

# An Offset Vector-Based Genetic Algorithm for Painted Body Storage Resequencing

Di Zhang

se\_diz@outlook.com

Zhengzhou Vocational College of Finance and Taxation

Yadi Li

Zhengzhou Vocational College of Finance and Taxation

Jiafang Han

Zhengzhou Vocational College of Finance and Taxation

Xueying Luo

Zhengzhou Vocational College of Finance and Taxation

---

## Research Article

### Keywords:

**Posted Date:** October 28th, 2026

**DOI:** <https://doi.org/10.21203/rs.3.rs-10456042/v1>

**License:** © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

**Additional Declarations:** No competing interests reported.

---

# An Offset Vector-Based Genetic Algorithm for Painted Body Storage Resequencing

Di Zhang<sup>1\*</sup>, Yadi Li<sup>1</sup>, Jiafang Han<sup>1</sup>, Xueying Luo<sup>1</sup>

<sup>1\*</sup>College of Big Data and Artificial Intelligence, Zhengzhou Vocational College of Finance and Taxation, Zhengzhou, 450001, Henan, China.

\*Corresponding author(s). E-mail(s): [se\\_diz@outlook.com](mailto:se_diz@outlook.com);

## Abstract

With the increasing demand for flexible and customized production in automotive manufacturing, the Painted Body Storage Resequencing (PBSR) problem has become a critical optimization challenge. However, existing studies often rely on heuristic or metaheuristic approaches with limited modeling clarity and insufficient ability to explicitly capture sequencing adjustments. To address these issues, this paper presents a more explicit and unified formulation of the PBSR problem, enabling consistent modeling and evaluation. Based on this formulation, we propose an Enhanced Genetic Algorithm with Offset Vector (EGA-OV), which integrates an offset vector representation to explicitly model sequence adjustments and a prior distribution mechanism to guide the search process. In addition, several heuristic-driven strategies are designed to improve solution quality and search efficiency. Extensive experiments on two real-world datasets demonstrate that the proposed method outperforms the state-of-the-art approaches in terms of both effectiveness and computational efficiency.

**Keywords:** painted body storage, car sequencing, genetic algorithm, scheduling optimization, manufacturing systems

## 1 Introduction

With the rapid advancement of e-commerce and mass customization, modern manufacturing systems are facing increasing demands for flexibility, responsiveness, and efficiency. In the automotive industry, production processes are typically organized into multiple interconnected workshops, such as stamping, welding, painting, and assembly [1, 2]. Among these stages, the transition between the Painting Workshop (PW) and

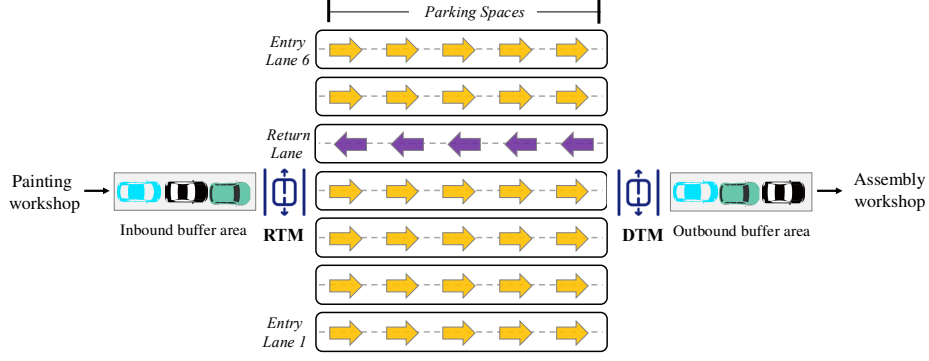


**Fig. 1** A toy example of painted body storage.

the Assembly Workshop (AW) plays a critical role in maintaining production continuity and efficiency. A fundamental challenge arises from the mismatch between the sequencing objectives of PW and AW [3, 4]. Specifically, the painting workshop aims to minimize color changeover costs by grouping vehicles with identical colors, whereas the assembly workshop requires a balanced workload distribution [5, 6] to avoid station overloads and ensure smooth operations. This inherent conflict leads to suboptimal production flow, increased operational costs, and reduced throughput if not properly managed. To address this issue, automotive manufacturers employ a Painted Body Storage (PBS) system [7] as an intermediate buffer, enabling the resequencing of car bodies before entering the assembly line.

The resulting Painted Body Storage Resequencing (PBSR) problem focuses on determining an optimal dispatching sequence from the buffer system, such that multiple operational objectives and constraints are satisfied simultaneously. From a modeling perspective, PBSR can be viewed as a constrained combinatorial optimization problem defined over a finite sequence space, where decisions involve vehicle ordering, buffer allocation, and dispatching operations under physical and operational constraints. Typical objectives [8–10] include minimizing sequencing violations (*e.g.*, hybrid spacing constraints), balancing workload-related attributes (*e.g.*, drive types), reducing buffer operations (*e.g.*, return lane usage), and improving overall scheduling efficiency. As illustrated in Figure 1, a standard PBS system consists of three main components: an inbound buffer, a multi-lane storage area, and an outbound buffer. The inbound buffer includes a Receiving Transverse Movement Machine (RTM), which assigns incoming vehicles to storage lanes. The multi-lane buffer typically contains several FIFO-structured lanes along with a return lane, providing limited flexibility for resequencing. The outbound buffer is managed by a Delivery Transverse Movement Machine (DTM), which determines the final dispatch sequence to the assembly workshop. The interaction between RTM and DTM, together with lane capacity and movement constraints, defines a complex and tightly coupled scheduling environment.

Traditionally, PBS scheduling has relied on manual decision-making or rule-based heuristics, where experienced operators construct feasible sequences based on domain knowledge [11–13]. However, such approaches are often inefficient and difficult to scale in the presence of large production volumes and diverse customization requirements.

Consequently, a variety of optimization methods have been proposed in the literature, including exact formulations, heuristic rules, and metaheuristic algorithms. Among them, metaheuristics such as Particle Swarm Optimization (PSO) [14–16], Genetic Algorithms (GA) [17–19], and Simulated Annealing (SA) [20, 21] have demonstrated strong performance in handling large-scale and complex scheduling problems due to their flexibility and computational efficiency.

Despite these advances, existing approaches still exhibit several limitations when applied to PBSR problems. First, many methods focus primarily on dispatching strategies without explicitly modeling sequence transformation mechanisms, which restricts their ability to explore the full solution space of resequencing decisions. Second, commonly used encoding schemes (*e.g.*, permutation-based representations) lack the flexibility to capture structured sequence adjustments under PBS constraints. Third, most metaheuristic methods rely on randomized search operators, leading to slow convergence and suboptimal performance, particularly in large-scale industrial scenarios where the search space is extremely large.

To address these challenges, this paper proposes an Enhanced Genetic Algorithm with Offset Vector (EGA-OV) for PBSR scheduling. The key idea is to transform the conventional sequence optimization problem into a structured search problem over an offset-based representation, where sequence adjustments are explicitly encoded as relative position shifts. This representation enables a more flexible and interpretable exploration of the solution space while preserving feasibility under PBS constraints. Furthermore, a prior distribution model is incorporated to guide the initialization and mutation processes, biasing the search toward promising regions and improving convergence efficiency. The main contributions of this paper are summarized as follows:

- We present a structured formulation of the PBSR problem, explicitly characterizing its objectives, constraints, and decision variables within a unified optimization framework.
- We propose an offset vector representation that transforms sequence optimization into a structured search problem, enabling more effective exploration of resequencing decisions.
- We develop a prior-guided genetic algorithm that improves convergence speed and solution quality by incorporating domain knowledge into the evolutionary process.
- We conduct extensive experiments on real-world datasets to validate the effectiveness and scalability of the proposed method compared with representative baseline approaches.

The remainder of this paper is organized as follows. Section 2 reviews related work on car sequencing and PBS scheduling. Section 3 presents the formal problem description and modeling framework. Section 4 introduces heuristic-driven sequencing strategies. Section 5 details the proposed EGA-OV algorithm. Section 6 reports experimental results and analysis. Finally, Section 7 concludes the paper and discusses future research directions.

## 2 Literature Review

The Painted Body Storage Resequencing (PBSR) problem is closely related to the broader research area of car sequencing and production scheduling in automotive manufacturing. In this section, we review the most relevant literature from three perspectives: classical car sequencing models, buffer-based resequencing systems, and metaheuristic optimization approaches.

**Classical Car Sequencing and Exact Methods.** The car sequencing problem has been extensively studied as a fundamental optimization problem in mixed-model assembly lines. Early research primarily focused on minimizing constraint violations under sequencing rules, such as capacity constraints and option spacing requirements. Comprehensive surveys by Boysen et al. [22] and Boysen et al. [23] provide detailed overviews of exact and heuristic approaches for car sequencing and resequencing problems. Exact methods, including mixed-integer linear programming (MILP) [24], dynamic programming [25], and branch-and-bound [26] techniques, have been proposed to obtain optimal or near-optimal solutions for small- to medium-scale instances. However, due to the combinatorial nature of the problem, these approaches often suffer from scalability limitations when applied to large-scale industrial settings.

**Buffer-Based Resequencing and PBS Systems.** To bridge the gap between upstream and downstream production stages, buffer-based resequencing systems, such as pull-off tables and painted body storage (PBS), have been widely adopted in practice. These systems introduce additional flexibility by allowing temporary storage and reordering of vehicles before final assembly. Research in this area has investigated both physical resequencing mechanisms [23, 27, 28] and hybrid strategies [29, 30] combining physical and virtual adjustments. For example, Boysen et al. [23] studied resequencing with limited buffer capacity, while Lai et al. [27] explored joint optimization of sequencing and buffer allocation. More recently, virtual resequencing approaches [29] decouple production sequencing from customer order fulfillment, improving system flexibility at the cost of increased planning complexity. Despite these advances, most existing studies focus on modeling system constraints and feasibility, while the representation of sequence transformation mechanisms remains relatively implicit.

**Metaheuristic Optimization for PBS Scheduling.** Given the complexity of PBSR problems, metaheuristic algorithms have been widely applied to obtain high-quality solutions within reasonable computational time. Techniques such as genetic algorithms (GA) [17, 18, 31, 32], particle swarm optimization (PSO) [14, 15], and simulated annealing (SA) [20, 21] have demonstrated strong performance in large-scale scheduling scenarios. In the context of PBS systems, Tian et al. [33] proposed a discrete small-world optimization algorithm to improve search efficiency, while Wang et al. [34] developed a GA-based method tailored for multi-lane buffer environments. Although these approaches effectively enhance dispatching strategies, they typically rely on permutation-based encodings and randomized search operators, which limit their ability to explicitly capture structured sequence adjustments and often lead to slow convergence.

**Research Gap and Motivation.** From the above review, it can be observed that existing studies either emphasize rigorous modeling of sequencing constraints (*e.g.*, exact methods and buffer-based models) or focus on algorithmic efficiency through

metaheuristic optimization. However, relatively limited attention has been paid to the design of structured representations that can explicitly model sequence transformation processes while maintaining computational efficiency. In particular, current encoding schemes lack the flexibility to describe fine-grained resequencing operations under PBS constraints, and most search processes do not effectively incorporate prior knowledge of feasible or promising sequence adjustments.

To address these limitations, this paper proposes a structured offset vector representation combined with a prior-guided genetic algorithm framework. By explicitly modeling sequence adjustments and guiding the search process toward promising regions, the proposed method aims to improve both solution quality and convergence efficiency for large-scale PBSR problems.

### 3 Problem Description

In this section, we present a structured formulation of the Painted Body Storage Resequencing (PBSR) problem. We first introduce the basic notation and decision variables, followed by the objective functions and system constraints.

#### 3.1 Notation and Decision Variables

Let  $V = \{v_1, v_2, \dots, v_n\}$  denote the set of car bodies processed in the system, where each vehicle  $v_i$  is associated with attributes such as power type (hybrid or non-hybrid) and drive type (4WD or 2WD). Let  $\pi = (v_{\pi(1)}, v_{\pi(2)}, \dots, v_{\pi(n)})$  denote the final dispatch sequence from the PBS to the assembly workshop.

The PBS system consists of a set of lanes  $L = \{1, 2, \dots, |L|\}$ , each with limited capacity. Let  $x_{i,l}$  denote the assignment of vehicle  $v_i$  to lane  $l$ , and let  $t_i^{in}$  and  $t_i^{out}$  denote the entry and exit times of vehicle  $v_i$  in the PBS system, respectively. In addition, let  $r_i \in \{0, 1\}$  indicate whether vehicle  $v_i$  utilizes the return lane.

The objective of PBSR is to determine a feasible dispatch sequence  $\pi$  together with scheduling decisions (lane assignment, movement, and dispatch timing) that satisfy system constraints while optimizing multiple performance criteria.

#### 3.2 Optimization Objectives

The PBSR problem is formulated as a multi-objective optimization problem. To facilitate evaluation and comparison, we adopt a weighted aggregation of multiple objectives.

**(1) Hybrid spacing constraint.** To ensure stable assembly operations, hybrid vehicles should be sufficiently separated. We define the violation cost as:

$$f_1 = \sum_{i < j} \mathbb{I}(v_{\pi(i)}, v_{\pi(j)} \text{ are hybrid and } j - i < 3), \quad (1)$$

where  $\mathbb{I}(\cdot)$  is an indicator function.

**(2) Drive type balancing.** To balance assembly workload, the ratio of 4WD and 2WD vehicles should be close to 1:1:

$$f_2 = \left| \frac{N_{4WD}}{N_{2WD}} - 1 \right|, \quad (2)$$

where  $N_{4WD}$  and  $N_{2WD}$  denote the number of vehicles of each type.

**(3) Return lane usage.** Excessive use of the return lane increases operational complexity:

$$f_3 = \sum_{i=1}^n r_i. \quad (3)$$

**(4) Scheduling time.** The total scheduling time is defined as:

$$f_4 = \sum_{i=1}^n (t_i^{out} - t_i^{in}). \quad (4)$$

The overall objective is defined as a weighted sum:

$$\min F = w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4, \quad (5)$$

where  $w_1, w_2, w_3, w_4$  are predefined weights reflecting operational priorities.

### 3.3 System Constraints

The PBSR problem is subject to operational constraints arising from the physical structure and control logic of the PBS system.

**(1) Lane capacity constraints:**

$$\sum_i x_{i,l} \leq C_l, \quad \forall l \in L, \quad (6)$$

where  $C_l$  is the capacity of lane  $l$ .

**(2) FIFO movement constraints:** Vehicles within each lane follow a first-in-first-out discipline, and overtaking is not allowed.

**(3) Machine operation constraints:** The Receiving Transverse Movement Machine (RTM) and Delivery Transverse Movement Machine (DTM) can process at most one vehicle at a time and operate in a non-preemptive manner.

**(4) Movement feasibility constraints:** Vehicle movements must follow the predefined directional flow in the PBS system, and once a movement starts, it cannot be interrupted or reassigned.

**(5) Dispatch constraints:** Vehicles can only be dispatched when they reach the first position of a lane, and the DTM cannot remain idle when dispatchable vehicles exist.

**(6) Return lane constraints:** Vehicles using the return lane must re-enter the scheduling process before being dispatched, and direct dispatch from the return lane is not allowed.

### 3.4 PBSR vs. Classical PBS

The PBSR problem is derived from the classical PBS scheduling framework [22, 23]. Compared with traditional models, the proposed formulation relaxes certain priority-based dispatching rules (*e.g.*, strict priority constraints in return lane and lane selection), thereby introducing additional flexibility for sequence optimization. This relaxation enlarges the feasible solution space and enables more effective resequencing, but also increases the complexity of the search process. Overall, PBSR can be viewed as a constrained sequence optimization problem with integrated routing and scheduling decisions, where the key challenge lies in balancing sequencing quality and operational efficiency under complex system constraints.

## 4 Heuristic-driven Optimization Strategies for PBSR Scheduling

This section presents the heuristic mechanisms used to efficiently explore the solution space of the PBSR problem. Instead of directly solving the optimization model defined in Section 3, we design structured neighborhood operations and heuristic evaluation strategies to guide the search process.

### 4.1 Sequencing Mechanisms as Neighborhood Operators

In the PBSR problem, a feasible solution is represented by a dispatch sequence  $\pi = (v_{\pi(1)}, \dots, v_{\pi(n)})$  together with corresponding scheduling decisions. Due to the physical constraints of the PBS system, not all sequence transformations are directly achievable. Therefore, we define a set of structured neighborhood operators that characterize feasible sequence adjustments under PBS operational rules. These operators are derived from practical scheduling mechanisms observed in real PBS systems and can be interpreted as elementary transformations of the sequence  $\pi$ . Specifically, we consider three types of neighborhood operators with increasing levels of flexibility: double-body replacement, three-body replacement, and return lane utilization.

**Double-Body Replacement Operator.** The double-body replacement operator performs a local swap between two adjacent vehicles in the sequence:

$$(v_{\pi(i)}, v_{\pi(i+1)}) \rightarrow (v_{\pi(i+1)}, v_{\pi(i)}). \quad (7)$$

This operator corresponds to a minimal perturbation of the sequence and can be implemented through precise coordination between the RTM and DTM without requiring additional buffer resources. The feasibility of this operation relies on the synchronization of inbound and outbound movements. Let  $t_e^l$  and  $t_d^l$  denote the entry and dispatch times for lane  $l$ , respectively. Under standard operating conditions, these satisfy:

$$t_e^l \approx t_d^l, \quad \forall l \in L, \quad (8)$$

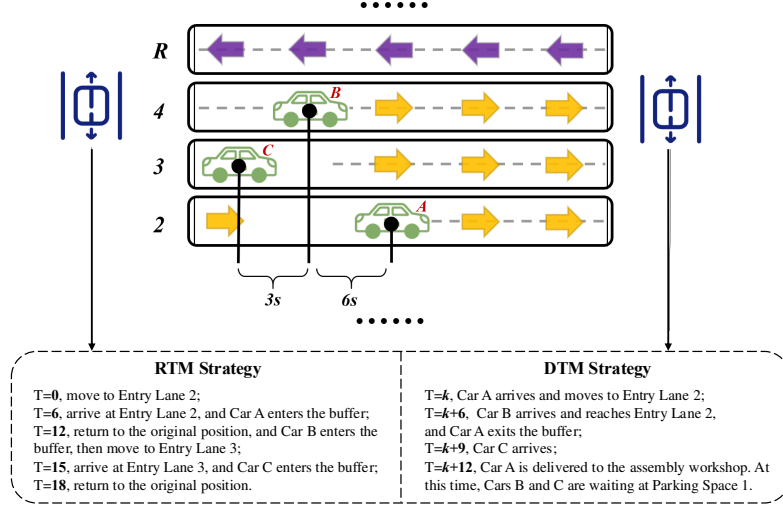


**Fig. 2** Illustration of the double-body replacement method, where cars B and C can be swapped in the final assembly sequence.

which implies that a limited time window exists for local resequencing:

$$\Delta t \approx \frac{t_e^l}{2}. \quad (9)$$

As illustrated in Fig. 2, vehicles A, B, and C are assigned to different lanes. By exploiting the timing gap when vehicles reach the first position of their lanes, the DTM can dispatch vehicles B and C in reversed order. This mechanism enables fine-grained corrections in the sequence, particularly useful for reducing local violations in the objective function.

**Three-Body Replacement Operator.** To achieve more flexible sequence transformations, we extend the local neighborhood to three consecutive vehicles:

$$(v_{\pi(i)}, v_{\pi(i+1)}, v_{\pi(i+2)}) \rightarrow \sigma(v_{\pi(i)}, v_{\pi(i+1)}, v_{\pi(i+2)}), \quad (10)$$

where  $\sigma(\cdot)$  denotes a permutation of three elements.

Compared with the double-body operator, this mechanism enlarges the neighborhood by allowing more complex reorderings. In practice, such transformations are realized by temporarily delaying one vehicle using the return lane, thereby creating additional scheduling flexibility. Let  $t_{dr}^l$  denote the processing time associated with return lane usage. The additional time window introduced can be approximated as:

$$t_{var} = \max_{l \in L} \left( t_{dr}^l - \frac{t_e^l}{2} \right). \quad (11)$$

As shown in Fig. 3, vehicle A is first redirected to the return lane to create a controlled delay, allowing vehicles B, C, and D to accumulate and be reordered at

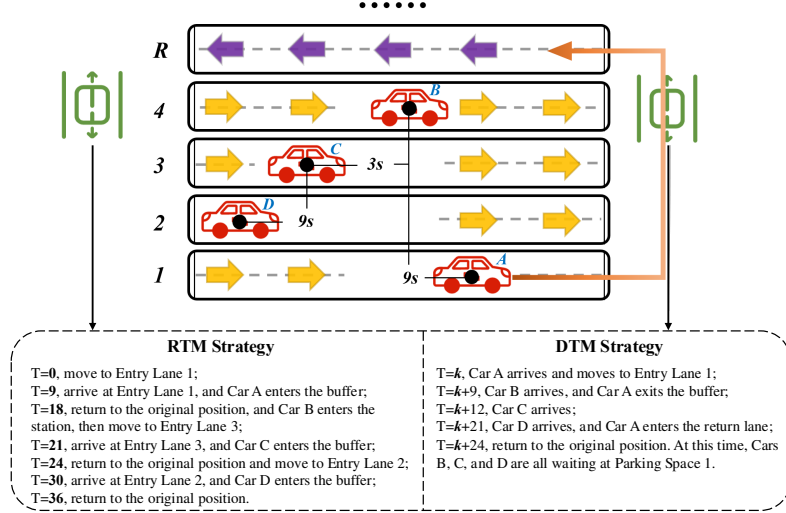


**Fig. 3** Illustration of the three-body replacement method, where cars B, C, and D can be reordered by leveraging the return lane.

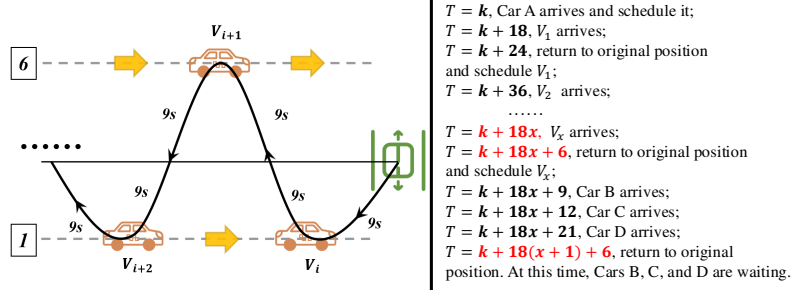


**Fig. 4** Illustration of the return lane utilization method, which allows for major sequencing modifications by temporarily storing cars before reinsertion.

the dispatch stage. This operator enables medium-scale sequence adjustments and is particularly effective for resolving multiple interacting constraint violations.

**Return Lane Operator.** The return lane operator provides a global transformation mechanism by allowing a vehicle to be removed from its current position and reinserted at a later position in the sequence:

$$v_{\pi(i)} \rightarrow v_{\pi(j)}, \quad j > i. \quad (12)$$

This operator significantly enlarges the feasible solution space but incurs additional operational costs. The available adjustment window can be estimated as:

$$\Delta t_{var} = \max_{l \in L} (t_{dr}^l - t_e^l). \quad (13)$$

As illustrated in Fig. 4, a vehicle is first moved to the return lane and temporarily stored, allowing subsequent vehicles to be rearranged before reinsertion. By maintaining a time buffer through coordinated DTM movements, this mechanism supports large-scale sequence restructuring that cannot be achieved by local operators alone.

**Summary.** The above three operators define a hierarchical neighborhood structure for the PBSR problem. The double-body operator enables fine-grained local adjustments, the three-body operator provides moderate flexibility through controlled delays, and the return lane operator supports global sequence transformations. Together, they allow efficient exploration of the solution space while respecting PBS operational constraints.

## 4.2 Heuristic Evaluation and Search Guidance

To guide the exploration of the solution space, we design a heuristic evaluation and selection mechanism based on the objective functions defined in Section 3. Given a candidate sequence  $\pi$ , its quality is assessed through the aggregated objective function:

$$F(\pi) = w_1 f_1(\pi) + w_2 f_2(\pi) + w_3 f_3(\pi) + w_4 f_4(\pi), \quad (14)$$

where  $f_1, f_2, f_3, f_4$  correspond to hybrid spacing violations, drive type imbalance, return lane usage, and scheduling time, respectively.

**Neighborhood Evaluation.** Given a current solution  $\pi$ , a neighborhood operator generates a candidate solution  $\pi' \in \mathcal{N}(\pi)$ . The quality improvement is measured by:

$$\Delta F = F(\pi') - F(\pi). \quad (15)$$

A candidate solution is considered beneficial if  $\Delta F < 0$ , indicating an improvement in overall objective value.

**Objective-aware Selection Strategy.** To balance multiple objectives, we further analyze the contribution of each component:

$$\Delta f_k = f_k(\pi') - f_k(\pi), \quad k = 1, 2, 3, 4. \quad (16)$$

A candidate solution  $\pi'$  is prioritized if it satisfies one of the following conditions:

- It strictly improves the overall objective ( $\Delta F < 0$ );
- It significantly reduces critical violations (*e.g.*,  $\Delta f_1 < 0$ ) even with slight increases in other components;
- It achieves a better trade-off between sequencing quality ( $f_1, f_2$ ) and operational cost ( $f_3, f_4$ ).

**Return Lane Cost Control.** Since the return lane operator introduces additional operational cost, its usage is regulated by a penalty-aware mechanism. Specifically, a candidate solution  $\pi'$  involving return lane operations is accepted only if:

$$w_1 \Delta f_1 + w_2 \Delta f_2 \geq \gamma(w_3 \Delta f_3 + w_4 \Delta f_4), \quad (17)$$

where  $\gamma > 0$  is a control parameter that balances sequencing improvement against operational cost.

**Scheduling Efficiency Regulation.** To avoid excessive delays, we impose a soft constraint on scheduling time:

$$f_4(\pi') \leq (1 + \epsilon)f_4(\pi), \quad (18)$$

where  $\epsilon$  is a tolerance parameter. This ensures that improvements in sequencing quality do not significantly degrade system throughput.

Therefore, the search process can be interpreted as a guided exploration of the solution space, where neighborhood operators define feasible transformations and the evaluation function  $F(\pi)$ , together with the above selection rules, determines the acceptance and prioritization of candidate solutions. This mechanism provides a principled bridge between the formal optimization model and the metaheuristic algorithm introduced in the next section.

## 5 Enhanced Genetic Algorithm with Offset Vector (EGA-OV)

In this section, we propose an Enhanced Genetic Algorithm with Offset Vector (EGA-OV) for solving the PBSR problem. The key idea is to transform the original sequence optimization problem into a structured search problem in an alternative representation space, where sequence adjustments are explicitly parameterized. This transformation enables more efficient exploration of the solution space and facilitates the incorporation of prior knowledge.

### 5.1 Offset Vector as a Structured Search Representation

Instead of directly optimizing the dispatch sequence  $\pi$ , we introduce an offset-based representation that reparameterizes the search space. Let  $O = (o_1, \dots, o_n)$  denote the initial positions of vehicles, where  $o_i = i$ . We define an offset vector:

$$S = (s_1, s_2, \dots, s_n), \quad (19)$$

where each  $s_i$  represents the displacement applied to vehicle  $v_i$ .

The modified position of each vehicle is given by:

$$o'_i = o_i + g(s_i), \quad (20)$$

where

$$g(s_i) = \begin{cases} 0, & s_i = 0, \\ s_i + 1, & s_i > 0. \end{cases} \quad (21)$$

The resulting dispatch sequence  $\pi(S)$  is obtained by sorting vehicles according to  $o'_i$ :

$$\pi(S) = \text{Sort}(\{v_i\} \text{ by } o'_i). \quad (22)$$

This formulation establishes a mapping:

$$S \rightarrow \pi(S), \quad (23)$$

which transforms the original combinatorial sequence optimization problem into a structured search over the offset space.

**Interpretation.** The offset vector provides a natural interpretation of sequence transformations:

- Small offsets ( $s_i \approx 0$ ) correspond to local adjustments (double-body operations);
- Moderate offsets correspond to multi-vehicle permutations (three-body operations);
- Large offsets correspond to global repositioning via the return lane.

Therefore, the offset representation unifies different neighborhood operators into a single parametric framework, enabling smooth transitions between local and global search.

## 5.2 Genetic Operations in Offset Space

Based on the above representation, we perform genetic operations directly in the offset space.

**(1) Initialization.** An initial population  $\{S^{(k)}\}$  is generated by sampling offset vectors, which are then mapped to sequences  $\pi(S^{(k)})$  and evaluated using  $F(\pi)$ .

**(2) Crossover.** Given two parent solutions  $S^{(p1)}$  and  $S^{(p2)}$ , offspring are generated by combining their offset components:

$$s_i^{(c)} = \begin{cases} s_i^{(p1)}, & \text{with probability 0.5,} \\ s_i^{(p2)}, & \text{otherwise.} \end{cases} \quad (24)$$

**(3) Mutation.** Mutation is performed by perturbing selected components:

$$s_i \leftarrow s_i + \delta_i, \quad (25)$$

where  $\delta_i$  is sampled from a predefined distribution.

**(4) Evaluation.** Each candidate solution  $S$  is mapped to  $\pi(S)$  and evaluated using:

$$F(S) := F(\pi(S)). \quad (26)$$

This ensures consistency with the optimization model defined in Section 3.

## 5.3 Prior Distribution-Guided Search

To improve search efficiency, we introduce a prior distribution over the offset variables. Instead of sampling offsets uniformly, we define a probability distribution:

$$P(s_i) \propto \phi(s_i), \quad (27)$$

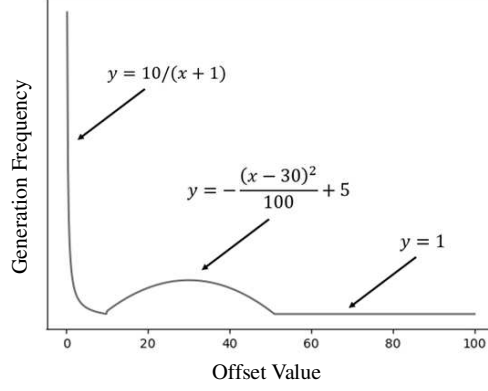


**Fig. 5** Prior distribution guiding offset sampling.

where  $\phi(\cdot)$  encodes domain knowledge about desirable sequence transformations.

Specifically, the distribution is designed as:

$$\phi(s_i) = \begin{cases} \frac{1}{s_i+1}, & 0 \leq s_i \leq \tau_1, \\ (s_i - \tau_1)^2, & \tau_1 < s_i \leq \tau_2, \\ \text{const}, & s_i > \tau_2, \end{cases} \quad (28)$$

where  $\tau_1$  and  $\tau_2$  are threshold parameters.

As illustrated in Fig. 5, this distribution favors small offsets while allowing occasional larger adjustments. The prior is incorporated into both initialization and mutation processes:

- Initialization: offsets are sampled according to  $P(s_i)$ ;
- Mutation: perturbations  $\delta_i$  follow the same distribution.

**Discussion.** The prior distribution introduces a probabilistic bias toward promising regions of the search space. Combined with the structured offset representation, it enables efficient exploration by balancing local refinement and global diversification. Compared with standard GA approaches based on permutation encoding, the proposed framework significantly improves convergence efficiency and solution quality in large-scale PBSR problems.

#### 5.4 Algorithmic Framework of EGA-OV

The proposed EGA-OV algorithm operates in the offset space and evaluates solutions in the sequence space via the mapping  $S \rightarrow \pi(S)$ . The overall framework (See Alg. 1) integrates representation, evaluation, and evolutionary search in a unified manner. The key idea of EGA-OV is that all evolutionary operations are performed in the offset space, while evaluation is conducted in the sequence space. This separation enables efficient exploration while preserving consistency with the original optimization objective.

---

**Algorithm 1:** The Framework of EGA-OV

---

**Input:** Car set  $V$ , population size  $pop$ , mutation rate  $mut$ , iterations  $iter$

**Output:** Best solution  $S^*$  and corresponding sequence  $\pi(S^*)$

- 1 Initialize population  $\{S^{(k)}\}$  using prior distribution  $P(s_i)$ ;
- 2 Evaluate each individual using  $F(S^{(k)}) = F(\pi(S^{(k)}))$ ;
- 3 Set best solution  $S^* = \arg \min F(S^{(k)})$ ;
- 4 **for**  $n = 1$  **to**  $iter$  **do**
- 5     Select parents from population via fitness-based selection;
- 6     Generate offspring via crossover and mutation in offset space;
- 7     Evaluate offspring using  $F(S)$ ;
- 8     Update population using elitism strategy;
- 9     Update best solution:

$$S^* \leftarrow \arg \min_{S \in \text{population}} F(S)$$

- 10 **return**  $S^*, \pi(S^*)$ ;
- 

**Time Complexity.** The computational complexity of EGA-OV is dominated by the evaluation step. For a population of size  $pop$ , each individual requires constructing  $\pi(S)$  and computing  $F(\pi)$ , which involves  $O(|V|)$  operations plus simulation cost  $T_{sim}$ . Therefore, each generation requires:

$$O(pop \cdot (|V| + T_{sim})). \quad (29)$$

Over  $iter$  iterations, the total complexity is:

$$O(iter \cdot pop \cdot (|V| + T_{sim})). \quad (30)$$

This complexity is comparable to standard GA-based approaches, while the structured representation improves search efficiency in practice.

## 5.5 Theoretical Analysis

In this section, we analyze several fundamental properties of the proposed EGA-OV framework, focusing on representation capability and search behavior.

**Proposition 1 (Representation Completeness).** The offset representation  $S$  is capable of generating any feasible dispatch sequence  $\pi$ .

*Sketch.* For any permutation  $\pi$ , there exists a corresponding set of offsets that can reorder the initial sequence into  $\pi$  via sorting based on  $o'_i$ . Since sorting with real-valued keys can represent arbitrary permutations, the mapping  $S \rightarrow \pi(S)$  is surjective over the solution space.

**Proposition 2 (Search Space Structure).** The offset representation induces a structured neighborhood over the sequence space.

*Explanation.* Small perturbations in  $S$  (i.e., small  $\Delta s_i$ ) lead to local changes in  $\pi(S)$ , corresponding to double-body or three-body operations. Larger perturbations result in global sequence adjustments. Therefore, the offset space naturally encodes a hierarchy of neighborhood structures, enabling both local and global search.

**Proposition 3 (Convergence Behavior).** Under standard genetic algorithm assumptions (elitism and nonzero mutation probability), the best fitness value in EGA-OV is non-increasing over iterations.

$$F(S_{n+1}^*) \leq F(S_n^*). \quad (31)$$

*Explanation.* The elitism strategy ensures that the best solution found so far is preserved. Therefore, the sequence of best objective values is monotonically improving (non-increasing in minimization form) and bounded below, implying convergence to a locally optimal solution.

**Proposition 4 (Effect of Prior Distribution).** The prior distribution  $P(s_i)$  biases the search toward low-cost regions of the solution space.

*Explanation.* Since small offsets are more likely under  $P(s_i)$ , the algorithm favors solutions with lower  $f_3$  and  $f_4$ . At the same time, nonzero probability of larger offsets ensures sufficient exploration, preventing premature convergence.

The above properties explain why EGA-OV achieves efficient and stable performance: the offset representation provides a structured search space, while the prior distribution balances exploitation and exploration.

## 6 Experiment

### 6.1 Experimental Setup

**Datasets.** To evaluate the effectiveness of our proposed EGA-OV algorithm for the PBSR problem, we utilize two datasets. Dataset 1 consists of 50,000 desensitized production records collected from real-world car workshop A, while Dataset 2 comprises 5,000 desensitized records from car workshop B. The records are formatted as presented in Table 1. To construct experimental instances with different scales, we randomly sample subsets from the original datasets. Specifically, instances with  $|V| = 10,000$  (Dataset 1) and  $|V| = 1,000$  (Dataset 2) are used in the main experiments, which allows us to evaluate both performance and scalability of the compared methods.

**Table 1** Example of data format in Datasets 1 and 2.

| Enter Car Index | Car Type | Power  | Drive      |
|-----------------|----------|--------|------------|
| 1               | B        | fuel   | two wheel  |
| 2               | A        | hybrid | two wheel  |
| 3               | A        | hybrid | four wheel |
| 4               | A        | fuel   | four wheel |
| 5               | B        | hybrid | two wheel  |
| ...             | ...      | ...    | ...        |

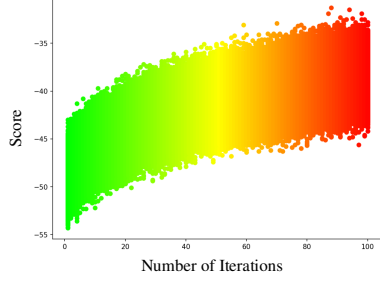


**Fig. 6** Results of GA on dataset 1.

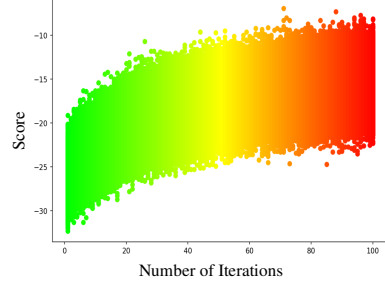


**Fig. 7** Results of GA on dataset 2.

**Compared Methods.** We compare the proposed EGA-OV algorithm with two baseline methods: Native and GA [35]. The Native method represents the basic and intuitive scheduling algorithm, where all cars are directly transported to the final assembly docking point via Inbound Lane 4 without any additional scheduling. The GA method refers to the scheduling plan generated by the genetic algorithm under the corresponding PBS constraints. Further, we compare EGA-OV with two state-of-the-art methods: MHA [36] and VRHA [29]. The MHA method is a meta-heuristic approach composed of the tabu search method with global perturbation operators (TSGPO) for initial sequence determination in PW and the variable neighborhood search method under the nested partition framework (NPVNS) for optimizing the PBS inbound process. The VRHA method is a heuristic-based resequencing algorithm that integrates virtual resequencing into the storage/retrieval subprocess, enhancing the resequencing process by jointly applying optimized fill and release approaches.

**Metrics.** In the PBSR problem, we define four optimization objectives: the sequencing of hybrid and non-hybrid cars, the sequencing of four-wheel-drive and two-wheel-drive cars, the utilization of return lanes, and the total scheduling time. Thus, We use weighted scores of four objectives and time cost to evaluate the effectiveness and efficiency of the above algorithm.

**Parameter Settings.** To ensure the effectiveness of the EGA-OV algorithm in solving the PBSR problem, we set the default hyperparameters through preliminary experimental tuning. The hyperparameters include a population size of  $pop = 6000$ , a mutation rate of  $mut = 0.01$ , and a total of  $iter = 500$  iterations. In Dataset 1, the number of cars  $|V|$  is set to 10,000, whereas in Dataset 2,  $|V|$  is set to 1,000. For the PBS setting, the number of entry lanes is set to 6 and the number of parking spaces is set to 10. The weight coefficients in the objective function are set to  $(w_1, w_2, w_3, w_4) = (0.4, 0.3, 0.2, 0.1)$ , reflecting the relative importance of different operational objectives. All algorithms are implemented under the same simulation environment, and each experiment is repeated multiple times with different random seeds.

## 6.2 Experimental results and analysis

**Exp-1: Effectiveness of strategy search.** To evaluate the effectiveness of the GA algorithm on strategy search, we record the first 100 iterations of the painting car sequence optimization across two datasets. As shown in Fig. 6 and Fig. 7, the strategy

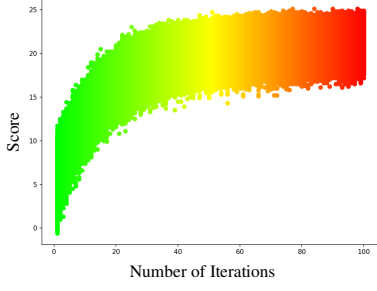


**Fig. 8** Results of EGA-OV on dataset 1.

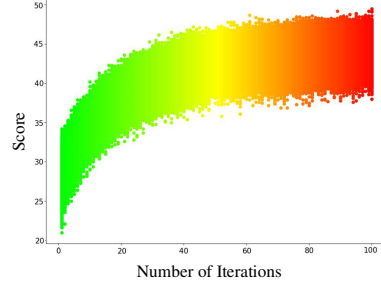


**Fig. 9** Results of EGA-OV on dataset 2.

scores for all individual genes within the population at each iteration are visualized as a scatter plot. From these figures, several limitations of the GA algorithm can be observed. First, the strategy scores remain generally low and negative even after 100 iterations. Second, the GA algorithm exhibits minimal improvement compared to the Native method and slow convergence, achieving an increase of merely 10 points over the entire iteration process. These findings suggest that the GA algorithm struggles to efficiently explore high-quality solutions.

Through an in-depth analysis of the solutions generated by the GA algorithm, we can find that the algorithm tends to overuse return lanes when exploring the solution space. This issue arises due to the random generation of offset vectors, where the offset values for each car body are sampled within predefined lower and upper bounds. Specifically, any car with an offset value greater than 3 is required to return. Given that the two datasets contain 10,000 and 1,000 cars, respectively, the initial population often includes scheduling plans that rely on thousands of return lanes. Furthermore, during gene mutations, each offset value has a probability of only 0.3% and 0.03% to be randomly assigned a value of 3 or less. This inherent limitation significantly contributes to the poor performance and slow convergence of the GA algorithm.

**Exp-2: Effectiveness of prior distribution.** To evaluate the effectiveness of prior distributions, we conduct experiments using EGA-OV algorithm on two datasets, presenting the score distributions constrained by prior distributions over the first 100 iterations. The results are shown in Fig. 8 and Fig. 9, and we can observe that the EGA-OV algorithm not only greatly improves the score of the initial solution, but also has a more obvious convergence effect in the iteration. This is because the prior distribution serve as a heuristic mechanism in the EGA-OV algorithm. Its introduction does not restrict the EGA-OV algorithm’s exploration of the solution space, but only makes it easier to explore part of the decomposition. Once the EGA-OV algorithm finds a high-quality solution with a high offset value, the chromosome, having a high score, can be preserved in the population for an extended period and continuously propagated. Thus, the use of the prior distribution can improve the ability of the EGA-OV algorithm to solve the PBSR problem.

**Exp-3: Optimization effect of EGA-OV.** To evaluate the optimization effect of EGA-OV, we conduct a comparative analysis against two baseline methods across two datasets. The results are presented in Tables 2, 3, 4, and 5. As observed from these

**Table 2** Results of EGA-OV vs. Native on Dataset 1.

| Objectives $\times$ weight | Native  | EGA-OV  | Difference     |
|----------------------------|---------|---------|----------------|
| O1 Score $\times 0.4$      | -28.800 | -14.000 | +14.800        |
| O2 Score $\times 0.3$      | 23.700  | 25.500  | +1.800         |
| O3 Score $\times 0.2$      | 20.000  | 18.600  | -1.400         |
| O4 Score $\times 0.1$      | 10.000  | 9.441   | -0.559         |
| Total score                | 24.900  | 39.541  | <b>+14.641</b> |

**Table 3** Results of EGA-OV vs. GA on Dataset 1.

| Objectives $\times$ weight | GA      | EGA-OV  | Difference     |
|----------------------------|---------|---------|----------------|
| O1 Score $\times 0.4$      | -20.000 | -14.000 | +6.000         |
| O2 Score $\times 0.3$      | 24.300  | 25.500  | +1.200         |
| O3 Score $\times 0.2$      | 16.000  | 18.600  | +2.600         |
| O4 Score $\times 0.1$      | 8.876   | 9.441   | +0.565         |
| Total score                | 29.176  | 39.541  | <b>+10.365</b> |

**Table 4** Results of EGA-OV vs. Native on Dataset 2.

| Objectives $\times$ weight | Native | EGA-OV | Difference     |
|----------------------------|--------|--------|----------------|
| O1 score $\times 0.4$      | -8.00  | 2.800  | +10.800        |
| O2 score $\times 0.3$      | 23.700 | 25.800 | +2.100         |
| O3 score $\times 0.2$      | 20.000 | 18.000 | -2.000         |
| O4 score $\times 0.1$      | 10.000 | 9.177  | -0.823         |
| Total score                | 45.700 | 55.777 | <b>+10.077</b> |

**Table 5** Results of EGA-OV vs. GA on Dataset 2.

| Objectives $\times$ weight | GA     | EGA-OV | Difference    |
|----------------------------|--------|--------|---------------|
| O1 Score $\times 0.4$      | 3.600  | 2.800  | +0.400        |
| O2 Score $\times 0.3$      | 23.700 | 25.800 | +2.100        |
| O3 Score $\times 0.2$      | 16.000 | 18.000 | +2.000        |
| O4 score $\times 0.1$      | 8.694  | 9.177  | +0.483        |
| Total score                | 51.944 | 55.777 | <b>+3.833</b> |

tables, the EGA-OV algorithm not only generates high-quality solutions but also effectively utilizes the optimization space freed by eliminating the two PBS constraints in the PBSR problem, further enhancing solution quality. By incurring minimal time and return costs, EGA-OV integrates our proposed double-body and three-body replacement methods into the scheduling strategy generation process. This enables efficient adjustment of the car sequences from PW, ultimately achieving a high score on the optimization objectives.

**Exp-4: Effect of number of cars  $|V|$ .** To analyze the impact of the number of cars on scheduling performance, we vary  $|V|$  from 5000 to 30000 on Dataset 1 (D1) and from

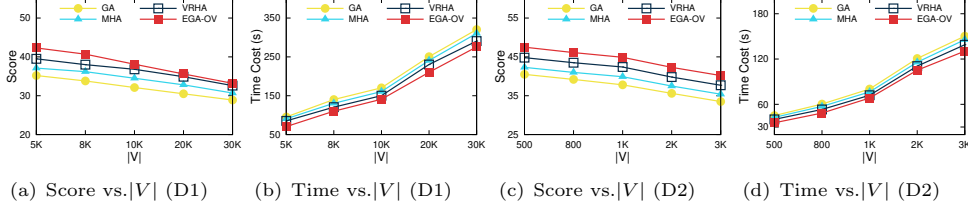


**Fig. 10** Results of varying the number of cars  $|V|$  on Dataset 1 and Dataset 2.

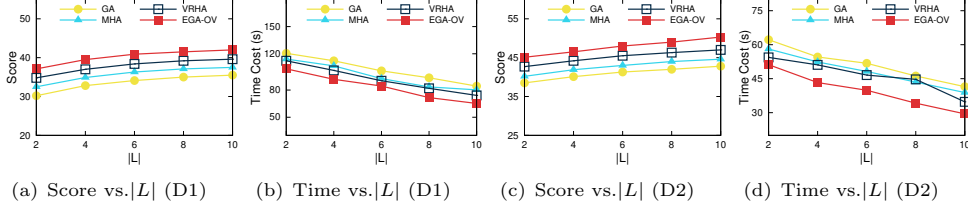


**Fig. 11** Results of varying the number of entry lanes  $|L|$  on Dataset 1 and Dataset 2.

500 to 3000 on Dataset 2 (D2). The results, as presented in Figs 10(a), 10(b), 10(c) and 10(d), indicate that EGA-OV consistently achieves higher scores and lower time costs compared to GA, MHA, and VRHA across different values of  $|V|$ . Specifically, as  $|V|$  increases, the performance gap between EGA-OV and other methods becomes more pronounced, showcasing its robustness in handling large-scale scheduling problems. This can be attributed to EGA-OV’s ability to dynamically optimize resource allocation and car resequencing, leveraging offset vector representation and prior distribution model. In contrast, GA and MHA struggle with scalability due to exponential search space expansion, while VRHA lacks the offset vector of EGA-OV. Although scheduling time increases for all methods as  $|V|$  grows, EGA-OV maintains a significant efficiency advantage by reducing redundant computations and minimizing processing overhead, making it well-suited for high-demand scenarios.

**Exp-5: Effect of number of entry lanes  $|L|$ .** We analyze the effect of varying the number of entry lanes  $|L|$  on scheduling performance. We experiment with values from 2 to 10 on both Dataset 1 and Dataset 2. The results, displayed in Figs 11(a), 11(b), 11(c) and 11(d), demonstrate that EGA-OV consistently outperforms other methods in terms of both score and time efficiency. Notably, as  $|L|$  increases, the performance improvement diminishes, suggesting a saturation point in efficiency gains. This phenomenon can be attributed to the fact that increasing  $|L|$  initially alleviates congestion and enhances vehicle throughput, leading to significant improvements in scheduling efficiency. However, as  $|L|$  continues to grow, the marginal benefit declines due to the inherent constraints of other system factors such as available parking spaces and the overall scheduling algorithm’s ability to process tasks effectively. When the number of entry lanes reaches a certain threshold, the bottleneck shifts from lane availability to downstream processing capacity, leading to the observed saturation effect.

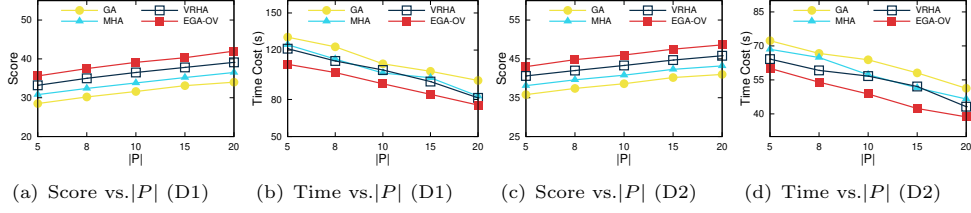


**Fig. 12** Results of varying the number of parking spaces  $|P|$  on Dataset 1 and Dataset 2.

**Exp-6: Effect of number of parking spaces  $|P|$ .** We analyze the impact of varying the number of parking spaces  $|P|$  on scheduling performance. We experiment with values from 5 to 20 on both Dataset 1 and Dataset 2. The results, displayed in Figs 12(a), 12(b), 12(c) and 12(d), show that EGA-OV consistently achieves superior performance in terms of both score and time efficiency. The improvement rate diminishes as  $|P|$  increases, indicating a plateau effect in optimization benefits. This is because increasing  $|P|$  enhances the flexibility of vehicle scheduling, reducing conflicts and congestion, which leads to an improvement in scheduling efficiency. However, after a certain threshold, the additional parking spaces no longer contribute significantly to performance gains because the system already operates near optimal capacity. This diminishing return suggests that an excessive number of parking spaces does not further enhance the optimization effect.

## 7 Conclusion

In this paper, we investigate the PBSR problem and propose an enhanced genetic algorithm, EGA-OV, to address it. A more rigorous and unified problem formulation is first established by explicitly defining objective functions and system constraints, enabling consistent evaluation across methods. Based on this formulation, EGA-OV integrates an offset vector representation and a prior distribution mechanism to effectively guide the resequencing process. Experimental results on two real-world datasets demonstrate that the proposed method consistently outperforms baseline and state-of-the-art approaches in terms of solution quality and computational efficiency, while maintaining stable performance across different problem scales.

Future work will consider dynamic PBSR settings with stochastic arrivals and real-time decision requirements. In addition, integrating learning-based models with heuristic optimization will be explored to further enhance adaptability and scalability.

## Declarations

- **Funding.** Zhengzhou Finance and Taxation Vocational College School-level Research Project ZCY2025004, and Henan Province Key Scientific Research Project Plan for Higher Education Institutions 23B520048.
- **Conflict of interest.** The authors declare no conflict of interest.
- **Ethics approval and consent to participate.** Not applicable.
- **Consent for publication.** Not applicable.

- **Data availability.** Data are available from: [https://github.com/Delia95/EGA\\_OV](https://github.com/Delia95/EGA_OV).
- **Code availability.** Code is available from: [https://github.com/Delia95/EGA\\_OV](https://github.com/Delia95/EGA_OV).
- **Author contributions.** Di Zhang designed the overall research framework and developed the main algorithm. Yadi Li contributed to algorithm implementation and experimental design. Jiafang Han assisted in data processing and experimental validation. Xueying Luo provided supervision and manuscript revision.

## References

- [1] Zhou, B., He, Z.: A static semi-kitting strategy system of JIT material distribution scheduling for mixed-flow assembly lines. *Expert Syst. Appl.* **184**, 115523 (2021)
- [2] Inman, R.: Asrs sizing for recreating automotive assembly sequences. *Int. J. Prod. Res.* **41**(5), 847–863 (2003)
- [3] Becker, C., Scholl, A.: A survey on problems and methods in generalized assembly line balancing. *Eur. J. Oper. Res.* **168**(3), 694–715 (2006)
- [4] Saeed, M., Demasure, T., Aghezzaf, E.-H., Cottyn, J.: Optimization-based estimation of the execution time of a robotic assembly task sequence. *Int. J. Adv. Manuf. Technol.* **130**(11), 5315–5328 (2024)
- [5] Takai, S.: An approach to integrate product and process design using augmented liaison diagram, assembly sequencing, and assembly line balancing. *J. Mech. Des.* **143**(10), 102002 (2021)
- [6] Özdemir, R., Kula, U., Helal, M.: Assembly line balancing with product and operator oriented sequence dependent task times using tabu search algorithm. *Int. J. Interact. Des. Manuf.* **18**(6), 4209–4222 (2024)
- [7] Chang, F., Dong, M., Liu, M., Wang, L., Duan, Y.: A lightweight appearance quality assessment system based on parallel deep learning for painted car body. *IEEE Trans. Instrum. Meas.* **69**(8), 5298–5307 (2020)
- [8] Xiang, C., Liu, B., Zhao, L., Zheng, X.: Three-dimension deep model for body mass index estimation from facial image sequences with different poses. *J. Vis. Commun. Image Represent.* **107**, 104381 (2025)
- [9] Ji, S., Xue, Y., Zhu, G.: Physical internet-enabled automobile production-distribution joint optimisation with multistage workshop. *Kybernetes* **52**(3), 898–920 (2023)
- [10] Li, D., Xie, F.: Design and debugging of transmission handling manipulator control system in automobile production line. In: *Proc. of the 2023 Int’l Conf. on AI and Metaverse in Supply Chain Management*, p. 8 (2023)
- [11] Moon, D.H., Song, C., Ha, J.H.: A dynamic algorithm for the control of

- automotive painted body storage. *Simul.* **81**(11), 773–787 (2005)
- [12] Li, Z., Shi, Y., Wu, L.: Research on mixed flow line balancing and scheduling optimization with multiple constraints. *Journal of System Simulation* **35**, 27–40 (2023)
  - [13] Yurtkuran, A., Yagmahan, B., Emel, E.: A novel artificial bee colony algorithm for the workforce scheduling and balancing problem in sub-assembly lines with limited buffers. *Appl. Soft Comput.* **73**, 767–782 (2018)
  - [14] Wang, X., Cao, D.: Multi-strategy particle swarm optimization algorithm based on evolution ability. *Computer Engineering and Applications* **59**, 78–86 (2023)
  - [15] Hu, G., Cheng, M., Houssein, E.H., Jia, H.: CMPSO: A novel co-evolutionary multigroup particle swarm optimization for multi-mission uavs path planning. *Adv. Eng. Informatics* **63**, 102923 (2025)
  - [16] Zhang, J., Chen, C., Su, S., Hu, W., Zhu, A.: Assembly sequence planning based on hybrid sos-pso algorithm. *Int. J. Adv. Manuf. Technol.* **136**(11), 5487–5504 (2025)
  - [17] Triki, H., Mellouli, A., Hachicha, W., Masmoudi, F.: A hybrid genetic algorithm approach for solving an extension of assembly line balancing problem. *Int. J. Comput. Integr. Manuf.* **29**(5), 504–519 (2016)
  - [18] Xu, Z., Shen, D., Nie, T., Kou, Y.: Hybrid feature selection algorithm combining information gain ratio and genetic algorithm. *Journal of Software* **33**, 1128–1140 (2022)
  - [19] Hao, L., Zou, Z., Liang, X.: Solving multi-objective energy-saving flexible job shop scheduling problem by hybrid search genetic algorithm. *Comput. Ind. Eng.* **200**, 110829 (2025)
  - [20] Du, L., Wang, Y., Xuan, Z., Ye, T., Zhang, Y.: Research on knitted production line scheduling based on improved simulated annealing algorithm. *Computer Engineering and Applications* **59**, 304–312 (2023)
  - [21] Oliveira Neto, R.F., Boratto, M., Alencar Filho, E.B.: Desao: A new approach for de novo drug using simulated annealing optimization. *Expert Syst. Appl.* **276**, 127099 (2025)
  - [22] Boysen, N., Scholl, A., Woppperer, N.: Resequencing of mixed-model assembly lines: Survey and research agenda. *Eur. J. Oper. Res.* **216**(3), 594–604 (2012)
  - [23] Boysen, N., Golle, U., Rothlauf, F.: The car resequencing problem with pull-off tables. *Bus. Res.* **4**, 276–292 (2011)
  - [24] Qian, J., Zhang, Z., Shi, L., Song, D.: An assembly timing planning method based

- on knowledge and mixed integer linear programming. *J. Intell. Manuf.* **34**(2), 429–453 (2023)
- [25] Zhang, J., Beck, J.C.: Domain-independent dynamic programming and constraint programming approaches for assembly line balancing problems with setups. *INFORMS J. Comput.* **37**(4), 977–997 (2025)
  - [26] Tian, S., Zhang, C., Wang, C., Li, X., Gao, L.: An iterative branch-and-bound approach for complex product assembly station scheduling problem. In: 28th Int’l Conf. on Computer Supported Cooperative Work in Design, pp. 1326–1331 (2025)
  - [27] Lai, X., Xu, Z.: Improved algorithms for joint optimization of facility locations and network connections. *Eur. J. Oper. Res.* **250**(3), 745–753 (2016)
  - [28] Huang, Y., Sheng, B., Fu, G., Yu, J., Lu, Y., Yin, X.: Solving car resequencing in automotive assembly shops based on multi-objective deep reinforcement learning. *Comput. Ind. Eng.* **211**, 111654 (2026)
  - [29] Sun, H.: Integrating virtual resequencing with car resequencing via selectivity banks for mixed-model assembly lines. *Comput. Ind. Eng.* **189**, 109990 (2024)
  - [30] Huang, Y., Sheng, B., Fu, G., Lu, Y., Yu, J.: Deep reinforcement learning for integrated virtual and physical car resequencing in automotive assembly shops. *Expert Syst. Appl.* **294**, 128770 (2025)
  - [31] Zhou, X., Li, R., Wu, Z.: Scheduling optimization for laminated door machining shop based on improved genetic algorithm. *Comput. Oper. Res.* **180**, 107078 (2025)
  - [32] Malek, N.G., Peng, Q.: Reinforcement learning for self-adaptive genetic algorithm in assembly sequence planning. *Int. J. Adv. Manuf. Technol.* **141**, 4803–4822 (2025)
  - [33] Tian, Z., Zhu, H., Shao, X.: A fast discrete small-world optimization algorithm. In: Proc. of the 5th Int’l Conf. on Computer Sciences and Automation Engineering, pp. 560–564 (2016)
  - [34] Wang, Z., Liu, M., Wang, X.: Joint scheduling of multilinear buffers based on improved genetic algorithm. *Modular Machine Tools & Automated Manufacturing Technique* **09**, 182–187 (2023)
  - [35] Zhang, H., Ding, W.: A decomposition algorithm for dynamic car sequencing problems with buffers. *Appl. Sci.* **13**(12), 7336 (2023)
  - [36] Wu, J., Ding, Y., Shi, L.: Mathematical modeling and heuristic approaches for a multi-stage car sequencing problem. *Comput. Ind. Eng.* **152**, 107008 (2021)