Journal of Global Optimization
https://doi.org/10.1007/s10898-024-01411-2

# Parallel machine scheduling with position-dependent processing times and deteriorating maintenance activities

Chaoming Hu[1,2] · Rui Zheng[1,2] · Shaojun Lu[1,2] · Xinbao Liu[1,2]

Received: 9 October 2023 / Accepted: 21 May 2024

## Abstract
Maintenance and production exert reciprocal influence in practical manufacturing applications. However, decisions regarding production scheduling and maintenance planning are often made separately, leading to frequent conflicts between production and maintenance plans. This paper investigates an integrated production scheduling and maintenance planning problem for a parallel machine system, considering both deteriorating jobs and deteriorating maintenance activities. Additionally, the problem features constraints on the number of available maintenance activities due to maintenance budget limitations. The objective is to determine the optimal scheduling and maintenance plan that minimizes the makespan. To tackle this complex problem, we initially delve into the special case where jobs and maintenance activities are already assigned to machines. In our endeavor to minimize the makespan for each machine, we uncover some crucial structural properties and present a polynomial-time algorithm. Subsequently, we develop a hybrid algorithm that combines Whale Optimization Algorithm and Variable Neighborhood Search (WOA–VNS) to address the assignment challenge encompassing jobs and maintenance activities within the parallel machine environment. A series of rigorous comparative experiments are conducted to assess the effectiveness of the proposed algorithm. The results conclusively demonstrate the superior performance of the WOA–VNS algorithm over the WOA, VNS, ABC, and ACO algorithms in addressing the presented problem.

**Keywords** Production scheduling · Maintenance planning · Deterioration effect · WOA–VNS algorithm

✉ Rui Zheng
richter.zheng@mail.utoronto.ca

✉ Shaojun Lu
lushaojun@hfut.edu.cn

✉ Xinbao Liu
lxb@hfut.edu.cn

1 School of Management, Hefei University of Technology, Hefei, Anhui, People's Republic of China

2 Key Laboratory of Process Optimization and Intelligent Decision-Making of the Ministry of Education, Hefei, Anhui, People's Republic of China

## 1 Introduction

Production scheduling plays a critical role in modern manufacturing, contributing to enhanced efficiency, cost reduction, meeting customer demands, and ensuring the smooth operation of production processes [1]. Extensive research efforts in production scheduling have yielded significant contributions. Nevertheless, existing research still exhibits certain limitations. Firstly, most production scheduling studies assume that job processing times are constant and known in advance. However, in actual manufacturing scenarios, the efficiency of machines can gradually decline as they operate, leading to an increase in job processing times. This phenomenon is widely recognized as the deteriorating effect [2]. Secondly, most production scheduling studies assume that machines are perpetually available for job processing, such as [3–5]. In reality, machines require preventive maintenance, rendering them temporarily unusable for production purposes [6, 7]. Consequently, there has been a notable surge in research interest in recent years focusing on integrated production scheduling and maintenance planning (IPSMP), with a particular emphasis on addressing the challenges posed by the deteriorating effect.

The initial studies on IPSMP with deteriorating effect were conducted in the context of a single machine. Wu and Lee [8] investigated a single machine scheduling problem that incorporates linearly deteriorating jobs and an unavailability period caused by maintenance, with the primary aim of minimizing the makespan. They successfully demonstrated that the problem they tackled could be solved using the 0–1 integer programming technique. Subsequently, Ji et al. [9] extended the work by Wu and Lee [8] and conclusively established that the problem under examination is NP-hard. Furthermore, they introduced an optimal approximation algorithm to address this challenging scheduling problem. Low et al. [10] conducted a research study akin to that of Wu and Lee [8]. But in their problem formulation, the deteriorating effect of the machine began to re-accumulate after maintenance. Building on this novel aspect, Low et al. [10] conducted an extensive analysis and successfully demonstrated that the problem they investigated is likewise NP-hard, and they also put forward several heuristic algorithms based on bin packing concepts.

Different from those studies with predetermined maintenance schedules, Lodree and Geiger [11] considered a flexible rate-modifying activity when investigating the single machine scheduling problem with a deteriorating effect. Their objective is to determine an integrated schedule for both jobs and the RMA that minimizes the makespan. In a related vein, Sun and Geng [12] also explored the single machine scheduling problem with deteriorating effects and a rate-modifying activity. They proved that the problem is polynomial time solvable whether the optimization goal is to minimize the maximum completion time or to minimize the sum of the total completion time. Rustogi and Strusevich [13] further presented a single machine scheduling problem with general position-based processing time and multiple rate-modifying activities. In this intricate problem, decision-makers were tasked with not only determining the job sequence but also strategically deciding the optimal number of maintenance activities to insert into the schedule. Zhang et al. [14] also discussed the single-machine scheduling problem with deteriorating effect and multiple maintenance activities. But in their study, the duration of a maintenance activity is considered as a non-decreasing function of the continuous running time of the machine. Subsequently, some researchers expanded the IPSMP problem with deteriorating effects into more complex parallel machine environments. Lu et al. [15] introduced a parallel machine scheduling problem featuring maintenance deterioration, with the primary aim of minimizing the makespan. In their problem formulation, each machine was mandated to undergo one mandatory maintenance activity. Zhang et al. [16] conducted a similar research study to that of Lu et al. [15], but their objective was to minimize the sum of completion times. Lee et al. [17] introduced a unique dimension to the parallel machine scheduling problem by considering job processing times as position-dependent. Additionally, their problem formulation allowed for at most one maintenance activity for each machine. The overarching goal of minimizing total earliness and tardiness costs. Da et al. [18] investigated a parallel machine scheduling problem with deteriorating jobs and flexible maintenance strategy, in which the actual processing time of a job depends on the effective age and the processing rate of the machine. Their two goals are to minimize the preventive maintenance mean cost rate and total completion time. Woo and Kim [19] delved into a parallel machine scheduling problem with deterioration effect and multiple maintenance activities, in which the actual processing time of job is time-dependent. Their research aimed at jointly determining the assignment of jobs to machines and the sequence of jobs and maintenance activities on each machine, all with the overarching goal of minimizing makespan.

Following the exhaustive examination of the existing literature, several significant observations come to light: (i) The majority of prior studies have generally overlooked the deterioration of maintenance activity. However, in actual production scenarios, delaying maintenance can lead to deteriorating machine conditions, resulting in longer maintenance duration. (ii) Many previous studies assumed the presence of only one maintenance activity per machine during the planning period. However, practical production environments frequently involve multiple maintenance activities. Embracing this aspect is imperative to ensure that scheduling models better reflect the complexities found in real-life production settings. (iii) When considering deteriorating effects, time-based processing time has received considerable attention, whereas position-based processing time has not been subject to equivalent scrutiny.

Therefore, in this paper, we investigate a new IPSMP problem, in which position-dependent processing times and multiple deteriorating maintenance activities are simultaneously considered. In addition, given the constraints of maintenance budget, the number of available maintenance activities in the problem is limited. The objective is to minimize the makespan by concurrently determining the allocation of jobs and maintenance activities to machines and establishing a joint sequence for both jobs and maintenance activities on each machine. Table 1 provides the comparison of previous studies and ours. To the best of our knowledge, the proposed problem has not been investigated in the literature.

The main contributions of this research are as follows:

(1) This paper investigates an IPSMP problem for a parallel machine system, in which both deteriorating jobs and deteriorating maintenance activities are considered.

(2) In the special case where jobs and maintenance activities are already allocated to each machine, we uncover several critical structural properties and present a polynomial-time algorithm aimed at minimizing the makespan for each machine.

(3) To solve the assignment of jobs and maintenance activities in parallel machine environment, we propose a hybrid algorithm that combines Whale Optimization Algorithm and Variable Neighborhood Search (WOA–VNS).

The remainder of this paper is organized as follows. Section 2 describes the studied problem. Section 3 analyzes the sequencing problem of jobs and maintenance activities on a given machine, and presents a polynomial time algorithm. Section 4 develops a hybrid WOA–VNS algorithm. Section 5 discusses the validity of the developed WOA–VNS through rigorous comparison experiments. Section 6 concludes this research.

**Table 1** Main characteristics of previous works and current work

| Authors | Production system | Deteriorating model | Number of maintenance activities | Deteriorating maintenance | Objective |
|---|---|---|---|---|---|
| Wu and Lee [8] | Single machine | $P_i = \alpha_i t$ | One | × | Makespan |
| Ji et al. [9] | Single machine | $P_i = \alpha_i t$ | One | × | Makespan |
| Low et al. [10] | Single machine | $P_i = \alpha_i t$ | One | × | Makespan |
| Lodree and Geiger [11] | Single machine | $p_j = \beta_j s_j$ | One | × | Makespan |
| Sun and Geng [12] | Single machine | $p_j^A = p_j + bs_j$ | One | × | Makespan |
| Rustogi and Strusevich [13] | Single machine | $p_j g(r)$ | Decision variables | × | Makespan |
| Zhang et al. [14] | Single machine | $p_j\,(t) = \alpha_j + \beta_j t$ | Decision variables | ✓ | Makespan; total completion time |
| Lu et al. [15] | Parallel machine | $p_j^A = p_j + b_j t$ | one for each machine | ✓ | Makespan |
| Zhang et al. [16] | Parallel machine | $p_{jt}^i = p_j^i(1 + \alpha_j^i t)$ | one for each machine | × | Expected sum of completion times |
| Lee et al. [17] | Parallel machine | $P_j = p_j r^a$ | at most one for each machine | × | Total earliness and tardiness costs |
| Da et al. [18] | Parallel machine | $p_i^j = \frac{p_{i0}^j + c^j t_i^j}{s^j}$ | Decision variables | × | Multi-objective |
| Woo and Kim [19] | Parallel machine | $p_i = \alpha_i + \beta_i s_i$ | Decision variables | × | Makespan |
| Current study | Parallel machine | $p_j^A = (1+\theta)^{r-1} p_j$ | Decision variables; limited; | ✓ | Makespan |

## 2 Problem description

The notations used in this paper and their definitions are presented as follows.

| Notions | Definitions |
|---|---|
| $n$ | Total number of jobs |
| $m$ | Total number of machines |
| $M_l$ | The machine $l$, $l = 1, \ldots, m$. |
| $k_{\max}$ | Number of available maintenance activities, $m \leq k_{\max} \leq n$ |
| $k_l$ | The number of maintenance activities performed on $M_l$ |
| $J_j$ | The job $j$, $j = 1, \ldots, n$. |
| $p_j$ | Normal processing time of $J_j$,$j = 1, \ldots, n$ |
| $p_j^A$ | Actual processing time of $J_j$ |
| $n_l$ | The number of jobs processed on $M_l$, where$\sum_{l=1}^{m} n_l = n$. |
| $G_{il}$ | The $i^{th}$ group of jobs processed on $M_l$, $i = 1, \ldots, k_l$. |
| $n_{il}$ | The number of jobs in $G_{il}$, where$\sum_{i=1}^{k_l} n_{il} = n_l$ |
| $P_{il}$ | The processing time of $G_{il}$, where$P_{il} = \sum_{J_j \in G_{il}} p_j^A$ |
| $\alpha$ | The normal duration of a maintenance activity |
| $\beta$ | The deteriorating maintenance factor, $\beta \geq 0$ |
| $ma_{il}$ | The actual duration of the $i^{th}$ maintenance activity on $M_l$ |
| $C_{\max}$ | The makespan |

The problem is described as follows. There are $n$ independent jobs to be processed on $m$ identical parallel machines. All of the jobs are available at time zero, and no preemption is allowed. Each machine can only process one job at a time, and the processing of each job cannot be interrupted. As the machine operates, its condition deteriorates, ultimately causing a gradual increase in the actual processing time of jobs. Hence, implementing maintenance becomes imperative to minimize actual production time. The maintenance is assumed to be perfect, ensuring the machine is restored to an "as good as new" condition. It is also presumed that all machines are initially new. To maintain this assumption in subsequent planning phase, a mandatory maintenance activity is executed immediately after the machine completes its final job. Throughout the planning horizon, the number of available maintenance activities is restricted due to the budget constraints. We use $k_{\max}$ ($m \leq k_{\max} \leq n$) to represent the total number of available maintenance activities and $k_l$ to indicate the number of maintenance activities assigned to $M_l$. Noting that, $k_l$ includes the final mandatory maintenance activity on $M_l$. Figure 1 is an illustrative diagram of the described problem. We designate the set of jobs between any two adjacent maintenance activities or prior to the first maintenance activity as a group. Hence, a schedule on machine $M_l$ can be represented by $\pi(M_l) = (G_{1l}, MA_{1l}, G_{2l}, MA_{1l}, \ldots, G_{k_l,l}, MA_{k_l,l})$, where $G_{il}$ represents the $i^{th}$ group on $M_l$.

According to Öztürkoğlu [20], the actual processing time of $J_j$ can be formulated as:

$$p_j^A = (1+\theta)^{r-1} p_j \tag{1}$$

where $p_j$ is the normal processing time of $J_j$, $r$ represents the position of $J_j$ within its corresponding group, and $\theta$ ( $\geq 0$) is the deteriorating rate of the machine.

Additionally, the duration of a maintenance activity is also deteriorating. Following Yang [21], the actual duration of the $i^{th}$ maintenance activity on $M_l$ can be formulated as:

$$ma_{il} = \alpha + \beta P_{il} \tag{2}$$

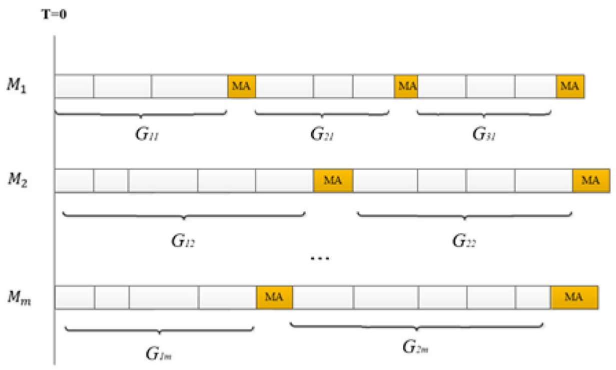


**Fig. 1** Schematic diagram of the proposed problem

where $\alpha$ is the basic duration of a maintenance activity, $\beta$ is the maintenance deteriorating factor, and $P_{il}$ represent represent the actual processing time of $G_{il}$.

The objective is to is to determine a combined schedule for both jobs and maintenance activities that minimizes the makespan. Using the three-filed notation scheme [22], the studied problem can be denoted as $Pm\ |p_j^A = (1+\theta)^{r-1} p_j,\ \ m \le MA \le k_{\max}|C_{\max}$, where $MA$ represents the number of maintenance activities actually performed.

## 3 Special case study

To address our problem, two crucial decisions need to be made: first, allocating jobs and maintenance activities to machines, and second, organizing the sequence for jobs and maintenance activities on each machine. Given the complexity of this problem, our initial focus will be on the special case where jobs and maintenance activities are already assigned to each machine.

With the known count of maintenance activities on each machine, the problem on a given single machine can be denoted as $1\ |p_j^A = (1+\theta)^{r-1} p_j,\ \ MA = k_l|C_{\max}$. To address this problem, we establish essential structural properties and then introduce a heuristic method in this section.

**Lemma 1** *For the problem* $1\ |p_j^A = (1+\theta)^{r-1} p_j,\ \ MA = k_l|C_{\max}$, *exchanging the positions of any two groups does not alter the makespan.*

***Proof*** For any given schedule $\pi$, the makespan on machine $M_l$ can be calculated as:

$$\begin{aligned}C_{\max}(M_l,\pi) &= \sum_{i=1}^{k_l} P_{il} + \sum_{i=1}^{k_l} ma_{il} = \sum_{i=1}^{k_l} P_{il} + \sum_{i=1}^{k_l}(\alpha + \beta P_{il}) \\ &= k_l\alpha + (1+\beta)\sum_{i=1}^{k_l} P_{il} \\ &= k_l\alpha + (1+\beta)\sum_{i=1}^{k_l}\left(\sum_{J_j\in G_{il}} p_j^A\right)\end{aligned} \tag{3}$$

According to the above equation, it's evident that the makespan solely relies on the grouping result and the job sequence within each group. Changing the order of groups doesn't impact the total job processing time or the overall duration of maintenance activities. Thus, Lemma 1 is proved. □

**Lemma 2** *For the problem* $1\ |p_j^A = (1+\theta)^{r-1}\ p_j,\quad MA = k_l|C_{\max}$, *if there exists an optimal schedule, the number of jobs within each group cannot exceed* $\lceil \frac{n_l}{k_l} \rceil$.

***Proof*** Suppose there exists an optimal schedule $\pi$ in which there is a group $G_{xl}$ containing $\varphi$ ($\varphi \geq \lceil \frac{n_l}{k_l} \rceil + 1$) jobs. Then there must be a group $G_{yl}$ containing fewer jobs than $\lceil \frac{n_l}{k_l} \rceil$. Denote the number of jobs in the group $G_{yl}$ as $\delta$ ($\delta < \lceil \frac{n_l}{k_l} \rceil$). Let $J_a$ represent the last job in the group $G_{xl}$, with a normal processing time of $p_a$. Move $J_a$ to the last position of the group $G_{yl}$ to obtain a new schedule $\pi^*$. According to Lemma 1, the difference in makespan between the schedule $\pi$ and $\pi^*$is:

$$\begin{aligned}
&C_{\max}\left(M_l, \pi\right) - C_{\max}\left(M_l, \pi^*\right)\\
&\quad = P_{xl}(\pi) - P_{xl}\left(\pi^*\right) + P_{yl}(\pi) - P_{yl}\left(\pi^*\right)\\
&\qquad + ma_{xl}(\pi) - ma_{xl}\left(\pi^*\right) + ma_{yl}(\pi) - ma_{yl}\left(\pi^*\right)\\
&\quad = (1+\theta)^{\varphi-1} p_a - (1+\theta)^{\delta} p_a + \beta(1+\theta)^{\varphi-1} p_a - \beta(1+\theta)^{\delta} p_a\\
&\quad = (1+\beta)\left[(1+\theta)^{\varphi-1} - (1+\theta)^{\delta}\right] p_a
\end{aligned} \tag{4}$$

As $\varphi - 1 \geq \delta$, it follows that $C_{\max}(\pi) - C_{\max}(\pi^*) \geq 0$. This indicates that relocating job $J_a$leads to a shorter makespan. Hence, $\pi$ cannot be the optimal schedule. With this, the proof of Lemma 2 is completed. □

Lemma 2 sets an upper limit for the number of jobs within a group but doesn't ascertain the exact count of jobs in each group. For example, if there are 10 jobs and 3 maintenance activities to be scheduled, the possible grouping results could be $\{n_{1l} = 4, n_{2l} = 3, n_{3l} = 3\}$ and $\{n_{1l} = 4, n_{2l} = 4, n_{3l} = 2\}$, as indicated by Lemma 2. To address this uncertainty regarding the number of jobs in each group, Lemma 3 is introduced.

**Lemma 3** *For the problem* $1\ |p_j^A = (1+\theta)^{r-1}\ p_j,\quad MA = k_l|C_{\max}$, *if there exists an optimal schedule, the number of jobs within a group cannot be less than* $\lfloor \frac{n_l}{k_l} \rfloor$.

***Proof*** Here are two cases.

Case 1: $n_l\ mod\ k_l\ = 0$

In this case, the number of jobs in each group is $\lfloor \frac{n_l}{k_l} \rfloor$, where$\lfloor \frac{n_l}{k_l} \rfloor = \lceil \frac{n_l}{k_l} \rceil$.

Case 2: $n_l\ mod\ k_l\ \neq 0$

Suppose there exists an optimal schedule $\pi$ in which there is a group $G_{xl}$ containing $\varphi$ ($\varphi < \lfloor \frac{n_l}{k_l} \rfloor$) jobs. Then there must be more than one group containing $\lceil \frac{n_l}{k_l} \rceil$ jobs. Randomly choose a group containing $\lceil \frac{n_l}{k_l} \rceil$ jobs and denote this group as $G_{yl}$. Let $J_a$ represent the last job in the group $G_{yl}$, with a normal processing time of $p_a$. Move $J_a$ to the last position of the group $G_{xl}$ to obtain a new schedule $\pi^*$. Then the difference in makespan between the schedule $\pi$ and $\pi^*$ is:

$$\begin{aligned}
&C_{\max}(M_l,\pi) - C_{\max}\left(M_l,\pi^*\right) \\
&\quad = P_{xl}(\pi) - P_{xl}\left(\pi^*\right) + P_{yl}(\pi) - P_{yl}\left(\pi^*\right) \\
&\qquad + ma_{xl}(\pi) - ma_{xl}\left(\pi^*\right) + ma_{yl}(\pi) - ma_{yl}\left(\pi^*\right) \\
&\quad = (1+\theta)^{\left\lceil \frac{n_l}{k_l} \right\rceil - 1} p_a - (1+\theta)^{\varphi} p_a + \beta(1+\theta)^{\left\lceil \frac{n_l}{k_l} \right\rceil - 1} p_a - \beta(1+\theta)^{\varphi} p_a \\
&\quad = (1+\beta)\left[(1+\theta)^{\left\lceil \frac{n_l}{k_l} \right\rceil - 1} - (1+\theta)^{\varphi}\right] p_a
\end{aligned} \tag{5}$$

As $\varphi < \left\lfloor \frac{n_l}{k_l} \right\rfloor = \left\lceil \frac{n_l}{k_l} \right\rceil - 1$, it follows that $C_{\max}(\pi) - C_{\max}(\pi^*) > 0$. This indicates that relocating job $J_a$leads to a shorter makespan. Therefore, $\pi$ cannot be the optimal schedule.

Combining case 1 and case 2, we complete the proof of lemma 3. □

**Lemma 4** *Given two sequences of positive numbers, $X$ and $Y$, the sum of products of their corresponding elements $\sum X_i Y_i$ is minimized when the two sequences are monotonic in the opposite sense.*

***Proof*** See in Hardy et al. [23]. □

**Lemma 5** *For the problem $1\,|p_j^A = (1+\theta)^{r-1}\, p_j,\quad MA = k_l|C_{\max}$, if there exists an optimal schedule, the jobs within a group should be processed in non-increasing order of their normal processing times.*

***Proof*** Suppose that the job sequence in the group $G_{il}$ is $s = \{J_1, J_2, \ldots, J_{n_{il}}\}$, and the normal processing time of each job is $p_j = \{p_1, p_2, \ldots, p_{n_{il}}\}$, then the processing time of the group $G_{il}$ and the duration of the closely-following maintenance activity is:

$$P_{il} + ma_{il} = P_{il} + \alpha + \beta P_{il} = \alpha + (1+\beta)\sum_{\sigma=1}^{n_{il}} p_\sigma (1+\theta)^{\sigma-1} \tag{6}$$

Clearly, $(1+\theta)^{\sigma-1}$ increases with $\sigma$, where $\sigma = 1, 2, \ldots, n_{il}$. Therefore, $P_{il} + ma_{il}$ is minimized when $p_\sigma = \{p_1, p_2, \ldots, p_{n_{il}}\}$ is sequenced in non-increasing order, as per Lemma 4. Thus, the proof of Lemma 5 is completed.

Utilizing the established lemmas, a polynomial time algorithm is developed to solve the single machine problem $1\,|p_j^A = (1+\theta)^{r-1}\, p_j,\quad MA = k_l|C_{\max}$. The pseudo-code for the proposed polynomial-time algorithm are outlined in Algorithm I. Noting that, $k_l$ includes the last mandatory maintenance activity on $M_l$. □

**Algorithm I:** Polynomial time algorithm for single machine problem

```
1: Input the number of jobs $n_l$ and the number of maintenance activities $k_l$
2: Sequence all jobs in descending order of their normal processing times and renumber them, such as
   $\pi = \{J_1, \cdots, J_x, \cdots, J_{n_l}\}$ which satisfying $p_1 \geq \cdots \geq p_x \geq \cdots \geq p_{n_l}$
3: For $x \to 1$ to $n_l$
4:     Let $a = x \bmod k_l$, and $b = \left\lceil \frac{x}{k_l} \right\rceil$
5:         If $a \neq 0$,then
6:             Put the job $J_x$ into the $b^{th}$ position in $a^{th}$ group
7:         Else
8:             Put the job $J_x$ into the $b^{th}$ position in $k_l^{th}$ group
```

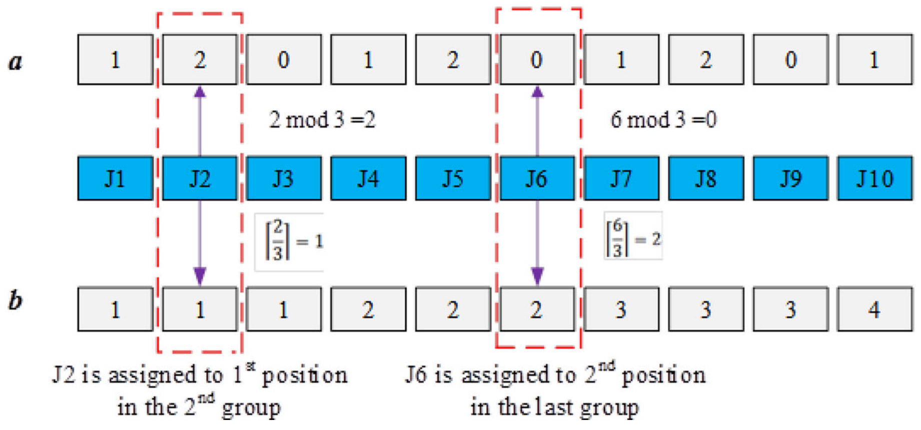

a
1 2 0 1 2 0 1 2 0 1
2 mod 3 =2
6 mod 3 =0
J1 J2 J3 J4 J5 J6 J7 J8 J9 J10
⌈2/3⌉ = 1
⌈6/3⌉ = 2
b
1 1 1 2 2 2 3 3 3 4
J2 is assigned to 1st position in the 2nd group
J6 is assigned to 2nd position in the last group

**Fig. 2** An example to illustrate Algorithm I

To provide a more intuitive understanding of Algorithm I, we present an example with 10 jobs and 3 maintenance activities (including the last mandatory maintenance activity). Let's consider a job set $J = \{J_1, J_2, \ldots, J_{10}\}$ to be processed on the machine $M_l$, with the job set following the condition $p_1 \geq p_2 \geq \cdots \geq p_{10}$. Based on Algorithm I, the scheduling result is depicted in Fig. 2.

**Theorem 1** *Algorithm I is optimal for solving the single machine problem* $1\,|p_j^A = (1+\theta)^{r-1}\,p_j,\ \ MA = k_l|C_{\max}$.

***Proof*** Denote the schedule generated by the Algorithm I as $\pi^{opt}$. Evidently, schedule $\pi^{opt}$ satisfies Lemmas 2 and 3. Now, let's shift our focus to the makespan of schedule $\pi^{opt}$. The makespan of schedule $\pi^{opt}$ can be calculated as follows:

$$\begin{aligned} C_{max}\left(M_l, \pi^{opt}\right) &= k_l\alpha + (1+\beta)\sum_{i=1}^{k_l}\left(\sum_{J_j \in G_{il}} p_j^A\right) \\ &= k_l\alpha + (1+\beta)\Big[p_1 + \cdots + p_{k_l} + p_{k_l+1}(1+\theta) + \cdots + p_{2k_l}(1+\theta) \\ &\quad + p_{2k_l+1}(1+\theta)^2 + \cdots + p_{3k_l}(1+\theta)^2 + p_{3k_l+1}(1+\theta)^3 \\ &\quad + \cdots + p_{4k_l}(1+\theta)^3 + \cdots + p_{n_l}(1+\theta)^{\left\lceil \frac{n_l}{k_l}\right\rceil - 1}\Big] \\ &= k_l\alpha + (1+\beta)\sum_{x=1}^{n_l} p_x(1+\theta)^{\left\lceil \frac{x}{k_l}\right\rceil - 1} \end{aligned} \tag{7}$$

Since the two sequences,$\{p_x\}$ and $\left\{(1+\theta)^{\left\lceil \frac{x}{k_l}\right\rceil - 1}\right\}$, demonstrate opposite monotonicity, the makespan is minimized, as per Lemma 4. Therefore, the proposed Algorithm I is optimal for solving the single machine problem. □

## 4 WOA–VNS algorithm

In Sect. 3, we delve into the special case where jobs and maintenance activities are pre-assigned to each machine. To solve the joint sequencing of jobs and maintenance activities on each machine, we present a polynomial time algorithm which is proxved to be optimal. In this section, our focus will be on the allocation of parts and maintenance activities within the parallel machine environment. Without accounting for the deteriorating effects and maintenance activities, the investigated parallel problem can be simplified to the problem $Pm||C_{\max}$, which has been proved to be NP-hard by Sethi [24]. Therefore, the proposed problem is at least NP-hard, and it is difficult to address it with an accurate algorithm. Consequently, to solve the proposed problem within a reasonable time, a hybrid algorithm that combines Whale Optimization Algorithm and Variable Neighborhood Search (WOA–VNS) is developed. More specific information about the WOA–VNS algorithm is detailed below.

### 4.1 Encoding method

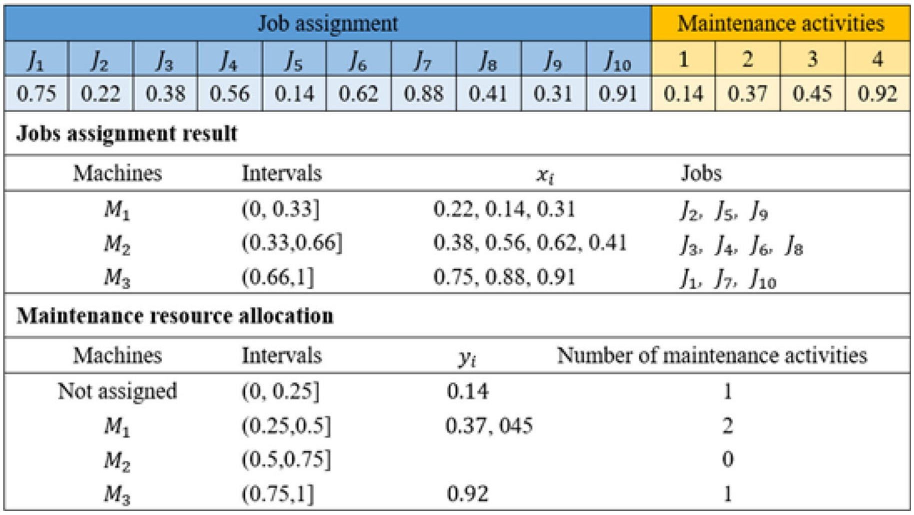
Due to the mandatory maintenance required after each machine completes its final job, the actual number of assignable maintenance activities becomes $k_{max} - m$. It's worth noting that, when the time spent on maintenance exceeds the reduction in processing time for jobs, the maintenance activity no longer remains beneficial. This implies that an optimal allocation plan doesn't necessarily require utilizing all available maintenance activities to their fullest extent. Consequently, the following one-dimensional vector is proposed to represent the solution to the addressed problem.

$$\vec{X} = \left(x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+k_{max}-m}\right), \; x_i \in (0, 1]$$

In the vector $\vec{X}$, the first $n$ elements represent the assignment results for each job, and the last $k_{max} - m$ elements represent the allocation results for each maintenance activity. For the first $n$ elements, if $\mathrm{x}_i$ falls within the interval $(\frac{k-1}{M}, \frac{k}{M}]$, where $k = 1, \ldots, M$, then job $i$ is assigned to the machine $M_k$. For the last $kmax - m$ elements, if $\mathrm{x}_i$ falls within the interval $(\frac{k}{M+1}, \frac{k+1}{M+1}]$, where $k = 1, \ldots, M$, then the corresponding maintenance activity is assigned to the machine $M_k$. It is worth noting that if $x_i$ falls within the interval $(0, \frac{1}{M+1}]$, the corresponding maintenance activity will not be assigned to any machine. This encoding permits maintenance activities to remain unassigned, ensuring the potential for discovering a balanced maintenance solution when maintenance resources are excessive. Figure 3 provides a more visual representation of this encoding method with an example involving 10 jobs, 3 machines, and 4 available maintenance activities (excluding the final mandatory maintenance activity on each machine).

### 4.2 The standard WOA

The Whale Optimization Algorithm (WOA) was introduced by Mirjalili and Lewis [25] in 2016, drawing inspiration from the hunting behavior of humpback whales. Since the WOA was developed, it has been widely used in various fields, such as feature selection, clustering, image classification, and scheduling optimization, due to its straightforward iterative approach and rapid convergence. The primary steps of the WOA are outlined as follows.



| Job assignment | | | | | | | | | | Maintenance activities | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ | $J_9$ | $J_{10}$ | 1 | 2 | 3 | 4 |
| 0.75 | 0.22 | 0.38 | 0.56 | 0.14 | 0.62 | 0.88 | 0.41 | 0.31 | 0.91 | 0.14 | 0.37 | 0.45 | 0.92 |

**Jobs assignment result**

| Machines | Intervals | $x_i$ | Jobs |
|---|---|---|---|
| $M_1$ | (0, 0.33] | 0.22, 0.14, 0.31 | $J_2$, $J_5$, $J_9$ |
| $M_2$ | (0.33,0.66] | 0.38, 0.56, 0.62, 0.41 | $J_3$, $J_4$, $J_6$, $J_8$ |
| $M_3$ | (0.66,1] | 0.75, 0.88, 0.91 | $J_1$, $J_7$, $J_{10}$ |

**Maintenance resource allocation**

| Machines | Intervals | $y_i$ | Number of maintenance activities |
|---|---|---|---|
| Not assigned | (0, 0.25] | 0.14 | 1 |
| $M_1$ | (0.25,0.5] | 0.37, 045 | 2 |
| $M_2$ | (0.5,0.75] | | 0 |
| $M_3$ | (0.75,1] | 0.92 | 1 |

**Fig. 3** An example to illustrate the encoding method

**(1) Encircling prey**
Since the exact location of the target prey is unknown in advance, the WOA algorithm operates under the assumption that the current optimal individual is the target prey. Subsequently, candidate whales update their positions in the direction of this presumed prey. This behavior can be described as follows:

$$\vec{D} = \left| C * \vec{X}^* (t) - \vec{X} (t) \right| \tag{8}$$

$$\vec{X} (t+1) = \vec{X}^* (t) - A * \vec{D} \tag{9}$$

where $\vec{X}^*$ is the current optimal individual, $t$ is the current iteration, $A$ and $C$ and are coefficient values, and $||$ signifies taking the absolute value of each element within the vector.

The calculation formulas for and are as follows:

$$A = 2a * r - a \tag{10}$$

$$C = 2 * r \tag{11}$$

$$a = 2 * \left( 1 - \frac{t}{T_{max}} \right) \tag{12}$$

where $T_{max}$ is the maximum number of iterations, and $r$ is a random value in [0,1]

**(2) Bubble-net attacking (Exploitation)**
While swimming around the prey, whales employ two mechanisms to update their positions: the shrinkage mechanism and the spiral mechanism. The shrinkage mechanism is accomplished by establishing the coefficient value within the range of [-1,1] while linearly decreasing the value of throughout the iterations. The spiral mechanism can be represented by the following formula.

$$\vec{D}' = \left| \vec{X}^* (t) - \vec{X} (t) \right| \tag{13}$$

$$\vec{X} (t+1) = \vec{X}^* + \vec{D}' * e^{bl} * \cos (2\pi l) \tag{14}$$

Because two mechanisms are used simultaneously, the entire model can be described as follows.

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - A * \vec{D}, & if \ \ pi < 0.5 \\ \vec{D}' * e^{bl} * \cos(2\pi l), & if \ \ pi \geq 0.5 \end{cases} \tag{15}$$

where $p_i$ is a random value in [0,1].

**(3) Search for prey**

Over the course of iteration, when $|A| > 1$, the whales update their position according to the randomly selected whale $\vec{X_{rand}}$. This mechanism motivates the whales to venture away from the current best solution, actively seeking potentially superior prey. This behavior can be expressed as follows.

$$\vec{D} = \left| C * \vec{X_{rand}} - \vec{X}(t) \right| \tag{16}$$

$$\vec{X}(t+1) = \vec{X_{rand}} - A * \vec{D} \tag{17}$$

## 4.3 Hybrid WOA–VNS

Despite the widespread adoption of the WOA algorithm across diverse applications, certain researchers have highlighted its potential challenges in effectively exploring extensive search spaces. This becomes especially pertinent when tackling intricate, large-scale problems [26–28]. To address these concerns, a VNS algorithm [29] is specifically developed and integrated into the standard WOA framework, aimed at augmenting its search capabilities.

As known, the key to success for VNS is the neighborhood structure it applies. Given the distinctive encoding method in this paper, classic local search operators may not be efficient. Consequently, a novel local search operator is designed, tailored to the specific characteristics of the encoding method utilized herein. The devised local search operator initiates by generating a random integer $\tau$ within the range of $(1, n+k_{max}-m)$. It subsequently determines whether to alter the job assignment or the maintenance activity assignment based on the generated integer. Algorithm II delineates the procedure steps of the designed local search operator.

**Algorithm II:** Local Search Operator for VNS

```
1:  Input the solution $\vec{X}$
2:  Let $\vec{X_{new}} \leftarrow \vec{X}$
3:  Randomly generate an integer $\tau$ in $(1, n + k_{max} - m)$
4:  If $\tau \leq n$ , then
5:      Randomly generate an integer $\sigma$ in $(1, \tau)$
6:      For $i$ in range $(\sigma, \tau)$
7:          Let $\vec{X_{new}}[i] = \vec{X}[\tau - i + \sigma]$
8:  Else
9:      Let $\vec{X_{new}}\ [\tau] = 1 - \vec{X}\ [\tau]$
10: End if
11: Output $\vec{X_{new}}$
```

Employing the tailored local search operator, a VNS algorithm is further developed and integrated into the WOA algorithm. Within each iteration, the VNS takes the optimal solution found by WOA as its initial solution, progressively refining it through iterations using the local search operator. Algorithm III illustrates the comprehensive process of the WOA–VNS algorithm, where lines 21 to 29 delineate the implementation steps of the designed VNS algorithm.

**Algorithm III:** Hybrid WOA-VNS algorithm

```
1:  Initialize the whales' population $\vec{X}_i,\ i = 1, \ldots, pop$
2:  Calculate the fitness of each search agent
3:  Set the best search agent as $\vec{X}^*$
4:  $t = 0$
5:  While $t < T_{\max}$
6:    For $i \leftarrow 1\ to\ pop$
7:      Generate the probability $p_i$
8:      Update $a,\ A,\ C$ based on (10),(11),(12)
9:      If $p_i < 0.5$
10:         If $|A| < 1$
11:           Update $\vec{X}_i$ by the formula (9)
12:         Else
13:           Randomly select a search agent $X_{\vec{rand}}$
14:           Update $\vec{X}_i$ by the formula (17)
15:         End if
16:     Else
17:           Update $\vec{X}_i$ by the formula (14)
18:     End if
19:   End for
20:   Calculate the fitness of each whale and Update $\vec{X}^*$
21:   Set $E = 1$
22:   While $E \leq 4$
23:     Execute Algorithm II for$\vec{X}^*$ and obtain $\vec{X}^*_{new}$
24:     If $fitness(\vec{X}^*_{new}) < fitness(\vec{X}^*)$
25:       $\vec{X}^* \leftarrow \vec{X}^*_{new}, E = 1$
26:     Else
27:       $E = E + 1$
28:     End if
29:   End while
30:   Let $t = t + 1$
31: End while
32: Return $\vec{X}^*$
```

## 5 Computational experiment

To evaluate the efficacy of our algorithm, we conduct a comparative analysis between the proposed WOA–VNS algorithm and several benchmarks, including the Gurobi solver, along with four other intelligent algorithms: the standard WOA algorithm [25], the VNS algorithm [30], the ant colony optimization (ACO) algorithm [31], and the artificial bee colony (ABC) algorithm [32]. These selected intelligent algorithms, except for the WOA algorithm, have been enhanced and demonstrated strong performance in solving similar parallel machine problems. For fair comparisons, the proposed Algorithm I is utilized in all of these algorithms to solve the single machine problem. Moreover, identical initial solutions are used for all algorithms on each problem instance. All algorithms are implemented in Python and executed on a computer with an Intel(R) Core (TM) i7-7770 CPU @3.60 GHz, 8GB RAM,

**Table 2** Experimental factors and their levels

| Parameters | Definitions | Levels |
|---|---|---|
| $p_j$ | The processing time of the job $J_j$ | [10, 20] |
| $\theta$ | The production deterioration factor | 0.1 |
| $\alpha$ | The normal duration of a maintenance activity | 10 |
| $\beta$ | The deteriorating maintenance factor, $\beta \geq 0$ | 0.2 |

and a Windows 10 operating system. Further details of the computational experiments are elaborated below.

### 5.1 Test instances and parameter settings

To ensure the robustness of the results, various scales of test instances are generated. Each instance is denoted as, where represents the number of jobs, represents the number of machines, and represents the actual number of assignable maintenance activities. Every algorithm utilizes a population size set at 10, with the termination condition set to reach 100 iterations. Each algorithm undergoes 10 executions for every test instance. For fair comparisons,the suggested parameters for the standard WOA algorithm [25], the VNS algorithm [30], ACO [31], and ABC [32] are employed. Table 2 details the parameters and their levels for the addressed problem.

### 5.2 Comparison with Gurobi solver

This subsection employs both the Gurobi solver and WOA–VNS on various small-scale problem instances, offering a comparative analysis of their operational outcomes. Before employing the Gurobi solver, establishing the mathematical model for the addressed problem is crucial. Given that single-machine problems can be solved by Algorithm I, our primary objective revolves around determining the allocation of jobs and maintenance activities. Therefore, the following decision variables are defined:

$$x_{[i][j]} = \begin{cases} 1, & job\ i\ is\ asigned\ to\ M_j \\ 0, & otherwise \end{cases}$$

$$y_{[k][j]} = \begin{cases} 1, & the\ k^{th}\ maintenance\ activity\ is\ asigned\ to\ M_j \\ 0, & otherwise \end{cases}$$

Further, the mathematical model of the simplified problem can be represented as follows:

$$Min \quad \max_{l=1 \longrightarrow m} \left\{ C_{max}(M_l, \pi^{opt}) \right\}$$

$$s.t. \begin{cases} \sum_{j=1}^{m} x_{[i][j]} = 1 \\ \sum_{j=1}^{m} y_{[k][j]} \leq 1 \\ x_{[i][j]} = \{0, 1\} \\ y_{[k][j]} = \{0, 1\} \end{cases}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, m$, and $k = 1, \ldots, k_{max} - m$.

In the aforementioned model, the first constraint signifies that a job can only be allocated to a single machine, and all jobs must be assigned. The subsequent constraint specifies that

**Table 3** Results generated by WOA–VNS and Gurobi solver

| Instance | WOA–VNS | | Gurobi | | GAP (%) |
|---|---|---|---|---|---|
| | Solution | Time | Solution | Time | |
| (10,2,3) | 102.401 | 1.09 | 102.401 | 1.33 | 0.00 |
| (10,3,4) | 81.220 | 1.16 | 81.22 | 1.74 | 0.00 |
| (15,2,3) | 168.072 | 1.37 | 166.233 | 11.39 | 1.11 |
| (15,3,4) | 111.725 | 1.43 | 110.988 | 16.45 | 0.66 |
| (20,2,3) | 229.973 | 1.75 | 223.582 | 41.51 | 2.86 |
| (20,3,4) | 155.485 | 1.72 | 152.236 | 102.15 | 2.13 |
| (20,4,5) | 119.569 | 1.94 | 117.601 | 103.60 | 1.67 |
| (50,2,4) | 674.598 | 3.68 | 656.602 | 1800 | 2.74 |
| (50,3,6) | 384.800 | 4.19 | 373.244 | 1800 | 3.10 |
| (50,4,6) | 309.917 | 4.48 | NA | 1800 | – |

NA indicates that no feasible solution is obtained within the specified time

a maintenance activity can be assigned to only one machine or remain unassigned to any machine. For any given feasible assignment plan, the optimal sequence of jobs and maintenance activities on each machine can be determined by Algorithm I, and the corresponding makespan on each machine ($C_{max}(M_l, \pi^{opt})$) can be calculated according to formula 7.

To solve the aforementioned model using the Gurobi solver, a maximum runtime of 1800s is set. Table 3 displays the results of the Gurobi solver and the proposed WOA–VNS for solving problems of different scales. The final column indicates the difference in outcomes between the WOA–VNS algorithm and the Gurobi solver, also known as the GAP. Examining Table 3, it's evident that the WOA–VNS algorithm significantly outperforms the Gurobi solver in terms of solution time. Furthermore, as the scale reaches (50, 4, 6), the Gurobi solver fails to yield a feasible solution within the designated time limit. Additionally, for all test instances, the difference (GAP) between the outcomes of the WOA–VNS algorithm and the Gurobi solver remains under 4%. Therefore, it can be inferred that the proposed WOA–VNS is effective in solving the given problem.

## 5.3 Comparison with other algorithms

In this subsection, the proposed WOA–VNS algorithm is further compared with other intelligent algorithms, namely WOA [25], VNS [30], ACO [31], and ABC [32]. To assess the advantages of the WOA–VNS algorithm over others, Friedman tests are conducted. These tests formulate a null hypothesis assuming that the distributions of solutions produced by WOA, VNS, ABC, ACO, and WOA–VNS are identical. By analyzing the solutions, the tests calculate the asymptotic significance. If the asymptotic significance is less than 0.05, the null hypothesis is rejected; otherwise, it is accepted. Table 4 summarizes the results of the Friedman test, with the asymptotic significant values above 0.05 indicated in bold. The table indicates that among the 21 test instances, there are significant differences in the solutions obtained by these algorithms in 19 test instances, excluding instances (150, 3, 8) and (200, 6, 12).

In addition to hypothesis tests, the Friedman tests also yield the mean ranks of WOA, VNS, ABC, ACO, and WOA–VNS. In the pursuit of minimizing the objective function, a lower mean rank indicates that the results obtained by a specific algorithm are closer to the

**Table 4** Summary of Hypothesis Test

| Instance | Total N | Test statistic | Degree of freedom | Asymp. sig. | Decision |
|---|---|---|---|---|---|
| (20,2,3) | 10 | 20.92 | 4 | 0.000328 | Reject the null hypothesis |
| (20,3,4) | 10 | 18.64 | 4 | 0.000925 | Reject the null hypothesis |
| (20,4,5) | 10 | 9.71 | 4 | 0.045634 | Reject the null hypothesis |
| (50,2,4) | 10 | 13.97 | 4 | 0.007396 | Reject the null hypothesis |
| (50,3,6) | 10 | 13.43 | 4 | 0.009337 | Reject the null hypothesis |
| (50,4,6) | 10 | 11.17 | 4 | 0.024700 | Reject the null hypothesis |
| (100,3,6) | 10 | 11.12 | 4 | 0.025295 | Reject the null hypothesis |
| (100,4,6) | 10 | 12.56 | 4 | 0.013638 | Reject the null hypothesis |
| (100,5,8) | 10 | 17.16 | 4 | 0.001801 | Reject the null hypothesis |
| (150,3,8) | 10 | 5.76 | 4 | **0.217803** | Retain the null hypothesis |
| (150,4,8) | 10 | 17.84 | 4 | 0.001326 | Reject the null hypothesis |
| (150,5,10) | 10 | 20.88 | 4 | 0.000334 | Reject the null hypothesis |
| (200,4,10) | 10 | 13.36 | 4 | 0.009644 | Reject the null hypothesis |
| (200,5,10) | 10 | 11.84 | 4 | 0.018582 | Reject the null hypothesis |
| (200,6,12) | 10 | 7.36 | 4 | **0.118044** | Retain the null hypothesis |
| (250,4,12) | 10 | 14.80 | 4 | 0.005135 | Reject the null hypothesis |
| (250,5,12) | 10 | 11.20 | 4 | 0.024406 | Reject the null hypothesis |
| (250,6,15) | 10 | 23.12 | 4 | 0.000120 | Reject the null hypothesis |
| (300,4,15) | 10 | 13.68 | 4 | 0.008390 | Reject the null hypothesis |
| (300,5,15) | 10 | 14.88 | 4 | 0.004957 | Reject the null hypothesis |
| (300,6,20) | 10 | 17.36 | 4 | 0.001645 | Reject the null hypothesis |

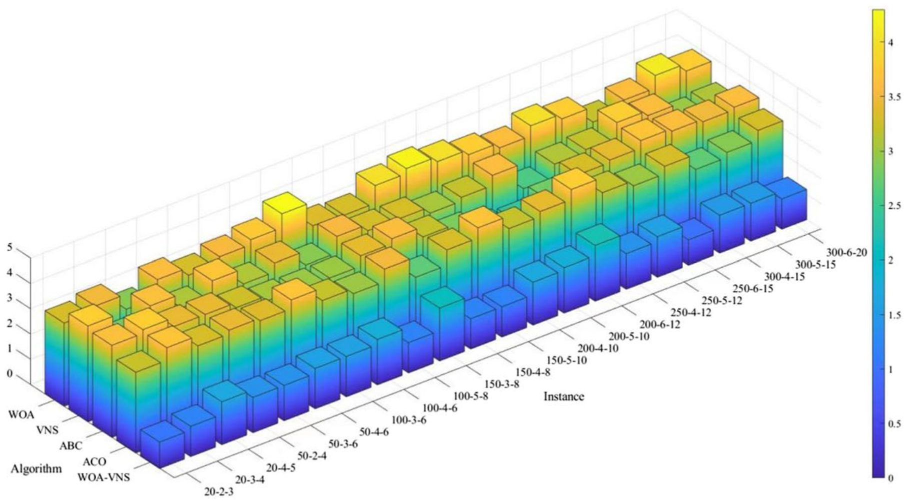


**Fig. 4** Mean ranks of each algorithm for different problem instances

optimal solution. Figure 4 vividly portrays the mean ranks of each algorithm across various problem instances. Remarkably, the WOA–VNS algorithm consistently attains the lowest mean rank for each test instance. This consistent pattern underscores the superiority of the proposed WOA–VNS algorithm over others, consistently delivering outstanding results.

To emphasize the superiority of WOA–VNS over other algorithms, the relative percentage deviations (*RPD*) analysis is also performed. The *RPD* is expressed as follows:

$$RPD = \frac{Z_A - Z_B}{Z_B}$$

where $Z_A$ is the result obtained by a given algorithm for an instance and $Z_B$ is the best result obtained by all selected algorithms for the same instance. Table 5 displays the *RPD* results of the maximum (Max), minimum (Min), average (AVE), and standard deviation (SD) of each algorithm for different problem instances. The better results are bold.

As shown in Table 5, in terms of minimum (Min) results, the WOA–VNS algorithm outperforms other selected algorithms in 17 out of 21 test problem instances. Additionally, when assessing the maximum (Max) and average (AVE) results, the WOA–VNS algorithm demonstrates superiority over the comparison algorithms across all test instances. Regarding standard deviation (SD), the WOA–VNS algorithm consistently delivers the best results, except for the specific problem instance (250, 6, 15). Therefore, it can be confidently concluded that the proposed WOA–VNS algorithm is both more efficient and stable compared to other selected algorithms in effectively addressing the presented problem.

Furthermore, Fig. 5 displays the convergence curves of these algorithms. Each point on the curve represents the average fitness of all individuals in each generation. Particularly noteworthy is that even when the iterations reach 80, the WOA–VNS algorithm exhibits a continuous decline, indicating ongoing improvement. This observation strongly suggests that the hybrid WOA–VNS algorithm effectively mitigates the original WOA algorithm's tendency to converge towards local optima, significantly addressing this limitation.

**Table 5** RPD results

| Problem instance | WOA | | | | VNS | | | | ABC | | | | ACO | | | | WOA–VNS | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MIN | MAX | AVE | SD | MIN | MAX | AVE | SD | MIN | MAX | AVE | SD | MIN | MAX | AVE | SD | MIN | MAX | AVE | SD |
| (20,2,3) | **0.00** | 0.02 | 0.01 | 12.7 | 0.00 | 0.02 | 0.01 | 16.9 | **0.00** | 0.02 | 0.01 | 17.5 | **0.00** | 0.02 | 0.01 | 17.4 | **0.00** | **0.00** | **0.00** | **0.00** |
| (20,3,4) | 0.03 | 0.11 | 0.06 | 2.69 | 0.00 | 0.07 | 0.04 | 1.90 | 0.03 | 0.07 | 0.06 | 0.41 | 0.03 | 0.09 | 0.06 | 1.06 | **0.00** | **0.03** | **0.02** | **0.00** |
| (20,4,5) | 0.01 | 0.24 | 0.12 | 5.97 | **0.00** | 0.24 | 0.15 | 5.87 | 0.03 | 0.24 | 0.13 | 4.10 | 0.03 | 0.24 | 0.14 | 4.16 | 0.01 | **0.06** | **0.03** | **0.00** |
| (50,2,4) | **0.00** | 0.02 | 0.01 | 1.87 | **0.00** | 0.01 | 0.00 | 0.53 | **0.00** | 0.03 | 0.01 | 6.93 | **0.00** | 0.03 | 0.01 | 4.24 | **0.00** | **0.01** | **0.00** | **0.00** |
| (50,3,6) | 0.01 | 0.06 | 0.05 | 1.76 | 0.01 | 0.14 | 0.05 | 4.07 | 0.01 | 0.10 | 0.05 | 2.17 | 0.01 | 0.08 | 0.04 | 1.89 | **0.00** | **0.03** | **0.01** | **0.00** |
| (50,4,6) | 0.01 | 0.18 | 0.10 | 3.80 | 0.00 | 0.18 | 0.06 | 2.91 | 0.01 | 0.15 | 0.07 | 2.25 | 0.01 | 0.15 | 0.10 | 2.81 | **0.00** | **0.04** | **0.02** | **0.00** |
| (100,3,6) | 0.00 | 0.18 | 0.10 | 6.56 | **0.01** | 0.18 | 0.08 | 6.36 | 0.00 | 0.17 | 0.07 | 5.10 | 0.00 | 0.18 | 0.08 | 6.24 | 0.01 | **0.03** | **0.02** | **0.00** |
| (100,4,6) | 0.03 | 0.37 | 0.19 | 3.31 | 0.00 | 0.20 | 0.08 | 1.01 | 0.01 | 0.20 | 0.09 | 1.37 | 0.01 | 0.25 | 0.11 | 1.65 | **0.00** | **0.08** | **0.04** | **0.00** |
| (100,5,8) | 0.06 | 0.24 | 0.17 | 1.92 | 0.06 | 0.24 | 0.16 | 1.59 | 0.01 | 0.24 | 0.14 | 1.91 | 0.01 | 0.24 | 0.17 | 2.01 | **0.00** | **0.08** | **0.03** | **0.00** |
| (150,3,8) | 0.00 | 0.65 | 0.15 | 13.4 | 0.01 | 0.27 | 0.08 | 4.93 | 0.01 | 0.37 | 0.12 | 8.22 | 0.01 | 0.15 | 0.05 | 2.52 | **0.00** | **0.04** | **0.02** | **0.00** |
| (150,4,8) | 0.11 | 0.39 | 0.23 | 3.72 | 0.00 | 0.29 | 0.16 | 3.14 | 0.01 | 0.28 | 0.16 | 3.61 | 0.01 | 0.36 | 0.17 | 3.69 | **0.00** | **0.07** | **0.03** | **0.00** |
| (150,5,10) | 0.11 | 0.47 | 0.32 | 0.89 | 0.13 | 0.36 | 0.26 | 0.25 | 0.17 | 0.33 | 0.25 | 0.07 | 0.17 | 0.44 | 0.29 | 0.72 | **0.00** | **0.21** | **0.07** | **0.00** |
| (200,4,10) | 0.09 | 0.82 | 0.32 | 5.71 | 0.04 | 0.37 | 0.16 | 2.44 | 0.04 | 0.33 | 0.13 | 2.25 | 0.04 | 0.36 | 0.20 | 2.21 | **0.00** | **0.09** | **0.06** | **0.00** |
| (200,5,10) | 0.12 | 0.84 | 0.40 | 2.01 | 0.08 | 0.94 | 0.42 | 1.77 | 0.05 | 0.42 | 0.22 | 0.61 | 0.05 | 0.58 | 0.36 | 0.68 | **0.00** | **0.29** | **0.12** | **0.00** |
| (200,6,12) | 0.12 | 0.39 | 0.22 | 0.17 | 0.03 | 0.31 | 0.17 | 0.66 | 0.00 | 0.44 | 0.18 | 1.09 | **0.00** | 0.43 | 0.23 | 0.92 | 0.02 | **0.22** | **0.11** | **0.00** |
| (250,4,12) | 0.21 | 0.33 | 0.27 | 0.21 | 0.04 | 0.44 | 0.21 | 2.47 | 0.09 | 0.52 | 0.25 | 2.00 | 0.09 | 0.47 | 0.24 | 2.44 | **0.00** | **0.13** | **0.07** | **0.00** |
| (250,5,12) | 0.13 | 0.53 | 0.31 | 1.70 | **0.00** | 0.47 | 0.25 | 1.85 | 0.08 | 0.60 | 0.33 | 2.64 | 0.08 | 0.47 | 0.24 | 1.89 | 0.04 | **0.19** | **0.11** | **0.00** |
| (250,6,15) | 0.24 | 0.31 | 0.27 | **0.00** | 0.19 | 0.31 | 0.29 | 0.72 | 0.19 | 0.31 | 0.29 | 0.60 | 0.19 | 0.31 | 0.28 | 0.68 | **0.00** | **0.18** | **0.08** | 1.37 |
| (300,4,15) | 0.01 | 0.60 | 0.30 | 4.27 | **0.00** | 0.46 | 0.22 | 3.14 | 0.10 | 0.76 | 0.26 | 4.66 | 0.10 | 0.56 | 0.20 | 4.77 | 0.01 | **0.12** | **0.05** | **0.00** |
| (300,5,15) | 0.13 | 0.42 | 0.27 | 0.39 | 0.04 | 0.47 | 0.21 | 1.20 | 0.11 | 0.39 | 0.26 | 0.78 | 0.11 | 0.34 | 0.21 | 0.47 | **0.00** | **0.23** | **0.09** | **0.00** |
| (300,6,20) | 0.12 | 0.32 | 0.20 | 0.41 | 0.04 | 0.30 | 0.16 | 0.85 | 0.07 | 0.33 | 0.19 | 0.82 | 0.07 | 0.28 | 0.19 | 0.42 | **0.00** | **0.14** | **0.08** | **0.00** |

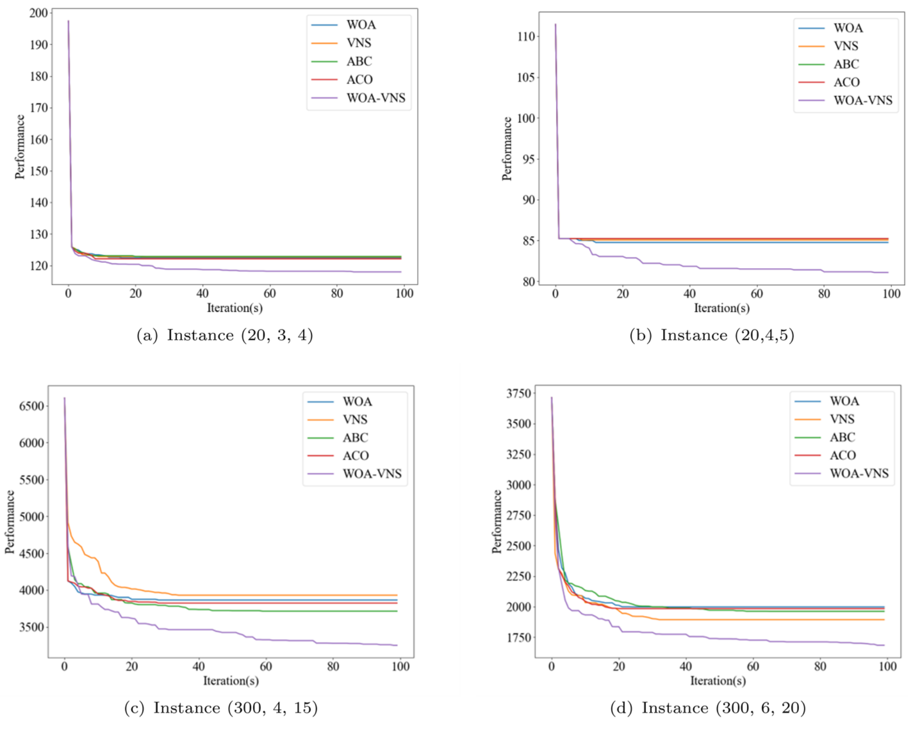


**Fig. 5** Convergence curves for different problem instances

## 6 Conclusion

In this paper, we investigate a joint production scheduling and maintenance planning problem in the parallel machine system, and the objective to minimize the makespan. Both deteriorating jobs and deteriorating maintenance activities are considered in our model. Considering the complexity of this problem, we initiate our study by focusing on a special case wherein the assignments of jobs and the number of maintenance activities allocated to each machine are predetermined. To tackle this challenge on a single-machine level, we introduce a heuristic approach that capitalizes on essential structural properties. Subsequently, we extend our investigation to address the assignment problem encompassing jobs and maintenance activities within the parallel machine environment, and develop a hybrid WOA–VNS algorithm. Computational experiments results unveil the superior performance of the WOA–VNS algorithm, which excels in both search capability and stability when compared to the WOA, VNS, ABC, and ACO algorithms.

Similar to numerous existing studies, our paper assumes a constant processing rate for the machines. However, in actual production settings, the equipment's rate is often adjustable. Future research endeavors can delve into the integration of controllable machining speeds to offer a more comprehensive analysis. Furthermore, our study does not encompass the consideration of potential random equipment failures. This omission represents another promising avenue for advancing research in this field, as it introduces an added layer of complexity and realism to the problem.

**Acknowledgements** This work is supported by the National Natural Science Foundation of China (Nos. 72331004, 72371095, 72071056, 72101077, 71922009, 72271077), Anhui Provincial Natural Science Foundation (No. 2308085MG225, 1908085MG223), Key Research and Development Project of Anhui Province (2022a05020023), and the Base of Introducing Talents of Discipline to Universities for Optimization and Decision-making in the Manufacturing Process of Complex Product (111 projects, B17014).

**Data availability** All the data (input and output) are reported in the paper.

## References

1. Liu, X., et al.: Optimization and management in manufacturing engineering: resource collaborative optimization and management through the Internet of Things, vol. 126. Springer (2017)
2. Gupta, J.N., Gupta, S.K.: Single facility scheduling with nonlinear processing times. Comput. Ind. Eng. **14**, 387–393 (1988)
3. Bertsimas, D., Öztürk, B.: Global optimization via optimal decision trees. J. Global Optim. **2023**, 1–41 (2023)
4. Homayouni, S.M., Fontes, D.B.: Production and transport scheduling in flexible job shop manufacturing systems. J. Global Optim. **79**, 463–502 (2021)
5. Tong, X., Li, M., Sun, H.: Decision bounding problems for two-stage distributionally robust stochastic bilevel optimization. J. Global Optim. **87**, 1–29 (2022)
6. Hu, C., Zheng, R., Lu, S., Liu, X., Cheng, H.: Integrated optimization of production scheduling and maintenance planning with dynamic job arrivals and mold constraints. Comput. Ind. Eng. **186**, 109708 (2023)
7. Zheng, R., Zhao, X., Hu, C., Ren, X.: A repair-replacement policy for a system subject to missions of random types and random durations. Reliab. Eng. Syst. Saf. **232**, 109063 (2023)
8. Wu, C.-C., Lee, W.-C.: Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine. Inf. Process. Lett. **87**, 89–93 (2003)
9. Ji, M., He, Y., Cheng, T.E.: Scheduling linear deteriorating jobs with an availability constraint on a single machine. Theoret. Comput. Sci. **362**, 115–126 (2006)
10. Low, C., Hsu, C.-J., Su, C.-T.: Minimizing the makespan with an availability constraint on a single machine under simple linear deterioration. Comput. Math. Appl. **56**, 257–265 (2008)
11. Lodree, E.J., Jr., Geiger, C.D.: A note on the optimal sequence position for a rate-modifying activity under simple linear deterioration. Eur. J. Oper. Res. **201**, 644–648 (2010)
12. Sun, X., Geng, X.-N.: Single-machine scheduling with deteriorating effects and machine maintenance. Int. J. Prod. Res. **57**, 3186–3199 (2019)
13. Rustogi, K., Strusevich, V.A.: Single machine scheduling with general positional deterioration and rate-modifying maintenance. Omega **40**, 791–804 (2012)
14. Zhang, X., Yin, Y., Wu, C.-C.: Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine. Eng. Optim. **49**, 84–97 (2017)
15. Lu, S., Liu, X., Pei, J., Thai, M.T., Pardalos, P.M.: A hybrid abc-ts algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity. Appl. Soft Comput. **66**, 168–182 (2018)
16. Zhang, X., Liu, S.-C., Lin, W.-C., Wu, C.-C.: Parallel-machine scheduling with linear deteriorating jobs and preventive maintenance activities under a potential machine disruption. Comput. Ind. Eng. **145**, 106482 (2020)
17. Lee, H.-T., Yang, D.-L., Yang, S.-J.: Multi-machine scheduling with deterioration effects and maintenance activities for minimizing the total earliness and tardiness costs. Int. J. Adv. Manuf. Technol. **66**, 547–554 (2013)
18. Da, W., Feng, H., Pan, E.: Integrated preventive maintenance and production scheduling optimization on uniform parallel machines with deterioration effect, pp. 951–955. IEEE (2016)
19. Woo, Y.-B., Kim, B.S.: Matheuristic approaches for parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. Comput. Oper. Res. **95**, 97–112 (2018)
20. Öztürkoğlu, Ö.: Identical parallel machine scheduling with nonlinear deterioration and multiple rate modifying activities. Int. J. Optim. Control Theor. Appl. **7**, 167–176 (2017)
21. Yang, S.-J., Yang, D.-L.: Minimizing the total completion time in single-machine scheduling with aging/deteriorating effects and deteriorating maintenance activities. Comput. Math. Appl. **60**, 2161–2169 (2010)

22. Graham, R.L., Lawler, E.L., Lenstra, J.K., Kan, A.R.: In: Optimization and approximation in deterministic sequencing and scheduling: a survey, vol. 5 287–326. Elsevier (1979)
23. Hardy, G.H., Littlewood, J.E., Pólya, G.: Inequalities. Cambridge University Press, Cambridge (1952)
24. Sethi, R.: On the complexity of mean flow time scheduling. Math. Oper. Res. **2**, 320–330 (1977)
25. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
26. Chakraborty, S., Saha, A.K., Sharma, S., Mirjalili, S., Chakraborty, R.: A novel enhanced whale optimization algorithm for global optimization. Comput. Ind. Eng. **153**, 107086 (2021)
27. Li, Y., Han, T., Zhao, H., Gao, H.: An adaptive whale optimization algorithm using gaussian distribution strategies and its application in heterogeneous ucavs task allocation. IEEE Access **7**, 110138–110158 (2019)
28. Vijayanand, R., Devaraj, D.: A novel feature selection method using whale optimization algorithm and genetic operators for intrusion detection system in wireless mesh network. IEEE Access **8**, 56847–56854 (2020)
29. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
30. Bathrinath, S., Saravana Sankar, S., Ponnambalam, S.G., Jerin Leno, I.: Vns-based heuristic for identical parallel machine scheduling problem, pp. 693–699. Springer (2015)
31. Tavares Neto, R. F., Godinho Filho, M., Da Silva, F. M.: An ant colony optimization approach for the parallel machine scheduling problem with outsourcing allowed. J. Intell. Manuf. **26**, 527–538 (2015)
32. Caniyilmaz, E., Benli, B., Ilkay, M.S.: An artificial bee colony algorithm approach for unrelated parallel machine scheduling with processing set restrictions, job sequence-dependent setup times, and due date. Int. J. Adv. Manuf. Technol. **77**, 2105–2115 (2015)