sized manufacturing enterprises with limited production capacity is essential. Formulating a rational and efficient machine renting and scheduling scheme is thus an urgent problem to be addressed. This paper investigates the offline scheduling problem of identical parallel machines, considering fixed rental costs, unit variable rental costs, and shared service costs. The objective is to minimize the sum of the maximum completion time (makespan) and the total shared costs. The study aims to provide a scientific basis for schedulers to make informed decisions regarding resource sharing and scheduling schemes. By doing so, it helps enterprises reduce sharing costs while improving production efficiency and customer satisfaction, ultimately promoting accurate and efficient matching of the supply and demand of manufacturing resources.

The main contributions of this work are as follows:

(1) In the context of shared manufacturing, we incorporate shared machines with rental costs and shared service costs into the classic parallel machine scheduling problem. We investigate how machine sharing influences the production scheduling decisions of enterprises.
(2) We establish an integer programming model, analyze the properties of the problem, and provide the optimal scheduling case without renting machines, the range of the number of machine rentals, and the upper bound of the problem.
(3) We propose a heuristic algorithm based on the number of shared machine rentals to address large-scale instances of the considered problem.

The rest of this paper is organized as follows. Section 2 reviews the relevant papers in the literature. Section 3 describes the considered problem and constructs an integer programming model. Section 4 analyzes the properties of the problem. Section 5 proposes a heuristic algorithm. In Section 6, the experimental results, together with the sensitivity analysis, are presented. Finally, Section 7 concludes this work and suggests future research directions.

2. Literature Review

Manufacturing resource sharing offers new insights for optimizing resource allocation in the scheduling domain. Efficient resource utilization is closely tied to the rational design of sharing strategies and scheduling optimization. Existing related studies have analyzed the sharing strategies for various manufacturing resources, including processing capacity [6], multi-skilled human resources [7], orders [8], and information [9]. Since this paper primarily investigates the sharing and scheduling scheme of machine resources, this section focuses on reviewing studies related to the scheduling optimization of shared machine resources in detail. Additionally, this section also reviews the scheduling optimization research on the use costs of identical parallel machines, which is closely related to this paper.

2.1. Shared Machine Resource Scheduling Problem

Dereniowski and Kubiak [10,11] considered the preemptive job situation, assuming that each job could be executed on its private processor and simultaneously on possibly many processors shared. Their goal was to maximize the total weighted overlap of all jobs. Ji et al. [12] proposed a shared parallel machine scheduling model with machine processing set restrictions, aiming at minimizing the sum of makespan and total shared service cost. They proved that this problem is NP-hard and designed a fully polynomial-time approximation scheme. Wei and Wu [13] studied two two-machine

 doi:10.20944/preprints202502.1852.v1

hybrid flow-shop problems with fixed processing sequences in the context of shared manufacturing. Xu et al. [14] assumed that when a manufacturer with a certain number of parallel machines uses shared machines, it will obtain a certain proportion of shared benefits. They considered the time value of money and the due date-to-deadline window, with the objective of maximizing the total future value of profits. They designed a genetic algorithm and a heuristic algorithm. Zheng et al. [15] described a sharing platform as one parallel machine scheduling problem with machine-order type matching and order splitting. Their objective was to minimize the sum of the total processing cost of machines used and the total completion time of orders. They designed a Capability-Based Greedy heuristic and an improved Genetic Algorithm. Fu et al. [16] proposed a stochastic bi-objective two-stage open shop scheduling problem for shared professional equipment, aiming to minimize total tardiness and processing costs subject to various resource constraints. They designed a hybrid multiobjective migrating bird optimization combined with a genetic operation and a discrete event system. Xu et al. [17,18] studied the online scheduling optimization problem of shared parallel machines, considering the shared machine rental costs and discounts. They designed a competitive strategy.

2.2. Scheduling Problem with the Use Cost of Identical Parallel Machines

Ruiz-Torres et al. [19] assumed that machine costs consist of a fixed cost and a variable cost. For machine costs with concave functions, they derived general characteristics of optimal solutions regarding decisions on the number of machines to use and the way to load machines. Rustogi and Strusevich [20] assumed that the machine cost is a linear function of the number of machines used. They examined the impact of increasing the number of machines on both makespan and total flow time. Lee et al. [21] assumed that each job-machine combination may have a different cost. They studied two bi-criteria scheduling problems, proposed fast heuristics, and established the corresponding worst-case performance bounds. Li Kai et al. believed that each machine has a cost per unit of time that differs from machine to machine. They considered the problem of minimizing the makespan or the total completion time [22] and the problem of minimizing the maximum lateness [23]. Jiang et al. [24] assumed that each machine has a different constant processing cost per unit of time and designed corresponding approximation algorithms for two different production objectives. Anghinolfi et al. [25] considered the energy consumption rate and the time-of-use electricity prices of different machines. They designed an ad-hoc heuristic method to minimize both the makespan and the total energy consumption. Jarboui et al. [26] assumed that each machine has a fixed energy consumption rate and a positive energy price in each period. They explored the multiobjective problem of minimizing both the makespan and the total energy cost simultaneously and proposed an Epsilon Oscillation Algorithm. Wu et al. [27] considered the fixed usage cost of each machine. ε-LO-MILP, ε-LBBD, and NSGA-II are presented to solve the bi-objective scheduling problem of minimizing the maximum completion time and the total cost including machine usage cost and resource consumption cost.

Existing scheduling literature considering machine costs assumes that the machines have the same cost components and that the scheduling scope is limited to self-owned machines. However, there are only a few studies on shared machine resource scheduling that consider both own machines' scheduling schemes and shared machines' renting and scheduling schemes. The differences from this paper mainly include the cost components, optimization objectives, problem characteristics, and optimization methods. This paper focuses on the fixed rental cost, unit variable rental cost, and shared service cost associated with shared machines. The goal is to provide a scientific basis for the formulation of resource-sharing and optimized scheduling schemes for small and medium-sized manufacturing enterprises with insufficient production capacity, thereby promoting the sustainable development of manufacturing enterprises.

3. Problem Description

Consider a machine set $\mathcal{M} = \{Mi | i = 1, 2, \ldots, m + k\}$ consisting of $m + k$ parallel machines with the same processing speed. Here, $m \geq 2$ represents the number of machines owned by the manufacturer, and $k \geq 1$ denotes the number of shared machines that can be rented on the sharing

 doi:10.20944/preprints202502.1852.v1

platform. Let h be the number of shared machines actually rented. Given a set of jobs $\mathcal{J} = \{J_j | j = 1, 2, \ldots, n\}$, p_j ($p_j > 0$) denotes the processing time of job J_j. It is assumed that any job J_j can be processed either on the manufacturer's own machines or on a rented external machine. The manufacturer must pay a fixed rental cost a (e.g., preparation and cleaning costs of the shared machine) and a unit variable rental cost b for renting a shared machine M_i ($m+1 \leq i \leq m+k$). Specifically, the rental cost of the machine is $a + bC_i$, where C_i represents the completion time of machine M_i. In addition, we consider the shared service cost c_{ij} (e.g., transportation cost for raw materials or finished products, processing preparation cost, service cost, etc., collectively referred to as shared service cost) incurred when renting machine M_i ($m+1 \leq i \leq m+k$) to process job J_j. Here it is assumed that all shared parallel machines have the same fixed rental cost and unit variable rental cost. That is, for $\forall i \in [m+1, m+k]$, $a_i = a$, $b_i = b$. Meanwhile, we do not consider the existing sunk cost associated with the manufacturer's own machines. Therefore, for $\forall i \in [1, m]$, $a_i = b_i = c_{ij} = 0$.

Without loss of generality, some basic assumptions are made [25]: (1) Each job is assigned to only one selected machine for processing; (2) Each machine only processes one job at a time; (3) Preemption is not allowed.

Let σ denote a feasible scheduling scheme. The problem is to find the optimal scheduling scheme σ^* that minimizes the sum of the makespan and the total shared cost. The three-parameter representation of the problem can be expressed as $P_{m+k} \| C_{\max} + \sum S_i$, where P_{m+k}, $C_{\max}$, and $\sum S_i$ denote the identical parallel machine, the maximum completion time and the total shared cost (the sum of the total machine rental cost and the total shared service cost), respectively. If job J_j is scheduled to be processed on machine M_i, let $x_{ij} = 1$, otherwise, $x_{ij} = 0$. If machine M_i is used, let $\alpha_i = 1$, otherwise, $\alpha_i = 0$. Therefore, the shared cost of machine M_i is $S_i = a_i\alpha_i + b_i \sum_{j=1}^{n} p_j x_{ij} + \sum_{j=1}^{n} c_{ij} x_{ij}$. When $\forall b_i = 0$, the problem reduces to a scheduling problem considering only the fixed rental cost and the shared service cost of shared machines. When $\forall c_{ij} = 0$, the problem reduces to a scheduling problem considering the fixed rental cost and the unit variable cost of shared machines. It can be seen that this problem is more general.

4. Model Formulation

In this section, we introduce the notations, formulate the mathematical model based on the fixed rental cost, unit variable rental cost, and shared service cost associated with the shared machines, and provide detailed explanations of the constraints.

4.1. Notations

The indices, parameters, and decision variables used in the model are defined as follows:

Indices:

i: the index of machines, $i = 1, 2, \ldots, m+k$;

j: the index of jobs, $j = 1, 2, \ldots, n$.

Parameters:

m: the number of own machines;

k: the number of shared machines;

n: the number of jobs;

a_i: the fixed rental cost, $a_i = a$ if machine M_i ($i = m+1, m+2, \ldots, m+k$) is a shared machine, and $a_i = 0$, otherwise;

b_i: the unit variable rental cost, $b_i = b$ if machine M_i ($i = m+1, m+2, \ldots, m+k$) is a shared machine, and $b_i = 0$, otherwise;

p_j: the processing time of job J_j, $j = 1, 2, \ldots, n$;

c_{ij}: the shared service cost, $c_{ij} > 0$ if job J_j is processed on shared machine M_i ($i = m+1, m+2, \ldots, m+k$), and $c_i = 0$, otherwise;

Decision variables

$C_{\max}$: the makespan;

S_i: the shared cost of machine M_i;

 doi:10.20944/preprints202502.1852.v1

α_i: binary variable, $\alpha_i = 1$ when machine M_i is used;
x_{ij}: binary variable, $x_{ij} = 1$ when job J_j is processed on machine M_i.

4.2. Mathematical Model

Based on the settings of the above parameters and variables, the established integer programming model is as follows:

$$\min Z = C_{\max} + \sum_{i=1}^{m+k} S_i \tag{1}$$

Subject to:

$$S_i = a_i\alpha_i + b_i \sum_{j=1}^{n} p_j x_{ij} + \sum_{j=1}^{n} c_{ij} x_{ij}, \quad i = 1, \ldots, m+k \tag{2}$$

$$\sum_{i=1}^{m+k} x_{ij} = 1, \quad j = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{n} p_j x_{ij} \le C_{\max}, \quad i = 1, \ldots, m+k \tag{4}$$

$$x_{ij} \le \alpha_i, \quad i = 1, \ldots, m+k; j = 1, \ldots, n \tag{5}$$

$$x_{ij} \in \{0,1\}, \alpha_i \in \{0,1\}, \quad i = 1, \ldots, m+k; j = 1, \ldots, n \tag{6}$$

The objective function in Equation (1) aims to minimize the sum of the maximum completion time and the total shared cost over the planning time period. Constraint (2) indicates the shared cost of machine M_i. Constraint (3) ensures that each job must be scheduled on a certain machine. Constraint (4) indicates that the completion time of machine M_i is not greater than the maximum completion time $C_{\max}$. Constraint (5) establishes the relationship between variables α_i and x_{ij}, indicating that only the machines that are used will be assigned jobs. Constraint (6) specifies the value ranges of the decision variables.

5. Problem property

In this section, some problem-specific properties are derived for in-depth understanding and solving the problem $P_{m+k}\|C_{\max} + \sum S_i$.

When $a \to \infty$ or $a = 0$, $b = 0$, $\forall c_{ij} = 0$, the problem simplifies to $P\|C_{\max}$, which is known to be strong NP-hard [28]. Consequently, the problem $P_{m+k}\|C_{\max} + \sum S_i$ is also a strong NP-hard problem. When $a = 0$, the problem can be regarded as a special case of the problem discussed in the literature [12]. Therefore, we will focus on the case where $a > b \ge 0$ in the following discussion. When job J_j is assigned to a shared machine M_i ($m+1 \le i \le m+k$) for processing, $C_{\max}$ can decrease by at most p_j, while the rental cost will increase by at least $bp_j + c_{ij}$. If $b \ge 1$, the optimal scheduling would not involve renting the shared machine M_i. Hence, we will restrict our discussion to the case where $0 \le b < 1$.

Let $p_{\max} = \max_{j=1}^{n}\{p_j\}$, $P = \sum_{j=1}^{n} p_j$. For the problem $P|pmtn|C_{\max}$, where $pmtn$ indicates that jobs are interruptible, McNaughton's wrap-around rule [29] provides the optimal solution $C^*_{\max} = \max\{p_{\max}, P/m\}$. When jobs are non-interruptible, the optimal maximal completion time satisfies $C^*_{\max} \ge p_{\max}$ and $C^*_{\max} \ge P/m$. Therefore, properties 1 and 2 can be readily derived. In Property 1, $\lceil P/p_{\max} \rceil$ denotes the minimum number of machines required such that the maximum processing time of all jobs is greater than or equal to the average completion time of the machines. Consequently, when the number of owned machines is greater than or equal to this value, the optimal scheduling does not involve renting shared machines. When the fixed rental cost a of the shared machine is

 doi:10.20944/preprints202502.1852.v1

sufficiently large, the optimal scheduling also avoids renting machines. Property 2 specifies the value range of a in this scenario.

Property 1. *For problem $P_{m+k}\|C_{\max}+\sum S_i$, if $m \geq \lceil P/p_{\max} \rceil$, then the optimal scheduling does not rent shared machines, and the optimal objective value is $p_{\max}$.*

Property 2. *For problem $P_{m+k}\|C_{\max}+\sum S_i$, if $a \geq \lfloor P/(m+1) \rfloor$, then the optimal scheduling does not rent shared machines.*

As mentioned above, if job J_j is assigned to the shared machine M_i ($m+1 \leq i \leq m+k$), $C_{\max}$ will decrease by at most p_j, and the rental cost will increase by at least $bp_j + c_{ij}$. If $c_{ij} \geq (1-b)p_j$, then $bp_j + c_{ij} \geq p_j$, and the optimal scheduling will not rent the shared machine M_i to process job J_j. Therefore, Property 3 can be obtained.

Property 3. *For problem $P_{m+k}\|C_{\max}+\sum S_i$, if $c_{ij} \geq (1-b)p_j$, the optimal scheduling will not assign job J_j to the shared machine M_i for processing.*

Before arranging the processing sequence of jobs, it is necessary to first determine the number of rented machines. Property 4 provides a range of the possible number of machines to be rented for optimal scheduling.

Property 4. *Let the optimal scheduling σ^* rent h^* machines, then h^* satisfies:*
(1) When $1-bm > 0$, if $\sqrt{P(1-bm)/a} - m \leq 0$, $h^* = 0$, if $0 < \sqrt{P(1-bm)/a} - m < k$, $h^* \leq \lceil \sqrt{P(1-bm)/a} \rceil - m$, and if $\sqrt{P(1-bm)/a} - m \geq k$, $h^* \leq k$.
(2) When $1-bm \leq 0$, $h^* = 0$.

In the research problem, the manufacturer decides whether to rent a machine based on actual conditions in order to reduce the maximum completion time of the jobs. Therefore, when renting a shared machine is not considered, the maximum completion time C^{LPT} obtained by scheduling all jobs on the manufacturer's own machines using the LPT rule is an upper bound of the problem, i.e. Property 5.

Property 5. *For problem $P_{m+k}\|C_{\max}+\sum S_i$, $Z^* \leq C^{LPT}$, where C^{LPT} is the maximum completion time obtained using the LPT rule on own machines.*

6. Solution Approach

Although the optimal solution can be obtained using mathematical programming models, the computation time may be excessively long, particularly for large-scale problems. To provide decision-makers with an efficient scheduling solution in a timely manner, this section designs a heuristic algorithm based on the number of shared machine rentals (Number of shared machine-dependent heuristic, NSMD-H). This algorithm builds upon the results of the problem property analysis. According to the properties discussed in Section 5, when $m \geq \lceil P/p_{\max} \rceil$ or $a \geq \lfloor P/(m+1) \rfloor$ or $1-bm \leq 0$ or $1-bm > 0$ and $\sqrt{P(1-bm)/a} - m \leq 0$, the optimal scheduling does not involve renting shared machines. At this time, the LPT rule is applied to schedule jobs on the manufacturer's own machines, ensuring that $C^{LPT} \leq (\frac{4}{3} - \frac{1}{3m})C^*_{\max}$ [30]. When none of the above four conditions hold, the heuristic algorithm NSMD-H is employed to solve the problem. If the LPT rule is still used to schedule jobs on the own machines, then the performance ratio is $Z(LPT)/Z^* = O(k/m)$. For example, consider a scenario with $n \to \infty$ jobs, each with a processing time of p, which can be processed on m own machines and k shared machines, where n is a multiple of m and $m+k$. Let $a \to 0$, $b = 0$, and $\forall c_{ij} = 0$. In this case, the performance ratio is given by: $Z(LPT)/Z^* = (np/m)/[(np/(m+k) + ak] = 1 + k/m$.

The main idea of the heuristic algorithm NSMD-H is to first determine the number of rented machines and then obtain the minimum value of the maximum completion time for the own machines.

 doi:10.20944/preprints202502.1852.v1

Based on this value and the ratio of service cost to job processing length, the algorithm determines which jobs should be assigned to the own machines and which to the shared machines. It then schedules and adjusts the jobs accordingly. Specifically, the algorithm proceeds as follows:

(1) **Initial Scheduling:** Schedule all jobs on the own machines using the LPT rule.

(2) **Determine Number of Rented Machines:** According to Property 4, determine the minimum number of rented machines H. If $H > 0$, iterate over the number of rented machines h from 1 to H and perform the following steps for each h:

(i) **Job Assignment:** Determine the jobs assigned to the own machines and each shared machine. Schedule jobs on the own machines using the LPT rule, and sort jobs on the shared machines in non-decreasing order based on the ratio of service cost to job processing time.

(ii) **Remove Duplicates:** Eliminate any jobs that are scheduled multiple times on the shared machines to ensure each job is processed only once.

(iii) **Adjust Shared Machine Jobs:** For shared machines with completion times exceeding those of the own machines, adjust the job assignments to balance the load.

(iv) **Optimize Machine Utilization:** Adjust jobs on shared machines with lower completion times to minimize the number of rented machines and reduce the overall objective value.

(v) **Reassignment Check:** Evaluate whether jobs initially assigned to the own machines can be reassigned to rented shared machines to further reduce the objective value.

(vi) **Output Results:** Record the scheduling result and the corresponding objective value for each h.

(3) **Select Optimal Solution:** After completing the above steps for all possible values of h, select the scheduling scheme with the smallest objective value as the final result of the algorithm.

The execution flow of NSMD-H is illustrated in Figure1. The specific algorithm is described as follows:

Step 0 Assign all jobs to own machines according to the LPT rule to obtain the initial schedule σ_0 and the target value Z_0. According to Property 4, if $\sqrt{P(1-bm)/a} - m \leq 0$, then let $H = 0$. If $0 < \sqrt{P(1-bm)/a} - m < k$, then set $H = \lceil \sqrt{P(1-bm)/a} - m \rceil$. If $\sqrt{P(1-bm)/a} - m \geq k$, then set $H = k$. If $H > 0$, then for each h from 1 to H, execute Steps 1 to 4 in sequence. Otherwise, proceed to Step 5.

Step 1 Sort the jobs in non-increasing order based on $\min_{i \in \{m+1,\ldots,m+k\}} c_{ij}/p_j$. Select the jobs sequentially until the sum of the processing times of the selected jobs exceeds $Pm/(m+h)$, and assign these selected jobs to own machines. Then, assign the jobs that satisfy $\min_{i \in \{m+1,\ldots,m+k\}} c_{ij} \geq (1-b)p_j$ to own machines according to Property 3. The jobs of own machines are scheduled according to the LPT rule, and the maximum completion time $C^0_{\max}$ is calculated. The jobs on shared machines are sorted in non-decreasing order according to c_{ij}/p_j, and the completion times of these machines are calculated. Let the set of completion times for the shared machines be $\mathcal{CI} = \{C_{m+1}, \ldots, C_{m+k}\}$. If there are jobs that are processed repeatedly on the shared machines, run Step 2. Otherwise, obtain the feasible schedule σ_h and determine whether there is a value greater than $C^0_{\max}$ in $\mathcal{CI}$, if so, run Step 3, otherwise, run Step 4.

Step 2 Delete the repeatedly processed jobs, which mainly involves the following two steps:

Step 2.1 Let the set of repeatedly processed jobs on the shared machines be denoted by $\mathcal{J}_r$, and let $\mathcal{CI}' = \mathcal{CI}$. If $\max\{\mathcal{CI}'\} > C^0_{\max}$, determine whether there are any repeatedly processed jobs on the machine with a completion time equal to $\max\{\mathcal{CI}'\}$. If so, delete the job with the largest processing position among the repeatedly processed jobs, update $\mathcal{CI}$, $\mathcal{CI}'$, and $\mathcal{J}_r$. If not, set the completion time of the corresponding machine in $\mathcal{CI}'$ equal to 0, and repeat this step until $\max\{\mathcal{CI}'\} \leq C^0_{\max}$.

Step 2.2 If $\mathcal{J}_r \neq \emptyset$, for each repeatedly processed job, retain only the instance on the machine with the smallest ratio of the job processing position to the number of jobs. If multiple machines have equal ratios, retain the job on the machine with the smallest machine number and delete the repeatedly processed jobs on the remaining machines. Sort the jobs on the shared machines by c_{ij}/p_j

 doi:10.20944/preprints202502.1852.v1

in a non-decreasing order, update $\mathcal{CI}$, and obtain the schedule σ_h. If $\max\{\mathcal{CI}\} > C^0_{\max}$, run Step 3, otherwise, run Step 4.

Step 3 Let M_I be the shared machine with a completion time equal to $\max\{\mathcal{CI}\}$. If there are already rented machines with completion times less than $C^0_{\max}$ (denoted by the set $\bar{\mathcal{M}}$), proceed to Step 3.1. Then, if $C_I > C^0_{\max}$, and there are unrented machines (denoted by the set $\mathcal{M}_0$), and $k - |\mathcal{M}_0| < h$, let the number of machines that need to be rented be $h_0 = \min\{\lceil (C_I - C^0_{\max})/C^0_{\max} \rceil, h - (k - |\mathcal{M}_0|)\}$, and run Step 3.2. Then, if $C_I > C^0_{\max}$, proceed to Step 3.3. Repeat this step until $\max\{\mathcal{CI}\} \leq C^0_{\max}$.

Step 3.1 Adjust the jobs on M_I in sequence. For example, consider the job J_g on M_I. Sort the machines in $\bar{\mathcal{M}}$ according to c_{ig} (where $i \in \bar{\mathcal{M}}$) in non-decreasing order, and determine whether J_g can be moved to each machine for processing in sequence. For instance, consider M_l in $\bar{\mathcal{M}}$. If J_g is moved to M_l, the following conditions must be satisfied: (1) $c_{lg} - c_{Ig} \leq z$ (where $z = p_g$ if $\max\{\mathcal{CI}\} - C^0_{\max} \geq p_g$, and $z = \max\{\mathcal{CI}\} - C^0_{\max}$ otherwise); (2) $C_l \leq C^0_{\max}$; (3) $c_{lg} < (1-b)p_g$. If all these conditions are met, move J_g to M_l, update $\mathcal{CI}$ and σ_h, and do not consider other adjustable machines after M_l. If $C_I \leq C^0_{\max}$, there is no need to adjust any other jobs on M_I after J_g.

Step 3.2 If $(C_I - C^0_{\max})/C^0_{\max} \geq 1$, set $D = C^0_{\max}$, otherwise, set $D = C_I - C^0_{\max}$. Each machine in $\mathcal{M}_0$ performs the following operations in sequence. For example, consider machine M_q in $\mathcal{M}_0$: The jobs on M_I are sorted by c_{qj}/p_j in non-decreasing order, and the processing times of the jobs that satisfy $c_{qj} < (1-b)p_j$ are added in sequence. When the cumulative sum of the jobs' processing times SP is greater than D, stop the summation. Let $\mathcal{J}_q$ be the set of jobs whose processing times are summed. If $C_I - SP \geq C^0_{\max}$, set $W_q = SP - (a + \Delta c)$, otherwise, set $W_q = (C_I - SP) - (a + \Delta c)$, where Δc represents the increase in service cost after adjusting the job. After all machines in $\mathcal{M}_0$ complete these operations, obtain the sets $\mathcal{W}$ and $\mathcal{J}$. If $\max\{\mathcal{W}\} \geq 0$, identify the machine M_u corresponding to $\max\{\mathcal{W}\}$. Move the jobs in $\mathcal{J}_u$ from M_I to M_u, update $\mathcal{CI}$, $\mathcal{M}_0$, and σ_h. Perform the above operation h_0 times.

Step 3.3 The jobs on M_I are sorted in non-increasing order based on c_{Ij}/p_j and are assigned sequentially to own machines. When the sum of the processing times of the assigned jobs exceeds $m(C_I - C^0_{\max})/(m+1)$, no further jobs from M_I are assigned to own machines. Update $\mathcal{CI}$, schedule the jobs on own machines according to the LPT rule, and update $C^0_{\max}$ and σ_h.

Step 4 Let the set of rented machines be $\mathcal{M}_1$, and initialize $\sigma'_h = \sigma_h$ and $\mathcal{CI}' = \mathcal{CI}$. If $\mathcal{M}_1 \neq \varnothing$, identify the machine with the smallest completion time in $\mathcal{M}_1$ as M_s, with completion time C_s. If $C_s > (|\mathcal{M}_1| - 1)C^0_{\max} - (\sum_{i \in \mathcal{M}_1} C_i - C_s)$, run Step 5. Otherwise, adjust the jobs on M_s in σ'_h as follows. For example, consider a job J_G on M_s. Sort the machines in $\mathcal{M}_1 \setminus \{M_s\}$ according to c_{iG} in non-decreasing order. Initialize $loop = 0$. Determine if J_G can be moved to them for processing in sequence. Take the machine M_L in $\mathcal{M}_1 \setminus \{M_s\}$ as an example. If moving J_G to M_L satisfies both $C_L + p_G \leq C^0_{\max}$ and $c_{LG} < (1-b)p_G$, then move J_G to M_L. Calculate the increase in service cost Δc. Set $loop = 1$, update $\mathcal{CI}'$ and σ'_h. No further machines need to be considered after M_L. If $loop = 0$, set $\sigma'_h = \sigma_h$, $\mathcal{CI}' = \mathcal{CI}$. No further jobs need to be judged after J_G. If $a - \sum \Delta c > 0$, set $\sigma_h = \sigma'_h$, $\mathcal{CI} = \mathcal{CI}'$, otherwise, set $\sigma'_h = \sigma_h$, $\mathcal{CI}' = \mathcal{CI}$. Remove M_s from $\mathcal{M}_1$ $\mathcal{M}_1 = \mathcal{M}_1 \setminus \{M_s\}$, repeat the above operations until $\mathcal{M}_1 = \varnothing$. Finally, calculate the target value Z_h for σ_h.

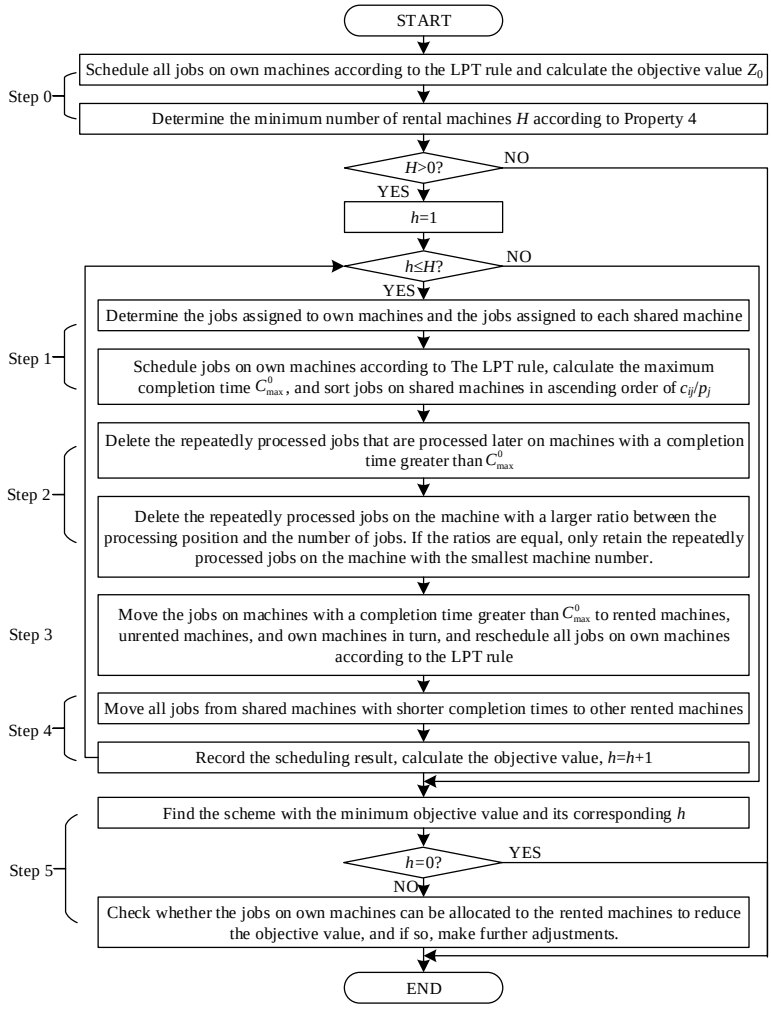
Step 5 Select the scheduling scheme with the minimum objective value and perform local adjustments to further optimize it.

Step 5.1 Select the scheduling σ and h corresponding to the minimum objective value from Z_0 and $Z = \{Z_1, Z_2, \ldots, Z_H\}$. If $h = 0$, then $\sigma = \sigma_0$ is the scheduling result of the algorithm, with the objective value being Z_0. Otherwise, let the completion time of all machines be $\mathcal{C} = \{C_1, \ldots, C_{m+k}\}$, set $c'_{ij} = c_{ij}$, let $\mathcal{M}_1$ be the set of rented machines, and let $\mathcal{J}_0$ be the set of jobs on own machines.

Step 5.2 Find the machine $M_{i'}$ and the job $J_{j'}$ that correspond to $\min_{i \in \mathcal{M}_1, j \in \mathcal{J}_0} c'_{ij}$. If assigning job $J_{j'}$ to the machine $M_{i'}$ and sorting the jobs on own machines according to the LPT rule results in a decrease in the objective value, then perform this operation and update $\mathcal{C}$ and $\mathcal{J}_0$. Otherwise, set $c'_{i'j'} = \sum p_j$.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 February 2025 doi:10.20944/preprints202502.1852.v1

Repeat Step 5.2 until $\forall c'_{i'j'} = \sum p_j, i' \in \mathcal{M}_1, j' \in \mathcal{J}_0$ or the maximum completion time of the shared machines is greater than or equal to the maximum completion time of the own machines.



Note: The corresponding conditions need to be met when executing Step 2 and Step 3

Figure 1. The heuristic algorithm flowchart.

In this heuristic algorithm, the computation times of Step 0 and Step 1 are $O(n \log n)$. The computation time of Step 2 is $O(nk)$. The computation times of Step 3.1, Step 3.2, and Step 3.3 are $O(nk)$, $O(nk^2)$, and $O(n \log n)$, respectively. Since Step 3.1 to Step 3.3 are executed at most k times, the computation time of Step 3 is $\max\{O(nk^3), O(kn \log n)\}$. The computation times of Step 4 and Step 5 are $O(nk^2)$ and $O(nk)$, respectively. Since Step 1 to Step 4 are executed at most k times, the computation time complexity of NSMD-H is $\max\{O(nk^4), O(k^2 n \log n)\}$.

7. Computation Experiments

This section analyzes the performance of the proposed NSMD-H algorithm through numerical experiments. The mathematical programming model and the algorithm are coded on Matlab R2014a and executed on a personal computer (Intel Core i7-8550U, 8GB memory, Windows 10 64 bit). CPLEX12.5 is employed to solve the integer programming model accurately. The calculation time limit of CPLEX is set to 7200 seconds (s). The following will introduce data generation, experimental results, and analysis in sequence.

The test cases are randomly generated, and the case scale is shown in Table 1.

 doi:10.20944/preprints202502.1852.v1

Table 1. The related parameters and the range of values of the instances.

Parameter	Value range
Number of jobs n	20, 50, 100, 200, 500, 800
Number of own machines m	2, 4, 6, 8, 10
Number of shared machines k	4, 8, 10
Processing time of job J_j p_j	$U[1, 20]$
Fixed rental cost of shared machine a	$U(0, 10]$
Unit variable rental cost of shared machine b	$U[0, 1/m)$
Shared service cost c_{ij}	$U[0, \lceil p_j \rceil]$

Note: $U[x, y]$ represents a uniform distribution on $[x, y]$.

For the above cases, CPLEX, NSMD-H, and a Genetic Algorithm (GA) are used to solve the problem, and the LPT rule is used to schedule jobs on own machines. The results obtained from these methods are compared and analyzed. The GA algorithm in this section is based on the work in [14], which shares similarities with this paper by considering jobs on both own and external machines. The GA algorithm from [14] is known for its high computational accuracy. Therefore, the GA algorithm in this section adopts the same individual gene coding representation, mutation operator, and selection operator as in [14]. However, extensive experiments have shown that the single-point crossover operator used in [14] is not well-suited for this problem. Consequently, the GA algorithm in this section employs a two-point crossover operator, specifically for the second half of the parent chromosome. Additionally, the scheduling result obtained from the NSMD-H algorithm is transformed into a chromosome to form part of the initial population for the GA algorithm. An elite retention strategy is also implemented to preserve the individual with the highest fitness after each iteration.

A total of 51 sets of experiments are conducted, varying the number of machines, job sizes, and the values of the fixed rental cost a and the unit variable cost b of shared machines. The specific results for small-scale and large-scale cases are presented in Table 2 and 3 respectively. Sensitivity analysis is performed on a and b to assess their impact on the experimental results. This analysis examines how changes in a and b (while keeping other parameters constant) affected the outcomes, using the control variable method. The experimental results are shown in Table 4. The columns Z, CPU, and h record the objective value, calculation time (in seconds), and the number of rented machines of each method, respectively. For the GA, the results shown are the average values obtained from five runs. The relative error gap between the algorithm's objective value and the optimal value is calculated as $gap = (Z - Z^*)/Z^* \times 100\%$. Since the calculation time for scheduling jobs on own machines using the LPT rule was consistently less than 0.01 seconds across all cases, the calculation time for LPT is not listed in the tables.

Analyzing Table 2 and 3 the following conclusions can be drawn:

(1) When the number of jobs and machines is small, CPLEX can accurately solve the problem. However, as the scale of the problem increases, its computation time tends to rise significantly, ranging from 0.65s to 7200s. When the number of jobs and machines is large, CPLEX becomes unstable in terms of computational efficiency. In some cases, CPLEX is unable to provide an accurate solution and corresponding scheduling scheme within two hours. For example, this occurs when there are 200 jobs, 6 own machines, 10 shared machines, and 800 jobs in total.

 doi:10.20944/preprints202502.1852.v1

Table 2. Computational results for small-scale problems.

n	m	k	CPLEX			NSMD-H				GA				LPT	
			Z^*	CPU	h^*	Z	CPU	gap	h	Z	CPU	gap	h	Z	gap
20	2	4	109.39	2.33	3	111.34	0.14	1.78	4	109.87	6.43	0.44	4	133.91	22.41
20	2	8	62.52	0.65	6	63.94	0.17	2.27	6	63.35	6.65	1.33	6	105.91	69.39
20	2	10	92.05	0.67	2	93.50	0.15	1.57	2	92.13	7.34	0.09	2	104.58	13.61
20	4	4	44.60	3.41	1	45.28	0.16	1.52	3	45.26	10.39	1.49	3	46.34	3.89
20	4	8	55.89	6.47	0	56.66	0.09	1.38	0	56.38	13.99	0.88	0	56.66	1.38
20	4	10	50.89	0.78	4	51.70	0.17	1.59	4	51.22	11.17	0.65	4	62.80	23.40
50	2	4	144.61	3.48	4	148.22	0.49	2.50	4	147.75	13.63	2.17	4	267.75	85.16
50	2	8	166.37	4.19	5	178.26	0.26	7.14	6	174.16	13.20	4.68	6	296.18	78.02
50	2	10	135.39	1.40	9	140.42	0.34	3.72	8	140.05	11.93	3.44	8	264.36	95.25
50	4	4	106.24	2.12	3	113.33	0.23	6.67	3	110.37	17.45	3.89	3	125.37	18.00
50	4	8	83.80	2.36	8	86.06	0.26	2.69	8	85.81	12.03	2.40	8	124.44	48.49
50	4	10	89.83	7.36	6	96.57	0.32	7.50	6	93.17	12.85	3.72	6	133.80	48.94
100	4	4	275.83	2.07	0	275.94	0.15	0.04	0	275.94	16.05	0.04	0	275.94	0.04
100	4	8	182.79	2.98	7	185.18	0.21	1.31	7	184.95	7.80	1.18	7	274.62	50.24
100	4	10	212.26	10.95	9	215.43	0.26	1.49	9	215.23	8.40	1.40	9	290.05	36.65
100	6	4	166.83	27.35	4	168.24	0.18	0.85	4	168.21	9.20	0.83	4	270.94	62.40
100	6	8	132.82	207.66	7	138.47	0.24	4.25	6	138.25	8.60	4.09	6	173.43	30.57
100	6	10	142.97	9.69	8	144.99	0.25	1.41	8	144.56	8.56	1.11	8	164.35	14.95
	Average		125.28	16.44	4.78	128.53	0.23	2.76	4.89	127.59	10.87	1.88	4.89	176.19	39.04

Table 3. Computational results for large-scale problems.

n	m	k	CPLEX			NSMD-H				GA				LPT	
			Z^*	CPU	h^*	Z	CPU	gap	h	Z	CPU	gap	h	Z	gap
200	6	4	272.07	9.35	4	276.04	0.26	1.46	4	276.02	11.70	1.45	4	349.87	28.60
200	6	8	292.25	24.07	8	297.24	0.35	1.71	8	296.96	10.65	1.61	8	355.80	21.75
200	6	10	—	—	—	198.82	0.38	—	10	198.80	9.78	—	10	331.79	—
200	8	4	230.63	1126.93	4	233.25	0.29	1.14	4	233.20	11.22	1.12	4	271.50	17.72
200	8	8	241.76	6739.49	5	250.41	0.37	3.58	7	250.30	11.67	3.53	7	263.22	8.88
200	8	10	—	—	—	194.88	0.37	—	10	194.88	10.69	—	10	259.48	—
500	8	4	576.97	147.11	4	579.68	0.84	0.47	4	579.44	18.20	0.43	4	658.87	14.20
500	8	8	—	—	—	667.66	0.97	—	8	667.48	18.25	—	8	670.36	—
500	8	10	500.11	1921.60	10	505.61	0.86	1.10	10	505.54	15.91	1.09	10	658.66	31.70
500	10	4	—	—	—	450.83	0.89	—	4	450.83	17.87	—	4	515.71	—
500	10	8	429.16	173.67	8	434.65	0.49	1.28	8	434.56	17.37	1.26	8	529.86	23.46
500	10	10	—	—	—	460.50	0.99	—	10	460.38	16.88	—	10	519.11	—
800	10	4	—	—	—	742.75	2.26	—	4	742.75	25.15	—	4	842.93	—
800	10	8	—	—	—	662.09	1.97	—	8	662.06	21.82	—	8	801.92	—
800	10	10	—	—	—	713.75	2.02	—	10	713.24	22.67	—	10	825.33	—
	Average		363.28	1448.89	6.14	444.54	0.89	1.53	7.27	444.43	15.99	1.50	7.27	523.63	20.90

'-' indicates that the example has not produced an exact solution within two hours.

(2) The NSMD-H algorithm demonstrates a short computation time, even for large-scale cases with 800 jobs, 10 own machines, and 10 shared machines, with a running time of only 2.02 seconds. In terms of computational accuracy, the NSMD-H algorithm performs well. The gap between the objective value obtained by the NSMD-H algorithm and the optimal value obtained by CPLEX ranges from 0.04% to 7.5%. The average gap values for small-scale and large-scale cases are 2.76% and 1.53%, respectively. The difference between the number of rented machines of the NSMD-H algorithm and the number of rented machines of optimal scheduling is not significant. The maximum difference observed is 2 machines. This indicates that the objective value is influenced not only by the number of rented machines but also by the scheduling arrangement of jobs.

(3) The computation time of the GA algorithm gradually increases from 6.43s to 25.15s, indicating relatively high running efficiency. However, compared to the NSMD-H algorithm, the GA algorithm does not significantly improve solution quality. The average gap values of small-scale and large-scale cases are 1.88% and 1.50%, respectively. Additionally, the number of shared machines rented by the GA algorithm is consistent with that of the NSMD-H algorithm. The primary reason for this is that the quality of solutions generated by the GA algorithm through random chromosome initialization differs significantly from that of the NSMD-H algorithm.

 doi:10.20944/preprints202502.1852.v1

Consequently, the better solutions obtained during GA iterations are mostly local adjustments to the solutions initially obtained by the NSMD-H algorithm. This results in minimal changes to the objective value. Therefore, the GA algorithm, while increasing computation time, does not significantly enhance the quality of solutions obtained by the NSMD-H algorithm.

(4) The LPT rule, which schedules jobs only on own machines without considering rented machines, has a very low computation time (no more than 0.01 seconds). However, it deviates significantly from optimal scheduling. In some cases, the maximum deviation reaches 95.25%.

Table 4. Computational results for different a and b.

n	m	k	a	b	CPLEX			NSMD-H				GA				LPT	
					Z^*	CPU	h^*	Z	CPU	gap	h	Z	CPU	gap	h	Z	gap
50	4	8	0.5	0.01	71.66	2.38	8	75.56	0.19	5.45	7	74.65	7.34	4.17	7	129.23	80.35
50	4	8	0.5	0.05	84.58	4.43	7	86.67	0.18	2.47	7	86.13	7.58	1.83	7	129.23	52.79
50	4	8	0.5	0.1	99.68	4.24	7	101.07	0.18	1.40	7	100.48	8.26	0.80	7	129.23	29.64
50	4	8	2.0	0.01	82.42	2.71	7	85.03	0.19	3.17	6	84.16	7.74	2.12	6	129.23	56.80
50	4	8	2.0	0.05	94.93	1.95	6	96.14	0.21	1.27	6	95.47	8.18	0.57	6	129.23	36.13
50	4	8	2.0	0.1	109.39	1.75	6	110.54	0.20	1.06	6	109.69	7.72	0.28	6	129.23	18.14
50	4	8	5.0	0.01	99.57	5.07	5	103.03	0.17	3.48	6	102.31	7.93	2.76	6	129.23	29.79
50	4	8	5.0	0.05	110.44	5.78	4	114.14	0.18	3.35	6	113.58	8.97	2.84	6	129.23	17.01
50	4	8	5.0	0.1	121.55	2.09	3	127.22	0.14	4.66	5	125.15	9.54	2.96	5	129.23	6.32
200	6	10	0.5	0.01	186.28	2628.48	10	194.99	0.36	4.67	10	193.16	10.79	3.69	10	341.28	83.21
200	6	10	0.5	0.05	—	—	—	242.31	0.34	—	10	240.24	11.28	—	10	341.28	—
200	6	10	0.5	0.1	297.03	64.29	10	298.39	0.44	0.46	10	297.43	11.56	0.14	10	341.28	14.90
200	6	10	2.0	0.01	201.29	141.02	10	209.99	0.37	4.32	10	209.47	10.85	4.06	10	341.28	69.54
200	6	10	2.0	0.05	251.18	36.56	10	257.31	0.35	2.44	10	254.53	10.65	1.33	10	341.28	35.87
200	6	10	2.0	0.1	311.01	22.40	8	313.39	0.45	0.77	10	312.47	11.11	0.47	10	341.28	9.73
200	6	10	5.0	0.01	230.27	357.82	9	238.25	0.36	3.47	9	237.05	13.12	2.95	9	341.28	48.21
200	6	10	5.0	0.05	278.83	25.64	8	285.57	0.37	2.42	9	283.71	11.17	1.75	9	341.28	22.40
200	6	10	5.0	0.1	330.15	36.87	5	341.28	0.24	3.37	0	341.25	20.14	3.36	0	341.28	3.37
Average					174.13	196.68	7.24	182.27	0.27	2.84	7.44	181.16	10.22	2.12	7.44	235.26	36.13

'-' indicates that the example has not produced an exact solution within two hours.

The following conclusions can be drawn from analyzing Table 4:

(1) When other parameters remain unchanged, an increase in the fixed rental cost a leads to a higher objective value and a gradual decrease in the number of shared machines rented. The same trend is observed for the unit variable rental cost b.

(2) The conclusions drawn from analyzing Tables 2 and 3 are consistent. CPLEX exhibits significant variability in computational efficiency, with computation times ranging from 1.75 seconds to 7200 seconds. The values of parameters a and b have a substantial impact on CPLEX's computational time, with no discernible pattern.

(3) The NSMD-H and GA algorithms demonstrate high computational efficiency, unaffected by the values of a and b. The average computational times are 0.27 seconds and 10.22 seconds, respectively. Both algorithms achieve higher computational accuracy, with average deviations of 2.84% and 2.12%, respectively.

(4) All jobs are scheduled on own machines using the LPT rule. The objective value is independent of the values of a and b. However, the deviation from the optimal value is significant, particularly in cases where the shared machine rental cost is low and the optimal number of rented machines is high(e.g., the 1st and 10th cases in the table). As previously noted, when parameters n, a, b, and c_{ij} take extreme values, the deviation becomes unbounded.

In summary, the NSMD-H and GA algorithms are effective in solving the problem, maintaining high computational accuracy and solution efficiency even as the problem size increases. While both algorithms achieve similar levels of computational accuracy, the NSMD-H algorithm demonstrates superior computational efficiency, meeting the dual requirements of solution quality and computational speed necessary for practical production environments.

The primary motivation for studying this machine-sharing scheduling problem is to provide a theoretical foundation for enterprise managers to make informed decisions regarding the rental of

 doi:10.20944/preprints202502.1852.v1

shared machines and the arrangement of job production schedules. Through mathematical modeling, property analysis, algorithm design, and problem-solving, the following management insights are offered to assist enterprise decision-makers:

(1) The rental and scheduling results of machines are influenced by multiple factors. Enterprises should comprehensively consider their own production conditions (such as the number of own machines), the characteristics of jobs to be processed (including the number of jobs and their processing times), the production goals of the enterprise, and the costs associated with shared machines (fixed rental costs, unit variable costs, shared service costs).

(2) Enterprises can utilize the proposed model and algorithm to determine the number of shared machines to be rented and to arrange the order of jobs on their own machines and shared machines. This approach helps reduce the maximum completion time of jobs (i.e., customer waiting time) while optimizing the enterprise's shared costs. For small-scale problems, the model can be used to obtain exact solutions. For large-scale problems, the proposed algorithm can efficiently generate near-optimal solutions to support decision-making.

8. Conclusions

This paper examines an offline scheduling problem for identical parallel machines, incorporating the practical scenario of renting shared machines in manufacturing enterprises. Manufacturers with a certain number of machines can rent additional shared machines to process jobs. However, they must incur fixed rental costs, unit variable rental costs, and shared service costs associated with both jobs and machines. The optimization objective is to minimize the sum of the makespan and the total shared cost. To address this problem, an integer programming model is first developed. Subsequently, the optimal scheduling without renting machines and the range of optimal rental quantities when renting machines are analyzed. Based on the problem's properties, the NSMD-H algorithm is proposed. Data experiments confirm that the proposed model can accurately solve small-scale cases, while the NSMD-H algorithm can quickly obtain near-optimal solutions for both small-scale and large-scale cases.

Future research directions may include (i) designing heuristics that can solve the problem more efficiently; (ii) considering more practical conditions, such as various rental discounts and multi-resource sharing; (iii) investigating scheduling problems where jobs are processed on shared machines of different machine types (e.g., uniform parallel machines, unrelated parallel machines, etc.); and (iv) studying the problem with other objective functions, such as scheduling problems with durations (maximum delay time, delay time, number of late jobs, etc.), or multi-objective functions.

Author Contributions: Conceptualization, Y.X. and R.Z.; methodology, R.Z. and F.Z.; software, R.Z.; validation, R.Z.; writing—original draft preparation, R.Z. and F.X.; writing—review and editing, R.Z. and F.Z.

Funding: This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 72271051, 72071144 and 71832001. This work was also supported by the Fundamental Research Funds for the Central Universities under Grant 2232018H-07.

Acknowledgments: The authors would like to thank the anonymous referees for their constructive comments.

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

Appendix A

Proof of Property 2. Let $C^*_{\max}$ denote the optimal maximum completion time of n jobs allocated on m own machines. If a shared machine is added, let $\tilde{C}^*_{\max}$ represent the optimal maximum completion time of n jobs allocated on $m+1$ machines, and let $C_{\min}$ be the minimum machine completion time. Then $C_{\min} \leq P/(m+1)$. Since jobs are non-interruptible, $C_{\min} \leq \lfloor P/(m+1) \rfloor$. Therefore, adding a machine can reduce $C_{\max}$ by at most $\lfloor P/(m+1) \rfloor$ (i.e. $C^*_{\max} - \tilde{C}^*_{\max} \leq \lfloor P/(m+1) \rfloor$). When the fixed rental cost

 doi:10.20944/preprints202502.1852.v1

a of the newly added shared machine is greater than or equal to the maximum reduction $\lfloor P/(m+1) \rfloor$ in the maximum completion time, the optimal scheduling does not rent a shared machine. □

Proof of Property 4. When the shared service cost c_{ij} is not considered, let Z_p^* denote the optimal value of the interruptible scheduling problem, with h_p^* representing the number of shared machines rented in the optimal scheduling. Similarly, let Z_{np}^* denote the optimal value of the non-interruptible scheduling problem, with h_{np}^* representing the number of machines rented in the optimal scheduling. We first prove that $h^* \leq h_{np}^* \leq h_p^*$. Next, we demonstrate that in the optimal scheduling of the interruptible problem, for $\forall i \in [1, m + h_P^*]$, the machine completion time is equal to $P/(m + h_P^*)$. Finally, we determine the value of h_P^* in the interruptible problem.

① It is evident that $Z_{np}^* \geq Z_p^*$. This inequality holds because the reduction in the maximum completion time when jobs can be interrupted is greater than or equal to the increase in rental costs. One possible reason for the increase in rental costs is the higher number of rented machines, implying that $h_{np}^* \leq h_P^*$. Moreover, since the problem accounts for the shared service cost c_{ij}, which can potentially reduce the number of rented machines, it follows that $h^* \leq h_{np}^* \leq h_P^*$.

② In the interruptible scheduling problem without considering the shared service cost. (i) Since renting a shared machine incurs a unit variable cost ($0 \leq b < 1$), the maximum completion time of own machines is greater than or equal to that of the rented machines; (ii) If the completion times of the own machines are not equal, adjusting them to be equal can reduce $C_{\max}$; (iii) If the completion times of the rented machines are not equal, adjusting them to be equal will not change the target value; (iv) If the completion times of own machines are greater than those of the rented machines, adjusting them to be equal. If the target value increases, then $h_p^* = 0$ and the completion times of own machines are equal to P/m. Otherwise, $h_p^* > 0$ and the completion times of the used machines are equal to $P/(m + h_p^*)$. For any scheduling σ_p, if the above four situations exist, the optimal scheduling σ_p^* can be obtained through appropriate adjustments. Figure A1 illustrates the effects of these adjustments schematically. Since the scheduling of jobs on each machine is not shown, the symbol "· · · · · · " is used in the diagram to simplify the representation.

③ In the interruptible scheduling problem without considering the shared service cost, for $\forall i \in [1, m + h_p^*]$, the machine completion time is equal to $P/(m + h_p^*)$. This leads to $Z_p^* = P/(m + h_p^*) + ah_p^* + bPh_p^*/(m + h_p^*)$. Define the function $f(x) = P/(m + x) + ax + bPx/(m + x)$. The first and second derivatives of $f(x)$ are $f'(x) = a - P(1 - bm)/(m + x)^2$ and $f''(x) = 2P(1 - bm)/(m + x)^3$, respectively. When $1 - bm > 0$, the second derivative $f''(x)$ is positive, and the first derivative $f'(x)$ increases monotonically. Currently, $f(x)$ minimizes at $x = 0$ when $\sqrt{P(1 - bm)/a} - m \leq 0$, at $x = \sqrt{P(1 - bm)/a} - m$ when $0 < \sqrt{P(1 - bm)/a} - m < k$, and at $x = k$ when $\sqrt{P(1 - bm)/a} - m \geq k$. When $1 - bm \leq 0$, the first derivative $f'(x)$ is positive, indicating that $f(x)$ is monotonically increasing. The minimum value of $f(x)$ occurs at $x = 0$.

 doi:10.20944/preprints202502.1852.v1

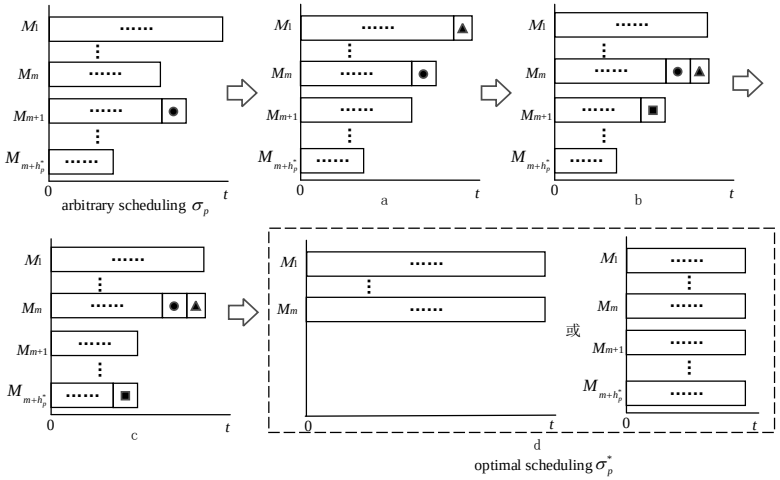


Figure A1. The scheduling effect diagram of arbitrary scheduling adjustment to optimal scheduling.

References

1. Wang, G.; Zhang, G.; Guo, X.; Zhang, Y.F. Digital twin-driven service model and optimal allocation of manufacturing resources in shared manufacturing. *J. Manuf. Syst.* **2021**, *59*, 165–179.
2. Liu, C.Y.; Liu, P. Dynamic allocation of manufacturing tasks and resources in shared manufacturing. *Intell. Autom. Soft. Co.* **2023**, *36*, 3221–3242.
3. Liu, P.; Wei, X.L. Three-party evolutionary game of shared manufacturing under the leadership of core manufacturing company. *Sustainability* **2022**, *14*, 13682.
4. Patterson, S.R.; Kozan, E.; Hyland, P. Energy efficient scheduling of open-pit coal mine trucks. *Eur. J. Oper. Res.* **2017**, *262*, 759–770.
5. Vélez-Gallego, M.C.; Maya, J.; Montoya-Torres, J.R. A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan. *Comput. Oper. Res.* **2016**, *73*, 132–140.
6. Zhao, D.Z.; Wang, Z.S. Scheduling optimization of cloud manufacturing platform processing capability sharing. *Ops. Res. Manag. Sci.* **2016**, *28*, 1–6. (In Chinese).
7. Yu, Y.N.; Xu, Z.; Liu, D.N. Distributed multi-project scheduling problem with multi-skilled staff. *Sys. Eng. Theor. Pract.* **2020**, *40*, 2921–2933. (In Chinese).
8. Tang, L.; Han, H.Y.; Tan, Z.; Jing, K. Centralized collaborative production scheduling with evaluation of a practical order-merging strategy. *Int. J. Prod. Res.* **2023**, *61*, 282–301.
9. Kuroda, M. Integration of product design and manufacturing through real-time due-date estimation and scheduling systems. *J. Adv. Mech. Des. Syst.* **2016**, *10*, JAMDSM42.
10. Dereniowski, D; Kubiak, W. Shared multi-processor scheduling. *Eur. J. Oper. Res.* **2017**, *261*, 503–514.
11. Dereniowski, D; Kubiak, W. Shared processor scheduling of multiprocessor jobs. *Eur. J. Oper. Res.* **2020**, *282*, 464–477.
12. Ji, M.; Ye, X.N.; Qian, F.Y.; Cheng, T.C.E.; Jiang, Y.W. Parallel-machine scheduling in shared manufacturing. *J. Ind. Manag. Optim.* **2022**, *18*, 681–691.
13. Wei, Q.; Wu, Y. Two-machine hybrid flow-shop problems in shared manufacturing. *CMES-Comp. Model. Eng.* **2022**, *131*, 1125–1146.
14. Xu, Y.F.; Zhi, R.T.; Zheng, F.F.; Liu, M. Parallel machine scheduling with due date-to-deadline window, order sharing and time value of money. *Asia Pac. J. Oper. Res.* **2022**, *39*, 2150024.
15. Zheng, F.F.;Jin, K.Y.; Xu, Y.F.; Liu, M. Parallel machine scheduling with order splitting and matching type. *Ops. Res. Manag. Sci.* **2023**, *32*, 1–7. (In Chinese).
16. Fu, Y.P.; Li, H.B.; Huang, M.; Xiao, H. Bi-objective modeling and optimization for stochastic two-stage open shop scheduling problems in the sharing economy. *IEEE T. Eng. Manage.* **2023**, *70*, 3395–3409.
17. Xu, Y.F.; Zhi, R.T.; Zheng, F.F.; Liu, M. Competitive algorithm for scheduling of sharing machines with rental discount. *J. Comb. Optim.* **2022**, *44*, 414–434.

 doi:10.20944/preprints202502.1852.v1

18. Xu, Y.F.; Zhi, R.T.; Zheng, F.F.; Liu, M. Online strategy and competitive analysis of production order scheduling problem with rental cost of shared machines. *Chinese J. Manage. Sci.* **2023**, *31*, 142–150. (In Chinese).
19. Ruiz-Torres, A.J.; López, F.J.; Wojciechowski, P.J.; Ho, J.C. Parallel machine scheduling problems considering regular measures of performance and machine cost. *J. Oper. Res. Soc.* **2010**, *61*, 849–857.
20. Rustogi, K.; Strusevich, V.A. Parallel machine scheduling: impact of adding extra machines. *Oper. Res.* **2013**, *61*, 243–257.
21. Lee, K.; Leung J.Y-T.; Jia, Z.H.; Li, W.H.; Pinedo, M.L.; Lin, B.M.T. Fast approximation algorithms for bi-criteria scheduling with machine assignment costs. *Eur. J. Oper. Res.* **2014**, *238*, 54–64.
22. Li, K.; Zhang, X.; Leung, J.Y-T.; Yang, S.L. Parallel machine scheduling problems in green manufacturing industry. *J. manuf. Syst.* **2016**, *38*, 98–106.
23. Li, K.; Xu, S.L.; Cheng, B.Y.; Yang, S.L. Parallel machine scheduling problem with machine cost to minimize the maximal lateness. *Sys. Eng. Theor. Pract.* **2019**, *39*,165–173. (In Chinese).
24. Jiang, Y.W.; Tang, X.L.; Li, K.; Cheng, T.C.E.; Ji, M. Approximation algorithms for bi-objective parallel-machine scheduling in green manufacturing. *Comput. Ind. Eng.* **2023**, *176*,108949.
25. Anghinolfi, D.; Paolucci, M.; Ronco, R. A bi-objective heuristic approach for green identical parallel machine scheduling. *Eur. J. Oper. Res.* **2021**, *289*, 416–434.
26. Jarboui, B.; Masmoudi, M.; Eddaly, M. Epsilon Oscillation Algorithm for the bi-objective green identical parallel machine scheduling problem. *Comput. Oper. Res.* **2024**, *170*, 106754.
27. Wu, P.; Wang, Y.; Chu, C. Logic-based Benders decomposition for bi-objective parallel machine selection and job scheduling with release dates and resource consumption. *Comput. Oper. Res.* **2024**, *164*, 106528.
28. Lenstra, J.K.; Rinnooy Kan, A.H.G.; Brucker, P. Complexity of machine scheduling problems. *Ann. Discrete Math.* **1977**, *1*, 343–362.
29. McNaughton, R. Scheduling with deadlines and loss functions. *Manage. Sci.* **1959**, *6*, 1–12.
30. Graham, R.L. Bounds on multiprocessing time anomalies. *SIAM J. Appl. Math.* **1969**, *17*, 416–429.