

Recall cost-time tradeoffs for remanufacturing shop lot streaming scheduling problem with non mixed production using an improved non-dominated sorting genetic algorithm

Gang Wang^{a*} and Minglun Ren^{a,b}

^aSchool of Management, Hefei University of Technology, Hefei, 230009, China

^bKey Laboratory of Process Optimization and Intelligent Decision-making of Ministry of Education, Hefei, 230009 China

CHRONICLE

Article history:

Received March 14 2025

Received in Revised Format

May 22 2025

Accepted July 12 2025

Available online July 12 2025

Keywords:

Recall cost

Remanufacturing shop

Lot streaming

Improved non-dominated
sorting genetic algorithm

Non-mixed scheduling

Completion time

ABSTRACT

In this paper, we study the problem of lot streaming scheduling in a remanufacturing shop with consistent sublots, where mixed production is not allowed between sublots possessing different types of remanufacturable parts. The problem is formulated as a multi-objective optimization problem with optimization objectives of recall cost and completion time. Such problems are NP-hard and need to be solved using an improved non-dominated sorting genetic algorithm. Two vectors regarding subplot size allocation and subplot processing order determination together form a solution. In order to improve the quality of the solution, the algorithm uses a randomization strategy and two heuristics to initialize the population and introduces dynamic genetic operations to advance the population diversity. On the one hand, the designed four types of genetic operators are dynamically selected according to the number of iterations. On the other hand, the elite retention strategy is improved, i.e., based on the probability that one of the individuals performing the crossover operation can come from the memory bank. Both numerical experiments and real case solving verify the effectiveness of the developed algorithms.

© 2025 by the authors; licensee Growing Science, Canada

1. Introduction

With the rapid development of the manufacturing industry, the replacement cycle of products is shortened, and a large number of retired products and used parts are subsequently generated (Zhao et al., 2019). Remanufacturing can complete the whole life cycle of the product closed loop, which is an ideal way to realize the green circular economy (Gong et al., 2020). Therefore, remanufacturing has gradually received extensive attention from all walks of life. The remanufacturing workshop is an important part to realize the digitalization of the remanufacturing industry. However, remanufacturing workshops generally suffer from manufacturing inefficiency. At present, many remanufacturing enterprises have unsatisfactory profitability and efficiency, mainly due to the failure to integrate and optimize the limited production resources in remanufacturing workshops, resulting in low remanufacturing efficiency (Guide et al., 1999). Therefore, many studies improve the remanufacturing efficiency by optimizing the scheduling scheme, but most of the literature now regard a single part as the scheduling object. However, in the actual remanufacturing workshop, considering the difference in quality grade between remanufacturable parts, after classification and screening, remanufacturable parts with the same quality grade are often processed in lots, i.e., a larger batch is divided into several sublots, and then sublots are produced and transferred in lot streaming, which not only eliminates the necessary auxiliary operations and saves time overhead, but also is conducive to remanufactured parts' post-tracing (Sun et al., 2018). In recent years, manufacturing recalls have gradually increased, causing significant losses to manufacturing companies (Chakraborty et al., 2023; Gorton and Stasiewicz, 2017). The variable quality status among remanufacturable parts and the lack of reliability of remanufacturing equipment are the key factors that lead to the probability of recalls of remanufactured parts exceeding that of other manufacturing industries (Ghazi et al., 2023). There has been a flurry of literature on reducing recall costs by constructing models that assume lots are indivisible and therefore only need to focus on the allocation of raw material lots with respect to order or product lots (Dupuy et al., 2005; Maity et al., 2021; Zhi-hui et al., 2012). However, in the lot streaming production environment of remanufacturing shops, not only the allocation of each subplot

* Corresponding author

E-mail 2019010076@mail.hfut.edu.cn (G. Wang)

ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)

2025 Growing Science Ltd.

doi: 10.5267/j.ijiec.2025.7.002

about the available machines in the process, i.e., the sublots sequencing problem, but also the lots division problem needs to be concerned. Therefore, this paper takes the recall cost as one of the optimization objectives, which is not only helpful for enterprises to reduce the recall loss, but also beneficial to improve product quality and consumer satisfaction.

Based on the above analysis, this paper constructs a multi-objective mixed-integer planning model for the remanufacturing shop lot streaming scheduling problem with non-mixed-schedule production among sublots (RSLSS_NMRP) with the objective of optimizing the completion time and recall cost at the same time, and a consistent sublots strategy is adopted. Since the lot streaming scheduling problem with 3 and more machines is NP-hard (Sang et al., 2018), the study in this paper belongs to the NP-hard problem. Methods to solve such problems include exact methods (Fang et al., 2021) and heuristic algorithms (H. Wang et al., 2019). Among them, heuristic algorithms are widely used because of their strong search ability and easy implementation (Tang et al., 2021). Therefore, an improved non-dominated sorting genetic algorithm (INSGA-II) is used to solve the problem.

Compared to previous work, this study makes the following three contributions.

- The necessity of reducing the recall cost of remanufactured parts is demonstrated, and the formula for calculating the recall cost is derived.
- Constructs a multi-objective mixed-integer planning model oriented to completion time and recall cost for RSLSS_NMRP, and proposes an improved non-dominated sorting genetic algorithm to solve the model.
- Demonstrates the validity of INSGA-II by comparing the IGD values of INSGA-II with those of GASA, HPSO, MOACA, HABC, and MOGWA under 21 cases of arithmetic.

This paper is organized as follows. Section 2 reviews the existing studies on lot streaming and recall cost optimization. Section 3 describes RSLSS_NMRP and gives the mathematical model. INSGA-II is discussed in Section 4. Section 5 presents simulation experiments and analysis. Section 6 summarizes the work and outlines future research directions.

2. Related work

2.1 Lot streaming

Lot streaming reduces completion time due to its ability to reduce equipment idle time and work-in-process inventory, among other effects. Therefore, since the concept of lot streaming was proposed by Reiter (1966), many scholars have studied lot streaming from different perspectives. At present, most of the optimization objectives of lot streaming models are related to time, cost and energy consumption. Time-related optimization objectives include maximum completion time (Božek & Werner, 2018; Mortezaei & Zulkifli, 2014; Singh et al., 2024; Ventura & Yoon, 2013; Yunusoglu & Topaloglu Yildiz, 2023), tardiness (Nejati et al., 2014; S. Wang et al., 2019; Zhu et al., 2024), flow time (Alfieri et al., 2021) and deviation time (B. Zhang et al., 2023; Zhu et al., 2024). In recent years, environmental protection has become increasingly important and the problem of optimal scheduling of energy consumption based on lot streaming has gained attention (Chakaravarthy et al., 2014; Wang et al., 2024). It is worth noting that many studies involve multiple objectives for lot streaming model optimization problems, such as simultaneously optimizing time and cost (Cheng et al., 2024) and simultaneously optimizing time and energy consumption (Pan et al., 2023a).

The determination of subplot sizing consists of three strategies, namely consistent sublots, equal sublots and variable sublots. Consistent sublots is defined as follows: $Q_{ik} = Q_{i+1k}$ for all i and k . i is the process index, k is the sublots index, and Q_{ik} denotes the size of the k th subplot size in the i th process. Equal sublots is defined as follows: $Q_k = Q_{k'}$ and $Q_{ik} = Q_{ik+1}$ for all i and k . Variable sublots is defined as follows: $Q_{ik} \neq Q_{i+1k}$ for some i and k . Pan et al. (2023b) compared the advantages and disadvantages of various heuristic algorithms for solving the completion time of a distributed flow shop using the equal sublots strategy. Tian et al. (2024) used a knowledge-based heuristic optimization algorithm with a variable sublots strategy to determine the value of the size of each subplot in each process to make the objectives of completion time and energy consumption under pareto optimality. Shao et al. (2024) investigated the effect of sublots size and processing order on the total delay time of a distributed hybrid flow shop under a consistent sublots strategy.

The solution of lot streaming models belongs to NP-hard problems. Its solution methods gradually transition from exact methods to population intelligent optimization algorithms. Although the exact method can obtain the global optimal solution, it can only solve the problem of smaller scale. Based on Cheng et al. (2013), Table 1 summarizes some of the population intelligence optimization algorithms involved in the lot streaming literature. In Table 1, the lot streaming model is described by the following features: production type/number of product types/sublot types/operational continuity/machine lead time/optimization objective. Where FJm denotes flexible job shop, Fm denotes flow shop and HFm denotes hybrid flow shop. Idling indicates that the machine is allowed to have idling, while the opposite is true for No-idling. EC denotes energy consumption, BD denotes start time deviation, DD is end time deviation, and RC denotes recall cost.

Table 1
Population intelligence optimization algorithm forsolving lot streaming models

Lot streaming model	References	Algorithm
HFm/1/Consistent subplot/No-idling/Cmax , EC	(Weiwei Wang et al., 2023)	artificial bee colony algorithm
HFm/1/Consistent subplot/Idling/Setup/Cmax	(Lu et al., 2023)	Adaptive Iterated Greedy Algorithm
HFm/n/Equal subplot/Idling/Setup/Cmax	(Han et al., 2019)	Migrating Birds Optimization algorithm
HFm/n/Consistent subplot/No-idling/Cmax , EC	(Chen et al., 2020)	genetic algorithm based approaches
Fm/n/Equal subplot/Idling/Setup/Cmax	(Sang et al., 2018)	Invasive weed optimization
HFm/n/Consistent subplot/Idling/Setup/Cmax , BD	(B. Zhang et al., 2023)	A decomposition-based evolutionary algorithm
HFm/n/Consistent subplot/Idling/Setup/Cmax , DD	(Zhu et al., 2024)	knowledge-based memetic algorithm
HFm/n/Consistent subplot/Idling/SD-Setup/Cmax	(Zhang et al., 2022)	An automatic evolutionary algorithm
Fm/n/Variable subplot/Idling/Setup/Cmax	(Wang et al., 2022)	water wave optimization algorithm
Fm/n/Equal subplot/Idling/Setup/Cmax	(Vijaychakaravarthy et al., 2014)	Improved Sheep Flock Heredity Algorithm and Artificial Bee Colony Algorithm
FJm/n/Consistent subplot/Idling/Setup/Cmax	(Y. Li et al., 2023)	A Reinforcement Learning-Artificial Bee Colony algorithm
HFm/n/Variable subplot/Idling/SD-Setup/Cmax	(Li et al., 2024)	A collaborative iterative greedy algorithm
HFm/n/Consistent subplot/Idling/SD-Setup/Cmax , RC	This paper	Multi-objective firefly algorithm

2.2 Recall cost

Dupuy et al. (2005) proposed a sausage recall cost optimization model with a 3-level "disassembly and assembly" bill of materials. The model uses batch dispersion to calculate the number of batches to be recalled. Batch dispersion is the number of raw material batches used in different product batches. The recall cost optimization model is constructed on this basis and it is shown that the model solution is an NP-hard problem. Dabbene et al. (2014), for a variety of production methods, state that recall costs may be affected by the number of lots, subplot size, and production methods. Maity et al. (2021) constructed a recall cost optimization model under stochastic demand for a production model with a 5-level "disassemble and assemble" bill of materials, also using batch dispersion as a measure of recall cost, and solved the model for a small-scale example using a nested L-shaped approach. Since many production managers are concerned about the maximum recall cost of defective products, Dabbene and Gay (2011) proposed the worst-case maximum recall cost (WCRC) to measure traceability. Qian et al. (2022) argued that batch dispersion phenomenon and mixing of sublots across production processes increase the recall probability, and therefore introduced in their literature the average recall cost (ARC) metric. The recall cost optimization model of a production system with "raw material lot - component lot - product lot" as the main link is generally an NP-hard problem, which is suitable to be solved by a population intelligence optimization algorithm. Dupuy et al. (2005) proved that the recall cost model with 3-level "disassembly and assembly" bill of materials production method is an NP-hard problem, and applied the branch-and-bound method in LINGO6.0 software to solve the model for a processing plant with 8 raw material batches, with a total time of 12 h. Zhi-hui et al. (2012) proved that the recall cost optimization model with 2 levels of "disassembly and assembly" is also an NP-hard problem, and the model is also solved by using the branch-and-bound method, and Bin et al. (2015) have effectively reduced the recall size of products by using genetic algorithms to make decisions about the order in which raw material batches are processed. Ren and Wang (2024) proposed a variable-neighborhood genetic algorithm to solve a recall cost optimization model for a machining shop, and demonstrated that the proposed algorithm is more suitable for solving large-scale problems than traditional optimization methods.

3. Problem formulation

3.1 Problem description

Remanufacturing refers to the manufacturing process in which remanufacturable parts are used as production blanks, and their quality characteristics are made to reach or not lower than the level of the original new products through specialized repair or upgrading. The remanufacturing process flow is shown in Fig. 1, which generally includes nondestructive disassembly, cleaning of scrap parts, damage monitoring, life assessment and remanufacturing processing (Sun et al., 2018). The pink part in Figure 1 shows the remanufacturing process involved in this paper. The initial quality control and classification process classifies the cleaned scrap parts into discarded parts, remanufacturable parts and direct-use parts, the discarded parts are directly scrapped, the direct-use parts are transported to the assembly plant for reassembly, and the remanufacturable parts are formed into remanufacturable parts with multiple quality levels after the secondary quality classification, as shown in Fig. 2.

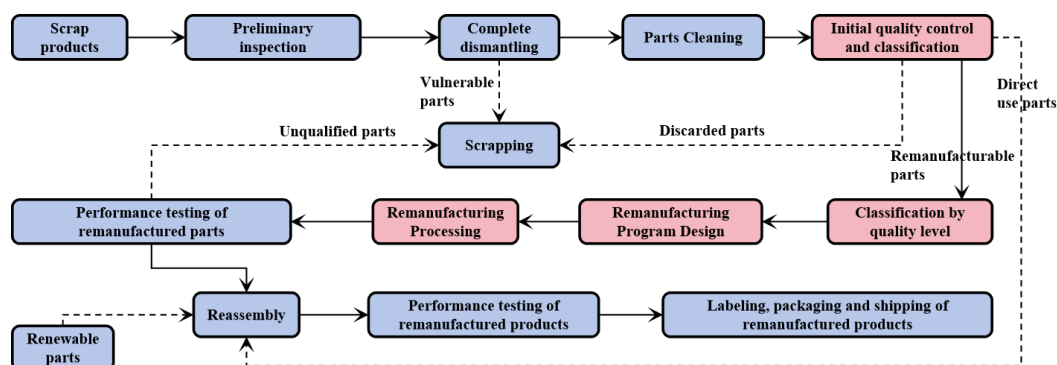


Fig. 1. Remanufacturing Process Flow

Different quality levels of remanufacturable parts correspond to different prices and reliability of remanufactured parts, so the recall probability and recall cost will be different. When designing the remanufacturing program, it is necessary to divide the remanufacturable parts of each quality grade into sublots according to the requirements of delivery and completion time, so as to realize batch production and transmission, which is not only conducive to the full utilization of remanufacturable resources, but also can significantly reduce the recall cost. After that, the remanufacturable parts are sent to the remanufacturing workshop in the form of sublots to start production according to the scheduling plan. In order to cope with the requirements of large-scale batch processing, remanufacturing companies generally adopt a hybrid flow shop layout, i.e., each subplot completes the remanufacturing process in a fixed machining sequence in each process (Gong et al., 2020).

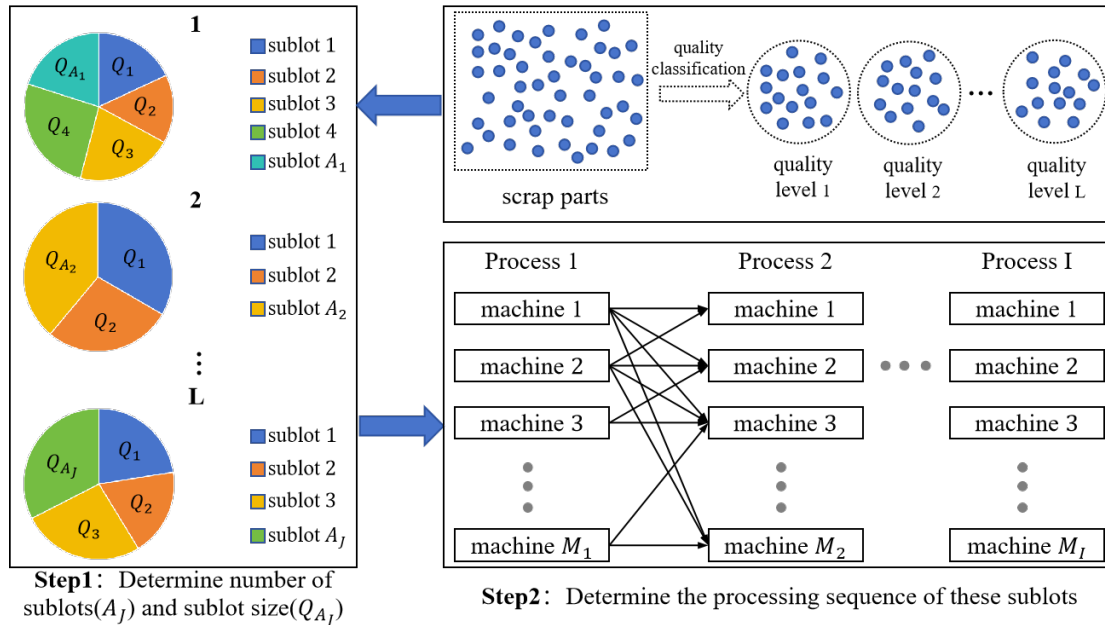


Fig. 2. Schematic diagram of the remanufacturing shop lot streaming scheduling

In summary, as shown in Figure 2, RSLSS_NMRP needs to determine not only the number of sublots and sublots size for each quality level of remanufacturable parts, but also the remanufacturing sequence of each subplot to maximize the minimization of completion time and recall cost.

RSLSS_NMRP can be described as follows: remanufacturable parts are categorized into L classes according to their quality status, and the size of remanufacturable parts with quality class l ($l = 1, 2, \dots, L$) is Q_l . These remanufacturable parts are repaired and modified in a hybrid flow remanufacturing shop, and the remanufacturing process of all remanufacturable parts contains I processes, and the number of machines available for the i th ($i = 1, 2, \dots, I$) process is M_i , and the performance of each machine is different. In each process, the machining of remanufactured parts of one quality grade is completed before the machining of remanufactured parts of other quality grades can begin, i.e., non-mixed-row production.

In order to shorten the remanufacturing cycle time, it is now necessary to process the remanufacturable parts in sublots, with the subplot being considered as the smallest scheduling unit. the maximum number of sublots corresponding to the remanufacturable parts of quality class L is B_l , and the subplot type is consistent subplot. The remanufacturing time is known to be T_{ikm} and the adjustment time to be $T_{ik' km}$. It is now necessary to determine the actual number of sublots A_l and subplot sizes Q_{A_l} of remanufacturable parts of quality class l and the remanufacturing sequence of each subplot to maximize the minimization of the recall cost and the completion time.

In order to focus the problem and reduce the modeling complexity, the assumptions about RSLSS_NMRP are as follows:

- (1) All remanufacturable parts can start remanufacturing at moment zero;
- (2) A machine can process only one remanufacturable part at a time;
- (3) The transportation time of sublots between processes is negligible and both the transportation capacity and buffer capacity are large enough;
- (4) Remanufacturable parts of the same quality grade are processed in equal time on the m th machine of the i th process;
- (5) The reliability of all machines obeys a Weibull distribution with the same shape and size parameters.

3.2 Optimization objectives

The optimization objectives of RSLSS_NMRP include maximum completion time C_{max} and recall cost RC . C_{max} is commonly used to measure the machining efficiency and its calculation formula is shown in Eq. (1).

$$C_{max} = \min \left(\max_{\forall (i,k,m)} (CK_{ikm}) \right) \quad (1)$$

What needs to be described in detail is the RC , which has been used by many literatures to measure traceability outcomes and achieve traceability optimization by constructing traceability optimization models (Qian et al., 2022). They use the product of the maximum subplot recall size and the unit part selling price to calculate the RC . However, in the remanufacturing shop, the factors that may contribute to recalls are not only related to the quality of remanufacturable parts, but also to the rate of progressive equipment failure (Maity et al., 2021). Specifically, in terms of equipment factors, machines may experience progressive failures due to the accumulation of processing time, which can be insidious and result in an increased probability of quality problems after delivery of the remanufactured part.

The progressive failure rate and reliability of the machine obey the Weibull distribution, which can be calculated by Eq. (2). where R_m and δ_m denote the reliability and failure rate of the m th machine, respectively, u_m denotes the machine service age, which refers to the cumulative time the machine has been involved in machining, α is the shape parameter of the Weibull distribution, and β is the size parameter. Therefore, the recall probability of subplot k in a remanufacturing shop caused by a progressive machine failure is shown in Eq. (4). Where x_{km} is a binary decision variable that takes the value of 1 if subplot k is processed on machine m and 0 otherwise.

$$R_m = \exp \left(- \left(\frac{u_m}{\beta} \right)^\alpha \right) \quad (2)$$

$$\delta_m = 1 - R_m \quad (3)$$

$$RC_1 = 1 - \sum_{m=1}^M x_{km} \times R_m \quad (4)$$

In terms of material factors, remanufacturable parts of different quality grades differ in terms of residual life, wear and aging, etc., and remanufactured parts resulting from the processing of remanufacturable parts of different quality grades will have different reliability and unit cost of sale, and therefore different probability of recall and cost of recall (Ghazi et al., 2023). Therefore, the recall probability due to the subplot quality level attribute is shown in Eq. (5). Where S_{kl} is a binary parameter that takes the value of 1 if subplot k belongs to quality class l and 0 otherwise. θ_l denotes the reliability of the remanufactured part corresponding to the remanufacturable part of l quality class.

$$RC_2 = \sum_{l=1}^L S_{kl} \times (1 - \theta_l) \quad (5)$$

In summary, the formula for RC is shown in Eq. (6). Where Q_k represents the number of remanufacturable parts included in the subplot k , and P_l represents the unit sales price of remanufactured parts corresponding to remanufacturable parts of l quality grade.

$$RC = \min \left(\sum_{k=1}^K \left(Q_k \times \left(\sum_{l=1}^L S_{kl} \times P_l \right) \times RC_1 \times RC_2 \right) \right) \quad (6)$$

A simple example to illustrate the calculation process is shown in Fig. 3. In the example, the remanufacturable parts are classified into 2 classes after quality grading, i.e., $l = 2$, and the remanufactured part corresponding to class 1 is better in quality than that of class 2. The reliabilities of the remanufactured parts corresponding to class 1 and class 2 are $\theta_1 = 0.9635$ and $\theta_2 = 0.9486$, respectively. The unit selling prices of remanufactured parts corresponding to grades 1 and 2 are $P_1 = 10$ and $P_2 = 7$, respectively. Each grade of remanufacturable parts contains three sublots. The number of remanufacturable parts contained in subplot 1 through subplot 6 are $Q_1 = 80$, $Q_2 = 80$, $Q_3 = 90$, $Q_4 = 90$, $Q_5 = 95$, and $Q_6 = 95$ in that order. The remanufacturing process consists of three steps. Each process consists of three machines with different performance. The reliability calculated from Eq. (2) and the service age of each machine is shown in Table 2, with a value of 3 for the shape parameter and 300 for the size parameter. x_{km} can be seen from Fig. 3. Therefore, $RC = 733.9733$ can be calculated from Eq. (6).

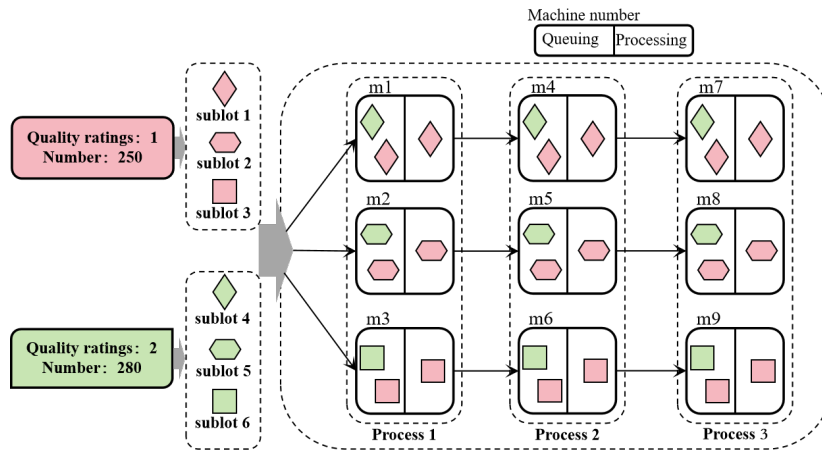


Fig. 3. Example of remanufacturing shop lot streaming scheduling

Table 2

Age and failure rate of machines

m	1	2	3	4	5	6	7	8	9
u	6	2	2	3	3	5	8	5	7
R_m	0.9418	0.9802	0.9802	0.9704	0.9704	0.9512	0.9231	0.9512	0.9324

In fact, recall costs can be reduced by changing the number of sublots of remanufacturable parts of each quality level, the size of each subplot, and the remanufacturing shop scheduling program. In the above case, for example, the changed subplot sizes of subplot 1 through subplot 6 are 60, 80, 110, 90, 75, and 115, in that order. The variation of machines processing each subplot in each process is shown in Fig. 4, and the variation of RC is observed by involving as many reliable machines as possible in remanufacturing. The $RC = 706.6556$ at this point can be calculated from Eq. (6), and the RC is significantly lower.

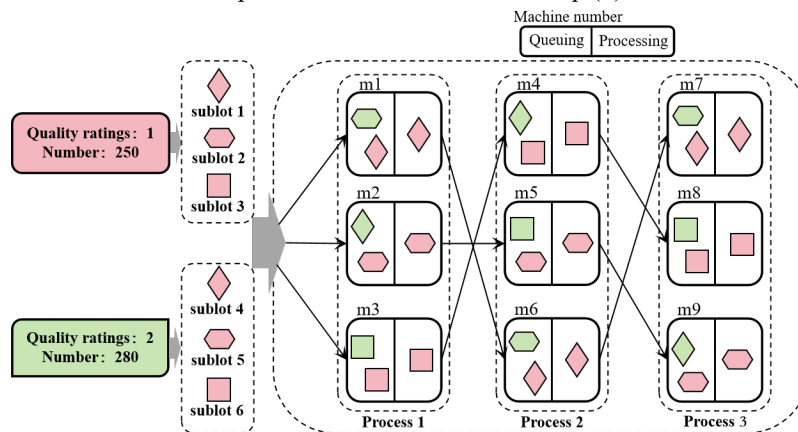


Fig. 4. Example of remanufacturing shop lot streaming scheduling improvement considering RC

3.3 Model formulation

To construct the mathematical model, the following symbols, parameters and variables were introduced:

I : total number of processes

i : process index, $i = 1, 2, \dots, I$

M_i : number of machines available for process i

M : total number of machines, $M = M_1 + M_2 + \dots + M_I$

m : machine index for process i , $m = 1, 2, \dots, M_i$

u_m : machine age, the length of time a machine has been in service

α, β : are the shape and size parameters of the Weibull distribution, respectively

R_m, δ_m : are the reliability and the asymptotic failure rate of the m th machine, respectively

L : total number of quality grades of remanufacturable parts

l : index of quality grades, $l = 1, 2, \dots, L$

θ_l : reliability of remanufactured parts corresponding to remanufacturable parts of quality grade l

B_l : maximum number of sublots of remanufacturable parts of quality grade l

P_l : unit selling price of remanufactured parts corresponding to remanufacturable parts of quality grade l

K : total number of sublots of remanufacturable parts, $K = B_1 + B_2 + \dots + B_L$

k : index of sublots, $k = 1, 2, \dots, K$

S_{kl} : 0-1 parameter, takes the value 1 if the subplot belongs to quality grade l , 0 otherwise

Q_l : total number of remanufacturable parts of quality grade l

T_{ikm} : processing time required for machine m in process i to process single remanufacturable parts in subplot k

$T_{ik'km}$: adjustment time required to process subplot k after subplot k' on machine m in process i

H : exceptionally large positive integer

A_l : actual number of sublots of remanufacturable parts of quality grade l

Q_k : number of remanufactured parts included in subplot k

CK_{ikm} : completion time of subplot k

CM_{ikm} : completion time of machine m

x_{km} : 0-1 variable that takes the value of 1 if the subplot k is processed on machine m , 0 otherwise

$x_{ikk'm}$: 0-1 variable that takes the value 1 if k is the immediately preceding subplot of k' on the i th process machine m , 0 otherwise

x_{i0km} : 0-1 variable that takes the value of 1 if subplot k is processed first on the i th process machine m , and 0 otherwise

x_{ik0m} : 0-1 variable that takes the value of 1 if subplot k was last processed on the i th process machine m , and 0 otherwise

Based on the above symbols, parameters and variables, the following mathematical model is constructed:

$$C_{max} = \min \left(\max_{\forall(i,k,m)} (CK_{ikm}) \right) \quad (7a)$$

$$RC = \min \left(\sum_{k=1}^K \left(Q_k \times \left(\sum_{l=1}^L S_{kl} \times P_l \right) \times \left(1 - \sum_{m=1}^M x_{km} \times R_m \right) \times \left(\sum_{l=1}^L S_{kl} \times (1 - \theta_l) \right) \right) \right) \quad (7b)$$

$$\sum_{l=1}^L B_l = K \quad (8)$$

$$1 \leq A_l \leq B_l, \forall l \quad (9)$$

$$\sum_{k=1}^K S_{kl} \times Q_k = Q_l, \forall l \quad (10)$$

$$Q_k \in N \cup \{0\}, \forall k \quad (11)$$

$$\sum_{k=0}^K \sum_{m=1}^{M_i} x_{ikk'm} = 1, \forall (i, k') \quad (12)$$

$$\sum_{k'=0}^K \sum_{m=1}^{M_i} x_{ikk'm} = 1, \forall (i, k) \quad (13)$$

$$\sum_{k=0}^K x_{ikk'm} - \sum_{k=0}^K x_{ik'km} = 0, \forall (i, k', m \in M_i) \quad (14)$$

$$x_{ikk'm} = 0, \forall (i, k, m) \quad (15)$$

$$CM_{ikm} + \sum_{m=1}^{M_i} x_{ikk'm} \times (Q_k \times T_{ikm} + T_{ikk'm}) + (\sum_{m=1}^{M_i} x_{ikk'm} - 1) \times H \leq CM_{ik'm}, \forall (i, k, k') \tag{16}$$

$$CK_{i+1km} = \begin{cases} CK_{ikm} + Q_k \times T_{ikm}, CM_{ikm} < CK_{ikm} - T_{ik'km} \\ CM_{ikm} + Q_k \times T_{ikm} + T_{ik'km}, CM_{ikm} \geq CK_{ikm} \end{cases}, \forall (i, k, k', m \in M_i) \tag{17}$$

$$CK_{0km} = 0, \forall (k, m \in M_i) \tag{18}$$

$$\sum_{m=1}^{M_i} \sum_{k \in l, k' \in l'} S_{kl} \times S_{k'l'} \times x_{ikk'm} = \begin{cases} M_i, B_l > M_i \\ B_l, B_l \leq M_i \end{cases} \tag{19}$$

$$S_{kl}, x_{km}, x_{ikk'm}, x_{i0k'm}, x_{ik0m} \in \{0,1\}, \forall (i, k, k', m, l) \tag{20}$$

The optimization objectives are to minimize the maximum completion time (C_{max}) and minimize the recall cost (RC), as shown in equations (7a) and (7b), respectively. Constraint (8) is the total number of sublots constraint. Constraint (9) is the actual number of sublots constraint, which limits the actual number of sublots to between 1 and the maximum number of sublots. Constraint (10) is the sublots size constraint. Constraint (11) is a subplot size non-negative integer constraint, i.e., the number of remanufacturable parts contained in any subplot is zero or a positive integer. Constraints (12) and (13) are tight-front subplot constraints and tight-back subplot constraints, respectively, i.e., there is one and only one tight-front subplot and one tightback subplot for any subplot to be processed on a certain machine corresponding to a certain process. Constraint (14) shows that when any subplot is processed on a particular machine corresponding to any process, the subplot is necessarily exposed to one of the following three situations: the subplot is processed first; the subplot is processed last; and there are tight-front and tight-following sublots for the subplot. Constraint (15) restricts the subplot to form a tight-front and tight-back relationship with itself. Constraint (16) shows the constraint between the completion times of subplot k and subplot k' when the machine processes these two sublots on a machine that forms the immediate preceding and following relationship. Constraint (17) shows the constraint relationship between the completion times of neighboring processes of the same subplot. Constraint (18) indicates that the completion time of each subplot before starting processing takes the value of 0. Constraint (19) is a non-mixed-schedule production strategy constraint. Constraint (20) is a 0-1 constraint.

4. Improved non-dominated sorting genetic algorithm

4.1 Encoding and Decoding

Coding is the first problem to be solved when applying genetic algorithms. Based on the research content of this paper, in the coding scheme, it is necessary to determine not only the machine selection problem and the sublots sorting problem, but also the problem of dividing the size of each subplot. Based on the existing coding methods, it is decided to use the truth value segmentation coding method. This coding method is not only easy to decode but also easy to understand and has been widely used (Chen et al., 2020; Wenjie Wang et al., 2023). In the truth-valued segmented coding method, the chromosome is divided into two parts, denoted as $\{X_q, X_s\}$, see a in Fig. 5. Where X_q is a subplot size vector, and the gene Q_k indicates the number of remanufacturable parts contained in the k th subplot, and X_s is a subplot processing order vector, and the gene sequence indicates the processing priority of each subplot in each process. Fig. 5, b, shows the coding scheme for the example used in Section 3.2. In this coding scheme, remanufacturable parts of both quality classes 1 and 2 are divided into three sublots, and the order in which each subplot is processed is shown in Part X_s .

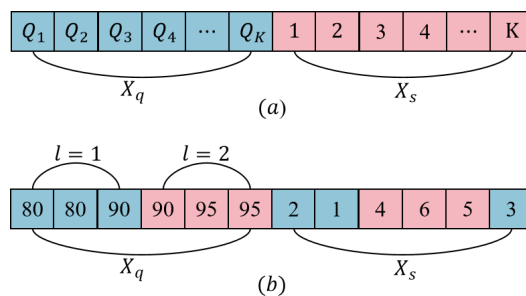


Fig. 5. Truth value segmentation coding mechanism and example

The coding process is the process of determining X_q and X_s . Decoding not only utilizes X_s to randomly match available processing machines for each subplot, but also uses X_q to iteratively calculate the subplot completion time and machine completion time for each process. The maximum completion time is equal to the maximum of the completion time of each subplot of the last process, while the recall cost can be calculated using the information of each subplot of the processing machine recorded during the decoding process as well as the relevant parameters. The decoding algorithm is shown in Algorithm 1. Note that when subplot k is being processed and the machine used to process the next process in subplot k is idle, the machine

can be brought forward into an adjustment state to minimize unnecessary waste of time.

Algorithm 1 Decoding algorithm pseudocode

Input: the feasible solution $X, M_i, B_l, T_{ik'km}, T_{ikm}, R_m, \theta_l, P_l, \alpha, \beta$

Output: C_{max}, RC

- (1) $CK_{ikm} \leftarrow 0, CM_{ikm} \leftarrow 0$
 - (2) $mach_for_slot \leftarrow []$ (Recording machine information for processing sublots)
 - (3) **For** $i \leftarrow 1$ **to** I **do**
 - (4) get the set of available machines M_i^*
 - (5) **For** $k \leftarrow 1$ **to** K **do**
 - (6) a machine is randomly selected from M_i^* for processing subplot k
 - (7) $CM_{ikm} \leftarrow CM_{ik'm} + Q_k \times T_{ik'm} + T_{ik'km}$ (machine completion time update)
 - (8) **If** $CM_{ikm} < CK_{ikm} - T_{ik'km}$ (
 - (9) $CK_{i+1km} \leftarrow CK_{ikm} + Q_k \times T_{ikm}$ (sublots completion time update)
 - (10) **Else if** $CM_{ikm} \geq CK_{ikm} \parallel 0 < CK_{ikm} - T_{ik'km} \leq CM_{ikm}$
 - (11) $CK_{i+1km} \leftarrow CM_{ikm} + Q_k \times T_{ikm} + T_{ik'km}$
 - (12) **End**
 - (13) update M_i^*
 - (14) $mach_for_slot \leftarrow []$
 - (15) **End**
 - (16) **End**
 - (17) $C_{max} \leftarrow \max(CK_{Ikm})$
 - (18) $RC \leftarrow \min\left(\sum_{k=1}^K (Q_k \times \left(\sum_{l=1}^L S_{kl} \times P_l\right)) \times \left(1 - \sum_{m=1}^M x_{km} \times R_m\right) \times \left(\sum_{l=1}^L S_{kl} \times (1 - \theta_l)\right)\right)$
-

4.2 Population initialization

In order to adequately cover the feasible domain, it was decided to use both randomization methods and heuristics to initialize the population. The randomization method (RS) allows the subplot sizes to take arbitrary values within the constraints and aims to allow the population to adequately cover the feasible domain. The pseudo-code for the randomization method is shown in Algorithm 2.

Algorithm 2 The pseudo-code for the randomization method

Input: number of individuals $NIND1, L, B_l, Q_l, K$

Output: *Population1*

- (1) **For** $i \leftarrow 1$ **to** $NIND1$ **do**
 - (2) **For** $l \leftarrow 1$ **to** L **do**
 - (3) $random_numbers \leftarrow \text{rand}(B_l) / \text{sum}(\text{rand}(B_l))$ (generate A random numbers that sum to 1)
 - (4) $Q_{S_l} \leftarrow \text{round}(Q_l \times \text{random_numbers})$ ($\text{round}()$ is a rounding function)
 - (5) **If** $Q_l < \text{sum}(Q_{S_l})$
 - (6) $Q_{S_l}(\text{randi}([1, Q_l])) \leftarrow Q_{S_l}(\text{randi}([1, Q_l])) - (\text{sum}(Q_{S_l}) - Q_l)$ ($\text{randi}([1, Q_l])$ can generate random integers that lie in the closed interval from 1 to Q_l)
 - (7) **Else**
 - (8) $Q_{S_l}(\text{randi}([1, Q_l])) \leftarrow Q_{S_l}(\text{randi}([1, Q_l])) + Q_l - \text{sum}(Q_{S_l})$
 - (9) **End**
 - (10) $col_index \leftarrow -1$
 - (11) $Population1(i, col_index: col_index + B_l - 1) \leftarrow Q_{S_l}$
 - (12) $col_index \leftarrow col_index + B_l$
 - (13) $individual_X_s \leftarrow []$
 - (14) randomize the machining sequence of remanufacturable parts (*seq*) within constraints
 - (15) generate X_s based on *seq*
 - (16) $individual_X_s \leftarrow [individual_X_s \text{ seq}]$
 - (17) **End**
 - (18) $Population1(i, K + 1: 2K) \leftarrow individual_X_s$
 - (19) **End**
-

The heuristic approach consists of two strategies. The first strategy is the balanced subplot size heuristic (BHS), which equalizes

the size of sublots belonging to the same quality level relative to each other so as to achieve a balanced machine load in order to reduce the completion time and recall cost. The pseudo-code and randomization methods under this strategy are roughly the same and will not be repeated here. The second strategy is the reduced number of sublots strategy (RHS), which is expected to reduce the adjustment time and thus the total completion time. The pseudo-code of the heuristics under the second strategy is shown in Algorithm 3.

Algorithm 3 RHS initialization population pseudocode

 Input: number of individuals $NIND3$, L , B_l , Q_l , K

 Output: $Population3$

```

(1)   For  $i \leftarrow 1$  to  $NIND3$  do
(2)   For  $l \leftarrow 1$  to  $L$  do
(3)     Generate a random integer no_zero_num that lies in the  $[1, B_l - 1]$  closed interval
(4)      $Q_{S_l} \leftarrow \text{floor}(Q_l/\text{no\_zero\_num})$  (floor() is a downward rounding function)
(5)     col_index  $\leftarrow 1$ 
(6)     If  $Q_{S_l} \times \text{no\_zero\_num} < Q_l$ 
(7)        $Population3(i, \text{col\_index} : \text{col\_index} + \text{no\_zero\_num}) \leftarrow Q_{S_l}$ 
(8)       col_index  $\leftarrow \text{col\_index} + B_l$ 
(9)     Else
(10)       $z \leftarrow Q_l - (\text{no\_zero\_num} - 1) \times Q_{S_l}$ 
(11)       $Population3(i, \text{col\_index} : \text{col\_index} + \text{no\_zero\_num}) \leftarrow Q_{S_l}$ 
(12)       $Population3(i, : \text{col\_index} + B_l - 2 : \text{col\_index} + B_l - 1) \leftarrow z$ 
(13)      col_index  $\leftarrow \text{col\_index} + B_l$ 
(14)    End
(15)  End
(16)  individual_ $X_s \leftarrow []$ 
(17)  randomize the machining sequence of remanufacturable parts ( $seq$ ) within constraints
(18)  generate  $X_s$  based on  $seq$ 
(19)  individual_ $X_s \leftarrow [\text{individual\_}X_s \text{ } seq]$ 
(20)   $Population3(i, K + 1 : 2K) \leftarrow \text{individual\_}X_s$ 
(21)  End
  
```

4.3 Adaptation value assignment strategies

Currently, common adaptation value assignment strategies include aggregation-based strategies (a in Fig. 6), criterion-based strategies (b in Fig. 6), and Pareto dominance-based strategies (c in Fig. 6).

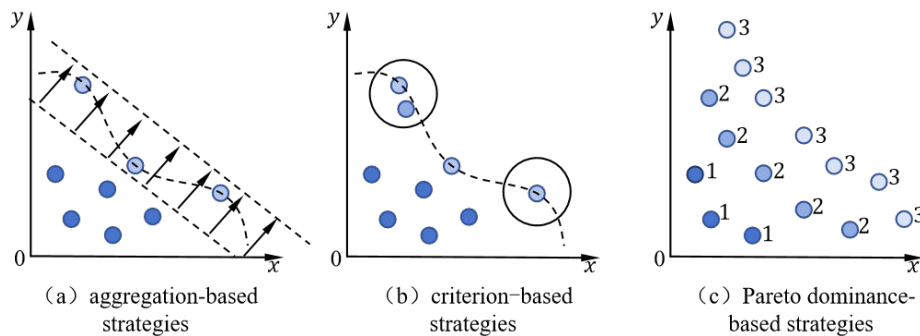


Fig. 6 Common adaptation value assignment strategies

The strategy based on Pareto domination achieves the classification of solutions based on the dominance relationship between them, and this type of adaptive value assignment strategy is especially suitable for the optimization of discrete combinatorial problems, and it is also inherent to the non-dominated sorting genetic algorithms, so the Pareto domination strategy is chosen. The Pareto dominance based strategy is executed as follows: where P denotes the population, P_r denotes the subpopulation, S_p is the set dominated by individual p , and n_p denotes the total number of individuals that dominate individual p .

Step 1: $\forall p, q \in P$, $\text{if } p < q$, $S_p = S_p \cup \{q\}$; if $q < p$, $n_p = n_p + 1$; if $n_p = 0$, $p_{rank} = 1$, $P_1 = P_1 \cup \{p\}$;

Step 2: $i = 1$, if $P_i \neq \emptyset$, assume $Z \neq \emptyset$; $\forall q \in S_p$, $n_p = n_p - 1$, if $n_q = 0$, $q_{rank} = i + 1$; $Z = Z \cup \{q\}$, $i = i + 1$, $P_i = Z$;

Step 3: if $P_i = \emptyset$, stop. Otherwise go to step 2.

4.4 Population diversity maintenance strategies

Currently, three common population diversity maintenance strategies exist, based on the kernel function (a in Fig. 7), based on the number of neighborhood solutions (b in Fig. 7), and based on the number of grids (c in Fig. 7).

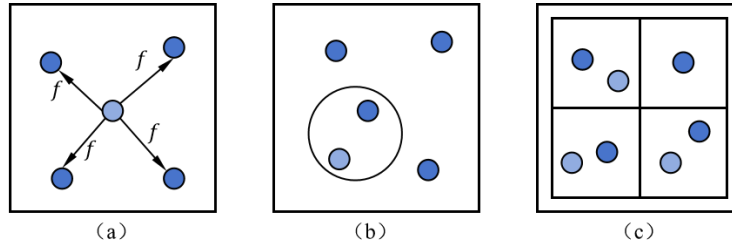


Fig. 7. diversity maintenance strategies

In this paper, we decided to use crowding distance and dynamic genetic operations to maintain the diversity of the population. Dynamic genetic manipulation includes dynamic crossover operation and mutation operation. Regarding the dynamic crossover operation, one means that the two individuals performing the crossover operation can be acquired by roulette selection among the parent individuals, or they can come from roulette selection and the memory bank, respectively, and the specific source of the individuals is controlled using the dynamic crossover probability; Secondly, two kinds of crossover operators are designed, the first kind of crossover operator is able to change most of the gene information in an individual, this kind of crossover operator is used in the first and middle period of the iteration, and the other kind of crossover operator is able to change only a small portion of the gene information in an individual, this kind of crossover operator is used in the middle and late period of the iteration. Regarding the dynamic variational operation, two variational operators, random step and fixed step, are designed, which are also used in different periods of population iteration, respectively. The calculation steps regarding the crowding distance are as follows:

Step 1: Initialize the crowding distance $d[i] = 0$ of individual i . Rank the sub-objective function values of individual i ;
 Step 2: Assign values to the boundary points and calculate the congestion distance from individual 2 to individual (N-1) with the following equation:

$$d[0] = d[N] = 0 \tag{21}$$

$$d[i] = d[i] + (d[i + 1].m - d[i - 1].m) / (f_m^{max} - f_m^{min}) \tag{22}$$

4.5 Crossover operator

According to the coding characteristics of the problem under study, the crossover operator performs crossover operations only for the individual X_s part. By improving the OX crossover operator, two crossover operators were obtained. Fig. 8 and Fig. 9 show the schematic diagrams of crossover operator 1 and crossover operator 2, respectively. In Fig. 8, the parent individual X_s part consists of 9 sublots, and the remanufacturable parts in these sublots can be categorized into 3 quality classes. Among them, sublots 1 and 2 are in one category, sublots 3, 4, 5 and 6 are in one category, and sublots 7, 8 and 9 are in one category. The process of crossover operator 1 is summarized in conjunction with Fig. 8. First intercept all subplot fragments corresponding to the quality class to which the first gene in the parent1_ X_s part belongs, denoted as part_parent1_ X_s , i.e., sublots 1 and 2, and place the genes in part_parent1_ X_s at the end of offspring1_ X_s after randomly disrupting the intrinsic order; After that, all the genes except part_parent1_ X_s are obtained from the parent1_ X_s part and placed into the null genes of offspring1_ X_s according to the intrinsic order of these genes in parent2_ X_s , finally generating offspring1_ X_s . Following the same method one can obtain offspring2_ X_s .

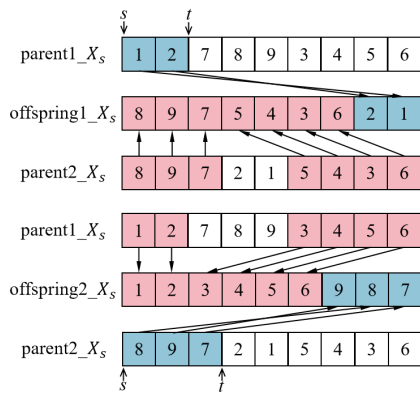


Fig. 8. Schematic diagram of crossover operator 1

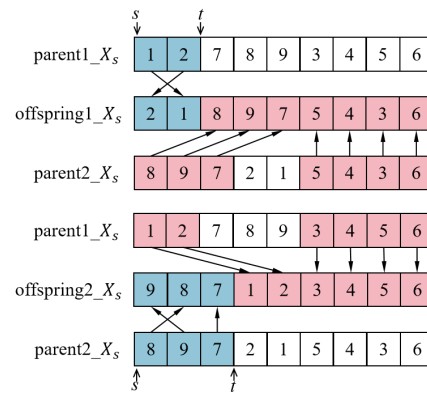


Fig. 9. Schematic diagram of crossover operator 2

Crossover operator 2 is mostly the same as crossover operator 1, except that part_parent1_X_s needs to be placed in the forward sequence in offspring1_X_s, see Fig. 9. From the corresponding descriptions, it can be seen that the offspring chromosomes produced by both crossover operators are feasible solutions. The crossover operator 1 is able to adjust the gene sequence of the parent in a wide range, which is favorable for the improvement of the global search ability of the algorithm, and is suitable to be used in the early stage of the algorithm iteration. On the other hand, crossover operator 2 adjusts the parent gene sequences in a small range, which facilitates the deep mining of the ideal solution attachment region, and is suitable for use in the later iterations of the algorithm. Therefore, crossover operator 1 is used if the current number of iterations is less than $G_{max}/2$, otherwise crossover operator 2 is used, and G_{max} is the maximum number of iterations.

4.6 Mutation operator

Mutation operators form new chromosomes by changing the genetic information of the chromosomes. According to the coding characteristics of the problem under study, the mutation operator performs the mutation operation only for the individual X_q part. Based on the reference of existing mutation operators, two types of mutation operators, random step size and fixed step size, are designed, which are shown in Fig. 10 and Fig. 11, respectively. In Fig. 10, the parent X_s section consists of nine sublots, and the information on the quality level to which the sublots belong is the same as in Fig. 8. The sizes of sublots 1 through 9 are shown in the parent_X_q section of Fig. 10. The random step size mutation operator is outlined in conjunction with Fig. 10. First, the intrinsic order of the genes in parent_X_q is disrupted to obtain parent*_X_q while following the subplot size constraints; after that, the random step size for the subplot size to which each quality level belongs is determined, and offspring_X_q is obtained by adding the variation step to each locus of the genes in parent*_X_q.

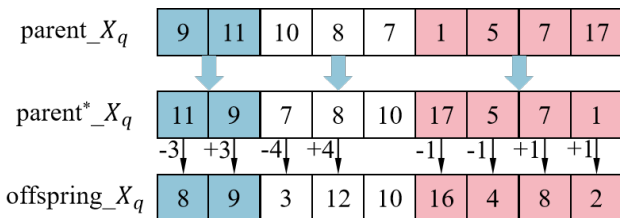


Fig. 10. The random step size mutation operator

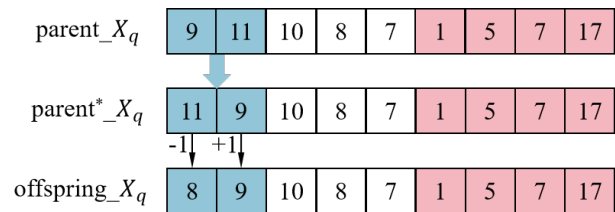


Fig. 11. The fixed step size mutation operator

Both mutation operators produce offspring that are feasible solutions. The random step size mutation operator is able to change the parent genes in a wide range, which is favorable for the global search ability of the algorithm and suitable for use early in the algorithm iteration. The fixed step size mutation operator changes the parent gene sequence in a small range, which facilitates deep exploration of the ideal solution attachment region and is suitable for use in the later stages of algorithm iteration. Therefore, if the current iteration count is less than $G_{max}/2$, use a random step size mutation operator, otherwise use a fixed step size mutation operator.

4.7 Elite retention strategies

At present, there are generally two elite retention strategies in multi-objective population intelligent optimization algorithms, and the two strategies are shown in (a) and (b) in Fig. 12, respectively. Based on the above two strategies, the Memory Bank II elite retention strategy is designed, see (c) in Fig. 12. The biggest difference with Memory Bank I is to make full use of the excellent genetic information contained in the non-inferior solutions in the memory bank to intervene in the generation of the offspring population in order to speed up the algorithm search efficiency, i.e., in the crossover operation, the decision of whether or not to introduce the excellent individuals in the memory bank is made on the basis of the dynamic crossover probability. In terms of memory bank updating, the memory bank absorbs the non-inferior solutions produced in each generation and achieves the selection of superior individuals in evolution by comparing the dominance relationships between the new solutions and those in the memory bank.

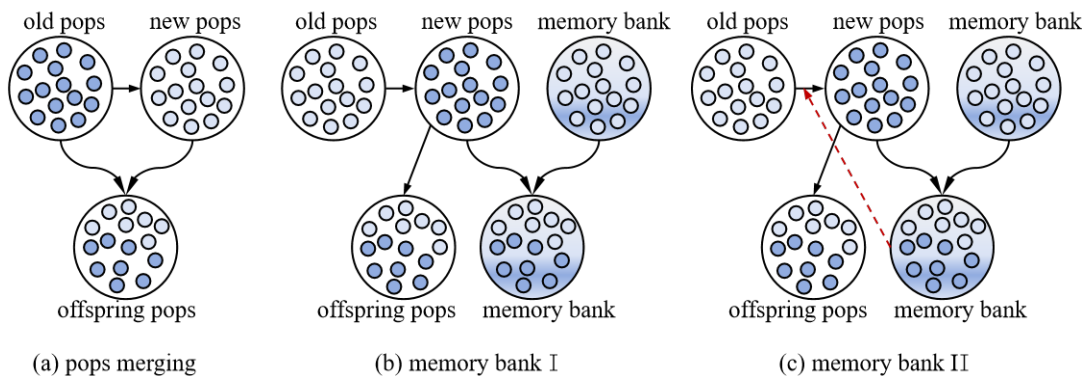


Fig. 12. Elite retention strategies

4.8 Algorithm flow

The INSGA-II algorithm flow is as follows:

Step 1: Initialize parameters: parameters include $L, Q_l, B_l, P_l, I, M_i, u_m, R_m, \delta_m, \alpha, \beta, T_{ikm}, T_{iklkm}$, population size N , maximum number of iterations G_{max} , crossover probability P_c , mutation probability P_m , ratio of the number of individuals generated by RS, BHS, and RHS strategies to the population P_{RS}, P_{BHS} , and P_{RHS} .

Step 2: Initialize the population using RS, BHS and RHS and calculate the multi-objective function values for each individual, perform non-dominated sorting based on the objective function values, and introduce non-inferior solutions into the memory bank.

Step 3: Compare the dominance relationship between the non-inferior solutions in the population and the non-inferior solutions in the memory bank and update the memory bank.

Step 4: Determine whether the number of iterations G_{cur} exceeds the maximum allowable number of iterations, if it has exceeded, stop the iteration and return the optimal solution; otherwise, go to step 5.

Step 5: Determine whether G_{cur} is greater than $G_{max}/2$. If it is not satisfied, perform genetic operation using crossover operator 1 and random step size mutation operator 1, respectively, and go to step 6; otherwise, perform genetic operation using crossover operator 2 and fixed step size mutation operator 2, respectively, and go to step 7.

Step 6: Initialize the crossover operation and mutation operation population storage templates. The upper limit of the number of crossover operations to be performed is $N \times P_c$. At each execution of a crossover operation, a random number located in the interval 0-1 is first generated, and if the random number is greater than P_c , two individuals are first randomly selected from the old population to perform the crossover operation. After that, one individual from the old population is randomly selected to perform the mutation operation. Then turn to step 8.

Step 7: Initialize the crossover operation and mutation operation population storage templates. The upper limit of the number of crossover operations to be performed is $N \times P_c$. At each execution of a crossover operation, a random number located in the interval 0-1 is first generated, and if the random number is greater than P_c , then first an individual is randomly selected from the old population and an individual is randomly selected from the memory bank to perform the crossover operation. After that an individual is randomly selected from the old population to perform the mutation operation. Then turn to step 8.

Step 8: Merge the old and new populations. Perform non-dominated sorting, congestion calculation and sorting on the merged population and select the top N individuals for the next round of iteration.

Step 9: Go to step 3.

5. Simulation experiments and analysis

In this section, simulation experiments are conducted to verify the effectiveness of the proposed algorithm. Both the associated algorithms and numerical experiments are programmed using MATLAB R2022b software and are executed on an Intel(R) Core(TM) i7-8565U (1.80GHz/8192MB RAM) PC with Windows 11 operating system.

5.1 Case generation and parameterization

The lot streaming scheduling problem of remanufacturing shop considering recall cost studied in this paper does not yet have a benchmarking arithmetic example that can be directly borrowed, therefore, based on the reference of "China's End-of-Life Vehicle Dismantling and Recycling Industry Development Prospect and Investment Strategy Consulting Report", "China's Waste Electrical and Electronic Equipment Recycling, Treatment, and Comprehensive Utilization Industry White Paper 2023", and related literatures (Chen et al., 2020; Sang et al., 2018), a large number of experimental arithmetic examples were generated. The experimental algorithm consists of nine main metrics, as shown in Table 3, where there are six quality classes, and the remanufacturable parts with quality class 1 have the best quality. The processing time is related to the quality level and is expressed as a linear function of l . In Table 3, the values of and are given according to the corresponding quality levels, as follows. In Table 3, the values of P_l and θ_l are given according to the corresponding quality level, e.g., 0.9635 is the reliability of the remanufactured part corresponding to quality level 1.

Table 3

Parameter settings for experimental algorithms

Symbol	Definition	Value
L	Total number of quality grades	1, 2, 3, 4, 5, 6
Q_l	Total number of remanufacturable parts	100, 200, 300
P_l	Unit selling price of remanufactured parts	10, 7, 6, 5.4, 5.2, 5
B_l	Maximum number of sublots	5, 10, 15, 20, 30
θ_l	Reliability of remanufactured parts	0.9635, 0.9486, 0.9317, 0.9214, 0.9136, 0.9021
I	Total number of processes	2, 3, 4
M_i	Number of machines available	3, 5, 8
u_m	Machine age (year)	[1, 10]
T_{ikm}	Processing time (min)	$[l \times 0.4 + 2, l \times 0.4 + 2.2]$
T_{iklkm}	Adjustment time (min)	[8, 11]

In order to validate the effectiveness of INSGA-II, five multi-objective algorithms in the literature, namely GASA (Z. Zhang et al., 2023), HPSO (Ma et al., 2023), MOACA (Amer et al., 2024), HABC (Zhang et al., 2024), and MOGWA (J. Li et al., 2023), are compared. The maximum number of iterations for each algorithm is capped at 500, and for each algorithm, the

algorithms are run independently for 20 times to obtain the final average. GASA, HPSO and HABC all obtain the solution by weighted summation, while the other algorithms obtain the Pareto solution by comparison of dominance relations. Based on a large number of preliminary experimental tests, the relevant parameters of the INSGA_II algorithm were set as follows: $P_c=0.6, P_m=0.1, P_{RS}=0.8, P_{BHS}=0.15, P_{RHS}=0.05, N=100$. Other algorithm parameter settings are shown in Table 4. where each algorithm takes the same value for the overlapping part of the same INSGA_II parameter.

Table 4
Parameter settings for each algorithm

Algorithm	Sysbol and value
GASA	Maximum number of iterations in the outer loop: 2000
	Maximum number of iterations of the inner loop: 300
	Initial temperature: $t=1000$
	Cooling factor : $\alpha=0.99$
	Probability of choosing the swap neighborhood structure: $p_{\text{swap}}=0.2$
	Probability of choosing the insertion neighborhood structure: $p_{\text{insertion}}=0.5$
HPSO	Probability of choosing the reversion neighborhood structure: $p_{\text{reversion}}=0.3$
	Population size: $N=100$
	Inertia weight: $w \in [0.8, 1.2]$
	Acceleration constant: $c_1=c_2=1.5$
MOACA	$N=50$
	Pheromone constants: $Q=10$
	Pheromone importance factor: $\alpha=1$
	Pheromone volatilization factor: $\rho=0.5$
HABC	Heuristic function factor: $\beta=5$
	$N=50$
MOGWA	Control parameter: $\text{limit}=100$
	$N=50$

5.2 Algorithm comparison

This section elects the Inverse Generation Distance (IGD) metric to evaluate the algorithms. IGD belongs to the comprehensive performance evaluation metrics, which are used to evaluate the convergence as well as the distributivity of the multi-objective optimization algorithms. It is used to evaluate the distance of the interval between PF_{true} and PF_{obtain} . The smaller the IGD, the better the convergence as well as the distribution of the algorithm. The IGD calculation formula is shown in equation (23). Where N^* denotes the number of optimal solutions contained in PF_{true} , and d_i^* denotes the Euclidean distance between the i th point on PF_{true} to the nearest point on PF_{obtain} .

$$IGD = \frac{1}{N^*} \sqrt{\sum_{i=1}^{N^*} d_i^{*2}} \tag{23}$$

The performance and effectiveness of the algorithms are analyzed by comparing the IGD values of INSGA-II, GASA, HPSO, MOACA, HABC, and MOGWA under 21 cases of algorithms. For each example, the algorithms were run independently 20 times to obtain the average value of IGD. The results are shown in Table 5, with the optimal evaluation metrics under each algorithm shown in bold. INSGA-II possesses the smallest value of IGD under almost all the arithmetic cases, which indicates that INSGA-II has strong optimization seeking ability. The values of IGD do not increase as the parameter size of the arithmetic cases increases, indicating that INSGA-II is also suitable for solving large-scale problems. In summary, INSGA_II is effective in solving the remanufacturing shop batch flow scheduling problem.

Table 5
Comparison of IGD indicator values

Serial No.	L	Q_i	I	M_i	B_i	GASA	HPSO	MOACA	HABC	MOGWA	INSGA II
C1	3	100	2	3	5	0.1449	0.0887	0.0667	0.1525	0.1359	0.0574
C2	3	100	2	3	10	0.1828	0.1265	0.0687	0.1904	0.1737	0.0594
C3	3	100	2	3	15	0.1821	0.1259	0.0555	0.1897	0.1731	0.0462
C4	3	100	2	5	10	0.1568	0.1006	0.0636	0.1644	0.1478	0.0544
C5	3	200	2	8	10	0.1825	0.1263	0.0329	0.1902	0.1735	0.0236
C6	3	200	3	5	20	0.1825	0.1263	0.0327	0.1901	0.1735	0.0382
C7	3	300	4	8	30	0.1582	0.1020	0.0687	0.1659	0.1492	0.0594
C8	4	100	2	5	15	0.1560	0.0998	0.0412	0.1636	0.1470	0.0319
C9	4	100	3	5	15	0.1692	0.1130	0.0419	0.1768	0.1602	0.0326
C10	4	100	4	5	15	0.1802	0.1240	0.0452	0.1878	0.1711	0.0359
C11	4	200	2	3	5	0.1583	0.1020	0.0689	0.1659	0.1492	0.0596
C12	4	200	4	5	15	0.1641	0.1079	0.0454	0.1717	0.1550	0.0361
C13	4	300	4	5	15	0.1687	0.1125	0.0507	0.1764	0.1597	0.0415
C14	4	300	4	8	30	0.1510	0.0948	0.0411	0.1587	0.1420	0.0318
C15	6	100	2	3	5	0.1785	0.1223	0.0473	0.1861	0.1695	0.0380
C16	6	200	3	5	10	0.1754	0.1192	0.0474	0.1830	0.1664	0.0381
C17	6	200	3	3	15	0.1472	0.0910	0.0466	0.1548	0.1382	0.0373
C18	6	200	3	5	20	0.1520	0.0958	0.0589	0.1596	0.1430	0.0496
C19	6	300	4	8	20	0.1604	0.1042	0.0562	0.1680	0.1514	0.0469
C20	6	200	3	5	30	0.1656	0.1094	0.0635	0.1732	0.1566	0.0542
C21	6	300	4	8	30	0.1459	0.0897	0.0369	0.1535	0.1369	0.0276

The corresponding PF_{obtain} for each algorithm under the C1 case is shown in Fig. 13, and the red "☆" indicates the non-inferior solutions obtained by INSGA-II. It can be clearly seen that the non-inferior solutions obtained by INSGA-II are better in all objective values and have the largest number of non-inferior solutions, which again proves the effectiveness of INSGA-II in solving the lot streaming scheduling problem of remanufacturing shops.

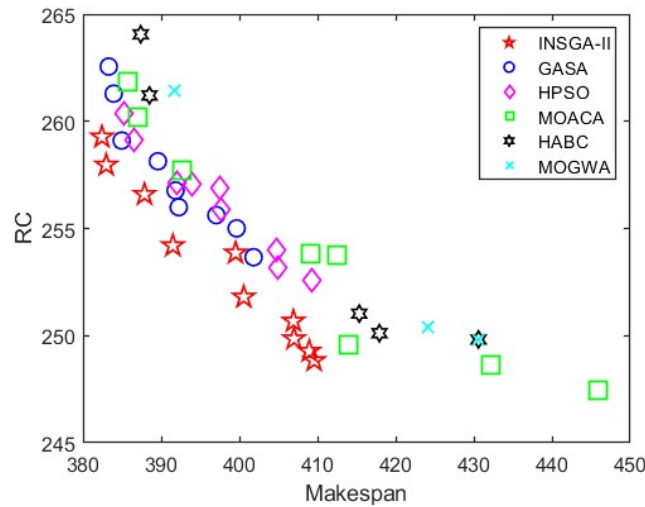


Fig. 13 PF_{obtain} for each algorithm under the C1 calculus

5.3 Case study

The proposed INSGA-II is also used to solve the problem of remanufacturing scheduling of printed circuit boards (PCB), which are the core components of electronic devices and are responsible for the electrical connections between electronic components. PCBs are widely used in a variety of products, such as computers, telecommunication devices, household appliances, and automotive electronics. With the widespread popularity of electronic products and the accelerated pace of replacement, remanufacturable PCB have increased dramatically. At the same time, environmental protection and resource reuse and other needs have prompted the rise of PCB remanufacturing industry. Enterprises will be based on the proportion of PCB components destroyed and the severity of damage to the substrate will be graded for quality, different quality levels corresponding to the remanufacturing of PCB products sold at different prices, and the processing time of the various processes are different, the general quality of the better grade of the processing time required is shorter. As an example, the remanufacturable FR-4 double-sided fiberglass PCB remanufacturing process is shown in Fig. 14, and the related processing parameters are shown in Tables 7-10. The proposed INSGA-II algorithms are again utilized to solve this problem, in which each of the algorithms is executed independently for 20 times.

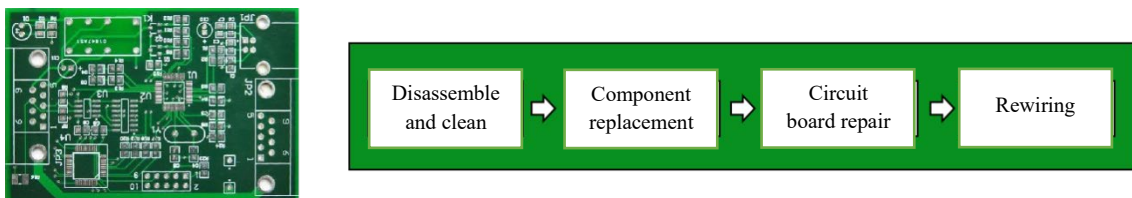


Fig. 14 PCB remanufacturing process

PF_{obtain} of each algorithm is shown in Fig. 15. From the figure, it can be seen that INSGA-II acquires 15 Pareto solutions, and although GASA and HPSO are closer to INSGA-II in the number of Pareto solutions acquired, both of them are not as good as INSGA-II in terms of the quality of solutions, which suggests that INSGA-II has a stronger capability of searching for excellence. In Fig. 15, points A and B correspond to the optimal completion time and recall cost, respectively. point C is an example solution that combines each optimization objective. points A, B, and C correspond to the objective values of (586.3960,1297.0343), (664.4263,1234.6022), and (600.0976,1256.9166), respectively. points A, B, and C points corresponding to the solutions are shown in Table 6.

Table 6

Solutions corresponding to points A, B and C

A	[15 9 24 8 2 12 8 8 6 8 9 12 12 15 10 11 10 7 9 15 5 9 16 19 14 14 14 6 1 17 7 5 8 1 2 9 4 6 3 10 28 23 29 26 25 22 24 27 30 21 16 14 12 20 18 17 11 13 15 19]
B	[9 12 0 2 14 14 13 14 12 10 12 16 18 9 1 7 24 7 10 6 11 6 6 9 0 11 9 14 29 20 26 22 28 25 21 27 30 24 29 23 8 6 4 5 3 9 1 2 7 10 11 14 13 19 17 15 16 20 18 12]
C	[15 9 24 8 2 12 8 8 6 8 9 12 12 15 10 11 10 7 9 15 5 9 16 19 14 14 14 6 1 17 10 7 3 4 2 6 1 5 8 9 28 23 29 26 25 22 24 27 30 21 16 14 12 20 18 17 11 13 15 19]

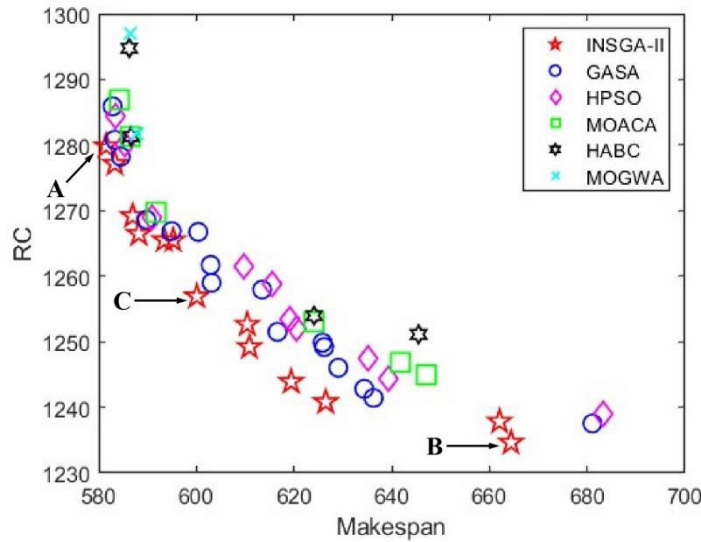


Fig. 15. PF_{obtain} for each algorithm

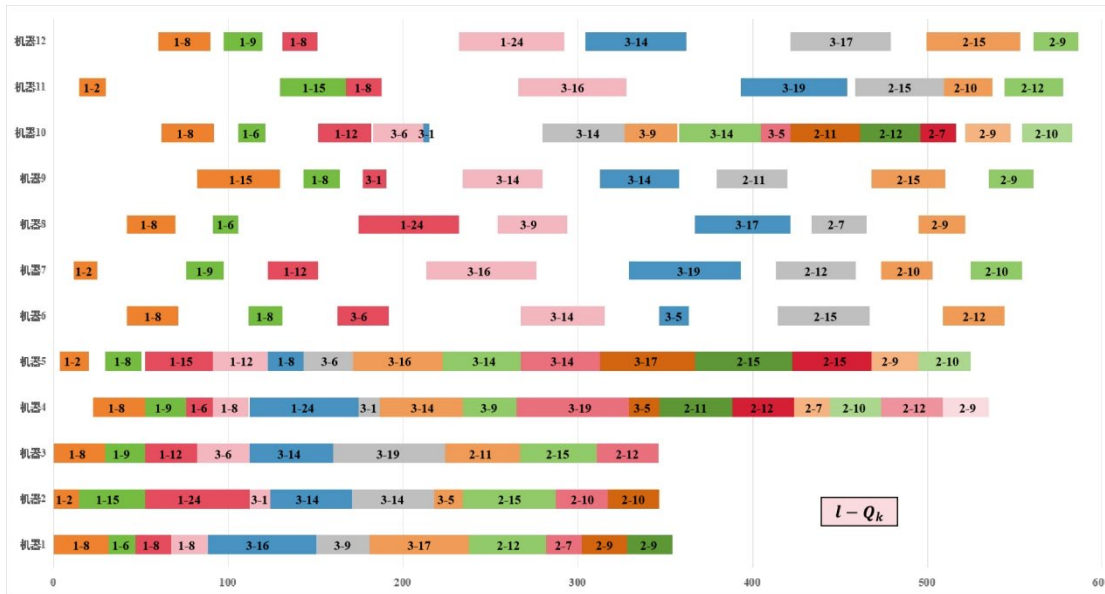


Fig. 16. Gantt chart of corresponding scheduling at point A

The reason point A has the optimal completion time is that the better performing machine in each process will take on more remanufacturable PCB component remanufacturing tasks. In Fig. 16, there is a clear difference in the remanufacturing tasks undertaken by machine 4 and machine 5, which are bottleneck devices. The number of remanufacturable PCB components processed by Machine 4 and Machine 5 with quality ratings of 1, 2 and 3 are 55, 61, 48 and 45, 49, 67, respectively. In Table 9, machine 4 performs better than machine 5 in processing remanufacturable PCB components of quality classes 1 and 2, and performs worse than machine 5 in processing remanufacturable PCB components of quality class 3. Therefore, machine 4 processes more remanufacturable PCB components of quality classes 1 and 2, and machine 5 processes more remanufacturable PCB components of quality class 3.

Point B has the optimal recall cost because it allows more reliable machines to take on the task of remanufacturing more remanufacturable PCB components with better quality grades in each step of the process. In Table 7, the reliability of each machine is ranked as follows: machine 2=machine 3=machine 4=machine 5=machine 11>machine 6=machine 8=machine 10>machine 1>machine 9>machine 7=machine 12. In FIG. 17, the number of remanufacturable PCB components of quality class 1 processed by machines 1 through 3 are 19, 43, and 38, respectively, with machines 2 and 3 processing significantly more than machine 1. This is also a good example of Eq. (7b), which states that the main variables affecting recall costs are machine reliability, selling price of remanufacturable parts, and subplot size.

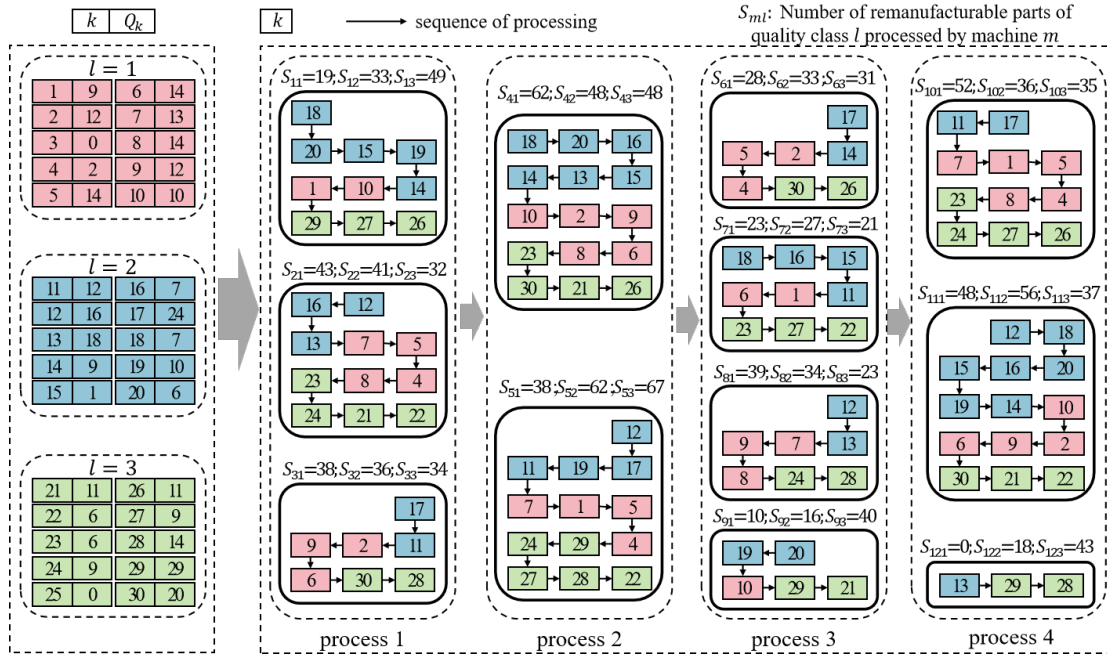


Fig. 17. Corresponding scheduling scheme for point B

Point C is obtained by weighing the propensity scheduling factors of points A and B on a particular optimization objective. In FIG. 18, the gap in the amount of remanufacturable PCB component remanufacturing tasks undertaken by the better-performing machine and the lesser-performing machine is mitigated, and the amount of remanufacturable PCB component remanufacturing tasks undertaken by the more reliable machine with the better quality rating is also reduced.

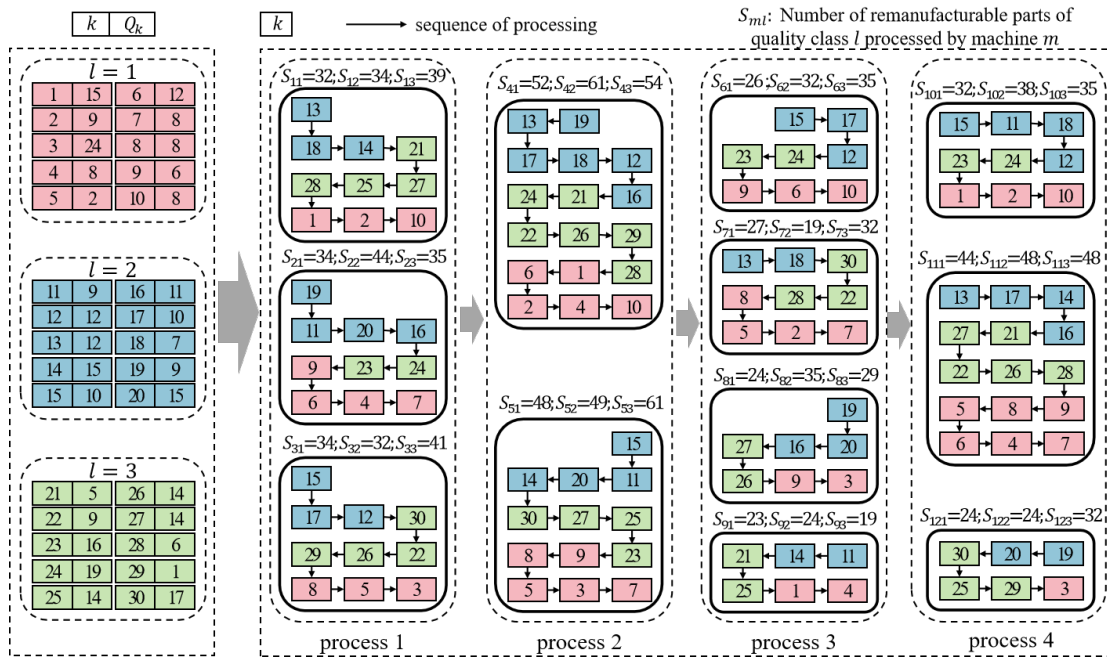


Fig. 18. Corresponding scheduling scheme for point C

Table 7

Machine reliability

Machine number	1	2	3	4	5	6
Process	1	1	1	2	2	3
machine age	6	2	2	3	3	5
Reliability	0.9418	0.9802	0.9802	0.9704	0.9704	0.9512
Machine number	7	8	9	10	11	12
Process	3	3	3	4	4	4
machine age	8	5	7	5	3	8
Reliability	0.9231	0.9512	0.9324	0.9512	0.9704	0.9231

Table 8

Remanufacturable FR-4 Double-Sided Fiberglass PCB Size, maximum number of sublots, unit sales price, and reliability

Quality level	1	2	3
size	100	110	115
maximum number of sublots	10	10	10
unit sales price (CNY)	22	20	16
reliability	0.9635	0.9486	0.9317

Table 9
Processing time

Quality level	process	Machine number	Processing time (min)	Quality level	process	Machine number	Processing time (min)
1	1	1	2.57	2	3	7	2.96
1	1	2	2.50	2	3	8	2.99
1	1	3	2.52	2	3	9	2.83
1	2	4	2.59	2	4	10	2.87
1	2	5	2.60	2	4	11	2.81
1	3	6	2.40	2	4	12	2.87
1	3	7	2.42	3	1	1	3.36
1	3	8	2.41	3	1	2	3.34
1	3	9	2.57	3	1	3	3.38
1	4	10	2.56	3	2	4	3.40
1	4	11	2.52	3	2	5	3.21
1	4	12	2.49	3	3	6	3.39
2	1	1	2.89	3	3	7	3.36
2	1	2	2.96	3	3	8	3.21
2	1	3	2.94	3	3	9	3.25
2	2	4	2.93	3	4	10	3.35
2	2	5	3.00	3	4	11	3.20
2	3	6	2.94	3	4	12	3.37

Table 10
Adjustment time

current job	previous job	process	Machine number	Adjustment time (min)	current job	previous job	process	Machine number	Adjustment time (min)
1	0	1	1	10.87	2	0	3	7	10.59
1	1	1	1	0.00	2	1	3	7	10.59
1	2	1	1	8.87	2	2	3	7	0.00
1	3	1	1	8.87	2	3	3	7	10.59
1	0	1	2	9.72	2	0	3	8	10.52
1	1	1	2	0.00	2	1	3	8	9.52
1	2	1	2	9.72	2	2	3	8	0.00
1	3	1	2	8.72	2	3	3	8	10.52
1	0	1	3	9.36	2	0	3	9	10.75
1	1	1	3	0.00	2	1	3	9	9.75
1	2	1	3	9.36	2	2	3	9	0.00
1	3	1	3	9.36	2	3	3	9	8.75
1	0	2	4	8.93	2	0	4	10	10.18
1	1	2	4	0.00	2	1	4	10	8.18
1	2	2	4	9.93	2	2	4	10	0.00
1	3	2	4	9.93	2	3	4	10	8.18
1	0	2	5	10.99	2	0	4	11	10.59
1	1	2	5	0.00	2	1	4	11	9.59
1	2	2	5	9.99	2	2	4	11	0.00
1	3	2	5	10.99	2	3	4	11	8.59
1	0	3	6	10.35	2	0	4	12	8.91
1	1	3	6	0.00	2	1	4	12	10.91
1	2	3	6	9.35	2	2	4	12	0.00
1	3	3	6	9.35	2	3	4	12	10.91
1	0	3	7	8.46	3	0	1	1	8.63
1	1	3	7	0.00	3	1	1	1	8.63
1	2	3	7	10.46	3	2	1	1	10.63
1	3	3	7	8.46	3	3	1	1	0.00
1	0	3	8	8.61	3	0	1	2	9.55
1	1	3	8	0.00	3	1	1	2	8.55
1	2	3	8	8.61	3	2	1	2	8.55
1	3	3	8	9.61	3	3	1	2	0.00
1	0	3	9	9.02	3	0	1	3	9.72
1	1	3	9	0.00	3	1	1	3	9.72
1	2	3	9	10.02	3	2	1	3	9.72
1	3	3	9	9.02	3	3	1	3	0.00
1	0	4	10	9.45	3	0	2	4	9.01
1	1	4	10	0.00	3	1	2	4	9.01
1	2	4	10	8.45	3	2	2	4	9.01
1	3	4	10	8.45	3	3	2	4	0.00
1	0	4	11	10.05	3	0	2	5	8.83
1	1	4	11	0.00	3	1	2	5	8.83

Table 10

Adjustment time (Continued)

current job	previous job	process	Machine number	Adjustment time (min)	current job	previous job	process	Machine number	Adjustment time (min)
1	2	4	11	8.05	3	2	2	5	9.83
1	3	4	11	8.05	3	3	2	5	0.00
1	0	4	12	10.00	3	0	3	6	10.58
1	1	4	12	0.00	3	1	3	6	8.58
1	2	4	12	8.00	3	2	3	6	8.58
1	3	4	12	9.00	3	3	3	6	0.00
2	0	1	1	8.62	3	0	3	7	8.36
2	1	1	1	9.62	3	1	3	7	9.36
2	2	1	1	0.00	3	2	3	7	9.36
2	3	1	1	9.62	3	3	3	7	0.00
2	0	1	2	10.80	3	0	3	8	9.77
2	1	1	2	10.80	3	1	3	8	10.77
2	2	1	2	0.00	3	2	3	8	8.77
2	3	1	2	8.80	3	3	3	8	0.00
2	0	1	3	8.33	3	0	3	9	8.85
2	1	1	3	10.33	3	1	3	9	9.85
2	2	1	3	0.00	3	2	3	9	10.85
2	3	1	3	10.33	3	3	3	9	0.00
2	0	2	4	10.83	3	0	4	10	8.80
2	1	2	4	9.83	3	1	4	10	8.80
2	2	2	4	0.00	3	2	4	10	9.80
2	3	2	4	9.83	3	3	4	10	0.00
2	0	2	5	9.88	3	0	4	11	10.21
2	1	2	5	9.88	3	1	4	11	10.21
2	2	2	5	0.00	3	2	4	11	9.21
2	3	2	5	10.88	3	3	4	11	0.00
2	0	3	6	8.48	3	0	4	12	10.69
2	1	3	6	10.48	3	1	4	12	10.69
2	2	3	6	0.00	3	2	4	12	9.69
2	3	3	6	8.48	3	3	4	12	0.00

6. Conclusions

In this paper, we study the problem of lot streaming scheduling in a remanufacturing shop with consistent sublots (RSLSS_NMRP), where mixed production is not allowed between sublots possessing different types of remanufacturable parts. First, a multi-objective mixed integer planning model is constructed with the objective of simultaneously optimizing completion time and recall cost. Second, an improved non-dominated sorting genetic algorithm (INSGA-II) is utilized to solve the model. Two vectors regarding subplot size allocation and subplot processing order determination together form a solution. In order to improve the quality of the solution, the algorithm uses a randomization strategy and two heuristics to initialize the population and introduces dynamic genetic operations to advance the population diversity. Thirdly, the effectiveness of INSGA-II is demonstrated by comparing the IGD values of INSGA-II with those of GASA, HPSO, MOACA, HABC, and MOGWA under 21 examples of algorithms. INSGA-II is also used to solve the actual Printed Circuit Board (PCB) remanufacturing scheduling problem.

Further research is needed in the future as follows: (1) to study the remanufacturing shop lot streaming scheduling problem with different subplot strategies for recall cost and time (S. Wang et al., 2019); (2) to study the remanufacturing shop lot streaming scheduling problem with mixed scheduling strategies for recall cost and time; and (3) to study the remanufacturing shop lot streaming scheduling problem with different production methods for recall cost and time (B. Zhang et al., 2023).

Declarations of competing interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant number 2021YFB3300900 and the National Natural Science Foundation of China under Grant number 72371087, 71531008, 72188101.

References

- Alfieri, A., Zhou, S., Scatamacchia, R., & van de Velde, S.L. (2021). Dynamic programming algorithms and Lagrangian lower bounds for a discrete lot streaming problem in a two-machine flow shop. *4OR-Q Journal of Operations Research*, 19, 265–288. <https://doi.org/10.1007/s10288-020-00449-8>
- Amer, D.A., Attiya, G., & Ziedan, I. (2024). An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing. *Cluster Computing-The Journal Of Networks Software Tools And Applications*, 27, 1799–1819. <https://doi.org/10.1007/s10586-023-04018-6>
- Bin, X., Xuexin, L., Jianping, Q., Jian, W., & Xiaoming, W. (2015). Establishment of materials batch mixing optimization model for traceability of fresh-cuts fruits and vegetables processing. *Transactions of the Chinese Society of Agricultural Engineering*, 31.

- Bożek, A., & Werner, F. (2018). Flexible job shop scheduling with lot streaming and subplot size optimisation. *International Journal of Production Research*, 56, 6391–6411. <https://doi.org/10.1080/00207543.2017.1346322>
- Chakaravarthy, G.V., Marimuthu, S., Ponnambalam, S.G., & Kanagaraj, G. (2014). Improved sheep flock heredity algorithm and artificial bee colony algorithm for scheduling m-machine flow shops lot streaming with equal size sub-lot problems. *International Journal of Production Research*, 52, 1509–1527. <https://doi.org/10.1080/00207543.2013.848304>
- Chakraborty, T., Mukherjee, A., & Chauhan, S. (2023). Should a powerful manufacturer collaborate with a risky supplier? Pre-recall vs. post-recall strategies in product harm crisis management. *Computers & Industrial Engineering*, 177. <https://doi.org/10.1016/j.cie.2023.109037>
- Chen, T.-L., Cheng, C.-Y., & Chou, Y.-H. (2020). Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. *Annals of Operations Research* 290, 813–836. <https://doi.org/10.1007/s10479-018-2969-x>
- Cheng, L., Tang, Q., & Zhang, L. (2024). Production costs and total completion time minimization for three-stage mixed-model assembly job shop scheduling with lot streaming and batch transfer. *Engineering Applications Of Artificial Intelligence*, 130. <https://doi.org/10.1016/j.engappai.2023.107729>
- Cheng, M., Mukherjee, N. j., & Sarin, S. C. (2013). A review of lot streaming. *International Journal of Production Research* 51, 7023–7046. <https://doi.org/10.1080/00207543.2013.774506>
- Dabbene, F., & Gay, P. (2011). Food traceability systems: Performance evaluation and optimization. *Computers and Electronics in Agriculture*, 75, 139–146. <https://doi.org/10.1016/j.compag.2010.10.009>
- Dabbene, F., Gay, P., & Tortia, C. (2014). Traceability issues in food supply chain management: A review. *Biosystems Engineering*, 120, 65–80. <https://doi.org/10.1016/j.biosystemseng.2013.09.006>
- Dupuy, C., Botta-Genoulaz, V., & Guinet, A. (2005). Batch dispersion model to optimise traceability in food industry. *Journal of Food Engineering*, 70, 333–339. <https://doi.org/10.1016/j.jfoodeng.2004.05.074>
- Fang, K., Luo, W., & Che, A. (2021). Speed scaling in two-machine lot-streaming flow shops with consistent sublots. *Journal of the Operational Research Society*, 72, 2429–2441. <https://doi.org/10.1080/01605682.2020.1796533>
- Ghazi, A., Karray, S., & Azad, N. (2023). Price and quality competition while envisioning a quality-related product recall. *European Journal Of Operational Research*, 311, 486–501. <https://doi.org/10.1016/j.ejor.2023.05.013>
- Gong, G., Deng, Q., Chiong, R., Gong, X., Huang, H., & Han, W. (2020). Remanufacturing-oriented process planning and scheduling: mathematical modelling and evolutionary optimisation. *International Journal of Production Research*, 58, 3781–3799. <https://doi.org/10.1080/00207543.2019.1634848>
- Gorton, A., & Stasiewicz, M. (2017). Twenty-Two Years of US Meat and Poultry Product Recalls: Implications for Food Safety and Food Waste. *Journal Of Food Protection*, 80, 674–684. <https://doi.org/10.4315/0362-028X.JFP-16-388>
- Guide, V.D.R., Jayaraman, V., & Srivastava, R. (1999). Production planning and control for remanufacturing: a state-of-the-art survey. *Robotics and Computer-Integrated Manufacturing*, 15, 221–230. [https://doi.org/10.1016/S0736-5845\(99\)00020-4](https://doi.org/10.1016/S0736-5845(99)00020-4)
- Han, Y., Li, J.-Q., Gong, D., & Sang, H. (2019). Multi-Objective Migrating Birds Optimization Algorithm for Stochastic Lot-Streaming Flow Shop Scheduling With Blocking. *IEEE Access*, 7, 5946–5962. <https://doi.org/10.1109/ACCESS.2018.2889373>
- Li, C., Han, Y., Zhang, B., Wang, Y., Li, J., & Gao, K. (2024). A novel collaborative iterative greedy algorithm for hybrid flowshop scheduling problem with batch processing machines and variable sublots. *International Journal of Production Research*, 62, 4076–4096. <https://doi.org/10.1080/00207543.2023.2253925>
- Li, J., Li, H., He, P., Xu, L., He, K., & Liu, S. (2023). Flexible Job Shop Scheduling Optimization for Green Manufacturing Based on Improved Multi-Objective Wolf Pack Algorithm. *Applied Sciences-Basel*, 13. <https://doi.org/10.3390/app13148535>
- Li, Y., Liao, C., Wang, L., & Xiao, Y. (2023). A Reinforcement Learning-Artificial Bee Colony algorithm for Flexible Job-shop Scheduling Problem with Lot Streaming. *Applied Soft Computing*, 146. <https://doi.org/10.1016/j.asoc.2023.110658>
- Lu, Y., Tang, Q., Pan, Q., Zhao, L., 2023. A Heuristic-Based Adaptive Iterated Greedy Algorithm for Lot-Streaming Hybrid Flow Shop Scheduling Problem with Consistent and Intermingled Sub-Lots. *Sensors* 23, 2808. <https://doi.org/10.3390/s23052808>
- Ma, X., Xu, X., Jiang, G., Qiao, Y., & Zhou, T. (2023). Hybrid adaptive particle swarm optimization algorithm for workflow scheduling. *Journal of Computer Applications*, 43, 474–483.
- Maity, M., Toloie, A., Sinha, A.K., & Tiwari, M.K. (2021). Stochastic batch dispersion model to optimize traceability and enhance transparency using Blockchain. *Computers & Industrial Engineering* 154, 107134. <https://doi.org/10.1016/j.cie.2021.107134>
- Mortezaei, N., & Zulkifli, N. (2014). A Study on Integration of Lot Sizing and Flow Shop Lot Streaming Problems. *Arabian Journal for Science and Engineering* 39, 9283–9300. <https://doi.org/10.1007/s13369-014-1416-9>
- Nejati, M., Mahdavi, I., Hassanzadeh, R., Mahdavi-Amiri, N., & Mojarad, M. (2014). Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint. *International Journal of Advanced Manufacturing Technology*, 70, 501–514. <https://doi.org/10.1007/s00170-013-5265-6>
- Pan, Y., Gao, K., Li, Z., & Wu, N., (2023a). Solving Biobjective Distributed Flow-Shop Scheduling Problems With Lot-Streaming Using an Improved Jaya Algorithm. *IEEE Transactions in Cybernetics*, 53, 3818–3828. <https://doi.org/10.1109/TCYB.2022.3164165>
- Pan, Y., Gao, K., Li, Z., & Wu, N. (2022). Improved meta-heuristics for solving distributed lot-streaming permutation flow shop scheduling problems. *IEEE Transactions on Automation Science and Engineering*, 20(1), 361–371.

- <https://doi.org/10.1109/TASE.2022.3151648>
- Qian, J., Dai, B., Wang, B., Zha, Y., & Song, Q. (2022). Traceability in food processing: problems, methods, and performance evaluations—a review. *Critical Reviews in Food Science and Nutrition* 62, 679–692. <https://doi.org/10.1080/10408398.2020.1825925>
- Reiter, S. (1966). A system for managing job shop production. *The Journal of Business* 39, 371–393.
- Ren, M., & Wang, G. (2024). A hybrid genetic algorithm with variable neighborhood search for batch dispersion problem to improve traceability. *International Journal Of Industrial Engineering Computations*, 15(1), 41–58. <https://doi.org/10.5267/j.ijiec.2023.12.002>
- Sang, H.-Y., Pan, Q.-K., Duan, P.-Y., & Li, J.-Q. (2018). An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. *Journal of Intellectual Manufacturing*, 29, 1337–1349. <https://doi.org/10.1007/s10845-015-1182-x>
- Shao, W., Shao, Z., & Pi, D. (2023). Lot sizing and scheduling problem in distributed heterogeneous hybrid flow shop and learning-driven iterated local search algorithm. *IEEE Transactions on Automation Science and Engineering*, 21(4), 6483–6497. <https://doi.org/10.1109/TASE.2023.3326301>
- Singh, S., Sarin, S. C., & Cheng, M. (2024). Single-lot, lot-streaming problem for a 1+ m hybrid flow shop. *Journal of Global Optimization*, 89(2), 435–455. <https://doi.org/10.1007/s10898-023-01354-0>
- Sun, H., Chen, W., Liu, B., & Chen, X. (2018). Economic lot scheduling problem in a remanufacturing system with returns at different quality grades. *Journal of Cleaner Production*, 170, 559–569. <https://doi.org/10.1016/j.jclepro.2017.09.184>
- Tang, J., Liu, G., & Pan, Q. (2021). A review on representative swarm intelligence algorithms for solving optimization problems: Applications and trends. *IEEE/CAA Journal of Automatica Sinica*, 8(10), 1627–1643. <https://doi.org/10.1109/JAS.2021.1004129>
- Tian, Z., Jiang, X., Tian, G., Li, Z., & Liu, W. (2023). Knowledge-based lot-splitting optimization method for flexible job shops considering energy consumption. *IEEE Transactions on Automation Science and Engineering*, 21(3), 4864–4875. <https://doi.org/10.1109/TASE.2023.3303915>
- Ventura, J.A., & Yoon, S.-H. (2013). A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. *Journal of Intelligent Manufacturing*, 24, 1185–1196. <https://doi.org/10.1007/s10845-012-0650-9>
- Vijaychakaravarthy, G., Marimuthu, S., & Naveen Sait, A. (2014). Comparison of Improved Sheep Flock Heredity Algorithm and Artificial Bee Colony Algorithm for lot Streaming in m-Machine Flow Shop Scheduling. *Arabian Journal for Science and Engineering*, 39, 4285–4300. <https://doi.org/10.1007/s13369-014-0994-x>
- Wang, H., Zhao, F., Gao, H., & Sutherland, J.W. (2019). A three-stage method with efficient calculation for lot streaming flow-shop scheduling. *Frontiers of Information Technology & Electronic Engineering*, 20, 1002–1020. <https://doi.org/10.1631/FITEE.1700457>
- Wang, S., Kurz, M., Mason, S.J., & Rashidi, E. (2019). Two-stage hybrid flow shop batching and lot streaming with variable sublots and sequence-dependent setups. *International Journal of Production Research*, 57, 6893–6907. <https://doi.org/10.1080/00207543.2019.1571251>
- Wang, Wenjie, Tian, G., Yuan, G., & Pham, D.T. (2023). Energy-time tradeoffs for remanufacturing system scheduling using an invasive weed optimization algorithm. *Journal of Intelligent Manufacturing*, 34, 1065–1083. <https://doi.org/10.1007/s10845-021-01837-5>
- Wang, W., Xu, Z., & Gu, X. (2022). A two-stage discrete water wave optimization algorithm for the flowshop lot-streaming scheduling problem with intermingling and variable lot sizes. *Knowledge-Based Systems*, 238. <https://doi.org/10.1016/j.knosys.2021.107874>
- Wang, W., Zhang, B., & Jia, B. (2023). A multiobjective optimization approach for multiobjective hybrid flowshop green scheduling with consistent sublots. *Sustainability*, 15(3), 2622. <https://doi.org/10.3390/su15032622>
- Wang, W., Zhang, B., Jiang, X., Jia, B., Sang, H., & Meng, L. (2024). Decomposition-based multi-objective approach for a green hybrid flowshop rescheduling problem with consistent sublots. *International Journal of Production Research*, 1–29. <https://doi.org/10.1080/00207543.2024.2333943>
- Yunusoglu, P., & Topaloglu Yildiz, S. (2023). Solving the flexible job shop scheduling and lot streaming problem with setup and transport resource constraints. *International Journal of Systems Science: Operations & Logistics*, 10, 2221072. <https://doi.org/10.1080/23302674.2023.2221072>
- Zhang, B., Pan, Q., & Li, J. (2022). An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots. *Knowledge-Based Systems*, 238. <https://doi.org/10.1016/j.knosys.2021.107874>
- Zhang, B., Pan, Q., Meng, L., Zhang, X., & Jiang, X. (2023). A decomposition-based multi-objective evolutionary algorithm for hybrid flowshop rescheduling problem with consistent sublots. *International Journal of Production Research*, 61, 1013–1038. <https://doi.org/10.1080/00207543.2022.2093680>
- Zhang, X., Sang, H., Li, Z., Zhang, B., & Meng, L. (2024). An efficient discrete artificial bee colony algorithm with dynamic calculation method for solving the AGV scheduling problem of delivery and pickup. *Complex & Intelligent Systems*, 10, 37–57. <https://doi.org/10.1007/s40747-023-01153-w>
- Zhang, Z., Shen, L., Gong, X., Zhong, X., & Yin, Y. (2023). A genetic-simulated annealing algorithm for stochastic seru scheduling problem with deterioration and learning effect. *Journal of Industrial and Production Engineering*, 40, 205–222. <https://doi.org/10.1080/21681015.2023.2167875>
- Zhao, J., Peng, S., Li, T., Lv, S., Li, M., & Zhang, H. (2019). Energy-aware fuzzy job-shop scheduling for engine remanufacturing at the multi-machine level. *Frontiers of Mechanical Engineering*, 14, 474–488.

<https://doi.org/10.1007/s11465-019-0560-z>

Zhi-hui, M., Xin-du, C., Chang-wei, H., & Zong-zhong, W. (2012). A Study on Raw Material Batch Mixing Problem for Optimal Quality Tracing of Shrimp Products. *Industrial Engineering Journal*, 15, 45.

Zhu, Y., Tang, Q., Cheng, L., Zhao, L., Jiang, G., Lu, Y., 2024. Solving multi-objective hybrid flowshop lot-streaming scheduling with consistent and limited sub-lots via a knowledge-based memetic algorithm. *Journal of manufacturing systems*, 73, 106–125. <https://doi.org/10.1016/j.jmsy.2024.01.006>



© 2025 by the authors; licensee Growing Science, Canada. This is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC-BY) license (<http://creativecommons.org/licenses/by/4.0/>).