Adaptive Large Neighborhood Search for Cost-Effective and Efficient Decentralized Remanufacturing

Yuanyuan Tan, Zihao Wei, Haibin Zhu, and Behzad Akbari

***Abstract*—Centralized manufacturing is increasingly facing challenges from technological advancements and global economic integration. In response, decentralized manufacturing has become a viable solution that reduces production, storage, and transportation costs by providing products to consumers. This article proposes an optimized multi-factory remanufacturing process that integrates dismantling factories, manufacturing factories, dismantling workstations, and third-party logistics to improve overall system performance. It focuses on dismantling line balancing, efficient transportation and route planning, as well as minimizing the cost of dismantling plant workstations. This article introduces a multi-objective optimization method that improves existing disassembly schemes and enhances delivery and transportation stages through Adaptive Large Neighborhood Search (ALNS). This method aims to optimize the overall execution profit and transportation efficiency within the dismantling plan. In addition, the study introduced a mixed integer programming model to achieve maximum profit and improve the overall performance of the reverse supply chain. The proposed mathematical model has been validated using the CPLEX solver to confirm its accuracy and feasibility.**

***Key Words*—Multi-plant remanufacturing process optimization, Reverse Supply Chain, Two-stage problem**

I. Introduction

The increasing scarcity of global resources and prominent environmental issues have highlighted the limitations of traditional centralized manufacturing models[1]. Decentralized manufacturing not only brings products closer to customers, but also reduces production, storage, and transportation costs. Under the dual pressure of the economy and environment, the manufacturing industry urgently needs to build a reverse supply chain network that balances ecological and economic benefits. In the process of reverse supply chain optimization, Disassembly and assembly Line Balancing Problem (DLBP) and Vehicle Routing Problem (VRP) are two key sub problems[2, 3, 4]. In this regard, this article combines two key sub problems for consideration,and formalizes. Integrated-Multi-factory Remanufacturing Process Optimization (I-MRPO) considering Delivery services Problem (I-MRPO-DP).

The MRPO Problem (MRPOP) is a distributed layout environment that adds dismantling factories, remanufacturing factories, and third-party transportation fleets on the basis of the dismantling line balance problem [5]. MRPOP not only involves complex DLBP, component transportation, and remanufacturing scheduling, but also needs to consider the comprehensive optimization of order requirements, delivery deadlines, and cost-effectiveness[6]. It is highly complex and challenging, aiming to approach dismantling problems from a more holistic perspective [7]. The important decision point of MRPOP is how to design an efficient scheduling plan, which involves optimizing the entire process from the recycling center to the dismantling factory, and then to the manufacturing factory [8]. This includes optimizing the sequence of dismantling tasks, allocating workstations, limiting the number and capacity of vehicles in third-party logistics fleets, and managing the flow and distance between delivery tasks[9, 10, 11].

The Delivery Vehicle Routing (DVR) Problem (DVRP) in the VRP problem family was first discussed by Habibi *et al.*[12] It imposes additional coupling constraints and priority constraints, requiring a pair of hybrid services (including pick-up and delivery) to be completed within one route, and the pick-up service must be completed before the delivery service begins [13, 14, 15].

In existing research, DLBP and VRP are mostly discussed separately, failing to reflect the overall collaborative requirements of distributed manufacturing systems in the context of Industry 4.0. For example, Hezer *et al.*[16] proposed a dismantling line balancing model aimed at reducing the number of workstations, while Guo *et al.*[17] explored the DLBP balancing problem while considering worker fatigue. Cui and others conducted[18] research with the goal of minimizing carbon emissions and energy consumption. Kenger *et al.*[19]

Manuscript received September 1, 2025; revised September 13 and September 18, 2025; accepted October 7, 2025. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

This work was supported in part by Liaoning Revitalization Talents Program under Grant XLYC1907166, in part by the Natural Science Foundation of Shandong Province under Grant ZR2024BF140, and in part by Archival Science and Technology Project of Liaoning Province under Grant 2021-B-004.

Y. Tan is with the College of Artificial Intelligence, Shenyang University of Technology, Shenyang 110870, China (e-mail: tanyuanyuan83@sina.com).

Z. Wei is with the College of Artificial Intelligence and Software, Liaoning Petrochemical University, Fushun 113001, China (e-mail: 952048182@qq.com).

H. Zhu is with the Department of Computer Science and Mathematics, Nipissing University, North Bay P1B 8L7, Canada (e-mail: haibinz@nipissingu.ca).

B. Akbari is with the Department of Electrical and Computer Engineering, Michigan Technological University, Houghton 49931, USA (e-mail: behzadak@mtu.edu).

Corresponding author: Yuanyuan Tan

first proposed an integrated model of DLBP and VRP, but it is only applicable to situations where a single disassembly center supplies multiple remanufacturing factories. At present, there is no literature on the integration of Multi-Factory Remanufacturing Process Optimization Problem (MRPOP) and Delivery Vehicle Routing Problem (I-MRPOP-DVRP)[20].

The contributions of this work are summarized as follows.

1) In order to consider the impact of dismantling costs and transportation costs (transportation efficiency, transportation path) on dismantling revenue, this work addresses the I-MRPOP-DVRPs problem, aiming to maximize profits and reduce costs.
2) It proposes the Adaptive Large Neighborhood Search (ALNS) algorithm to solve the proposed problem. Aim to gradually increase the target value by applying a deletion heuristic and an insertion heuristic to the solution in each iteration.
3) Construct the two major sub problems in the reverse supply chain optimization process as a two-stage problem, and input the results of the first stage (MRPOP) into the second stage (DVRP). Further compare the results of different scale cases and CPLEX results using the proposed method.

The paper is structured as follows. I-MRPOP-DVRP is described in Section II. Its solution method, ALNS, is explained in Section III. Experimental results are presented in Section IV. Future work is summarized in Section V of this paper.

II. Problem Description

A. Problem description

This article explores the integration of DLBP and DVRP in MRPOP. Multi-factory remanufacturing is part of the Reverse Supply Chain (RSC) network, including dismantling plants, manufacturing plants, workstations, and transport vehicles[21]. It aims to address various challenges, including factory site selection, product positioning, sequencing of dismantling processes, product remanufacturing, and vehicle route planning. The dismantling task is supervised by the dismantling factory, and then the dismantled sub components are transported to the manufacturing factory by vehicles. An important focus of optimizing the RSC network is to effectively design efficient reverse logistics channels. This requires determining the number and location of dismantling and manufacturing factories, optimizing the sequence of dismantling tasks, allocating workstations, setting capacity limits for third-party logistics vehicles, determining service times for task nodes, and managing the processes and distances between delivery tasks[22, 23, 24]. As shown in Fig. 1, the problem can be divided into three stages:

1) Selection and dismantling scheduling of dismantling factories: Multiple dismantling factories and manufacturing factories distributed in different locations have been designed. The dismantling factories have multiple workstations and dismantling lines, as shown in Fig . 2 which can simultaneously carry out parallel dismantling tasks for different products[25]. By reasonably allocating disassembly tasks to workstations, optimizing the cycle and open time of factories and workstations based on task priority relationships and workstation cycle time constraints, and finding the optimal solution for linear disassembly offline incomplete disassembly problems[26].
2) Remanufacturing factory selection: This stage focuses on component allocation[27]. Based on the sub component prices of different manufacturing factories and the distance between disassembly and manufacturing factories, sub components obtained from different products are allocated to appropriate manufacturing factories[28].
3) Plan the optimal transportation route: This stage deals with the transportation of disassembled parts, especially DVRP issues. The goal is to plan the most effective route from dismantling plants to manufacturing plants, ensuring the successful transportation of all dismantled parts and meeting the requirements of all businesses along the way[29]. Considerations include vehicle capacity limitations and service time, where each part request must be picked up and delivered by the same vehicle. In addition, the load of each transportation node must comply with the maximum capacity limit of visiting vehicles. The load can be integrated for transportation to minimize transportation costs[30, 31, 32]. Service time is defined as the shortest time between arrival at each node and departure, including the time for loading and unloading goods. For the warehouse, the service time is set to 0. The schematic diagram of the multi-factory remanufacturing process for transportation resource sharing is shown in Fig .1.

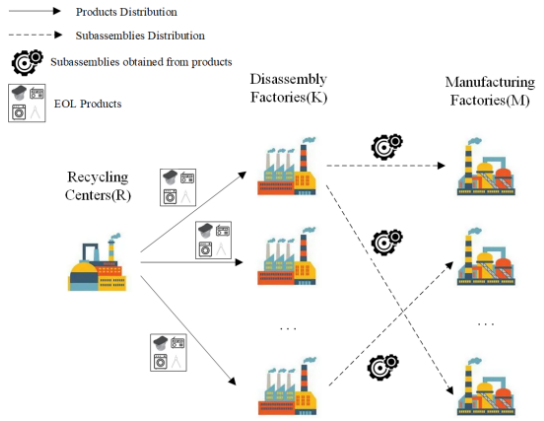


Fig. 1. The workflow of multi-factory remanufacturing process optimization

In addition, this work makes the following assumptions:

1) The location of each disassembly factory and remanufacturing factory is known.
2) Each End-of-life (EOL) parameter is known, including the cost of disassembling the product, the profit obtained, and the AND/OR chart. The weight and output of all sub components of the product are known (See Section II.B).
3) The requirements for components in each factory are determined and must be met.
4) The disassembly relationship matrix (D), priority matrix (S), and conflict matrix (R) are known.

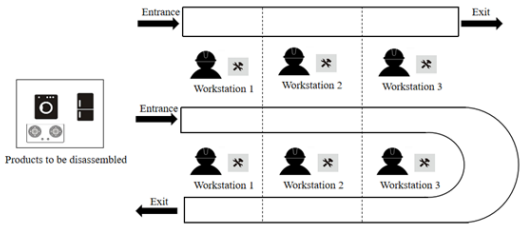


Fig. 2. Linear and U-shaped hybrid disassembly line

5) The dismantling process is incomplete dismantling.
6) The time spent on dismantling tasks and the cost per unit time are known, and each dismantling task can be processed on any workstation.
7) Each vehicle has a maximum capacity limit, which requires that the vehicle load after accessing each task node does not exceed its maximum capacity.
8) The service time of the transport vehicle when accessing task nodes is known, and the load requests of all task nodes in the network are known, represented by positive and negative signs for pick-up/delivery requests.
9) The running time of each workstation should not exceed the cycle time of the dismantling factory it is located in.

B. Mathematical Model

A mathematics model formally specifies a system under study and supports analysis and verification [33][34]. To establish the model for I-MRPOP-DVRP, we need to first understand three fundamental matrices:

1) Basic Matrix:

The correlation matrix $D = d_{ij}^{p}$ describes the relationship between components and disassembly tasks, where i represents components, j represents people, and p represents product numbers.

$$d_{ij}^{p} = \begin{cases} 1, & \text{If the disassembly task } j \text{ of product } p \text{ can obtain component } i. \\ -1, & \text{If the disassembly task } j \text{ of product } p \text{ cannot obtain component } i. \\ 0, & \text{otherwise.} \end{cases}$$

The priority matrix $S = s_{j1,j2}^{p}$ describes the relationship between two tasks, where $j1$ and $j2$ represent disassembly tasks, and p represents product number.

$$s_{j1j2}^{p} = \begin{cases} 1, & \text{If task } j1 \text{ of product } p \text{ is a prerequisite task for task } j2. \\ 0, & \text{otherwise.} \end{cases}$$

Conflict matrix $R = r_{j1,j2}^{p}$ describes the conflict relationship between two tasks, where $j1$ and $j2$ represent disassembly tasks, and p represents product number.

$$r_{jq}^{p} = \begin{cases} 1, & \text{If there is a conflict between task } j \text{ and task } q \text{ of product } p. \\ 0, & \text{otherwise.} \end{cases}$$

The disassembly allocation relationship variable $\alpha_{k,m,p,i}$ describes the allocation relationship of components from the disassembly factory to the manufacturing factory, where k represents the disassembly factory, m represents the manufacturing factory, p represents the product number, and i represents the component number.

$$\alpha_{kmpi} = \begin{cases} 1, & \text{Component } i \text{ of product } p \text{ is transported from dismantling factory } k \text{ to remanufacturing factory } m. \\ 0, & \text{otherwise.} \end{cases}$$

The distance matrix $d_{e,f}$ describes the Euclidean distance between two task nodes, where e and f represent two different nodes.

The travel time matrix $t_{e,f}$ describes the travel time taken by a vehicle to pass through two task nodes, where e and f represent two different nodes.

2) Sets:

$\mathbb{K}$	Dismantling factory assembly, $\mathbb{K}$={1,2,...,K}.
$\mathbb{M}$	Assembly of manufacturing factories, $\mathbb{M}$={1,2,...,M}.
$\mathbb{P}_n$	Pick node set, $\mathbb{P}_n$={1,2,...,n}.
$\mathbb{D}_n$	Collection of delivery nodes, $\mathbb{D}_n$={n+1,n+2,...,2n}.
$\mathbb{N}'$	Collection of all task nodes, $\mathbb{N}'=\mathbb{P}_n \cup \mathbb{D}_n$={1,2,...,n,n+1,n+2,...2n}.
$\mathbb{N}$	All node sets contain starting and ending points, $\mathbb{N}$={0,1,2,...,n,n+1,n+2,...2n,2n+1},0 and 2n+1 both represent warehouses .
$\mathbb{V}$	Collection of transportation vehicles, $\mathbb{V}$={1,2,...,V}.
$\mathbb{P}$	Product Collection, $\mathbb{P}$={1,2,...,V}.
$\mathbb{I}_p$	Collection of components for the product, $\mathbb{I}_p$={1,2,...,I_p}.
$\mathbb{J}_p$	Task set for Product P, $\mathbb{J}_p$={1,2,...,J_p}.
$\mathbb{W}^k$	The collection of workstations (linear dismantling lines) for the first dismantling factory, $\mathbb{W}^k$={1,2,...,W^k}.

3) Parameters:

v_{mpi}	The price of component i of product p purchased by the mth factory.
w_{pi}	The weight of component i in product p.
d_{ef}	Euclidean distance from node e to node f.
t_{ef}	Travel time from node e to node f.
c_{km}^{T}	The transportation cost from dismantling factory k to manufacturing factory m is based on distance;.
Q^{v}	Indicate the maximum load capacity of vehicle v.
t_{pj}	Disassembly time for task j of product p.
c_{kpj}^{D}	Unit time dismantling cost of task j for product p.
c_{k}^{o}	Unit time cost for opening dismantling factory k.
c_{kw}^{s}	Open the fixed cost of dismantling factory k workstation w.
W_{kmpi}	The weight of component i of product p transported from dismantling factory k to manufacturing factory m.
T^{k}	Cycle time for dismantling factory k .
q_e	Load demand of task node e,$q_e > 0$ represents a pick-up request, $q_e < 0$ represents a delivery request.
Q_e^{v}	Current load of vehicle v after accessing node e .
t_e^{v}	The time after vehicle v accesses node e .
Q_{km}	Load demand from dismantling factory k to manufacturing factory m .

4) Decision variables

$$x_{pjkw} = \begin{cases} 1, & \text{Product } p \text{ task } j \text{ assigned to dismantling factory } k \\ & \text{workstation } w \ . \\ 0, & \text{Others.} \end{cases}$$

$$\theta_{efv} = \begin{cases} 1, & \text{If vehicle } v \text{ travels from node } e \text{ to node } f. \\ 0, & \text{Others.} \end{cases}$$

$$y_k = \begin{cases} 1, & \text{Open dismantling factory } k. \\ 0, & \text{Others .} \end{cases}$$

$$Z_{pk} = \begin{cases} 1, & \text{If product } p \text{ is assigned to dismantling factory } k \ . \\ 0, & \text{Others .} \end{cases}$$

$$U_{kw} = \begin{cases} 1, & \text{Open the workstation } w \text{ for dismantling factory } k \ . \\ 0, & \text{Others .} \end{cases}$$

$$\alpha_{kmpi} = \begin{cases} 1, & \text{Product } p \text{ component } i \text{ is transported from} \\ & \text{dismantling factory } k \text{ to manufacturing fac-} \\ & \text{tory } m \ . \\ 0, & \text{Others} \end{cases}$$

$$\beta_{km} = \begin{cases} 1, & \text{There is a distribution task from dismantling} \\ & \text{factory } k \text{ to manufacturing factory } m \ . \\ 0, & \text{Others .} \end{cases}$$

4) Objective of optimization

Based on the above symbols and decision variables, the objective function and constraints of the I-MRPOP-DVRP model for this problem are as follows.

$$\begin{aligned} \max f = & \sum_{k\in\mathbb{K}} \sum_{m\in\mathbb{M}} \sum_{p\in\mathbb{P}} \sum_{i\in\mathbb{I}} v_{mpi}\alpha_{kmpi} - \sum_{v\in\mathbb{V}} \sum_{e\in\mathbb{N}} \sum_{f\in\mathbb{N}} d_{ef}\theta_{efv} - \\ & \sum_{k\in\mathbb{K}} \sum_{p\in\mathbb{P}} \sum_{j\in\mathbb{J}^p} \sum_{w\in\mathbb{W}^k} c^D_{kpj} t_{kpj} x_{pjkw} - \sum_{k\in\mathbb{K}} c^o_k T_k - \\ & \sum_{k\in\mathbb{K}} \sum_{w\in\mathbb{W}^k} c_{kw} u_{kw}. \end{aligned} \tag{1}$$

The I-MRPOP-VRPPD mathematical model in this article aims to maximize the expected profit of the entire network, and formula (1) aims to maximize the benefits of disassembly and minimize transportation costs. The dismantling cost includes the fixed operating costs of the dismantling factory and its related workstations, as well as the cost required to perform the dismantling task.

5) Model Constraints

$$\sum_{m\in\mathbb{M}} \alpha_{kmpj} \le \sum_{w\in\mathbb{W}_k} \sum_{j\in\mathbb{J}^p} d_{pij} x_{pjkw}, \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall i \in \mathbb{I}^{\mathbb{P}}. \tag{2}$$

$$\sum_{k\in\mathbb{K}} z_p k = 1, \forall p \in \mathbb{P}. \tag{3}$$

$$z_p k \le y_k, \forall w \subseteq W^k, \forall k \in \mathbb{K}. \tag{4}$$

$$u_{kw} \le y_k, \forall w \subseteq W^k, \forall k \in \mathbb{K}. \tag{5}$$

$$x_{pjkw} \le z_{pk}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_P, \forall w \subseteq W^k, \forall k \in \mathbb{K}. \tag{6}$$

$$x_{pjkw} \le u_{kw}, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_P, \forall w \subseteq W^k, \forall k \in \mathbb{K}. \tag{7}$$

$$\sum_{k\in\mathbb{K}} \sum_{W\in\mathbb{W}^k} x_{pjkw} \le 1, \forall p \in \mathbb{P}, \forall j \in \mathbb{J}_P. \tag{8}$$

$$\sum_{J\in\mathbb{J}_P} t_{Pj} x_{pjkw} =\le T^k, \forall k \in \mathbb{K}, \forall w \subseteq W^k. \tag{9}$$

Constraint (2) ensures that only task components obtained through disassembly can be allocated from the disassembly factory to the remanufacturing factory constraint. Constraint (3) ensures that only components obtained through disassembly can be allocated from the disassembly factory to the remanufacturing factory; Constraint (4) ensures that each product can only be assigned to a specific dismantling factory; Constraints (5) - (7) indicate that the product can only be assigned to dismantling factory workstations that are in operation; Constraint (8) stipulates that each product's disassembly task can only be performed once; Constraint (9) limits the working hours of each workstation in the dismantling line to not exceed the specified cycle time.

$$\begin{aligned} & \sum_{w\in\mathbb{W}^k} w(x_{pjkw} - x_{pqkw}) + W^k\Big(\sum_{w\in\mathbb{W}^k} x_{pqkw} - 1\Big) \le 0, \\ & \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall j, q \in \mathbb{J}_P, s_{pjq} = 1. \end{aligned} \tag{10}$$

$$\begin{aligned} & \sum_{w\in\mathbb{W}^k} x_{pqkw} \le \sum_{j\in\mathbb{J}_P} \sum_{w\in\mathbb{W}^k} x_{pjkw} s_{pjq}, \\ & \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall j, q \in \mathbb{J}_P, d_{piq} = 0. \end{aligned} \tag{11}$$

.

$$\begin{aligned} & \sum_{w\in\mathbb{W}^k} x_{pjkw} + x_{pqkw} \le 1, \\ & \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall j, q \in \mathbb{J}_P, r_{piq} = 1. \end{aligned} \tag{12}$$

.

$$z_{pk} \in (0,1), \forall k \in \mathbb{K}, \forall p \in \mathbb{P}. \tag{13}$$

$$x_{pjkw} \in (0,1), \forall j \in \mathbb{J}_P, \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall w \subseteq W^k. \tag{14}$$

$$y_k \in (0,1), \forall k \in \mathbb{K}. \tag{15}$$

$$u_{kw} \in (0,1), \forall k \in \mathbb{K}, \forall p \in \mathbb{P}, \forall m \in \mathbb{M}, \forall i \in \mathbb{I}. \tag{16}$$

$$\alpha_{kmpi} \in (0,1), \forall k \in \mathbb{K}, \forall p \in \mathbb{P}. \tag{17}$$

$$T_k \in \mathbb{R}^+, \forall k \in \mathbb{K}. \tag{18}$$

Constraints (10) and (11) require that the disassembly task allocation for each product must match its internal complexity to meet priority requirements; Constraint (12) ensures that the allocation of disassembly tasks meets conflict relationship constraints; Constraints (13) - (18) define the range of values for each decision variable.

$$\sum_{p\in\mathbb{P}}\sum_{i\in\mathbb{I}^p} w_{kmpi} = q_e, \tag{19}$$

$$\forall k \in \mathbb{K}, \forall m \in \mathbb{M}, \forall e \in \mathbb{P}_n, |\beta_{km} = 1. \tag{20}$$

.

$$q_e \leq \beta_{km} * M, \forall e \in \mathbb{P}_n, \theta_{efv} = 1, \forall e \in \mathbb{P}_n, |\beta_{km} = 1. \tag{21}$$

.

$$\sum_{f\in\mathbb{N}} \theta_{fev} - \sum_{f\in\mathbb{N}} \theta_{efv} = 0, \forall e \in \mathbb{N}^{'}, \forall v \in \mathbb{V}. \tag{22}$$

$$\sum_{(f\in\mathbb{P}_N+2*n+1)} \theta_{ofv} = 1, \forall v \in \mathbb{V}. \tag{23}$$

$$\sum_{(e\in\mathbb{D}_N+0)} \theta_e, 2*n+1, v = 1, \forall v \in \mathbb{V}. \tag{24}$$

$$\sum_{(e\in\mathbb{D}_N+0)} \theta_e, 2*n+1, v = 1, \forall v \in \mathbb{V}. \tag{25}$$

$$\sum_{f\in\mathbb{N}} (\theta_{efv} - \theta_{f,n+e,v}) = 0, \forall e \in \mathbb{P}_n, \forall v \in \mathbb{V}. \tag{26}$$

$$Q_e^v + q_f - Q_f^v \leq M * (1 - \theta_{efv}), \forall e, f \in \mathbb{N}, \forall v \in \mathbb{V}. \tag{27}$$

.

$$q_e \leq Q_e^v \leq Q^v, \forall v \in \mathbb{V}. \tag{28}$$

.

$$Q_0^v = 0, \forall v \in \mathbb{V}. \tag{29}$$

.

$$T_e^v + t_e f - T_f^v \leq M * (1 - \theta_{efv}), \forall e, f \in \mathbb{N}, \forall v \in \mathbb{V}. \tag{30}$$

.

$$T_e^v + t_{(e,n+e)} \leq T_{n+e}^v, \forall e \in \mathbb{P}_n, \forall v \in \mathbb{V}. \tag{31}$$

.

$$\theta_e \in (0,1), f_v \in (0,1), \forall e, f \in \mathbb{N}, \forall v \in \mathbb{V}. \tag{32}$$

.

$$Q_e^v \geq 0, \forall e \in \mathbb{N}, \forall v \in \mathbb{V}. \tag{33}$$

.

$$T_e^v \geq 0, \forall e \in \mathbb{N}, \forall v \in \mathbb{V}. \tag{34}$$

.

Constraint (19) is used to determine the load request of the picking node; Constraint (20) connects two symbol systems, representing the allocation relationship between disassembly factory K and manufacturing factory M; Constraint (21) ensures that each request is only accessed once; Constraint (22) ensures flow conservation and path continuity, requiring vehicles to meet both entry and exit conditions at node e; Constraints (23) and (24) ensure that each vehicle departs from the starting point and arrives at the destination, respectively; Constraint (25) ensures that pickup and delivery nodes related to the same request are served by the same vehicle; Constraint (26) ensures that the vehicle updates the load correctly during its operation, where M is a sufficiently large positive integer; Constraints (27) and (28) respectively prevent vehicles from exceeding their maximum load capacity when passing through pickup and delivery nodes; Constraint (29) requires an initial load of 0 for each vehicle; Constraint (30) ensures that the time for the vehicle to reach the subsequent node is at least the sum of the arrival time of the previous node and the travel time between two points; Constraint (31) ensures that the pickup node of the same request receives service before the corresponding delivery node; Finally, constraints (32) - (34) provide the range of values for each decision variable.

III. Algorithm design

The core idea of ALNS is to use a series of designed destroy operators and repair operators to perform large-scale perturbations on a feasible solution, thereby escaping from local optima. Unlike traditional genetic algorithms that rely on crossover operations, ALNS mainly focuses on local and large-scale destruction and reconstruction of individual solutions, and improves search efficiency by adaptively adjusting the frequency of use of each operator.As shown in Fig. 3

A. ALNS framework

Algorithm 1 Adaptive Large Neighborhood Search

Input: $s^* \leftarrow$ feasible initial solution; $REM \leftarrow$ set of removal operators; $INS \leftarrow$ set of insertion operators; $q \leftarrow$ number of requests to be removed; maxIter $\leftarrow$ maximum iterations
Output: Optimized solution s^*
Begin
for iteration = 1 to maxIter **do**
Select removal operator from REM and insertion operator from INS
$(s^{'}, R_{\text{removed}}) \leftarrow \text{Remove}(s, q)$
$s^{'} \leftarrow \text{Insert}(s^{'}, R_{\text{removed}})$
if $f(s^{'}) < f(s^*)$ **then**
$s^* \leftarrow s^{'}$
end if
if acceptance criterion met **then**
$s \leftarrow s^*$
end if
if iteration mod segIter = 0 **then**
Update operator scores in REM and INS
end if
end for
return s^*
End

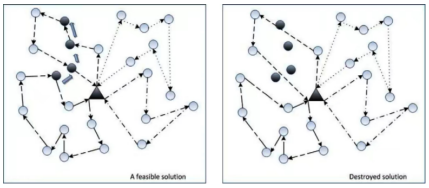


Fig. 3. Specific execution scenarios of the ALNS algorithm

B. Delete heuristic

This section introduces three different removal heuristic methods: 1) random removal: 2) Shaw removal considering priority: 3) worst removal. Each heuristic uses unique criteria to determine which requests need to be removed, and defines a function in Algorithm 1. This function takes a feasible solution s and an integer q as inputs, and outputs a partial solution s (in which q requests have been removed) and the set of removed requests. It is more preferable to remove requests that are close to each other at the same time rather than removing requests from each path individually or in pairs. The Shaw removal method aims to remove requests that are highly similar in a single iteration by quantifying the similarity between pairs of requests. Ropke and Pisinger *et al.*[35] studied an extended framework for the Adaptive Large Neighborhood Search (ALNS) algorithm, improved the Shaw removal heuristic, and enhanced search diversity by prioritizing customer similarity; Christiaansen *et al.*[36] embedded a multi-attribute priority scoring system in the ALNS disruption stage for the Vehicle Routing Problem with Time Window (VRPTW) to reduce the proportion of delayed orders

Algorithm 2 Shaw Removal with Priority (Solution s, q)

Input: Feasible solution s, number of requests to be removed q
Output: Partial demodulated solution s'
  Randomly select a request r as the initial removal request
  Set $R_{\text{removed}} = \{r\}$
  while $|R_{\text{removed}}| < q$ **do**
    Select a random request r' from R_{removed}
    Generate unremoved request list L
    Sort L in ascending order of request similarity $R(r, r')$
    Generate random number $y \in (0, 1)$
    Select request r at index position $\lfloor y^p \cdot |L| \rfloor$ in L
    $R_{\text{removed}} = R_{\text{removed}} \cup \{r\}$
  end while
  $s' = \text{Remove}(s, R_{\text{removed}})$
  if R_{removed} is not empty **then**
    return s'
  else
    return Empty solution
  end if

The request cost r is defined as the difference between the total objective function values that include the pick-up and drop-off pairs that do not include the request, and is denoted as:

$$\Delta f_r = f(s) - f^{-r}(s) \tag{35}$$

where s is the current solution, f is the objective function, and $f^{-r}(s)$ is the target value after the request r is removed from s. A large Δf_r value indicates that request r is in a disadvantageous position, implying that an improved solution is more likely to be found by substituting such a request. The worst-case removal algorithm is designed to identify and remove requests with a significant value of Δf_r in each iteration. Algorithm 3 gives a specific process for the algorithm, which is similar to the Shaw removal algorithm that takes priority into account, but the key difference is that the requests are sorted in descending order of request cost in each iteration.

Similar to the Shaw removal algorithm that takes priority into account, the worst removal algorithm introduces a random parameter p to increase randomness. This randomness is essential to prevent the algorithm from repeatedly trying to remove the same set of requests in successive iterations, thus ensuring efficient exploration of the neighborhood space.

Algorithm 3 Worst Removal (Solution s, q)

Input: Feasible solution s, number of requests to be removed q
Output: Partial solution s' and removal request set R_{removed}
  $R_{\text{removed}} = \varnothing$
  while $|R_{\text{removed}}| < q$ **do**
    Generate unremoved request list L
    Sort L in descending order of incremental cost $\Delta f(r)$
    Generate random number $y \in (0, 1)$
    Select request r at index position $\lfloor y^p \cdot |L| \rfloor$ in L
    $R_{\text{removed}} = R_{\text{removed}} \cup \{r\}$
  end while
  $s' = \text{Remove}(s, R_{\text{removed}})$
  return s', R_{removed}

C. Insert heuristics

This section provides a detailed description of various insertion heuristics, each of which introduces a set of rules to determine the position for inserting requests, and defines function insert($s^{'}$,$R_{removed}$) in Algorithm 1. These methods take a partial solution $s^{'}$ and the request set $R_{removed}$ as input, and output a solution after reinserting the requests.

The Simple Greedy Insertion (SGI) algorithm is designed to identify the best request that increases the least on the value of the objective function in each iteration. In this algorithm, the insertion cost $\Delta f_{r,k}$ represents the minimum increase in the objective function resulting from inserting the requestr into path k. Algorithm 4 outlines the process for achieving that goal. In the initial phase, the cost of inserting each request into each path was calculated; The request r^* with the lowest insertion cost is then inserted into the partial solution and the request is removed from the collection of unprocessed requests. If there are still requests that fail to be inserted successfully, an empty solution is returned, indicating that the algorithm has failed to construct a feasible solution. Otherwise, the output is descripted as s.

Algorithm 4 Simple Greedy Insert (Solution s, R_{removed})

Input: Feasible solution s, Set of removed requests R_{removed}
Output: Updated solution s'

```
if R_removed = ∅ then
  return s {No requests to insert}
else
  s' ← s
  for each request r in R_removed do
    minCost ← ∞
    bestPosition ← null
    bestRoute ← null
    for each route k in solution s' do
      Calculate insertion cost Δf(r, k) for all feasible
      positions in route k
      if feasible insertion positions exist then
        Find position with minimal cost Δf_min(r, k)
        if Δf_min(r, k) < minCost then
          minCost ← Δf_min(r, k)
          bestPosition ← position with min cost
          bestRoute ← k
        end if
      end if
    end for
    if bestRoute ≠ null then
      Insert request r into bestRoute at bestPosition
    else
      {No feasible insertion found for request r}
    end if
  end for
  return s'
end if
```

However, the SGI heuristic only considers the effect of a single step after insertion, which has a limited field of view, which often leads to the high cost of some "bad" requests in the later stage of insertion. In contrast, the Regret Insert strategy extends the evaluation to multiple subsequent steps (i.e., the Regret-m Insert method) to make more informed decisions.

Santini *et al.*[37] compared the performance of different order regret values (k-regret) in vehicle routing problems and proposed for the first time a dynamic order adjustment mechanism to improve the quality of solutions and computational efficiency. Zhou *et al.*[38] designed a multi node collaborative regret insertion strategy for the electric vehicle routing problem, which solved the problem of calculating regret values with multiple constraints in the electric vehicle routing. Wang *et al.*[39] combined regret insertion with deep reinforcement learning and proposed a state aware regret function to dynamically adjust order insertion priority.

The regret value is used to quantify the degree of degradation that may result from delaying the insertion of a request, so as to give preference to requests with higher regret values; If there is the same regret value, the request with the lower insertion cost is preferred. It is important to note that the SGI actually corresponds to the Regret-1 Insert in that it only performs a single-step evaluation at the time of insertion.

In order to handle requests with limited possible insert locations, we set $\Delta f_{rkn} = M$ to prioritize those requests that have fewer feasible insert locations. The overall process is similar to a simple greedy insertion, except that in line 10 the request is selected to make the regret value c_r the largest.

In this study, the set of insertion operators consists of Regret 1 (simple greed) Regret-2, Regret-3, Regret-4, and .

D. The acceptance criteria are based on linear threshold acceptance

The acceptance criteria used in this study are derived from linear Threshold Acceptance (TA), with a termination temperature set to 0. The temperature threshold italic is used to determine whether the difference in the objective function is acceptable. In each iteration, the improved neighborhood solution is always accepted; If the degenerate solution satisfies $[f(s^{'}) - f(s^{*})]/f(s^{*}) < T$, it will also be accepted. The initial temperature italic is set to T_{start}, and the rate of $\triangle T = T_{start}/maxIter$ gradually decreases.

Linear TA has been proven to have high computational efficiency and can generate high-quality solutions. Another commonly used acceptance criterion comes from simulated annealing (SA), which always accepts the better solution, while the degenerate solution is accepted with probability $e^{-((f(s)-f(s))/T)}$. Santini et al. pointed out that in the Adaptive Large Neighborhood Search (ALNS) framework, the performance of the two criteria is comparable. The reason for choosing TA in this study is:

1) The threshold is relatively intuitive: acceptance can be directly judged by the threshold T, without the need to generate random numbers and calculate probabilities.
2) Parameter tuning is simple: simply adjust the initial temperature to determine the cooling rate.

We have studied the influence of initial temperature on the search process. In Fig. 4, the target values of the accepted solution (current target) and the target values of the best solution to date are plotted. Several observation results can be obtained.*Firstly, the target value of the accepted solution decreases linearly due to the linear decrease of the threshold T. Secondly, every time the current target value and the optimal solution target value coincide, the global optimal target value will be updated. At the beginning of the search process, when the quality of the solution is not high, the update frequency is higher. Finally, when T_{start} is low, The local search process accepts fewer solutions, as can be seen from the increased sparsity in Figs. 4 - 6. A lower threshold requires solutions with a smaller target gap to be accepted.*

E. Adaptive weight adjustment

For each operator i in the removal and insertion operator sets, a weight w_i will be assigned. The operator selection adopts the roulette wheel selection principle, so that the probability of operator i being selected is $w_i/\sum_{i\in\mathbb{O}} w_i$, where 0=REM or 0=INS.

In each iteration, the score Π_i of operator i will be updated based on the quantities determined by parameters σ_1, σ_2, and σ_3 in different scenarios:

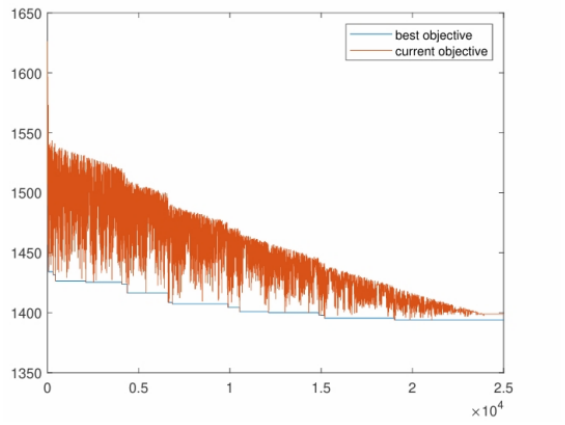


Fig. 4. The optimal solution and the currently accepted solution during the iterative process at T_{start}=0.08

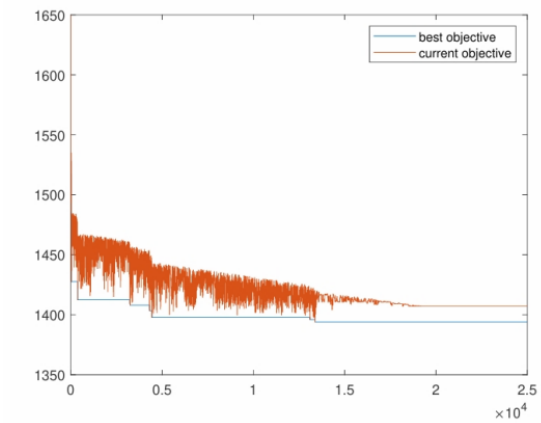


Fig. 5. The optimal solution and the currently accepted solution during the iterative process at T_{start}=0.04

1) If the new solution produces a new global optimal solution to date, then add σ_1.
2) If the new solution improves the target value and was not previously accepted, add σ_2.
3) If the new solution was not accepted before, but was accepted when the target value decreased, then add σ_3.

The entire search process is divided into several stages. At the beginning of each stage, the scores of all operators are reset to 0. At the end of each stage, update the weights using the following formula:

$$w_{i,j+1} = (1 - r)w_{i,j} + r * \pi_i/\theta_i. \quad (36)$$

Among them,$w_{i,j}$ is the weight of operator i in stage j,π_i is the score obtained by operator i in this stage, and θ_i is the number of times operator i is used in this stage.

IV. Simulation Experiment and Analysis

To evaluate the accuracy of the model and the effectiveness of the proposed algorithm, an experimental case was solved using IBM ILOG CPLEX Optimization Studio to obtain the standard optimal solution, and the same case was solved

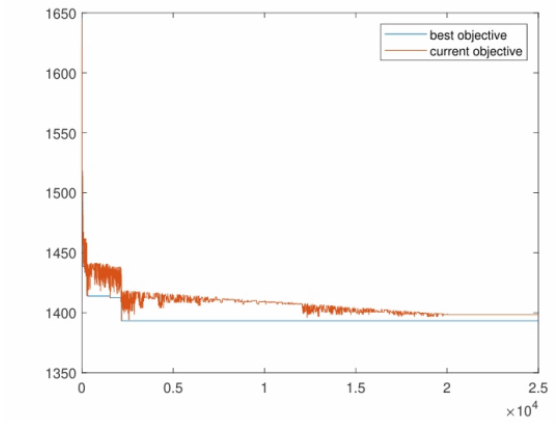


Fig. 6. The optimal solution and the currently accepted solution during the iterative process at T_{start}=0.02

using ALNS algorithm in IntelliJ IDEA 2024.3.1.1 * 64 integrated development environments to compare the experimental results. The calculation experiment was conducted on a computer equipped with Intel (R) Core (TM) i9-13900HX processor (operating frequency 5.40GHz) and 32.00GB RAM.

A. Test Case

To make the experimental research more comprehensive, three different product sizes were selected: washing machine, computer, and radio. We create multiple product cases for testing by combining these products with different configurations. Table I provides specific size information for the combination cases. In terms of work environment, three dismantling factories and three manufacturing factories have been established, with a maximum of five workstations on each factory's dismantling line. The specific product parameter settings are shown in Table I, and the experimental test case information is shown in Table II. Among them, cases 1-4 are small-scale dismantling cases, while cases 5-7 are large-scale dismantling cases. The profit generated from recycling parts may fluctuate depending on the unique location of each factory in the remanufacturing multi-factory setup. Due to different disassembly products, the related disassembly tasks and profitable parts will also be different. This diversity reflects the complexity of the dismantling process and its potential profitability.

TABLE I Product information.

Disassemble the product	Number of tasks	Number of components	Dismantling profit(RMB)	Component weight(KG)	Disassembly time(s)
Computer	13	25	102-166	6-41	5-11
Washing machine	13	15	106-179	3-33	4-10
Radio	30	29	63-162	5-74	4-11

B. Experimental Results And Algorithm Performance Analysis

We tested the experimental case using CPLEX, and the results are shown in Table III.Case 7 of the Multi- Factory Remanufacturing Process Optimization Problem (MRPOP)

TABLE II Case information

Case Number	Number of products	Product			Dismantling factories	Remanufact-uring factories	Number of tasks	Number of components	Component profit(RMB)
		Computer	Washing machine	Radio					
1	2	1	1	0	2	2	43	44	208-345
2	3	1	1	1	2	2	56	69	271-507
3	4	2	1	1	3	3	69	94	373-673
4	5	2	2	1	3	3	82	109	479-852
5	6	2	2	2	3	3	112	138	542-1014
6	9	3	3	3	3	3	168	207	813-1521
7	10	3	3	4	3	3	198	236	876-1668

involves 10 products: 3 washing machines, 3 computers, and 4 radios. The experimental results are shown in Table III and Fig. 7. Product A is assigned to disassembly center B for disassembly. Component 2 disassembled from product C is transported to manufacturing center D, while components 7, 11, and 12 are transported to manufacturing factory E. Fig. 7 visually illustrates the complete workflow of case 5.

TABLE III CPLEX solution results

Product ID	Dismantling factory allocation	Distribution of remanufacture factories	Factory transportation weight
1	2	*<2→2>/<7,11,12→3>*	8→2,5→3
2	3	<2,3→2>	33→2
3	2	<8,11→1>/<2,13,14,15→3>	11→1,2→3
4	3	<9,12→2>/<7,10→3>	19→2,22→3
5	3	<9,11,12→2>/<10,13→3>	26→2,15→3
6	3	<5,9,13→3>	41→3
7	2	<2,23→1>	74→1
8	1	<9,19,23,24,26→1>	79→1
9	1	<2,23→1>	74→1
10	1	<9,19,23,24,26→1>	79→1

By comparison, it was found that CPLEX's results were only based on the profit of the components provided by the manufacturing factory for the distribution and transportation of disassembled components, indicating significant deficiencies in its comprehensive transportation process. Thus we take the one-stage MRPOP scheduling scheme as the optimal solution obtained by re inputting the required parameters for solving the two-stage delivery vehicle routing problem (VRPPD) model. The case parameters are shown in Table IV.

The comparison between Fig. 7 and Fig. 8 clearly shows that compared with the first stage CPLEX solution, the two-stage DVRP transportation model considers the load integration of transportation vehicles, which can integrate multiple batches of parts and deliver them to the dismantling manufacturing factory in sequence. After optimization, it significantly reduces distribution costs, total vehicle travel distance, and required vehicle quantity. This is more in line with the actual benefit requirements of the multi-factory remanufacturing environment.

C. Algorithm validation of cases of different scales

To verify the performance of the proposed ALNS framework in solving VRPPD, smaller scale instances were generated and evaluated. Compare the results with those obtained by CPLEX within a time limit of 120 minutes. The objective function of the two-stage model is to maximize revenue and minimize transportation costs. The experiment randomly selects N pairs of requests from the adjusted project instances to form a smaller dataset.Experiments were conducted on instances with N values of 5, 10, and 15, and the results are shown in Tables V, VI, and VII, respectively.

For small-scale cases with N=5, the target values calculated by CPLEX and ALNS are consistent, and CPLEX's calculation time is slightly better than ALNS. However, ALNS can still obtain the optimal solution for all instances at a relatively fast speed. When N=10, ALNS found a better solution within 5 seconds in 6 cases, and performed the best in case 3, where CPLEX took 2044.5 times longer to calculate the target value than ALNS. In two cases, CPLEX failed to find the optimal solution within the 60-minute time limit. In the case of finding the optimal solution, ALNS is 145.8 to 2044.5 times faster than CPLEX. In other cases, both ALNS and CPLEX can find the optimal solution, but ALNS is faster. In addition, in experiments with N=10 and 15, ALNS requires much faster computation time than CPLEX, and the difference in computation time also increases as the problem size increases. When N=15, CPLEX can only solve 5 cases, and Case 2 cannot obtain a solution within the limited 120-minutes, while ALNS can find feasible solutions for all instances without a significant increase in computation time. In Case 3 and Case 4, there is a significant difference in the target value of CPLEX compared to ALNS, which effectively proves the superiority of the ALNS algorithm. Due to the fact that the results obtained by ALNS in all instances are the same as the optimal target value successfully found in CPLEX, the target value obtained by ALNS in large-scale cases is better than that of CPLEX under a limited running time of 120-minutes. Therefore, it can be concluded that the proposed ALNS has robustness when applied to the model proposed in this study, and ALNS has higher computational efficiency and robustness in experiments with larger scale problems.

TABLE IV CPLEX solves transportation plan information

Demand	Task Node	Weight requirement	Service Time(s)	Coordinate
starting point	S	0	0	(35, 35)
1	1	232	30	(92, 81)
	1	-232	30	(44, 87)
2	2	85	10	(20, 88)
	1	-85	10	(44, 87)
3	2	8	10	(20, 88)
	2	-8	10	(98, 50)
4	2	47	10	(20, 88)
	3	-47	10	(22, 32)
5	3	78	10	(16, 12)
	2	-78	10	(98, 50)
6	3	78	10	(16, 12)
	3	-78	10	(22, 32)
End	E	0	0	(35, 35)

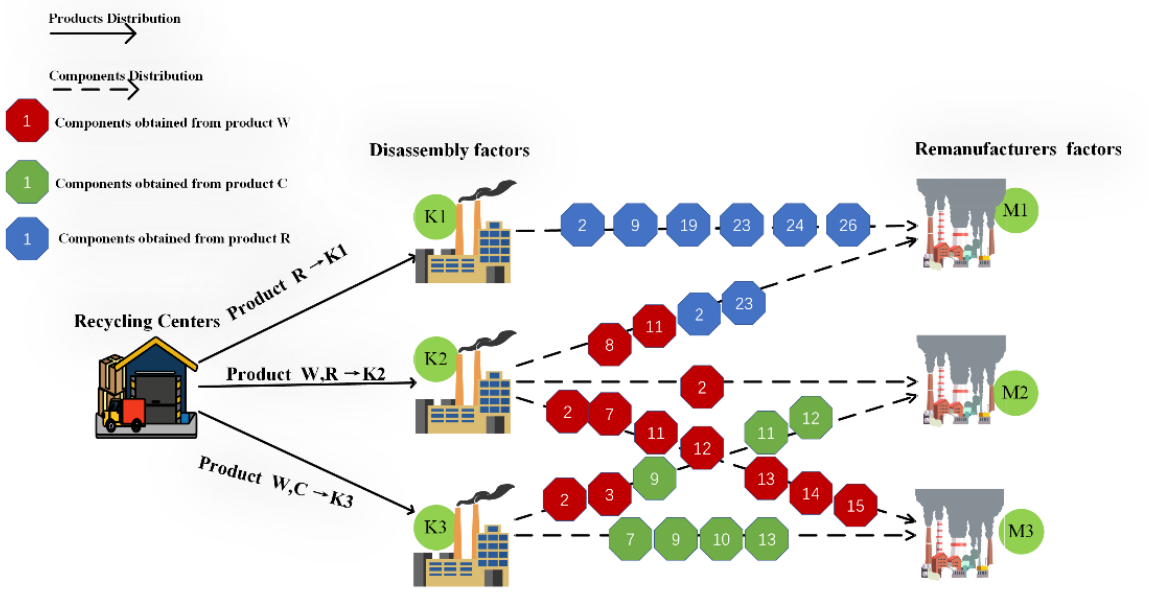


Fig. 7. One-stage disassembly scheduling transport scheme

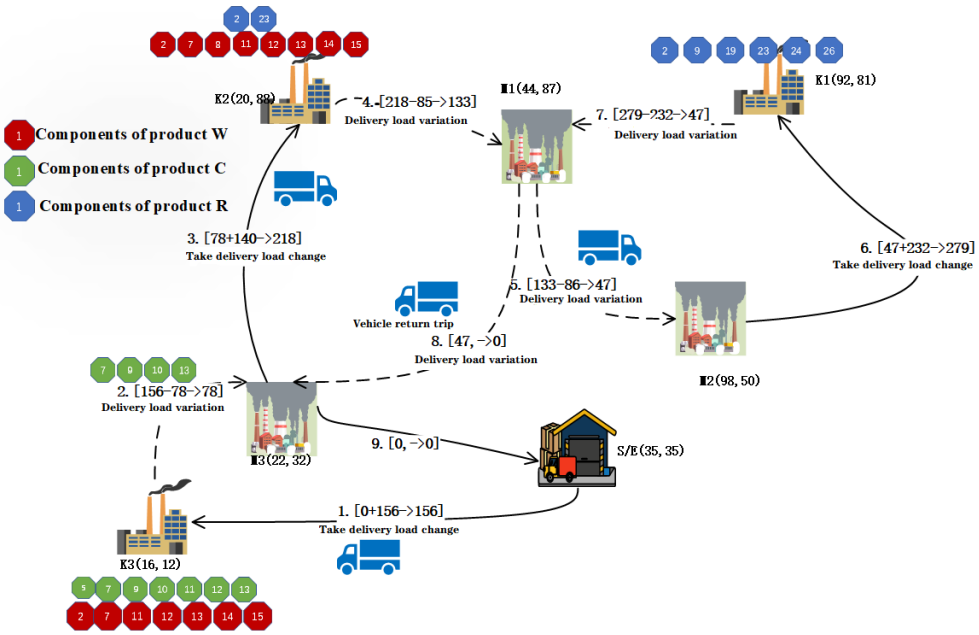


Fig. 8. Two-stage optimal disassembly and distribution scheme.

TABLE V Comparison of results between CPLEX and ALNS N=5

Test Examples	Target value		Running time (s)	
	CPLEX	ALNS	CPLEX	ALNS
1	196.499	196.499	0.693	1.14
2	265.647	265.647	0.479	1.13
3	211.094	211.094	0.204	0.71
4	275.335	275.335	0.305	1.01
5	518.822	518.822	0.207	0.83
6	356.407	356.407	0.506	1.23

TABLE VI Comparison of results between CPLEX and ALNS N=10

Test Examples	Target value		Running time (s)	
	CPLEX	ALNS	CPLEX	ALNS
1	781.062	763.768	*7200	4.15
2	633.427	603.877	*7200	1.59
3	354.714	354.714	1921.5	0.97
4	547.470	547.470	720.47	4.94
5	433.965	433.965	504.31	1.64
6	326.235	326.235	1260.59	3.59

TABLE VII Comparison of results between CPLEX and ALNS N=15

Test Examples	Target value		Running time (s)	
	CPLEX	ALNS	CPLEX	ALNS
1	572.504	557.224	*7200	1.11
2	———	659.206	*7200	1.96
3	2824.479	611.233	*7200	5.59
4	902.876	544.052	*7200	4.91
5	1253.413	986.798	*7200	1.82
6	1216.911	1049.695	*7200	3.15

V. CONCLUSION

This paper investigates a two-stage I-MRPOP-DVRP problem that considers the planning of delivery vehicles. The optimization problem of multi-factory remanufacturing process is a research hotspot in the field of supply chain. This article proposes for the first time the multi-factory remanufacturing process optimization problem I-MRPOP-DVRP considering delivery services, and establishes a mixed integer programming model aimed at maximizing profits to describe this problem. Decompose the problem into two sub problems: overall disassembly scheduling and optimal transportation route planning. In the future, we will continue to study two-stage problems. Unlike this article, the decisions in the first stage will affect the decisions in the second stage, and the decisions in the second stage will be iterated back to the first stage[40].

REFERENCES

[1] Y. Ji, S. Liu, M. Zhou, Z. Zhao, X. Guo, and L. Qi, "A machine learning and genetic algorithm-based method for predicting width deviation of hot-rolled strip in steel production systems," *Information Sciences*, vol. 589, pp. 360–375, 2022.

[2] Z. Zhao, S. Liu, M. Zhou, D. You, and X. Guo, "Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 251–261, 2022.

[3] Z. Zhao, M. Zhou, and S. Liu, "Iterated greedy algorithms for flow-shop scheduling problems: A tutorial," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1941–1959, 2022.

[4] Z. Zhao, S. Liu, M. Zhou, and A. Abusorrah, "Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 6, pp. 1199–1209, 2022.

[5] L. Qi, M. Li, X. Guo, and W. Luan, "Multi-objective optimization for robotaxi dispatch with safety-carpooling mode in pandemic era," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 1, pp. 878–891, 2024.

[6] Z. Zhao, S. Liu, M. Zhou, X. Guo, and L. Qi, "Decomposition method for new single-machine scheduling problems from steel production systems," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1376–1387, 2019.

[7] K. Zhang, R. M. Zhou, J. Wang, Y. Xiao, X. Guo, and C. Shi, "Transmission line component defect detection based on uav patrol images: A self-supervised hc-vit method," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 11, pp. 6510–6521, 2024.

[8] C. Xu, H. Wei, X. Guo, S. Liu, L. Qi, and Z. Zhao, "Human-robot collaboration multi-objective disassembly line balancing subject to task failure via multi-objective artificial bee colony algorithm," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 1–6, 2020.

[9] X. Cui, X. Guo, M. Zhou, J. Wang, S. Qin, and L. Qi, "Discrete whale optimization algorithm for disassembly line balancing with carbon emission constraint," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 3055–3061, 2023.

[10] S. Qin, J. Li, J. Wang, X. Guo, S. Liu, and L. Qi, "A salp swarm algorithm for parallel disassembly line balancing considering workers with government benefits," *IEEE Transactions on Computational Social Systems*, 2023.

[11] S. Qin, S. Zhang, J. Wang, S. Liu, X. Guo, and L. Qi, "Multi-objective multi-verse optimizer for multi-robotic u-shaped disassembly line balancing problems," *IEEE Transactions on Artificial Intelligence*, 2023.

[12] X. Guo, C. Fan, M. Zhou, S. Liu, J. Wang, S. Qin, and Y. Tang, "Human–robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm," *IEEE Transactions on Automation Science and Engineering*, 2023.

[13] X. Guo, Z. Bi, J. Wang, S. Qin, S. Liu, and L. Qi, "Reinforcement learning for disassembly system optimization problems: A survey," *International Journal of Network Dynamics and Intelligence*, pp. 1–14, 2023.

[14] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin, and L. Qi, "Multiobjective u-shaped disassembly line balancing problem considering human fatigue index and an efficient solution," *IEEE Transactions on Computational Social Systems*, 2022.

[15] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, and Z. Zhao, "Stochastic hybrid discrete grey wolf optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1744–1756, 2021.

[16] X. Guo, M. Zhou, A. Abusorrah, F. Alsokhiry, and K. Sedraoui, "Disassembly sequence planning: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 7, pp. 1308–1324, 2020.

[17] X. Guo, M. Zhou, S. Liu, and L. Qi, "Multiresource-constrained selective disassembly with maximal profit and minimal energy consumption," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 804–816, 2020.

[18] X. Guo, Z. Zhang, L. Qi, S. Liu, Y. Tang, and Z. Zhao, "Stochastic hybrid discrete grey wolf optimizer for multi-objective disassembly sequencing and line balancing planning in disassembling multiple products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1744–1756, 2022.

[19] Y. Tan, M. Zhou, Y. Wang, X. Guo, and L. Qi, "A hybrid mip–cp approach to multistage scheduling problem in continuous casting and hot-rolling processes," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1860–1869, 2019.

[20] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 12, pp. 5037–5048, 2019.

[21] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 894–903, 2019.

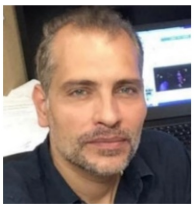
[22] Y. Tan, M. Zhou, Y. Zhang, X. Guo, L. Qi, and Y. Wang, "Hybrid scatter search algorithm for optimal and energy-efficient steelmaking-continuous casting," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1814–1828, 2020.

[23] L. Qi, W. Luan, X. S. Lu, and X. Guo, "Shared p-type logic petri net composition and property analysis: A vector computational method," *IEEE Access*, vol. 8, pp. 34 644–34 653, 2020.
[24] K. Zhang, Y. Su, X. Guo, L. Qi, and Z. Zhao, "Mu-gan: Facial attribute editing based on multi-attention mechanism," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 9, pp. 1614–1626, 2020.
[25] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Stochastic multi-objective integrated disassembly-reprocessing-reassembly scheduling via fruit fly optimization algorithm," *Journal of Cleaner Production*, vol. 278, p. 123364, 2021.
[26] Z. Bi, X. Guo, J. Wang, S. Qin, and G. Liu, "Deep reinforcement learning for truck-drone delivery problem," *Drones*, vol. 7, no. 7, p. 445, 2023.
[27] J. Gu, J. Wang, X. Guo, G. Liu, S. Qin, and Z. Bi, "A metaverse-based teaching building evacuation training system with deep reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2209–2219, 2023.
[28] J. Wang, "Patient flow modeling and optimal staffing for emergency departments: A petri net approach," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 2022–2032, 2023.
[29] J. Wang and J. Wang, "Real-time adaptive allocation of emergency department resources and performance simulation based on stochastic timed petri nets," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 4, pp. 1986–1996, 2023.
[30] J. Zhou, J. Wang, and J. Wang, "A simulation engine for stochastic timed petri nets and application to emergency healthcare systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 4, pp. 969–980, 2019.
[31] H. Han, W. Li, J. Wang, G. Qin, and X. Qin, "Enhance explainability of manifold learning," *Neurocomputing*, vol. 500, pp. 877–895, 2022.
[32] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 110–120, 2020.
[33] J. Wang, "Charging information collection modeling and analysis of GPRS networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 4, pp. 473–481, 2007.
[34] Y. Tian, G. Liu, J. Wang, and M. Zhou, "ASA-GNN: Adaptive sampling and aggregation-based graph neural network for transaction fraud detection," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 3, pp. 3536–3549, 2024.
[35] B. Hu and J. Wang, "Deep learning based hand gesture recognition and uav flight controls," *International Journal of Automation and Computing*, vol. 17, no. 1, pp. 17–29, 2020.
[36] L. Qi, L. Liang, W. Luan, T. Lu, X. Guo, and Q. T. A. Talukder, "Integrated control strategies for freeway bottlenecks with vehicle platooning," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 46–56, 2025.
[37] X. Guo, M. Zhou, S. Liu, and L. Qi, "Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3307–3317, 2020.
[38] J. Gu, Z. Guo, J. Wang, L. Qi, S. Qin, and S. Zhang, "Optimization of multi-factory remanufacturing processes with shared transportation resources using the alns algorithm," *International Journal of Artificial Intelligence and Green Manufacturing*, vol. 1, no. 1, pp. 1–13, 2025.
[39] N. Yang, Z. Zheng, M. Zhou, and et al., "A domain-guided noise-optimization-based inversion method for facial image manipulation," *IEEE Transactions on Image Processing*, vol. 30, pp. 6198–6211, 2021.
[40] Z. Zhao, S. Liu, M. Zhou, D. You, and X. Guo, "Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 251–261, 2022.

YuanYuan Tan received her B.S. degree in information and computing science from Bohai University, Jinzhou, China, in 2007, M.S. degree and Ph. D. degree in the College of Information Science and Engineering from Northeastern University, Shenyang, China, in 2009 and 2013 respectively. She is currently an associate professor of the School of Artificial Intelligence at Shenyang University of Technology. Her research focuses on scheduling and production planning, and intelligent optimization algorithms. She has published over several journal and conference proceedings papers in the above research areas.

Zihao Wei received his degree in computer science and technology from Anyang Institute of Technology in China, in 2023. Now he is a graduate student of College of Artificial Intelligence and Software,Liaoning University of Petrochemical Technology. His main research direction is reinforcement learning.

Haibin Zhu is a Full Professor at Nipissing University, Canada. He is also an affiliate full professor of Concordia Univ. and an adjunct professor of Laurentian Univ., Canada. He has accomplished over 300+ research publications, including 60+ IEEE Transactions articles. He is a fellow of IEEE, AAIA (Asia-Pacific Artificial Intelligence Association) and I2CICC (International Institute of Cognitive Informatics and Cognitive Computing), a senior member of ACM. He is Vice President, Systems Science and Engineering (SSE) (2023-), a co-chair (2006-) of the technical committee of Distributed Intelligent Systems, and a Distinguished Lecturer of IEEE Systems, Man and Cybernetics Society (SMCS), Associate Editor (AE) of IEEE Transactions on SMC: Systems (2018-), IEEE Transactions on Computational Social Systems (2018-), IEEE Systems Journal (2024-), Frontiers of Computer Science (2021-), IEEE Canada Review (2017-), and Deputy Editor-in-Chief of Artificial Intelligence Science and Engineering (2025-). He was General Chair: E-CARGO 2025, China, 2024, China, 2023, online, ScalCom 2023, UK; ISEEIE 2023, Singapore, SPCS 2022, China, ICCSIT 2021, France, and Program Chair: ICFTIC 2025, 2024, China, CSCWD 2020, CSCWD 2018, China, ICNSC 2019, and CSCWD 2013, Canada. He is the founding researcher of Role-Based Collaboration and the creator of the E-CARGO model. He has offered 38 keynote speeches for international conferences and 93 invited talks internationally.



Behzad Akbari is an IEEE member who earned his bachelor's and master's degrees in Computer Software and Computer Architecture from Tehran University and Knowledge and Research University, Iran, in 1996 and 1999, respectively. He continued his academic journey in Canada, completing a second master's in Computer Science and a Ph.D. in Electrical and Computer Engineering (ECE) at McMaster University in 2015 and 2021. Dr. Akbari is currently an Assistant Professor at Michigan Technological University in Houghton, Michigan, USA. He has served as a reviewer for prominent journals including IEEE Access, IEEE Transactions on Robotics, and IEEE Transactions on Systems, Man, and Cybernetics. His research focuses on state estimation algorithms, collaborative multi-agent systems, multi-target tracking, multi-output Gaussian processes, iterative localization and mapping, factor graph optimization, and reinforcement learning.