

Article

Distributed Assembly Permutation Flowshop Scheduling with Capacity-Triggered Batch Transportation and Dedicated Closed-Loop Vehicles

Jingcao Cai ^{1,2,3,*}, Junkui Han ¹, Deming Lei ⁴, Lei Wang ^{1,*}, Yingjie Wang ¹, Duohao Geng ¹, Shuai Yang ¹, Mengrui Luo ¹, Chang Liu ¹, Xiangqiang Zhong ¹, Qi Hao ^{3,†}, Chuang An ⁵ and Chao Huang ⁶

¹ School of Mechanical and Automotive Engineering, Anhui Polytechnic University, Wuhu 241000, China; hanjunkui@foxmail.com (J.H.); wangyingjie@foxmail.com (Y.W.); dkamoxmu733944@foxmail.com (D.G.); yangshuai@foxmail.com (S.Y.); luomengrui@foxmail.com (M.L.); changliu2001@foxmail.com (C.L.); xqzh@nuaa.edu.cn (X.Z.)

² The Key Laboratory of Electric Drive and Control of Anhui Province, Anhui Polytechnic University, Wuhu 241000, China

³ Anhui Province Key Laboratory of Machine Vision Detection and Perception, Wuhu 241000, China; fm2359611157@foxmail.com

⁴ School of Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China; deminglei11@163.com

⁵ School of International Education, Hebei University of Technology, Tianjin 300400, China; chuang.an@student.lut.fi

⁶ Wuhu Industrial Innovation Center, Wuhu 241000, China; cycxzx@126.com

* Correspondence: caijingcao@foxmail.com (J.C.); wangdalei2000@126.com (L.W.)

† These authors contributed equally to this work.

Abstract

This paper investigates a distributed assembly permutation flowshop scheduling problem with first-in-first-out (FIFO) batch transportation triggered by vehicle capacity and heterogeneous dedicated vehicles operating in closed loops. Components are processed at multiple factories and transported in batches to a central assembly station, where a product can be assembled only after all of its required components have arrived. The objective is to minimize the makespan. Factory assignment and sequencing within each factory determine processing completion times and, through the prescribed transportation rules, affect batch departures, vehicle return times, component arrivals, product readiness, and assembly timing. To address these coupled temporal effects, a Timing Propagation Cooperative Population-Based Iterated Greedy algorithm (TPCPIG) is proposed. It combines multisource population construction guided by temporal features, hierarchical joint reinsertion of factory assignment and sequencing, and bilateral cross-factory cooperative reconstruction; all candidate solutions are evaluated by a unified schedule decoder. Computational experiments on 120 test instances show that TPCPIG achieves an overall average relative percentage deviation of 0.906%, compared with 3.006–8.607% for the four comparison algorithms, and obtains the lowest mean makespan for all tested job sizes from $n = 8$ onward. Further ablation experiments and analyses of search behavior clarify the respective roles of the three proposed mechanisms in population construction, evaluation effort, and adjustment across factories.

Keywords: distributed assembly permutation flowshop; dedicated closed-loop vehicles; capacity-triggered FIFO batch transportation; timing propagation; population-based iterated greedy; makespan



Academic Editor: Vasilis K. Oikonomou

Received: 10 September 2026

Revised: 26 September 2026

Accepted: 28 September 2026

Published: 30 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

In manufacturing systems with several factories, components of the same product may be processed at different factories and then transported to a central assembly station. In a permutation flowshop, the same job sequence is used on all machines [1]. In its distributed form, each job must be assigned to a factory and the jobs at each factory must be sequenced [2]. When a product requires several components, assembly can start only after all of them have arrived at the assembly station [3]. Thus, factory assignment and job sequencing affect final completion time not only through processing but also through the arrival of components needed for assembly.

When transport resources are considered, component arrival times depend on production schedules and transport operations. Limited vehicle capacity can affect delivery timing by requiring orders to be grouped into batches and served in multiple trips [4]. In distributed assembly scheduling, transportation capacity has been studied together with factory eligibility and setup times [5], and a fuzzy distributed assembly model has been developed to integrate processing, transportation, and assembly [6]. Recent studies have also examined batch delivery after assembly [7], transport resource scheduling [8], and the batching of completed jobs before assembly [9]. However, these works use vehicle settings, batching rules, and assembly arrangements that differ from those considered here [5–9]. This paper considers a distributed assembly permutation flowshop scheduling problem with first-in-first-out (FIFO) batch transportation triggered by vehicle capacity (CTBDAPFSP). Each factory has one dedicated vehicle; capacities and travel times may differ across factories, and every vehicle operates in a closed loop. Factory assignment and job sequencing affect processing completion times and, through the transport rules, batch formation, vehicle return times, component arrivals, product readiness, and assembly timing. These timing effects propagate from processing through transportation to assembly.

In CTBDAPFSP, the explicit scheduling decisions are factory assignment and job sequencing within each factory. Once these decisions are specified, batch formation, vehicle operations, product readiness, and assembly timing are determined by the operating rules. A change in assignment or sequencing therefore requires the resulting production, transportation, and assembly schedule to be recalculated. Iterated greedy methods that maintain a population use multiple candidate solutions and improve them through destruction, reconstruction, and local search [10,11]. In distributed permutation flowshop scheduling, removing and reinserting jobs can change both factory assignment and job sequencing [12]. This framework is therefore suitable for jointly exploring the two decisions with complete schedule evaluation. Based on this framework, we propose a Timing Propagation Cooperative Population-Based Iterated Greedy algorithm (TPCPIG). It uses information related to processing, transportation, and product readiness to build the initial population, applies hierarchical joint reinsertion to adjust factory assignment and job sequencing, and uses bilateral reconstruction across factories for larger structural changes. The same decoder is used to evaluate the complete production, transportation, and assembly schedule of every candidate solution.

The main contributions of this paper are summarized as follows:

1. A mathematical model is developed for a distributed assembly permutation flowshop scheduling problem with FIFO batch transportation triggered by vehicle capacity. Each factory has a dedicated vehicle operating in a closed loop, and vehicle capacities and travel times may differ across factories. The model minimizes the makespan and captures the timing relationships among processing, batch transportation, vehicle return times, product readiness, and assembly.
2. TPCPIG is developed for CTBDAPFSP. It combines population construction using information on processing, transportation, and product readiness; hierarchical joint

reinsertion of factory assignment and job sequencing; and bilateral reconstruction across factories. A single decoder evaluates the complete production, transportation, and assembly schedule for every candidate solution.

3. A diagnostic study of the individual mechanisms is conducted under a fixed central processing unit (CPU) time budget. Beyond solution comparisons and ablation tests, it quantifies the reduction in evaluation effort and the recall rate of hierarchical screening, the role of adjustments across factories, and the contribution of sequence tuning after exchanges across factories.

The remainder of this paper is organized as follows. Section 2 reviews distributed permutation flowshop scheduling, scheduling with transport resources, and related solution methods. Section 3 describes CTBDAPFSP and presents its mathematical formulation. Section 4 details TPCPIG and its main search mechanisms. Section 5 reports the computational experiments and analyzes the results. Section 6 concludes the paper and outlines directions for future research.

2. Literature Review

Relevant studies are reviewed from three perspectives: developments in distributed permutation flowshop scheduling, flowshop scheduling with transport resources, and related solution methods. The review focuses on how factory assignment and job sequencing are modeled, how transport resources affect delivery and assembly timing, and how these problem structures are handled algorithmically.

2.1. Developments in Distributed Permutation Flowshop Scheduling

Previous reviews cover permutation flowshop heuristics and the broader development of flowshop scheduling [13,14]. Naderi and Ruiz [2] extended this setting to the distributed permutation flowshop scheduling problem (DPFSP), in which jobs are assigned to several identical factories and sequenced within each factory. Factory assignment and job sequencing jointly determine the resulting schedule and makespan. Naderi and Ruiz [15] later developed a scatter search for DPFSP using a reference set, local search, and restarts. A complete solution is represented by one job list for each factory; together, these lists record both factory assignment and job sequencing.

Subsequent studies extended DPFSP by considering additional production constraints and transportation factors. Studies [16,17] investigated setup times that depend on the processing sequence. Zhao et al. [18] considered group scheduling, in which jobs in each group are processed consecutively without interruption. Li et al. [19] considered a common deadline constraint for part of the jobs, whereas Xiong et al. [20] modeled direct transportation with production and transportation costs that vary across factories. These studies extend the basic DPFSP through setup relations, group constraints, deadline requirements, and transportation costs.

2.2. Flowshop Scheduling with Transport Resources

Transportation has been incorporated into flowshop scheduling in several forms. Gao et al. [4] studied integrated batch production and distribution with a single facility and one dedicated vehicle of limited capacity. Customer orders are grouped into batches, and the vehicle may need to make multiple trips. In assembly flowshop scheduling, Wang et al. [21], Yang and Xu [22], and Li et al. [7] considered batch delivery of completed products to customers after assembly, with sufficient vehicles available for delivery. Qin et al. [23] studied distributed hybrid flow shops in which finished products are delivered in batches by limited manufacturer vehicles or vehicles provided by third-party logistics.

These studies illustrate different ways of incorporating batch delivery, vehicle capacity or availability, and transportation mode into production and assembly scheduling.

Transportation can also act as a constrained resource within production and assembly systems. Lei and Li [24] considered a distributed assembly problem with three stages in which several transportation machines at each factory transfer components from fabrication to assembly under capacity and availability constraints. Wang et al. [5] considered multiple identical transportation machines at each factory in distributed assembly flowshop scheduling; transportation machine assignment, capacity, and round trips are included in the schedule. Luo et al. [8] further treated vehicle assignment and vehicle sequencing for component delivery as explicit decisions in a distributed assembly hybrid flowshop. At the shop floor level, Li et al. [25] considered one robot at each factory to move jobs between machines, Xin et al. [26] jointly scheduled processing machines and automated guided vehicles, and Liu et al. [27] considered transport robots and shared buffers. Liu et al. [28] jointly determined automated guided vehicle assignment and sequencing, factory assignment, and job sequencing in distributed production. Berterottière et al. [29] likewise integrated machine assignment, vehicle assignment, sequencing, and routing in a flexible job shop. Related flowshop studies have also integrated processing schedules with job logistics by considering transportation times in the scheduling model [30]. These studies show that transportation resources can introduce capacity, occupancy, assignment, sequencing, and waiting relations that interact directly with production schedules.

Some studies incorporate transportation between production and assembly through transfer operations without explicitly scheduling transportation equipment. Shi et al. [31] considered a three-stage distributed heterogeneous hybrid flowshop with flexible assembly, in which completed jobs are transferred individually to their target assembly stations with given transportation times. Hao et al. [9] studied a distributed heterogeneous assembly hybrid flowshop in which completed jobs are batched before assembly. Job division into batches, batch sequencing, and batch transfer timing are included as decisions in the transportation stage. These settings differ from the problem studied here. In CTBDAPFSP, batch formation is not an independent decision. Jobs at each factory are ordered by their completion times at the final processing stage and grouped into FIFO batches according to vehicle capacity. Each factory has one dedicated vehicle operating in a closed loop, and the vehicle must return after delivering the current batch before the next batch can depart. Factory assignment and job sequencing therefore determine processing completion times, batch formation, vehicle availability, component arrivals, product readiness, and assembly timing. The transportation and assembly schedule is induced by production decisions and prescribed operating rules; batch partitioning, vehicle assignment, and transportation sequencing are not independent decision variables.

2.3. Related Solution Methods

Manufacturing scheduling has been addressed using a broad range of intelligent optimization methods, including metaheuristics, evolutionary algorithms, and learning-based approaches [32]. Several studies use problem information to construct and improve solutions in distributed assembly scheduling. Pan et al. [33] developed constructive heuristics and variable neighborhood search methods for product and part sequences. Huang et al. [34] designed destruction, construction, and local search procedures separately for products and jobs. Yu et al. [35] proposed an NEH initialization incorporating problem knowledge, destruction and construction procedures using such knowledge, and a local search at the product level for a distributed assembly hybrid flowshop with dual-resource constraints. Zhang et al. [36] combined an initial population with high quality and diversity, a matrix cube probabilistic model, and a neighborhood search based on the critical path for

energy-efficient distributed assembly permutation flowshop scheduling. Deng et al. [37] used an encoding that represents factory assignment and job processing sequence, together with local search incorporating problem knowledge, for distributed two-stage assembly flowshop scheduling. Wu et al. [6] also used hybrid initialization and neighborhood operators tailored to the problem in a fuzzy distributed assembly hybrid flowshop that includes transportation.

Moves between factories provide another way to modify a distributed schedule. Cui et al. [38] moved the operations of a critical job from the critical factory to a non-critical factory selected according to its makespan and then inserted them greedily. For CTBDAPFSP, changes in factory assignment or job sequencing must be evaluated through the resulting production, transportation, and assembly schedule. These studies therefore provide methodological background for solution construction informed by problem characteristics, search at different scheduling levels, and moves between factories that alter assignment and sequence structures.

For CTBDAPFSP, factory assignment and job sequencing affect not only the processing stage but also batch formation, vehicle return times, component arrivals, product readiness, and assembly timing under the prescribed operating rules. This structure raises three search issues. First, initial solutions should use problem information related to processing, transportation, and product readiness rather than rely only on random construction. Second, reinserting a job requires considering both its target factory and its position, while every candidate must be evaluated through the resulting complete schedule. Exhaustively evaluating all factory and position combinations can therefore be costly. Third, single-job relocation is less direct when simultaneous structural changes in two factories are desirable. These considerations motivate TPCPIG, which uses informed population construction, screened joint reinsertion, and cooperative adjustment across factories. All candidate solutions are evaluated by the same full schedule decoder.

3. Problem Description and Mathematical Formulation

3.1. Problem Description

This subsection describes CTBDAPFSP. Let J , F , H , and K denote the sets of jobs, factories, machine stages, and products, respectively. Each job belongs to one product and must be assigned to one factory, where it is processed through all machine stages. Jobs assigned to the same factory follow the same permutation sequence on all machines.

After completing the final processing stage, jobs at each factory are ordered by their completion times to form a FIFO sequence. Factory f has one dedicated vehicle with capacity Q_f and a travel time of τ_f in one direction. Each group of Q_f consecutive jobs in the FIFO sequence forms a full batch, while any remaining jobs form the final batch. Once formed, a batch cannot be split or rearranged. A batch departs immediately when it is ready and the vehicle is available. After delivering a batch to the central assembly station, the vehicle immediately returns to its factory and cannot serve the next batch until it has returned. A product becomes ready when all of its components have arrived at the central assembly station. The readiness time of product k is denoted by E_k . Products are assembled without delay on a single assembly station according to the lexicographic order of (E_k, k) . Given the factory assignments and processing sequences within factories, batch transportation, vehicle operation, product readiness, and assembly timing are uniquely determined. The objective is to minimize the makespan.

To further clarify the above operating rules and their temporal interactions, an illustrative example is provided in Section 3.3.

3.2. Mathematical Formulation

The formulation is organized around the coupled processing, transportation, and assembly stages of CTBDAPFSP. Timing relations based on processing positions are used to represent processing, while the transportation and assembly constraints are formulated according to the prescribed operating rules of the problem. In particular, batch formation and vehicle operation are induced by factory assignment and sequencing rather than optimized as independent scheduling decisions.

To formulate the mixed-integer linear programming (MILP) model, a set of potential processing positions is defined for each factory. Variable x_{jfr} simultaneously represents the assignment of job j to factory f and its processing position r . To simplify the processing time constraints, $\bar{p}_{frh}$ denotes the processing time associated with position (f, r) on machine h . It equals p_{jh} when position r is occupied by job j , and zero otherwise. According to vehicle capacity Q_f , the potential positions are further divided into potential transportation batches. Because occupied positions at each factory are constrained to be contiguous, the actual batches are uniquely determined by the position order and vehicle capacity.

The processing schedule of each factory is therefore represented by a position sequence with contiguous occupied positions, with factory assignment and sequencing within each factory both captured by x_{jfr} . Based on vehicle capacity Q_f , the position set is partitioned into potential batches, and I_{fb} denotes the positions associated with batch b at factory f . Variable g_{fb} identifies whether batch b is nonempty, while $y_{kk'}$ represents the assembly precedence between products under the specified readiness rule. These variables are used to represent deterministic operating rules rather than additional scheduling decisions. Table 1 summarizes the notation used in the MILP formulation.

The MILP formulation is given by Equations (1)–(26):

$$\min C_{\max} \quad (1)$$

Equation (1) minimizes the makespan.

$$\sum_{f \in F} \sum_{r \in O} x_{jfr} = 1, \quad \forall j \in J \quad (2)$$

$$\sum_{j \in J} x_{jfr} \leq 1, \quad \forall f \in F, r \in O \quad (3)$$

$$\sum_{j \in J} x_{jfr} \geq \sum_{j \in J} x_{jfr+1}, \quad \forall f \in F, r = 1, \dots, n-1 \quad (4)$$

Equations (2)–(4) define the factory and position assignments. Equation (2) assigns each job to exactly one position at one factory, Equation (3) allows at most one job at each position, and Equation (4) ensures that the occupied positions at each factory are consecutive. Therefore, x_{jfr} jointly determines the factory assignment of each job and the permutation sequence within each factory.

$$C_{fr1} \geq C_{f,r-1,1} + \bar{p}_{fr1}, \quad \forall f \in F, r = 2, \dots, n \quad (5)$$

$$C_{f,1,1} \geq \bar{p}_{f,1,1}, \quad \forall f \in F \quad (6)$$

$$C_{frh} \geq C_{f,r-1,h} + \bar{p}_{frh}, \quad \forall f \in F, r = 2, \dots, n, h = 2, \dots, m \quad (7)$$

$$C_{frh} \geq C_{f,r,h-1} + \bar{p}_{frh}, \quad \forall f \in F, r = 1, \dots, n, h = 2, \dots, m \quad (8)$$

Equations (5)–(8) model processing in the permutation flowshop. The processing time $\bar{p}_{frh}$ is associated with position (f, r) on machine h . Equations (5) and (6) define processing on the first machine, while Equation (7) enforces the precedence between consecutive

positions on the same machine, and Equation (8) enforces the technological precedence between consecutive machine stages for the same position. Together, these constraints define the processing completion times associated with the position sequence at each factory. The use of position completion times in these constraints follows a structure similar to that of Model 3 of Naderi and Ruiz [2].

$$g_{fb} \geq x_{jfr}, \quad \forall j \in J, r \in I_{fb} \quad (9)$$

$$g_{fb} \leq \sum_{r \in I_{fb}} \sum_{j \in J} x_{jfr} \quad (10)$$

$$U_{fb} \geq C_{frm} - M \left(1 - \sum_{j \in J} x_{jfr} \right), \quad r \in I_{fb} \quad (11)$$

Table 1. Notation for the MILP Formulation.

Symbol	Description
J	Job set, $J = \{1, \dots, n\}$
n	Number of jobs
F	Factory set, $F = \{1, \dots, f\}$
f	Number of factories
H	Machine stage set, $H = \{1, \dots, m\}$
m	Number of machine stages at each factory
K	Product set, $K = \{1, \dots, k\}$
k	Number of products
J_k	Set of jobs belonging to product k
p_{jh}	Processing time of job j on machine h
a_k	Assembly time of product k
Q_f	Capacity of the dedicated vehicle at factory f
τ_f	Travel time in one direction for the dedicated vehicle at factory f
O	Potential processing position set at each factory, $O = \{1, \dots, n\}$
B_f	Potential transportation batch set at factory f , $B_f = \left\{ 1, \dots, \left\lceil \frac{n}{Q_f} \right\rceil \right\}$
I_{fb}	Set of processing positions associated with batch b at factory f
M	A sufficiently large constant
x_{jfr}	Equals 1 if job j is assigned to position r at factory f , and 0 otherwise
$\bar{p}_{f rh}$	Processing time associated with position r at factory f on machine h
g_{fb}	Equals 1 if batch b at factory f is nonempty, and 0 otherwise
$y_{kk'}$	Equals 1 if product k is assembled before product k' , and 0 otherwise
$C_{f rh}$	Completion time of position r at factory f on machine h
U_{fb}	Ready time of batch b at factory f
D_{fb}	Departure time of batch b at factory f
A_{fb}	Arrival time of batch b at the central assembly station
R_{fb}	Time when the vehicle at factory f becomes available again after serving batch b
E_k	Readiness time of product k
S_k	Assembly start time of product k
T_k	Assembly completion time of product k
$C_{\max}$	Makespan

Equations (9)–(11) describe batch activation and readiness. Based on the contiguous occupied positions and vehicle capacity Q_f , the position sequence is divided into potential transportation batches. Equations (9) and (10) identify whether batch b is nonempty. Equation (11) ensures that the ready time of a nonempty batch cannot precede the comple-

tion time on the final machine of any occupied position in that batch. Therefore, except for the final nonempty batch, each nonempty batch contains Q_f consecutive jobs.

$$D_{f1} \geq U_{f1} - M(1 - g_{f1}) \quad (12)$$

$$D_{fb} \geq U_{fb} - M(1 - g_{fb}), \quad b \geq 2 \quad (13)$$

$$D_{fb} \geq D_{f,b-1} + 2\tau_f - M(1 - g_{fb}), \quad b \geq 2 \quad (14)$$

$$A_{fb} = D_{fb} + \tau_f g_{fb}, \quad R_{fb} = D_{fb} + 2\tau_f g_{fb} \quad (15)$$

$$E_k \geq A_{fb} - M(1 - x_{jfr}), \quad \forall k, j \in J_k, r \in I_{fb} \quad (16)$$

Equations (12)–(16) model transportation by the dedicated vehicle operating in a closed loop at each factory. A nonempty batch can depart only after it is ready. For every batch after the first, Equation (14) additionally requires the vehicle to complete the round trip of the preceding batch before the next departure. Equation (15) defines the batch arrival time at the central assembly station and the time at which the vehicle becomes available again. Equation (16) ensures that the readiness time of each product cannot precede the arrival of any of its components. Related distributed assembly models explicitly optimize the assignment and sequencing of transportation resources [8]. In contrast, the present constraints describe predetermined FIFO batches served successively by one dedicated vehicle operating in a closed loop at each factory.

$$E_k - E_{k'} \leq M(1 - y_{kk'}), \quad k < k' \quad (17)$$

An auxiliary binary variable $z_{kjfr} \in \{0, 1\}$ is introduced for $k \in K, j \in J_k, f \in F$, and $r \in O$; it equals 1 if job j at position r in factory f is selected as the component whose arrival determines the readiness time of product k , and 0 otherwise.

$$\sum_{j \in J_k} \sum_{f \in F} \sum_{r \in O} z_{kjfr} = 1, \quad \forall k \in K \quad (18)$$

$$z_{kjfr} \leq x_{jfr}, \quad \forall k \in K, j \in J_k, f \in F, r \in O \quad (19)$$

$$E_k \leq A_{fb} + M(1 - z_{kjfr}), \quad \forall k \in K, j \in J_k, f \in F, b \in B_f, r \in I_{fb} \quad (20)$$

Together with Equation (16), Equations (18)–(20) determine the readiness time of each product. Equation (16) provides a lower bound based on the arrival time of every component, while Equations (18)–(20) select one component associated with an occupied position to provide the corresponding upper bound. Consequently, the readiness time of a product equals the latest arrival time among all of its components.

$$E_{k'} - E_k + 1 \leq M y_{kk'}, \quad k < k' \quad (21)$$

$$S_k \geq E_k, \quad \forall k \in K \quad (22)$$

Equations (17) and (21) determine the assembly precedence between each pair of products according to the lexicographic order of (E_k, k) . For $k < k'$, $y_{kk'} = 1$ when $E_k \leq E_{k'}$; when the readiness times are equal, the smaller product index therefore has priority. Equation (22) ensures that assembly cannot start before the corresponding product is ready.

$$S_{k'} \geq T_k - M(1 - y_{kk'}), \quad k < k' \quad (23)$$

$$S_k \geq T_{k'} - M y_{kk'}, \quad k < k' \quad (24)$$

$$T_k = S_k + a_k, \quad \forall k \in K \quad (25)$$

$$C_{\max} \geq T_k, \quad \forall k \in K \quad (26)$$

Equations (23)–(25) model assembly on the single assembly station according to the precedence determined above. Equations (23) and (24) prevent two products from overlapping in assembly, and Equation (25) defines the assembly completion time. Finally, Equation (26) relates the product completion times to the makespan.

The relations describing the single assembly station and makespan are consistent with the general production–assembly structure of Hatami et al. [3], while product readiness in the present model is determined by component arrivals and the assembly order follows the prescribed lexicographic rule of (E_k, k) . Thus, factory assignment and sequencing within factories affect the makespan through their successive effects on processing completion, batch transportation, component arrivals, product readiness, and assembly timing.

3.3. Illustrative Example

To further illustrate the operating rules described in Section 3.1, consider the illustrative instance shown in Tables 2 and 3. It consists of two factories, two serial machines, six jobs, and three products. Table 2 gives the processing and assembly data, while Table 3 gives the vehicle capacity and travel time in one direction at each factory.

Table 2. Job and Product Data for the Illustrative Instance.

Job	Machine 1	Machine 2	Product	Assembly Time
J_1	1	3	P_1	2
J_2	4	2		
J_3	2	4	P_2	2
J_4	3	1		
J_5	2	2	P_3	1
J_6	3	4		

Table 3. Factory and Vehicle Data for the Illustrative Instance.

Factory	Vehicle Capacity	Travel Time in One Direction
F_1	2	3
F_2	2	1

All $6! \times 7 = 5040$ combinations of a job permutation and a factory split are evaluated by complete enumeration, yielding the exact optimum $C_{\max} = 15$. Thus, this value is obtained by complete enumeration rather than by a heuristic algorithm. One optimal solution assigns the sequence $\{J_1, J_4, J_5\}$ to F_1 and $\{J_3, J_2, J_6\}$ to F_2 . The resulting Gantt chart is shown in Figure 1. At F_1 , J_1 and J_4 form the first full batch. This batch departs at time 5, arrives at the central assembly station at time 8, and the vehicle returns to F_1 at time 11. Although J_5 completes processing at time 8, it must wait until the vehicle returns and therefore departs in the second batch at time 11, arriving at time 14. This illustrates how the required return trip of the dedicated vehicle can delay the departure of a later batch even after its jobs have completed processing. Products P_1 and P_2 both become ready at time 9. According to the lexicographic order of readiness time and product index, P_1 is assembled from time 9 to 11, followed by P_2 from time 11 to 13. Product P_3 becomes ready at time 14 and is assembled from time 14 to 15. The resulting makespan is therefore $C_{\max} = 15$.

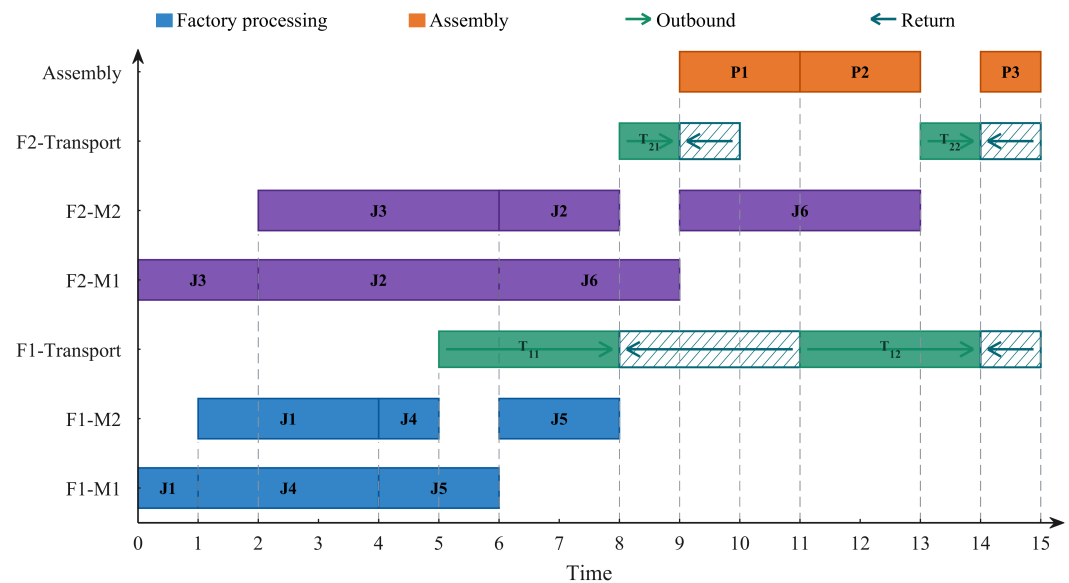


Figure 1. Gantt chart for the illustrative instance.

4. Timing Propagation Cooperative Population-Based Iterated Greedy Algorithm

Iterated greedy (IG) improves a current solution through repeated destruction and greedy reconstruction. Classical IG usually maintains a single current solution and removes and reinserts jobs to explore different permutations [39]. In distributed permutation flowshop scheduling, reconstruction can change both factory assignment and the processing sequence within each factory [12]. IG methods that use a population extend this framework by maintaining multiple candidate solutions with different scheduling structures during the search [10,19].

In CTBDAPFSP, factory assignment and sequencing within each factory affect not only processing completion times but also batch formation triggered by vehicle capacity, vehicle return times, component arrivals, product readiness, and assembly timing. TPCPIG is designed to account for these successive temporal effects. Its search consists of three main components: initial population construction, local improvement through joint adjustment of factory assignment and sequencing, and structural adjustment across factories. All candidate solutions are evaluated by the same full schedule decoder. The overall workflow of TPCPIG is shown in Figure 2.

4.1. Two-Layer Random Key Encoding and Unified Decoding

TPCPIG uses two layers to encode factory assignment and processing sequences within factories. A candidate solution is denoted by $X = (\pi^F, \pi^S)$, where π_j^F specifies the factory assigned to job j , and π_j^S is its priority key. Jobs assigned to the same factory are sorted in ascending order of π_j^S to obtain the permutation sequence of that factory. Ties, if any, are broken by job index. Thus, the two layers represent the two explicit scheduling decisions defined in Section 3: factory assignment and sequencing within each factory. Transportation batches, vehicle operations, product readiness, and assembly timing are induced by the operating rules rather than encoded as independent scheduling decisions.

Figure 3 illustrates how the two layers are mapped to a complete schedule. The factory indices π_j^F determine the assignment of the six jobs, while the priority keys π_j^S determine their relative order within each factory. Once the representation is mapped to factory sequences, the unified decoder calculates processing completion times, batch transportation, vehicle operations, product readiness, and assembly timing and returns

$C_{\max}$. TPCPIG therefore modifies only (π^F, π^S) during the search, while every candidate is evaluated under the same scheduling rules.

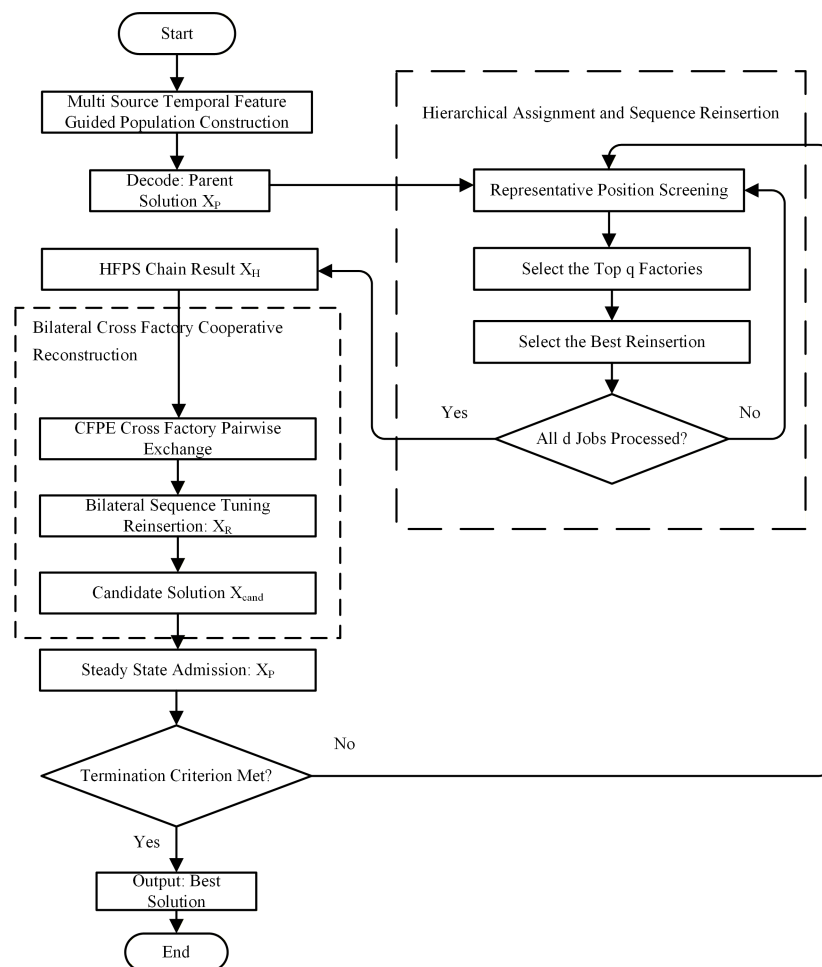


Figure 2. Overall workflow of TPCPIG.

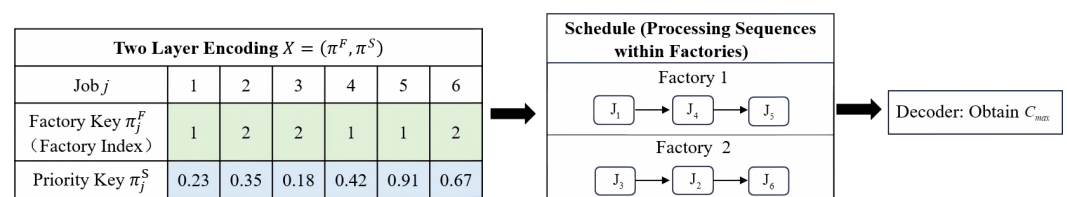


Figure 3. Encoding and decoding with two-layer representation.

4.2. Multisource Temporal Feature-Guided Population Construction

Random initialization can generate different factory assignments and processing sequences, but it does not explicitly use problem information related to processing, transportation, and product readiness. Therefore, TPCPIG constructs the initial population from three sources, denoted by R , K , and D . Source R uses jobs as the basic unit and evaluates candidate factories according to processing load, estimated batch transportation burden, batch triggering, and the distribution of components belonging to the same product. Source K uses products as the basic unit and considers product processing load and estimated batch transportation burden, with components of the same product assigned to the same factory during construction. Source D again uses jobs as the basic unit and considers processing load, vehicle capacity, and estimated batch transportation burden. It also applies constrained random selection among highly ranked factories to generate different assign-

ment structures. Each source includes controlled randomness and can therefore generate multiple candidates.

For each source, the first r trial candidates are compared with their opposite candidates [40]. The opposite candidate is generated by complementing the factory assignments and priority keys. Both candidates are fully decoded, and the candidate with the smaller makespan is retained. When the two makespans are equal, the original candidate is retained.

After the opposition screening, each source continues to generate candidates according to its construction rule until it contains $N/3$ individuals. The three groups are then combined to form the initial population. In this study, $N = 36$ and $r = 5$. Thus, each source contributes 12 individuals, including five individuals obtained through opposition screening and seven generated directly by the corresponding construction rule. The complete procedure is given in Algorithm 1.

Algorithm 1 Multisource Temporal Feature-Guided Population Construction

Input: instance I , population size N , number of trial candidates per source r

Output: initial population P

```

1.    $P \leftarrow \emptyset$ ;
2.   for  $b \in \{R, K, D\}$  do
3.      $P_b \leftarrow \emptyset$ ;
4.     for  $t = 1, \dots, r$  do
5.       Generate a trial candidate  $X$  according to the construction rule of source  $b$ ;
6.       Generate the opposition candidate  $\bar{X}$  of  $X$ ;
7.       Fully decode  $X$  and  $\bar{X}$ ;
8.       if  $C_{\max}(\bar{X}) < C_{\max}(X)$  then
9.          $X \leftarrow \bar{X}$ ;
10.      end if
11.       $P_b \leftarrow P_b \cup \{X\}$ ;
12.    end for
13.    Continue generating candidates according to the construction rule of source  $b$ 
    until  $|P_b| = N/3$ ;
14.     $P \leftarrow P \cup P_b$ ;
15.  end for
16.  return  $P$ .
```

4.3. Hierarchical Joint Reinsertion of Factory Assignment and Sequencing

After constructing the initial population, TPCPIG maintains multiple candidate solutions during the search [10,19]. Reinserting a job in CTBDAPFSP requires selecting both a target factory and an insertion position. Because each candidate requires full schedule decoding, evaluating all combinations of factories and insertion positions can be computationally expensive. Therefore, a hierarchical factory position screening procedure, denoted by HFPS, is embedded in the joint reinsertion of factory assignment and sequencing. In each iteration, the parent solutions are considered in a random order. For the current parent solution X_p , d different jobs are randomly selected without replacement to form the reinsertion sequence J_d . The selected jobs are reinserted one at a time. After each reinsertion, the current solution is updated before the next job is considered. This sequential update forms a chained local improvement process. The number of jobs reinserted in one parent event is defined as

$$d = d_0(n) = \min\{n - 1, \max(2, \lceil \log_2 n \rceil)\}. \quad (27)$$

For each selected job, the job is first removed from the current solution. The HFPS procedure shown in Figure 4 is then applied. In the first stage, the first, middle, and last insertion positions at each factory are fully evaluated, and duplicate positions are merged.

The minimum $C_{\max}$ among the representative insertion positions is used as the screening value S_f for factory f . The q factories with the lowest S_f values are selected for the second stage. The remaining untested insertion positions at these factories are then evaluated. Finally, the candidate with the minimum $C_{\max}$ among all candidates evaluated in both stages is selected to update the current solution. This procedure is repeated for all d jobs in J_d , yielding the HFPS chain result X_H . All candidates are evaluated by the same full schedule decoder. Thus, HFPS reduces the number of factory and position combinations requiring full evaluation without changing the evaluation rule for an individual candidate solution. The complete procedure is given in Algorithm 2.

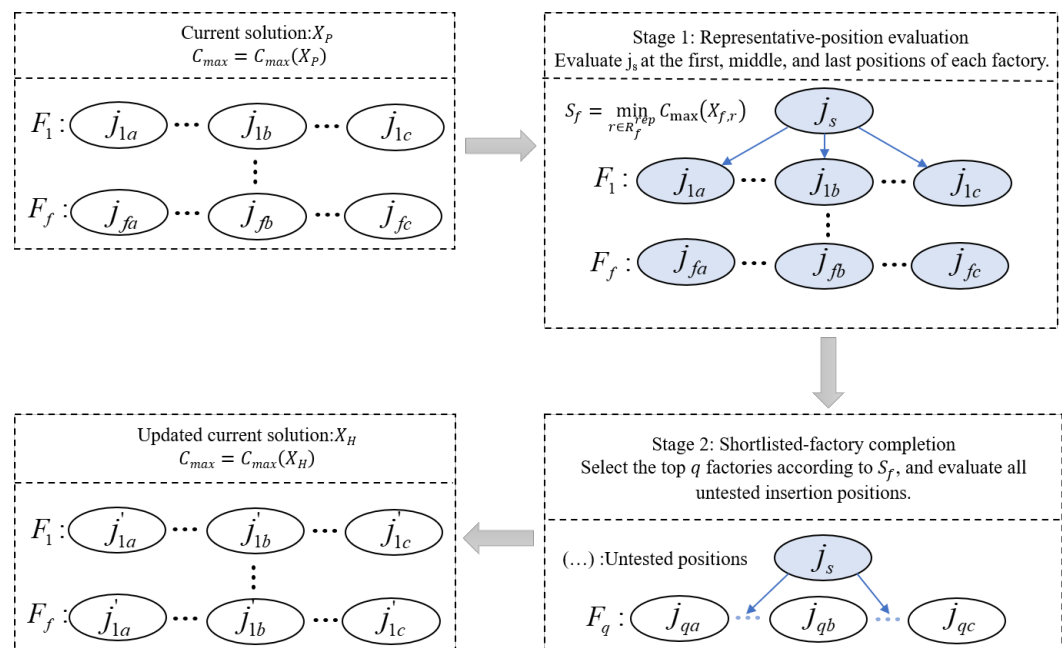


Figure 4. Hierarchical Factory Position Screening and Joint Reinsertion.

Algorithm 2 Hierarchical Joint Reinsertion of Factory Assignment and Sequencing

Input: parent solution X_P , number of processed jobs d , number of candidate factories q
Output: HFPS chain result X_H

1. $X \leftarrow X_P$;
2. Randomly select d different jobs from X without replacement to obtain the processing sequence J_d ;
3. **for each** $j \in J_d$ **do**
4. Remove j from the current solution X ;
5. **for each factory** f **do**
6. Construct the first, middle, and last representative positions;
7. Merge duplicate positions;
8. Fully decode the candidates at all representative positions;
9. Use the minimum $C_{\max}$ as the screening value S_f of factory f ;
10. **end for**
11. Sort all factories in ascending order of S_f ;
12. Select the first q factories;
13. Fully decode the untested positions of the selected factories;
14. Select the solution with the minimum $C_{\max}$ from all evaluated candidates;
15. $X \leftarrow$ the selected solution;
16. **end for**
17. $X_H \leftarrow X$;
18. **return** X_H .

4.4. Bilateral Cross-Factory Cooperative Reconstruction and Steady-State Update

The joint reinsertion in Section 4.3 adjusts factory assignment and sequencing through individual job relocation. To generate structural changes involving two factories, TPCPIG applies bilateral cross-factory cooperative reconstruction to the HFPS chain result X_H . First, two different nonempty factories f_a and f_b are selected, and one job is randomly selected from each factory. The two jobs are exchanged through cross-factory pairwise exchange (CFPE), yielding the exchanged solution X_C . Sequence tuning reinsertion (STR) is then applied sequentially to the two exchanged jobs in their receiving factories. For each job, candidate insertion positions are constructed around its exchange anchor r_a within a tuning radius ρ :

$$P(r_a) = \{r_a - \rho, \dots, r_a, \dots, r_a + \rho\}. \quad (28)$$

Cross-factory candidate positions are clipped to the feasible range, and duplicates are removed. Each remaining candidate position is evaluated using the full schedule decoder. The current solution is updated only if the best insertion position strictly improves its objective value. STR considers the two receiving sides in a fixed order and yields the bilateral STR result X_R . The procedure is illustrated in Figure 5.

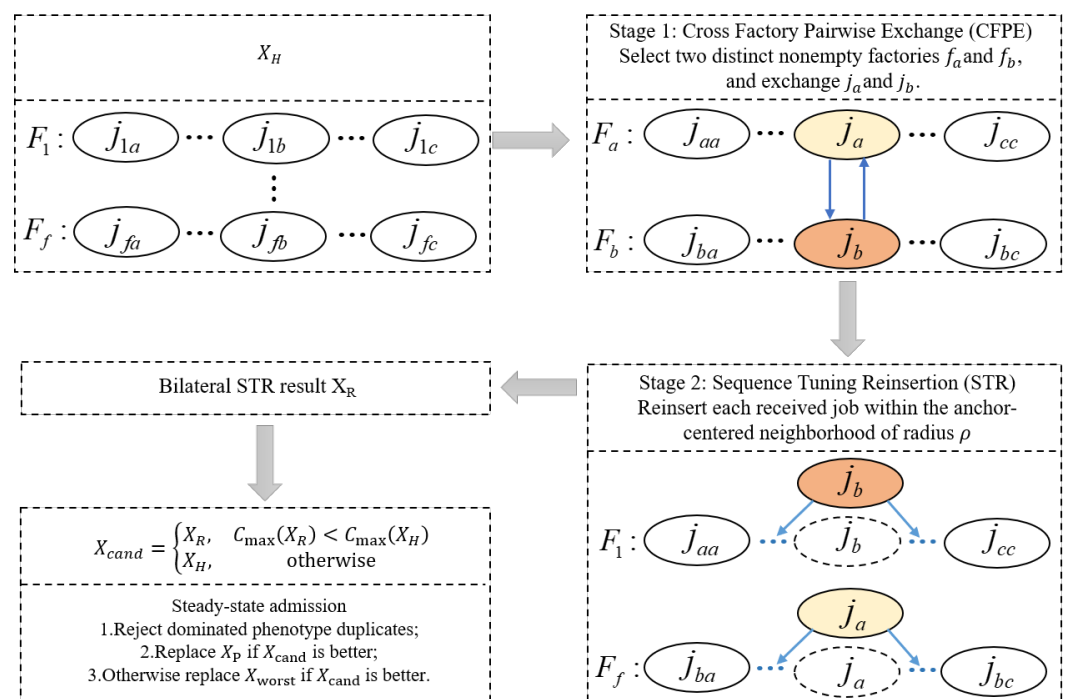


Figure 5. Schematic Diagram of Bilateral Cross-Factory Cooperative Reconstruction and Steady-State Update.

After bilateral STR, X_R is used as the candidate for the current parent event only if $C_{max}(X_R) < C_{max}(X_H)$; otherwise, X_H remains the candidate. The resulting candidate first undergoes a phenotype duplicate check, where the phenotype is defined by factory assignment and sequences within factories. If the population already contains the same phenotype with an objective value no worse than that of the candidate, the candidate is rejected. Otherwise, a candidate that strictly improves the parent replaces it directly. If it does not improve the parent but is strictly better than the current worst nonelite solution, it replaces that solution while the current elite is preserved. Thus, the bilateral reconstruction procedure is performed sequentially after hierarchical joint reinsertion rather than as an independent branch from the parent solution. The complete procedure is given in Algorithm 3.

Algorithm 3 Pseudocode of Bilateral Cross-Factory Cooperative Reconstruction and Steady-State Update

Input: parent solution X_P , HFPS chain result X_H , population P , tuning radius ρ
Output: updated population P

```

1. Randomly select two different nonempty factories  $f_a$  and  $f_b$  from  $X_H$ ;
2. Randomly select  $j_a \in f_a$  and  $j_b \in f_b$ ;
3. Exchange  $j_a$  and  $j_b$  to obtain the CFPE candidate  $X_C$ ;
4. Fully decode  $X_C$ ;
5.  $X \leftarrow X_C$ ;
6. Process the two receiving sides in a fixed order:
7.   Determine the exchange anchor for the current receiving job  $j$ ;
8.   Temporarily remove  $j$  from the current solution;
9.   Construct the position set;
10.  Clip positions outside the feasible range and remove duplicates;
11.  Fully decode all feasible positions;
12.  if the best position strictly improves the current solution then
13.    Accept the best position;
14.  else
15.    Keep the current position;
16.  end if
17. end;
18. Obtain the bilateral STR result  $X_R$ ;
19. if  $C_{\max}(X_R) < C_{\max}(X_H)$  then
20.    $X_{\text{cand}} \leftarrow X_R$ ;
21. else
22.    $X_{\text{cand}} \leftarrow X_H$ ;
23. end if
24. if an individual in  $P$  has the same phenotype as  $X_{\text{cand}}$  and an objective value no
    worse than  $X_{\text{cand}}$  then
25.   Reject  $X_{\text{cand}}$ ;
26. else if  $C_{\max}(X_{\text{cand}}) < C_{\max}(X_P)$  then
27.   Replace  $X_P$  with  $X_{\text{cand}}$ ;
28. else
29.   Identify the current worst nonelite individual  $X_{\text{worst}}$ ;
30.   if  $C_{\max}(X_{\text{cand}}) < C_{\max}(X_{\text{worst}})$  then
31.     Replace  $X_{\text{worst}}$  with  $X_{\text{cand}}$ ;
32.   end if
33. end if
34. return  $P$ .
```

To describe the computational effort of TPCPIG, let n , m , f , and N denote the numbers of jobs, machines, factories, and individuals in the population, respectively. A full schedule decoding evaluates processing over m machine stages together with batch transportation, product readiness, and assembly sequencing, with complexity $O(nm + n \log n)$. For one parent event, HFPS sequentially reinserts d jobs. For each reinsertion, the first stage evaluates at most $3f$ representative insertion positions, while the second stage evaluates the remaining positions in q selected factories. An upper bound on the number of full schedule evaluations is therefore

$$d[3f + q(n + 1)]. \quad (29)$$

Bilateral reconstruction additionally requires one CFPE evaluation, while STR evaluates at most $2\rho + 1$ positions on each receiving side. This adds at most $4\rho + 3$ full evaluations. The computational complexity of one complete population scan is therefore

$$O(N[d(3f + q(n + 1)) + 4\rho + 3](nm + n \log n)). \quad (30)$$

Multisource population construction is performed only once at the beginning of the search, whereas HFPS joint reinsertion and bilateral reconstruction are repeatedly executed. TPCPIG uses CPU time as the stopping criterion. Therefore, the numbers of completed parent events and search iterations depend on the evaluation cost for instances of different sizes.

5. Experimental Results and Analysis

To evaluate TPCPIG, CTBDAPFSP instances are generated by extending the standard distributed permutation flowshop instances of Naderi and Ruiz [2] under a common set of generation rules. Product groupings and assembly times are added to the original processing data, and each factory is assigned a dedicated vehicle with a specified capacity and travel time in one direction. The final test set, denoted by Benchmark120, contains 120 instances with $n \in \{4, 6, 8, 10, 12, 14, 16, 20, 50, 100, 200, 500\}$. The instances cover different problem sizes and transportation constraint levels. Parameter calibration uses a separate set, CAL30, which is independently generated under the same rules and does not overlap with Benchmark120. All algorithms are implemented in C++17 and compiled using Microsoft Visual Studio 2022 version 17.14 with the x64/O2 option. The experiments are conducted on a computer with an Intel Core i7 14650HX processor, 31.74 GiB of memory, and Windows 10 (64-bit). Each randomized algorithm is run independently 10 times on each instance. Process CPU time is used as the stopping criterion, with the time limit defined as

$$t_{\max} = \max\{1, 0.10nm\} \text{ s.}$$

The experiments examine parameter settings, the effects of the main mechanisms, solution performance, and search behavior.

5.1. Parameter Settings

TPCPIG has four main parameters: population size N , number of chained jobs d , number of candidate factories q in HFPS, and STR tuning radius ρ . The reference value of d depends on the number of jobs:

$$d_0(n) = \min\{n - 1, \max(2, \lceil \log_2 n \rceil)\}. \quad (31)$$

Parameter calibration uses CAL30, which is generated independently from Benchmark120. Stage A and Stage B each use 15 nonoverlapping instances. The reference configuration is

$$B_0 = (N = 36, d = d_0, q = 2, \rho = 1),$$

and two candidate levels are considered for each parameter, as shown in Table 4. To measure the deviation of a candidate solution from the specified reference solution, the relative percentage deviation (RPD) is defined as

$$RPD = 100 \frac{C_{\max} - C_{\max}^{ref}}{C_{\max}^{ref}}. \quad (32)$$

A lower RPD indicates better performance.

In Stage A, the reference configuration and the eight candidate parameter settings are each tested on 15 instances using five common random seeds, resulting in 675 runs in total. A candidate level is retained only if its overall RPD is below -0.03% and its number of wins exceeds its number of losses. Figure 6 presents the results of screening one factor at a time. The RPD values are calculated relative to the reference configuration, with negative values indicating improvement. Although $N = 30$ obtains an overall RPD of -0.0864% , its win/loss result is 5/6; therefore, it does not satisfy the retention criterion. The overall RPD values of $q = 1$, $\rho = 2$, and $\rho = 3$ are -0.1803% , -0.0577% , and -0.1553% , respectively. Accordingly, $q = 1$ and $\rho = 3$ are retained, yielding

$$B^* = (N = 36, d = d_0, q = 1, \rho = 3).$$

The configurations B_0 and B^* are then independently validated on the remaining 15 CAL30 instances using 10 new common random seeds for each configuration, resulting in 300 runs. As reported in Table 5, the mean $C_{\max}$ decreases from 3255.13 for B_0 to 3244.70 for B^* . The RPD reported in Table 5 relative to B_0 is -0.1866% , and the win/tie/loss result over the 15 instances is 8/3/4. The one-sided exact Wilcoxon test gives $p = 0.024658$. The independent validation supports the configuration selected in Stage A. Therefore, $N = 36$, $d = d_0$, $q = 1$, and $\rho = 3$ are used in all subsequent experiments.

Table 4. Parameter Levels for Calibrating TPCPIG.

Parameter	Reference Value	Candidate Level 1	Candidate Level 2
Population size N	36	30	42
Number of chained jobs d	d_0	$d_0 - 1$	$d_0 + 1$
Number of candidate factories q in HFPS	2	1	3
STR tuning radius ρ	1	2	3

Table 5. Independent Validation Results for Candidate Configurations.

Configuration	Parameter Setting (N, d, q, ρ)	Mean $C_{\max}$	RPD Relative to $B_0/\%$	One-Sided Exact Wilcoxon p -Value	Result
B_0	$(36, d_0, 2, 1)$	3255.13	0	—	Reference
B^*	$(36, d_0, 1, 3)$	3244.70	-0.1866	0.024658	Accepted

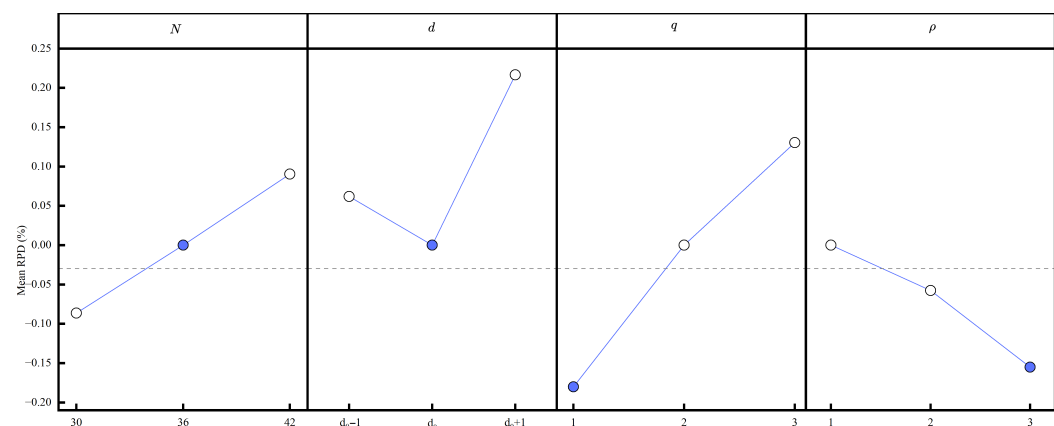


Figure 6. Results of Screening One Factor at a Time for TPCPIG.

5.2. Validation of the CTBDAPFSP Model

To assess consistency between the CTBDAPFSP mathematical model and the algorithm decoder, we use representative instances with $n \in \{6, 10, 16, 20, 50\}$ and $m \in \{2, 3, 5, 10\}$. The tests also examine the computational behavior of CPLEX as the problem size increases. The selected instances include small cases that CPLEX can solve to proven optimality and larger cases that approach or reach the prescribed time limit. CPLEX is run using a single thread with a time limit of 600 s for each instance. TPCPIG uses the final parameter configuration and is run independently five times on each instance.

As shown in Table 6, TPCPIG obtains the same best $C_{\max}$ as CPLEX for all instances on which CPLEX proves optimality. This agreement provides computational evidence for the consistency between the mathematical model and the algorithm decoder on these instances. As the numbers of jobs and machines increase, the CPU time required by CPLEX increases substantially, and several instances reach the 600 s time limit. For the cases that reached the time limit with $n = 20$, TPCPIG obtains better objective values than the CPLEX incumbents. For the cases with $n = 50$, CPLEX does not obtain a feasible integer solution within the prescribed time limit, whereas TPCPIG returns feasible solutions within its shorter CPU budget. These results also show that the computational effort required by CPLEX increases rapidly with instance size, supporting the use of TPCPIG for the subsequent medium and large instances.

Table 6. Comparison of CPLEX and TPCPIG on Representative CTBDAPFSP Instances.

(n, f, m)	CPLEX Incumbent $C_{\max}$	CPLEX CPU Time (s)	TPCPIG Best $C_{\max}$	TPCPIG Mean $C_{\max}$	TPCPIG CPU Budget (s)
(6, 2, 2)	233	0.19	233	233.0	1.2
(10, 2, 2)	407	1.35	407	407.0	2.0
(16, 2, 2)	411	201.45	411	415.4	3.2
(20, 2, 5)	906	600.01	890	896.2	10.0
(50, 2, 5)	—	—	1585	1592.6	25.0
(6, 2, 3)	377	0.56	377	377.0	1.8
(10, 2, 3)	408	14.70	408	408.0	3.0
(16, 2, 3)	596	600.01	596	596.0	4.8
(20, 2, 10)	1363	600.01	1255	1261.4	20.0
(50, 2, 10)	—	—	1843	1890.6	50.0

5.3. Algorithm Comparison

To evaluate the overall performance of TPCPIG, it is compared with IG, CPIG, Jaya-QL, and Q-ANSGA. IG provides a basic iterated greedy reference. CPIG [18] is a cooperative population-based iterated greedy method for distributed permutation flowshop group scheduling. Jaya-QL [41] is a method enhanced by Q-learning for distributed assembly permutation flowshop scheduling. Q-ANSGA [31] is included as a recent NSGA-II approach for a related distributed scheduling problem with flexible assembly and discrete transportation. All five algorithms are tested on Benchmark120. Each algorithm is run independently 10 times on every instance using the same full schedule decoder for CTBDAPFSP and the same CPU time limit. For instance i , RPD is calculated using the common best reference value B_i , defined as the smallest $C_{\max}$ among all raw runs included in the unified comparison. For one algorithm on an instance, the mean RPD over the 10 runs is defined as the average relative percentage deviation (ARPD) for that instance. The overall ARPD is the mean of these instance ARPD values over all 120 instances. For the statistical comparisons, the mean $C_{\max}$ from the 10 runs on each instance is used as the paired observation.

Table 7 reports the mean $C_{\max}$ values for different numbers of jobs. TPCPIG obtains the lowest overall mean $C_{\max}$, 2448.07, and the lowest overall ARPD, 0.906%. The overall ARPD values of IG, CPIG, Jaya-QL, and Q-ANSGA are 8.607%, 4.829%, 4.217%, and 3.006%, respectively. Across the 120 instances, the win/tie/loss results of TPCPIG against IG, CPIG, Jaya-QL, and Q-ANSGA are 117/3/0, 95/14/11, 109/11/0, and 99/16/5, respectively. The corresponding one-sided Wilcoxon tests are all statistically significant. Thus, using the same instances, decoder, and CPU budget, the paired results favor TPCPIG over the four compared algorithms.

Table 7. Mean $C_{\max}$ by Number of Jobs.

Number of Jobs n	Number of Instances	IG	CPIG	Jaya-QL	Q-ANSGA	TPCPIG
4	6	276.00	268.03	267.83	267.83	267.83
6	6	304.38	296.15	297.00	295.50	295.50
8	6	332.87	322.37	327.38	322.55	322.15
10	6	393.02	376.50	382.65	377.12	374.97
12	6	447.70	417.82	433.55	419.37	415.93
14	6	509.53	483.72	498.57	485.28	482.90
16	6	505.42	465.25	483.52	468.65	464.17
20	16	1058.12	1015.38	1028.47	1027.77	1013.03
50	16	1679.41	1591.81	1616.05	1598.14	1550.21
100	16	2923.88	2835.05	2806.79	2737.29	2663.54
200	24	5470.24	5409.23	5156.32	5139.02	4954.02
500	6	14,067.10	14,194.18	12,816.07	13,116.90	12,583.70
Overall	120	2690.70	2648.68	2533.43	2530.56	2448.07

At $n = 4$ and $n = 6$, TPCPIG shares the lowest mean $C_{\max}$ with some compared algorithms. From $n = 8$ onward, it obtains the lowest mean $C_{\max}$ for every larger tested job size. Figure 7 presents the overall ARPD values of the five algorithms. The error bars correspond to a 95% least significant difference of 0.6091 percentage points for pairwise comparisons of mean ARPD values; they are not confidence intervals for individual algorithm means. Figure 8 shows the search trajectories on a representative instance with $n = 500$. On this instance, TPCPIG successively improves its best solution and reaches the lowest final $C_{\max}$ within the prescribed CPU budget.

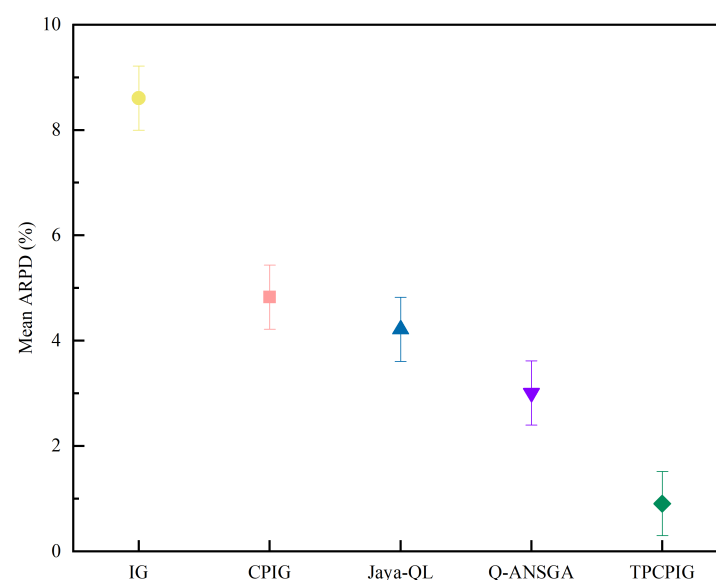


Figure 7. Overall ARPD of the Compared Algorithms.

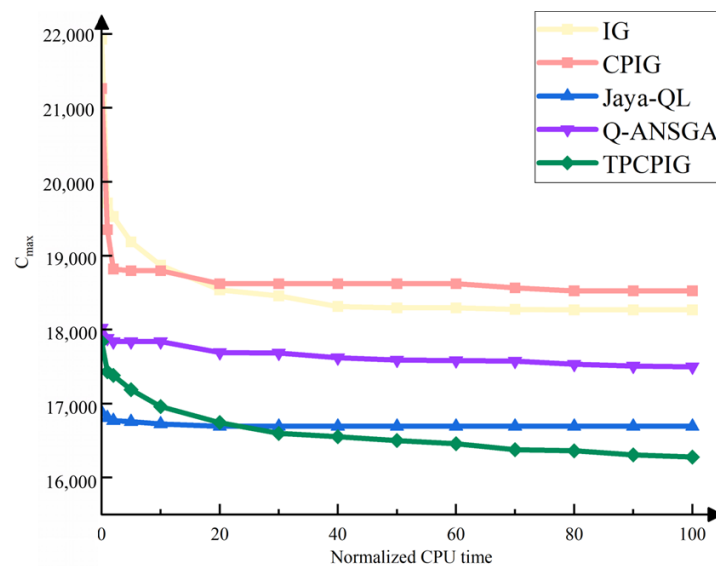


Figure 8. Search Trajectories of the Compared Algorithms.

5.4. Effectiveness Analysis of Key Mechanisms

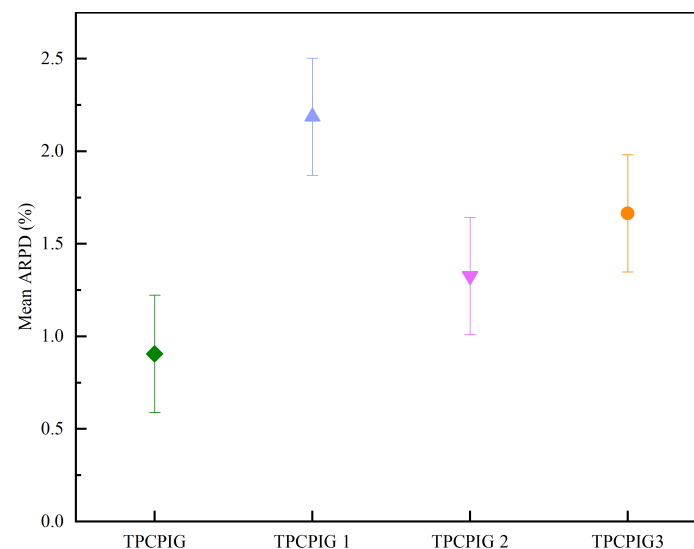
To assess the contribution of the main mechanisms in TPCPIG, three ablation variants are constructed by removing one mechanism at a time. TPCPIG1 replaces the multi-source temporal feature-guided population construction with random feasible initialization. TPCPIG2 replaces the HFPS hierarchical joint reinsertion with a baseline exact chained reinsertion that evaluates all factories and insertion positions. This variant retains the basic joint reinsertion capability and allows the effect of hierarchical screening to be examined. TPCPIG3 removes CFPE and bilateral STR and therefore disables bilateral cross-factory cooperative reconstruction. All four variants use the same Benchmark120 instances, random seeds, CPU time limit, full schedule decoder, phenotype duplicate elimination, and steady-state admission rule.

Table 8 reports the mean $C_{\max}$ values by number of jobs. The overall mean $C_{\max}$ of TPCPIG is 2448.07, compared with 2535.23, 2462.99, and 2456.95 for TPCPIG1, TPCPIG2, and TPCPIG3, respectively. Their overall ARPD values are 0.906%, 2.187%, 1.326%, and 1.665%, respectively. Figure 9 presents these overall ARPD values. The error bars correspond to a 95% least significant difference of 0.3169 percentage points for pairwise comparisons of mean ARPD values; they are not confidence intervals for individual algorithm means. The ARPD differences between TPCPIG and all three ablation variants exceed this value. The win/tie/loss results of TPCPIG against TPCPIG1, TPCPIG2, and TPCPIG3 are 76/20/24, 82/22/16, and 96/14/10, respectively. The corresponding one-sided Wilcoxon tests are all statistically significant. Under the current parameter configuration and fixed CPU budget, these results indicate that all three mechanisms contribute to the overall performance of TPCPIG. Removing the multisource population construction causes the largest overall degradation.

The contributions of the mechanisms vary across problem sizes. TPCPIG1 is close to the complete algorithm when $n \leq 20$, whereas the gap becomes more apparent from $n = 50$. At $n = 500$, TPCPIG1 obtains 13,599.02, compared with 12,583.70 for TPCPIG. This pattern indicates that the contribution of initialization guided by temporal features from multiple sources is more pronounced for larger instances. TPCPIG2 achieves mean makespans close to those of TPCPIG at smaller values of n , but higher mean makespans at $n \in \{50, 100, 200, 500\}$. This suggests that the hierarchical organization of HFPS provides an effective balance between candidate evaluation and search progress under the fixed CPU budget.

Table 8. Mean $C_{\max}$ in the Ablation Study.

Number of Jobs n	Number of Instances	TPCPIG	TPCPIG1	TPCPIG2	TPCPIG3
4	6	267.83	267.83	267.83	267.83
6	6	295.50	295.50	295.50	295.50
8	6	322.15	322.03	322.05	324.77
10	6	374.97	374.92	375.08	380.40
12	6	415.93	416.98	416.57	423.98
14	6	482.90	482.87	482.77	491.10
16	6	464.17	464.15	464.55	471.50
20	16	1013.03	1012.62	1014.69	1023.48
50	16	1550.21	1559.66	1555.34	1560.72
100	16	2663.54	2698.11	2677.12	2678.32
200	24	4954.02	5106.74	4999.82	4966.15
500	6	12,583.70	13,599.02	12,643.63	12,585.82
Overall	120	2448.07	2535.23	2462.99	2456.95

**Figure 9.** Overall ARPD in the Ablation Study.

TPCPIG3 obtains higher mean $C_{\max}$ values from $n = 8$ to $n = 200$, but it is close to TPCPIG at $n = 500$. Thus, the bilateral reconstruction mechanism provides complementary search over a broad range of problem sizes, while its effect is small at $n = 500$ in this experiment. For individual problem sizes, an ablation variant can be close to or slightly better than the complete algorithm. Therefore, the contribution of each mechanism should be assessed mainly from the overall paired results over 120 instances rather than interpreted as consistent improvement at every tested job size.

5.5. Search Behavior and Mechanism Execution Analysis

To describe the search behavior of TPCPIG under the fixed CPU budget, 36 diagnostic instances are selected from Benchmark120. The diagnostic set covers 12 values of n , $n \in \{4, 6, 8, 10, 12, 14, 16, 20, 50, 100, 200, 500\}$, with three instances for each value. Ten common random seeds are used for each instance, resulting in 360 diagnostic runs. The diagnostic process records search checkpoints, HFPS screening, reinsertion across factories, and the execution of CFPE and STR. The improvement rate (IR) measures the cumulative improvement relative to the initial solution. The recall rate (RR) measures the proportion of selections preferred by full enumeration that are retained by HFPS, while the evaluation reduction rate (ER) measures the reduction in full schedule evaluations. The cross-factory

adjustment ratio (CFAR) is the proportion of HFPS reinsertion events involving adjustments across factories, and the improvement contribution ratio (ICR) is the share of total positive objective improvement contributed by these adjustments. For each value of n , IR, ER, CFAR, and ICR are summarized as medians across the 30 runs. RR is calculated by dividing the total number of recall hits by the total number of recall audits across these runs. Figure 10 shows the median IR over normalized search progress. The shaded bands in Figures 11 and 12 show the interquartile ranges of the corresponding diagnostic values at the run level.

Figure 10 shows that most improvements occur early in the search, although further improvements can still occur at later stages for larger instances. At $n = 50$, the median IR increases from 13.060% at 10% search progress to 16.370% at the end of the search, while at $n = 500$ it increases from 0.950% to 1.708%. Figure 11 further shows that ER generally increases with the number of jobs, reaching 68.348% and 74.549% at $n = 100$ and $n = 500$, respectively, while the corresponding RR values remain at 93.533% and 81.708%. These observations indicate that the search does not rely exclusively on early improvements and that, under the fixed CPU budget, HFPS reduces the number of full evaluations required for reinsertion while retaining a high proportion of the selections preferred by full enumeration.

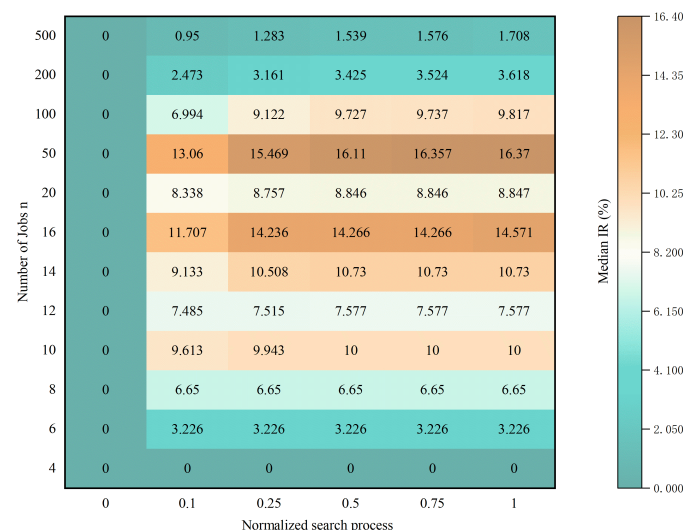


Figure 10. Median IR over Normalized Search Progress.

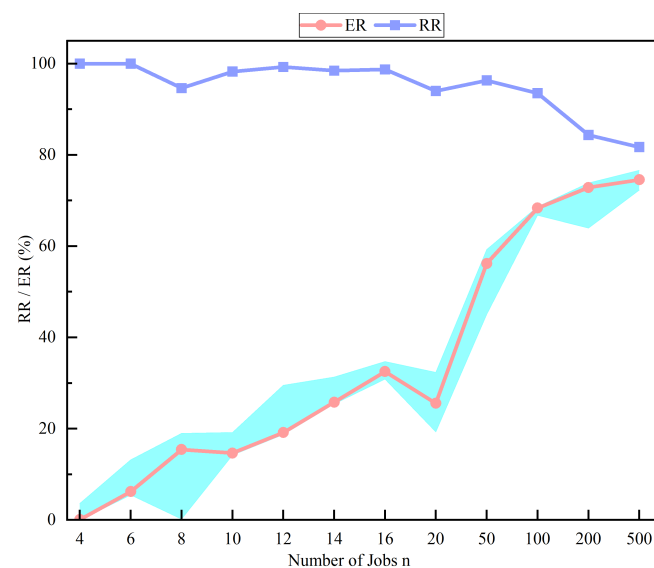


Figure 11. HFPS Recall Rate and Evaluation Reduction Rate.

Figure 12 describes the role of adjustments across factories in joint reinsertion. Although CFAR can be low, ICR can still be substantial. At $n = 16$, for example, CFAR is only 1.689%, whereas ICR reaches 66.482%. Thus, adjustments across factories are not the most frequent form of reinsertion, but they can contribute a substantial share of the total positive objective improvement.

Figure 13 further describes the relationship between CFPE and STR in the bilateral reconstruction procedure. Let C_H , C_C , and C_R denote the objective values of the HFPS chain output, the solution after CFPE, and the solution after bilateral STR, respectively. A direct CFPE improvement satisfies $C_C < C_H$, whereas an STR adaptive conversion satisfies $C_C \geq C_H$ and $C_R < C_H$. The bars show the shares of these two event types among successful reconstruction events, while the labels above the bars show the proportion of successful reconstruction events among all executed reconstruction events involving CFPE and STR. At $n = 50, 100$, and 200 , STR adaptive conversion accounts for about 60% of successful events. This indicates that local sequence adjustment can convert some exchanges across factories that do not initially outperform the HFPS result into improved candidates. Overall, the diagnostics describe a sequential interaction among HFPS screening, adjustment across factories, and subsequent local sequence adjustment.

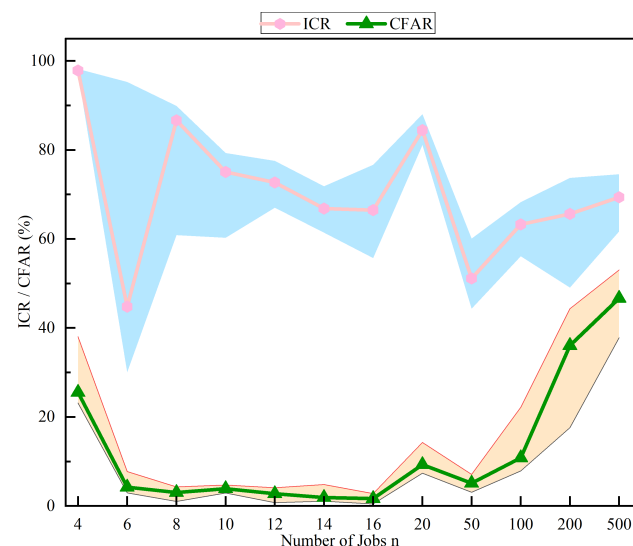


Figure 12. Cross-Factory Adjustment Ratio and Improvement Contribution Ratio in HFPS.

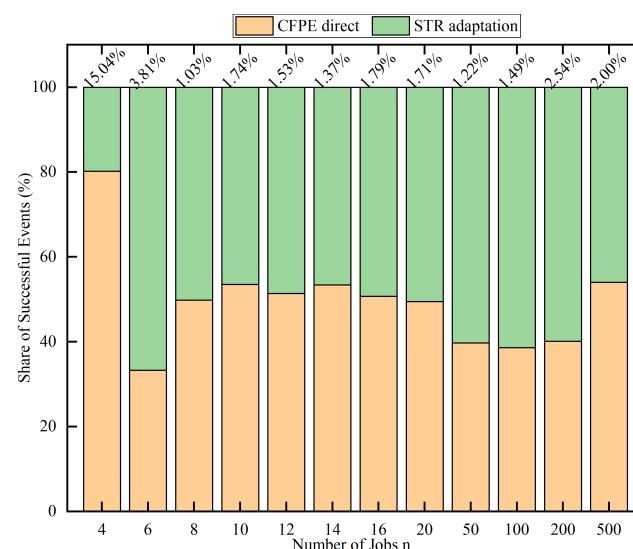


Figure 13. Composition of Successful Bilateral Cross-Factory Cooperative Reconstruction Events.

6. Conclusions

This paper investigates a distributed assembly permutation flowshop scheduling problem in which batches are formed according to vehicle capacity and dedicated vehicles operate in closed loops. Components are processed at multiple factories, transported in batches to a central assembly station, and assembled only after all components required by a product have arrived. A unified optimization model describes processing, batch transportation, vehicle cycles, product readiness, and assembly. To account for the effects of factory assignment and sequencing within factories on subsequent transportation and assembly stages, this paper proposes the Timing Propagation Cooperative Population-Based Iterated Greedy algorithm (TPCPIG). TPCPIG constructs its initial population from complementary temporal feature sources, uses hierarchical joint reinsertion to adjust factory assignment and sequencing, and applies bilateral cross-factory cooperative reconstruction as an additional structural adjustment procedure.

On Benchmark120, TPCPIG achieves the lowest overall mean $C_{\max}$ of 2448.07 and an overall average relative percentage deviation of 0.906%, compared with 3.006–8.607% for the four comparison algorithms. From $n = 8$ onward, TPCPIG obtains the lowest mean $C_{\max}$ for every tested job size. The ablation results further support the contributions of multisource population construction, hierarchical joint reinsertion with HFPS, and bilateral cross-factory cooperative reconstruction under the fixed CPU budget. The search diagnostics show that HFPS reduces the number of full schedule evaluations while retaining a high recall rate, and further reveal the roles of adjustments across factories and subsequent local sequence adaptation during the search.

From a practical perspective, the proposed model is relevant to distributed assembly systems in which components are processed at multiple factories and transported to a central assembly station by dedicated vehicles with limited capacity operating in closed loops. The results highlight that factory assignment and sequencing within each factory should not be considered independently of transportation operations, because changes in production decisions can propagate through batch formation and vehicle availability to component arrival and product readiness. The proposed model and TPCPIG therefore provide a framework to support coordinated production, transportation, and assembly decisions in manufacturing systems with such operating characteristics.

Future research may proceed in two directions. From the algorithmic perspective, future studies may investigate adaptive selection of construction sources and dynamic adjustment of the number of chained jobs d , the number of candidate factories q , and the STR tuning radius ρ . Incremental evaluation, parallel computation, and more efficient candidate screening methods may also reduce the computational cost of full schedule decoding. From the modeling perspective, future studies may consider coordination among multiple vehicles, transportation times that vary over time, vehicle failures, setup times that depend on the processing sequence, finite assembly buffers, due dates, and energy consumption.

Author Contributions: Conceptualization, J.C. and L.W.; methodology, J.C., D.L., C.H. and Q.H.; software, J.H., Y.W., D.G., S.Y., M.L. and C.L.; validation, J.H.; writing—original draft preparation, J.C. and J.H.; writing—review and editing, J.C. and X.Z.; visualization, J.C. and J.H.; supervision, J.C. and J.H.; data curation, C.A.; funding acquisition, C.H. and Q.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Key Laboratory of Electric Drive and Control of Anhui Province (DQKJ202412), Anhui Province Key Laboratory of Advanced Numerical Control and Servo Technology (XJSK2025012), the Open Research Fund of Anhui Province Key Laboratory of Machine Vision Detection and Perception (KLMVI-2025-HIT-02 and KLMVI-2024-HIT-15), and the Open Fund of State Key Laboratory of Digital Intelligent Technology for Unmanned Coal Mining (Grant No. SZQZ2025-2-026).

Data Availability Statement: Benchmark120 and CAL30 were randomly generated from the DPFSP instances available at https://github.com/chgogos/DPFSP_CP/raw/main/DPFSP.7z, following Section 5.1 (accessed on 28 August 2026).

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

References

1. Johnson, S.M. Optimal two- and three-stage production schedules with setup times included. *Manag. Sci.* **1954**, *1*, 61–68. [\[CrossRef\]](#)
2. Naderi, B.; Ruiz, R. The distributed permutation flowshop scheduling problem. *Comput. Oper. Res.* **2010**, *37*, 754–768. [\[CrossRef\]](#)
3. Hatami, S.; Ruiz, R.; Andrés-Romano, C. The distributed assembly permutation flowshop scheduling problem. *Int. J. Prod. Res.* **2013**, *51*, 5292–5308. [\[CrossRef\]](#)
4. Gao, S.; Qi, L.; Lei, L. Integrated batch production and distribution scheduling with limited vehicle capacity. *Int. J. Prod. Econ.* **2015**, *160*, 13–25. [\[CrossRef\]](#)
5. Wang, J.; Tang, H.; Lei, D. A Q-learning artificial bee colony for distributed assembly flow shop scheduling with factory eligibility, transportation capacity and setup time. *Eng. Appl. Artif. Intell.* **2023**, *123*, 106230. [\[CrossRef\]](#)
6. Wu, R.; Li, Q.; Cheng, B.; Chen, Y.; Li, X. A Q-learning-based adaptive NSGA-II for fuzzy distributed assembly hybrid flow shop scheduling problem. *Processes* **2026**, *14*, 500. [\[CrossRef\]](#)
7. Li, D.; Gao, K.-Z.; Duan, P.; Suganthan, P.N.; Wu, N. Reinforcement learning assisting artificial bee colony algorithm for scheduling distributed assembly flowshops with batch delivery. *IEEE Trans. Syst. Man Cybern. Syst.* **2025**, *55*, 9295–9308. [\[CrossRef\]](#)
8. Luo, Q.; Deng, Q.; Guo, X.; Gong, G.; Zhao, X.; Chen, L. Modelling and optimization of distributed assembly hybrid flowshop scheduling problem with transportation resource scheduling. *Comput. Ind. Eng.* **2023**, *186*, 109717. [\[CrossRef\]](#)
9. Hao, H.; Zhu, H.; Luo, Y. A multi-objective immune balancing algorithm for distributed heterogeneous batching-integrated assembly hybrid flowshop scheduling. *Expert Syst. Appl.* **2025**, *259*, 125288. [\[CrossRef\]](#)
10. Chen, S.; Pan, Q.-K.; Gao, L.; Sang, H.-Y. A population-based iterated greedy algorithm to minimize total flowtime for the distributed blocking flowshop scheduling problem. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104375. [\[CrossRef\]](#)
11. Zhao, F.; Xu, Z.; Wang, L.; Zhu, N.; Xu, T.; Jonrinaldi, J. A population-based iterated greedy algorithm for distributed assembly no-wait flow-shop scheduling problem. *IEEE Trans. Ind. Inform.* **2023**, *19*, 6692–6705. [\[CrossRef\]](#)
12. Ruiz, R.; Pan, Q.-K.; Naderi, B. Iterated greedy methods for the distributed permutation flowshop scheduling problem. *Omega* **2019**, *83*, 213–222. [\[CrossRef\]](#)
13. Ruiz, R.; Maroto, C. A comprehensive review and evaluation of permutation flowshop heuristics. *Eur. J. Oper. Res.* **2005**, *165*, 479–494. [\[CrossRef\]](#)
14. Gupta, J.N.D.; Stafford, E.F. Flowshop scheduling research after five decades. *Eur. J. Oper. Res.* **2006**, *169*, 699–711. [\[CrossRef\]](#)
15. Naderi, B.; Ruiz, R. A scatter search algorithm for the distributed permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2014**, *239*, 323–334. [\[CrossRef\]](#)
16. Huang, J.-P.; Pan, Q.-K.; Gao, L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *Swarm Evol. Comput.* **2020**, *59*, 100742. [\[CrossRef\]](#)
17. Zhao, F.; Zhou, G.; Wang, L. A cooperative scatter search with reinforcement learning mechanism for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 4899–4911. [\[CrossRef\]](#)
18. Zhao, H.; Pan, Q.-K.; Gao, K.-Z. A cooperative population-based iterated greedy algorithm for distributed permutation flowshop group scheduling problem. *Eng. Appl. Artif. Intell.* **2023**, *125*, 106750. [\[CrossRef\]](#)
19. Li, Q.-Y.; Pan, Q.-K.; Sang, H.-Y.; Jing, X.-L.; Framiñán, J.M.; Li, W.-M. Self-adaptive population-based iterated greedy algorithm for distributed permutation flowshop scheduling problem with part of jobs subject to a common deadline constraint. *Expert Syst. Appl.* **2024**, *248*, 123278. [\[CrossRef\]](#)
20. Xiong, F.; Shi, J.; Jing, L.; Ping, A. Logic-based Benders decomposition methods for the distributed permutation flow shop scheduling problem with production and transportation cost. *Comput. Oper. Res.* **2025**, *179*, 107044. [\[CrossRef\]](#)
21. Wang, K.; Ma, W.-Q.; Luo, H.; Qin, H. Coordinated scheduling of production and transportation in a two-stage assembly flowshop. *Int. J. Prod. Res.* **2016**, *54*, 6891–6911. [\[CrossRef\]](#)
22. Yang, S.; Xu, Z. The distributed assembly permutation flowshop scheduling problem with flexible assembly and batch delivery. *Int. J. Prod. Res.* **2021**, *59*, 4053–4071. [\[CrossRef\]](#)
23. Qin, H.; Li, T.; Teng, Y.; Wang, K. Integrated production and distribution scheduling in distributed hybrid flow shops. *Memet. Comput.* **2021**, *13*, 185–202. [\[CrossRef\]](#)
24. Lei, D.; Li, J. Distributed energy-efficient assembly scheduling problem with transportation capacity. *Symmetry* **2022**, *14*, 2225. [\[CrossRef\]](#)

25. Li, W.; Chen, X.; Li, J.-Q.; Sang, H.-Y.; Han, Y.; Du, S.-L. An improved iterated greedy algorithm for distributed robotic flowshop scheduling with order constraints. *Comput. Ind. Eng.* **2022**, *164*, 107907. [\[CrossRef\]](#)
26. Xin, B.; Lu, S.; Wang, Q.; Deng, F.; Shi, X.; Cheng, J.; Kang, Y. Simultaneous scheduling of processing machines and automated guided vehicles via a multi-view modeling-based hybrid algorithm. *IEEE Trans. Autom. Sci. Eng.* **2024**, *21*, 4753–4767. [\[CrossRef\]](#)
27. Liu, X.; Wang, Y.; Liu, H.; Zhao, H.; Zhang, L. Transportation-integrated flexible job shop scheduling with a shared buffer. *Symmetry* **2026**, *18*, 1038. [\[CrossRef\]](#)
28. Liu, Z.; Sang, H.; Zhang, B.; Meng, L.; Meng, T. A learning-based two-stage multi-thread iterated greedy algorithm for co-scheduling of distributed factories and automated guided vehicles with sequence-dependent setup times. *IEEE Trans. Emerg. Top. Comput. Intell.* **2025**, *9*, 2208–2218. [\[CrossRef\]](#)
29. Berterottière, L.; Dauzère-Pérès, S.; Yugma, C. Flexible job-shop scheduling with transportation resources. *Eur. J. Oper. Res.* **2024**, *312*, 890–909. [\[CrossRef\]](#)
30. Mu, H.; Wang, Z.; Chen, J.; Zhang, G.; Wang, S.; Zhang, F. A flow shop scheduling method based on dual BP neural networks with multi-layer topology feature parameters. *Systems* **2024**, *12*, 339. [\[CrossRef\]](#)
31. Shi, Z.; Chen, H.; Yan, F.; Deng, X.; Hao, H.; Zhang, J.; Yin, Q. An improved NSGA-II for three-stage distributed heterogeneous hybrid flowshop scheduling with flexible assembly and discrete transportation. *Symmetry* **2025**, *17*, 1306. [\[CrossRef\]](#)
32. Momenikorbekandi, A.; Kalganova, T. Intelligent scheduling methods for optimisation of job shop scheduling problems in the manufacturing sector: A systematic review. *Electronics* **2025**, *14*, 1663. [\[CrossRef\]](#)
33. Pan, Q.-K.; Gao, L.; Li, X.-Y.; Framiñán, J.M. Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem. *Appl. Soft Comput.* **2019**, *81*, 105492. [\[CrossRef\]](#)
34. Huang, Y.-Y.; Pan, Q.-K.; Huang, J.-P.; Suganthan, P.N.; Gao, L. An improved iterated greedy algorithm for the distributed assembly permutation flowshop scheduling problem. *Comput. Ind. Eng.* **2021**, *152*, 107021. [\[CrossRef\]](#)
35. Yu, F.; Lu, C.; Zhou, J.; Yin, L. Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints. *Expert Syst. Appl.* **2024**, *239*, 122434. [\[CrossRef\]](#)
36. Zhang, Z.-Q.; Hu, R.; Qian, B.; Jin, H.-P.; Wang, L.; Yang, J.-B. A matrix cube-based estimation of distribution algorithm for the energy-efficient distributed assembly permutation flow-shop scheduling problem. *Expert Syst. Appl.* **2022**, *194*, 116484. [\[CrossRef\]](#)
37. Deng, J.; Wang, L.; Wang, S.-Y.; Zheng, X.-L. A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem. *Int. J. Prod. Res.* **2016**, *54*, 3561–3577. [\[CrossRef\]](#)
38. Cui, H.; Li, X.; Gao, L. An improved multi-population genetic algorithm with a greedy job insertion inter-factory neighborhood structure for distributed heterogeneous hybrid flow shop scheduling problem. *Expert Syst. Appl.* **2023**, *222*, 119805. [\[CrossRef\]](#)
39. Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *Eur. J. Oper. Res.* **2007**, *177*, 2033–2049. [\[CrossRef\]](#)
40. Tizhoosh, H.R. Opposition-based learning: A new scheme for machine intelligence. In Proceedings of the 2005 International Conference on Computational Intelligence for Modelling, Control and Automation, Vienna, Austria, 28–30 November 2005; pp. 695–701.
41. Yu, H.; Gao, K.-Z.; Ma, Z.-F.; Pan, Y.-X. Improved meta-heuristics with Q-learning for solving distributed assembly permutation flowshop scheduling problems. *Swarm Evol. Comput.* **2023**, *80*, 101335. [\[CrossRef\]](#)

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