SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2023, Vol. 66 212202:1–212202:23
https://doi.org/10.1007/s11432-022-3714-4

A bi-level optimization approach for joint rack sequencing and storage assignment in robotic mobile fulfillment systems

Xiang SHI[1], Fang DENG[1,2*], Sai LU[1], Yunfeng FAN[1], Lin MA[3] & Jie CHEN[1,4]

[1]*Key Laboratory of Intelligent Control and Decision of Complex Systems, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*Beijing Institute of Technology Chongqing Innovation Center, Chongqing* 401120*, China;*
[3]*Zhejiang Cainiao Supply Chain Management Co., Ltd., Hangzhou* 311101*, China;*
[4]*Shanghai Research Institute for Intelligent Autonomous Systems, Shanghai* 200092*, China*

Received 16 September 2022/Revised 3 January 2023/Accepted 16 February 2023/Published online 19 October 2023

Abstract This paper studies a novel rack scheduling problem with multiple types of multiple storage locations (RS-MTMS), which can decide the retrieval sequence of racks and assign each rack a storage location after visiting a picking station. A major challenge in RS-MTMS is that the storage assignment problem and the retrieval sequence decision are closely coupled. If the RS-MTMS is solved directly, the storage assignment scheme and the retrieval sequence of racks are generally generated separately, thus resulting in poor performance. To overcome this difficulty, we propose a bi-level optimization approach for jointly optimizing the storage assignment and retrieval sequence (BiJSR). In BiJSR, the storage assignment problem is solved by variable neighborhood search (VNS) in the upper-level optimization. Effective candidate modes are incorporated into VNS to improve solution quality and computational efficiency. The sequencing optimization is obtained in the lower-level according to the given storage location set. A transformation strategy with sufficient problem-specific knowledge is developed to identify the lower-level optimization as the traveling salesman problem and its variants. Then these identified problems are solved using the loop-based strategy. Experimental results show that the proposed BiJSR is more effective and efficient than the representative algorithms in solving the RS-MTMS problem.

Keywords rack scheduling, sequence decision, storage assignment, bi-level optimization, robotic mobile fulfillment system

Citation Shi X, Deng F, Lu S, et al. A bi-level optimization approach for joint rack sequencing and storage assignment in robotic mobile fulfillment systems. Sci China Inf Sci, 2023, 66(11): 212202, https://doi.org/10.1007/s11432-022-3714-4

1 Introduction

The rapid development of the e-commerce economy introduces many challenges with respect to the order picking system of the operating warehouses. In the traditional picker-to-parts order picking systems, order picking is the most time-consuming and labor-intensive process. Recently, a novel parts-to-picker order picking system, called the robotic mobile fulfillment system (RMFS), has been widely used in e-commerce companies such as Amazon, Scallog, and Alibaba [1–5].

When operating RMFS to handle the daily order picking process, each picker in the picking station will first receive a list of customer orders with the required products. Then, some racks will be assigned to the different picking stations to process these orders [6–8]. These racks are presented to pickers sequentially through robot handling. To improve the order picking efficiency, each robot has to complete the rack set's delivery task in the shortest time possible [9–11]. As shown in Figure 1, many racks, open storage locations, and picking stations exist in RMFS. The delivery task of a rack mainly involves three parts.

* Corresponding author (email: dengfang@bit.edu.cn)

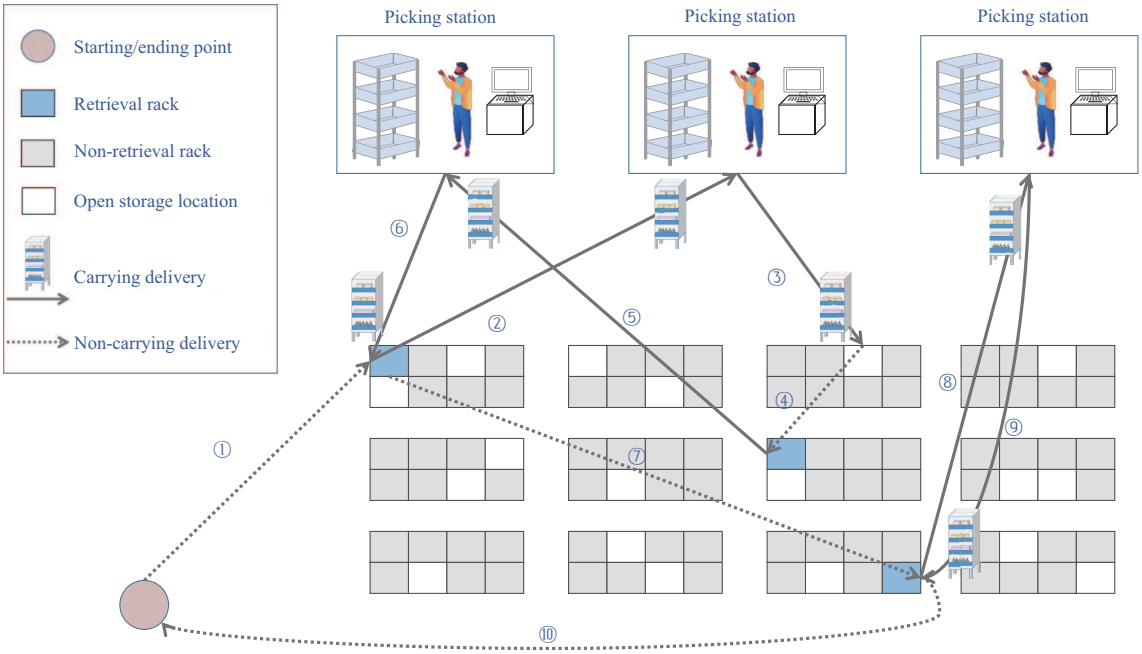

Figure 1 (Color online) The description of the RS-MTMS in the RMFS-based order picking process, where the robot starts from the starting point and continuously performs the delivery tasks of retrieval racks.

(1) Drive to the rack's retrieval location from the robot's current location. For an order picking process, a robot may start at the starting point or the storage location of the last retrieval rack (such as steps ①, ④, and ⑦) in each delivery task.

(2) Deliver the rack from its retrieval location to the picking station. We assume that the visits of the racks at the picking stations are already fixed. This assumption is reasonable because the assignment of racks to picking stations can be obtained by the rack assignment problem, which is the upper layer of RS-MTMS in the operations planning hierarchy [10].

(3) Return the rack to a storage location. Once all products are retrieved from the current rack, this rack can be stored at a location that is available in the set of storage locations.

Initially, the robot starts from the starting point and continuously performs the delivery tasks (steps ①–⑨). The robot moves to the ending point when all retrieval racks are picked and returned to storage locations (step ⑩). The starting/ending points are located in the same charging zone because the robot is usually charged when there is no delivery task.

In this paper, we focus primarily on the rack scheduling problem with multiple types of multiple storage locations (RS-MTMS), which is a unique scenario not considered before. More specifically, RS-MTMS is a sophisticated and interactive process where the storage assignment problem and retrieval sequence decision are intertwined. The storage assignment decides which storage location is to be selected to store the retrieval rack after it visits a picking station. In our RS-MTMS, the retrieval racks can be stored in multiple storage locations, including open storage locations, their own retrieval location, and other empty retrieval locations. The sequence decision problem decides the retrieval sequence of racks to be adopted at picking stations. Instead of visiting continuously between the retrieval locations, the robot moves from the storage location to the retrieval location of a new rack.

In the rack scheduling problem studied previously, the racks' storage assignment scheme and retrieval sequence are generally generated separately, and only a single type of storage location is assumed [12–14]. However, these studies limit the information for both decisions, and their assumption may result in the higher-quality solutions being discarded. Therefore, it is essential to study the RS-MTMS and develop a bi-level optimization approach for jointly optimizing the storage assignment and retrieval sequence (BiJSR).

In BiJSR, the storage locations of racks are obtained in the upper-level. Then, the optimal retrieval sequence is obtained in the lower-level according to the given storage locations. The upper-level optimization problem is solved using a variable neighborhood search (VNS). Sufficient problem-specific knowledge and effective candidate modes are incorporated into VNS to improve solution quality and computational

efficiency. A transformation technique is developed to solve the lower-level optimization problem. It transforms the retrieval sequence decision into the traveling salesman problem (TSP) and its variants. The originality and contributions of this paper are summarized as follows.

(1) A mathematical programming model is built for the novel RS-MTMS problem. The model reflects the interdependence between the storage location and retrieval sequence in determining the rack scheduling, and incorporates different kinds of practical constraints.

(2) We propose a bi-level optimization approach, BiJSR, to tackle the RS-MTMS problem effectively. In BiJSR, the upper-level storage assignment problem is solved by the VNS with problem-specific knowledge and effective candidate modes, and the lower-level retrieval sequence decision is solved by an effective transformation heuristic technique.

(3) We compare the proposed algorithm using popular optimization software and state-of-the-art algorithms to verify its effectiveness.

The remainder of this paper is structured as follows. The related studies are discussed in Section 2. Section 3 introduces the mathematical formulation of the RS-MTMS problem and its linear version. Section 4 illustrates the details of our proposed BiJSR. Sections 5 and 6 describe the details of developed algorithms for the lower-level and upper-level optimization problem, respectively. Computational experiments are shown in Section 7, and we conclude our work and offer insights for future research in Section 8. For clarity, the main notations employed in this paper are listed in Appendix A.

2 Related studies

Among the typical operations of a warehousing system, the order picking is the most time-consuming and labor-intensive process [15–17]. Traditional order picking is executed by order pickers walking or driving along the aisles to pick inventory products required by the orders. It will require a robust workforce and personnel because unproductive travel times reduce order pickers' productivity. Introducing unmanned and intelligent technology is one of the most promising ways of improving picking efficiency [18, 19]. In recent years, a novel parts-to-picker order picking system RMFS has been gaining popularity, and is used by more than 30 well-known e-commerce suppliers because of its superior picking performance. The existing literature on RMFS can be classified into three categories: system analysis [20–23], design optimization [24–27], and operations planning and control [28–31]. Our research belongs to the third category as it focuses on the scheduling of delivery tasks on the operational level.

Task scheduling is an essential decision problem that aims to achieve appropriate deployment of task resources under certain constraints and optimal performance indexes [32–35]. The storage assignment and rack sequencing optimization are two most basic and important parts of task scheduling in the RMFS. Merschformann [36] redefined the storage assignment as the rack repositioning problem and considered the active and passive repositioning situation. Yuan et al. [37] presented a paper that focused on the velocity-based storage assignment problem. A fluid model was developed to analyze the performance of velocity-based storage strategies. Weidinger et al. [12] researched the rack storage assignment problem and formalized the original problem as a special interval scheduling problem. Ji et al. [38] established the optimization models, including two cases: single rack storage assignment and multiple-rack storage assignments. They also designed the three-class-based strategy and Kuhn-Munkras algorithm to solve these problems. However, the studies mentioned above deal with storage assignments alone. In their studies, the retrieval sequence of racks is fixed, or only some simple sequencing decision strategies are embedded to verify the performance of the proposed algorithms.

For the rack sequencing optimization problem, the minimum number of rack visits to all picking stations is considered the objective in some studies. Boysen et al. [13] decided on the retrieval sequencing problem for a single picking station. To solve this problem, they presented a mixed-integer programming model and a decomposing frame based on the given order sequence. Yang et al. [14] and Zhuang et al. [28] extended the work of Boysen to the multiple picking stations, and developed different hybrid heuristic algorithms to find such a retrieval sequence. Valle and Beasley [7] investigated the rack sequencing problem at each single picking station as an integer program. The other way to consider the sequencing optimization is to minimize the total retrieval time. Ouzidan et al. [39] optimized the arriving sequence of racks while minimizing the time needed to fulfill all orders. Wang et al. [40] studied the rack scheduling problem using approximate dynamic programming with branch-and-price algorithms. Yuan et al. [41] presented a mixed integer programming model with the objective of make-span minimization. They also

proposed the two heuristic rules and an ant colony optimization algorithm to solve the problem. All of the studies above assume that each retrieval rack has a fixed location in the storage area. These studies do not consider the storage assignment and only deal with the sequencing optimization problem.

Because the storage assignment and sequencing optimization problem are both related to the order picking efficiency, these two problems need to be studied simultaneously. Merschformann et al. [10] structured the decisions at the operational level and developed the heuristic decision rules to solve these two problems. However, in their work, a sequential method was applied to first formulate the retrieval sequence and then assign the storage locations. Gharehgozli et al. [11] investigated a joint storage assignment and sequencing optimization problem, and presented the mathematical formulation and meta-heuristic solution algorithm. In their paper, instead of explicitly considering multiple different types of storage locations, they assume that racks can only be stored in open storage locations after visiting the picking stations.

This paper addresses the RS-MTMS, which is a case of a joint storage assignment and retrieval sequence optimization problem with multiple different types of multiple storage locations.

3 Problem description

The RS-MTMS problem is formulated as follows. There are a set of retrieval racks $R = \{r_1, r_2, \ldots, r_m\}$ and a set of open storage locations $\mathrm{Os} = \{\mathrm{os}_1, \mathrm{os}_2, \ldots, \mathrm{os}_n\}$ available for storing retrieval racks. Let $\Gamma = \{\zeta_1, \zeta_2, \ldots, \zeta_m, \zeta_{m+1}, \ldots, \zeta_{m+n}\}$ be the set of position of all locations consisting of retrieval locations and storage locations. Each rack r_i is assigned to a fixed picking station p_i, whose location is ζ_p^i. The starting point and ending point are l_0 and l_0', both of which are at ζ_0. In our scenario, the distance of a move depends on whether it is a carrying move. Specifically, for the non-carrying move of a delivery task (the first part), we can directly use the Manhattan distance as $t(i,j) = |\zeta x_i - \zeta x_j| + |\zeta y_i - \zeta y_j|$, where $(\zeta x_i, \zeta y_i)$ and $(\zeta x_j, \zeta y_j)$ are the Cartesian coordinates of two endpoint locations i and j of a non-carrying move. However, the robot is carrying a retrieval rack in the second and third parts for a delivery task. Some additional travel distances need to be added to their distances. Lamballais et al. [20] reported in detail how to calculate the distance of a carrying move in an RMFS. Taking as an example a simple case of the second part, the picking station is located north of the scenario and the entrance of the retrieval location is situated in an aisle with a northward travel direction. The distance is calculated as $u + d_{\mathrm{le,si}} + |\zeta x_i - \zeta x_j| + |\zeta y_i - \zeta y_j| + \Delta_{\mathrm{le},p}$, where u is the distance between the retrieval location and its location entrance, and $d_{\mathrm{le,si}}$ is the distance between the location entrance and the start intersection. A detour distance $\Delta_{\mathrm{le},p}$ should be considered owing to the adjustment of travel directions in the hall in front of the picking station. Once the travel distances are obtained, they should be divided by the robot speed to calculate the travel time. In addition, the time related to lifting or storing rack is considered for the carrying move.

For the RS-MTMS problem, the different scheduling schemes can directly impact the travel time necessary to deliver all retrieval racks. Figure 2 depicts an example with two retrieval racks r_1 and r_2, two open storage locations os_1 and os_2, and $\Gamma = \{(1,0),(3,0),(2,0),(5,0)\}$. The locations of the starting point and ending point are both (0, 0). For the sake of brevity, the travel time of the carrying move between the picking station p_1 and Γ is given as $\mathrm{Tp}_1 = [5,5,3,7]$ and $\mathrm{Tp}_2 = [8,5,6,5]$ for p_2 according to [20]. The robot speed is assumed to be 1 and the acceleration and deceleration are not considered. There are four different solutions. In Solution (a), the storage locations of r_1 and r_2 are selected as os_1 and os_2, and the moves of the mobile robot are as follows: $l_0 \to r_1 \to p_1 \to \mathrm{os}_1 \to r_2 \to p_2 \to \mathrm{os}_2 \to l_0'$. Hence, the total travel time of the mobile robot in Solution (a) is 25. Similarly, the total travel times of Solutions (b)–(d) are 32, 26, and 24, respectively. Minimizing the total travel time is the objective of RS-MTMS because it can guarantee a higher picking efficiency [10,11,20].

Applying the notations in Table 1, we provide the mathematical formulation of the RS-MTMS problem.

$$\mathrm{P}: \min \ \ \mathrm{Te}_m + \mathrm{Tsl} \tag{1}$$

$$\text{s.t.} \ \sum_{i=1}^{m} x_{ij} = 1, \ \forall j \in \{1, \ldots, m\}, \tag{2}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \ \forall i \in \{1, \ldots, m\}, \tag{3}$$

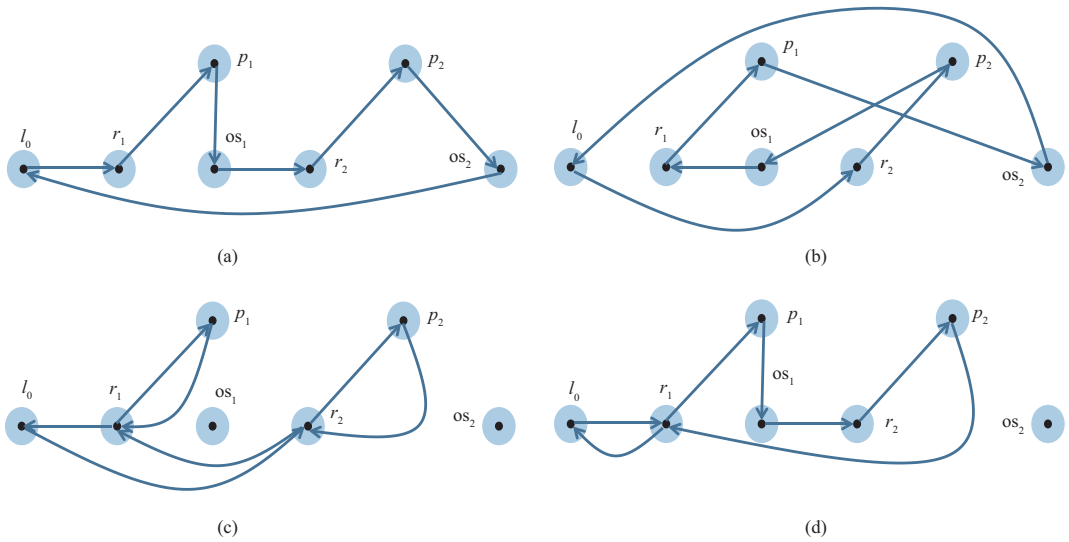

Figure 2 (Color online) Four scheduling example schemes. (a) Solution with cost = 25; (b) solution with cost = 32; (c) solution with cost = 26; (d) solution with cost = 24.

Table 1 Notation declaration of mathematical formulation

Parameter	Description
m	Number of retrieval racks
n	Number of open storage locations
TP	The travel time matrix between all picking stations and all locations Γ
x_{ij}	Binary decision variables: 1, if the rack j is delivered in the i task; otherwise, 0, $\forall i, j \in \{1, \ldots, m\}$
y_{iw}	Binary decision variables: 1, if the rack delivered in the i task is stored in the w location; otherwise, 0, $\forall i \in \{1, \ldots, m\}, w \in \{1, \ldots, m+n\}$
Rl_i	The index of the retrieval rack for the ith delivery task, $\forall i = 1, 2, \ldots, m$
Sl_i	The index of the storage location for the ith delivery task, $\forall i = 1, 2, \ldots, m$
Tlr	The travel time between the l_0 and Rl_1
Tsr_i	The travel time between the Sl_{i-1} and Rl_i, $\forall i = 2, \ldots, m$
Trp_i	The travel time between the Rl_i and p_i, $\forall i = 1, \ldots, m$
Tps_i	The travel time between the p_i and Sl_i, $\forall i = 1, \ldots, m$
Tsl	The travel time between the l_0' and Sl_m
Ts_i	The time to arrive at x_i for the ith delivery task, $\forall i = 1, \ldots, m$
Te_i	The time to arrive at y_i for the ith delivery task, $\forall i = 1, \ldots, m$

$$\sum_{i=1}^{m} y_{iw} \leqslant 1, \ \forall w \in \{1, \ldots, m+n\}, \tag{4}$$

$$\sum_{w=1}^{m+n} y_{iw} = 1, \ \forall i \in \{1, \ldots, m\}, \tag{5}$$

$$\mathrm{Ts}_1 = \mathrm{Tlr}, \tag{6}$$

$$\mathrm{Te}_i = \mathrm{Ts}_i + \mathrm{Trp}_i + \mathrm{Tps}_i, \ \forall i \in \{1, \ldots, m\}, \tag{7}$$

$$\mathrm{Ts}_i = \mathrm{Te}_{i-1} + \mathrm{Tsr}_i, \ \forall i \in \{1, \ldots, m\}, \tag{8}$$

$$\mathrm{RL}_i = \max\{j \times x_{ij}\}, \ \forall i, j \in \{1, \ldots, m\}, \tag{9}$$

$$\mathrm{SL}_i = \max\{w \times y_{iw}\}, \ \forall i \in \{1, \ldots, m\}, w \in \{1, \ldots, m+n\}, \tag{10}$$

$$\mathrm{Te}_i - \mathrm{Ts}_j \geqslant \tau, \ \text{if } \mathrm{RL}_i = \mathrm{SL}_j, \ \forall i, j \in \{1, 2, \ldots, m\}, \tag{11}$$

$$x_{ij}, y_{iw} \in \{0, 1\}, \ \forall i, j \in \{1, \ldots, m\}, w \in \{1, \ldots, m+n\}. \tag{12}$$

Objective (1) is to minimize the total travel time to complete all delivery tasks. Constraints (2) and (3) guarantee that each retrieval rack can only be assigned to one delivery task for processing. Constraints (4)

and (5) state that each retrieval rack must be stored, and that each storage location can store at most one retrieval rack. Constraints (6)–(8) are constraints on the delivery task's starting time and ending time when carrying the retrieval rack. Constraint (6) is the constraint on the starting time upon arriving at the location of the first retrieval rack. The Tlr can be calculated as (13) because it is a non-carrying move.

$$\mathrm{Tlr} = \left| \sum_{j=1}^{m} (\zeta x_j \times x_{1j}) - \zeta x_0 \right| + \left| \sum_{j=1}^{m} (\zeta y_j \times x_{1j}) - \zeta y_0 \right|. \tag{13}$$

Constraints (7) and (8) state that the ending/starting time of the delivery task can be calculated after its starting/ending time has been determined. The travel time Trp_i between the retrieval rack r_i and its corresponding picking station p_i is

$$\mathrm{Trp}_i = \sum_{p=1}^{m} \sum_{j=1}^{m} \mathrm{Tp}_{pj} \times \mathrm{aux1}_{i,p,j},\ \forall i \in \{1, \ldots, m\}, \tag{14}$$

$$\sum_{p=1}^{m} \mathrm{aux1}_{i,p,j} = x_{i,j}, \forall i, j \in \{1, \ldots, m\}, \tag{15}$$

$$\mathrm{aux1}_{i,p,j} \in \{0, 1\}, \forall i, j, p \in \{1, \ldots, m\}, \tag{16}$$

where aux1 is an auxiliary variable. For the storage move of r_i, Tps_i can also be calculated as follows:

$$\mathrm{Tps}_i = \sum_{p=1}^{m} \sum_{w=1}^{m+n} \mathrm{Tp}_{pw} \times \mathrm{aux2}_{i,p,w},\ \forall i \in \{1, \ldots, m\}, \tag{17}$$

$$\sum_{p=1}^{m} \mathrm{aux2}_{i,p,w} = y_{i,w},\ \forall i \in \{1, \ldots, m\}, w \in \{1, \ldots, m+n\}, \tag{18}$$

$$\mathrm{aux2}_{i,p,w} \in \{0, 1\},\ \forall i, p \in \{1, \ldots, m\}, w \in \{1, \ldots, m+n\}. \tag{19}$$

Similar to formula (13), Tsr_i represents a non-carrying move from the storage location of r_{i-1} to the retrieval location of r_i, and is given as follows:

$$\begin{aligned} \mathrm{Tsr}_i = &\left| \sum_{j=1}^{m} \zeta x_j \times x_{ij} - \sum_{w=1}^{m+n} \zeta x_w \times y_{i-1,w} \right| \\ &+ \left| \sum_{j=1}^{m} \zeta y_j \times x_{ij} - \sum_{w=1}^{m+n} \zeta y_w \times y_{i-1,w} \right|,\ \forall i \in \{2, \ldots, m\}. \end{aligned} \tag{20}$$

Constraints (9)–(11) indicate that a retrieval rack r_i can be stored in the retrieval location of r_j only after the retrieval rack r_j has been delivered. τ is a small positive value used for this precedence constraint. The robot returns to the ending point without carrying the rack after performing all the delivery tasks. Hence, Tsl in objective (1) can be calculated as

$$\mathrm{Tsl} = \left| \sum_{w=1}^{m+n} (\zeta x_w \times y_{mw}) - \zeta x_0 \right| + \left| \sum_{w=1}^{m+n} (\zeta y_w \times y_{mw}) - \zeta y_0 \right|. \tag{21}$$

We can observe that P is a nonlinear optimization problem because there are absolute value terms (14) and (21) and the logical constraint (11) in P. To make it easier to solve this problem with the optimization solver, we transform the nonlinear constraints into linear constraints. To linearize the absolute value terms, we use t_x to represent $|\zeta x_i - \zeta x_j|$, and we introduce an auxiliary variables aux3 and a large positive value M_1. The linearized computation of the x-coordinate is shown in (22)–(26). Similarly, we can also linearize the computation of the y-coordinate.

$$\zeta x_i - \zeta x_j \leqslant t_x, \tag{22}$$

$$\zeta x_j - \zeta x_i \leqslant t_x, \tag{23}$$

$$\zeta x_i - \zeta x_j \geqslant t_x - M_1 \times (1 - \text{aux3}), \tag{24}$$
$$\zeta x_j - \zeta x_i \geqslant t_x - M_1 \times \text{aux3}, \tag{25}$$
$$\text{aux3} \in \{0, 1\}. \tag{26}$$

For the logical constraint (11), we use an auxiliary variable aux4 and two different large positive values M_2 and M_3 to obtain its linear formulation.

$$\text{Te}_i - \text{Ts}_j \geqslant M_2 \times (\text{Sl}_i - \text{Rl}_j) - M_3 \times (1 - \text{aux4}_{ij}) + \tau, \tag{27}$$
$$\text{Te}_i - \text{Ts}_j \geqslant M_2 \times (\text{Rl}_j - \text{Sl}_i) - M_3 \times \text{aux4}_{ij} + \tau, \tag{28}$$
$$M_2 \ll M_3, \tag{29}$$
$$\text{aux4}_{ij} \in \{0, 1\},\ i, j \in \{1, \ldots, m\}. \tag{30}$$

We further develop a relaxation model to obtain the lower bound. Specifically, we relax the original problem by removing the precedence constraints (9)–(11). The objective is bounded below by the optimal value of this relaxation model.

4 Proposed bi-level optimization model

In this section, we describe BiJSR, which is a bi-level optimization model for jointly optimizing the storage assignment and retrieval sequence. Given the notations, the BiJSR model is stated as follows:

$$\min_{S \in \Phi, \text{Rs} \in \Omega_{\text{Rs}}(S)} F(S, \text{Rs}) = \phi(\text{Rs}) \tag{31}$$
$$\text{s.t. :Rs} \in \arg \min_{\text{Rs}' \in \Omega_{\text{Rs}}(S)} \phi(\text{Rs}'), \tag{32}$$

where

$$\phi(\text{Rs}) = \sum_{i=1}^{m} f(\text{Rs}_i) + t(\zeta_0, \zeta_{r_1}) + t(\zeta_{s_m}, \zeta_0), \tag{33}$$

$$f(\text{Rs}_i) = \begin{cases} \text{Tp}(r_i, r_i) + \text{Tp}(r_i, s_i + m), & i = 1, \\ t(\zeta_{s_{i-1}}, \zeta_{r_i}) + \text{Tp}(r_i, r_i) + \text{Tp}(r_i, s_i + m), & i > 1. \end{cases} \tag{34}$$

The $F(S, \text{Rs})$ is the objective function representing the total travel time to complete all delivery tasks. Eqs. (31) and (32) are respectively the upper-level and lower-level optimization. The upper-level optimization aims to find the optimal storage assignment set S^*, while the lower-level optimization is to find the optimal retrieval sequence $\text{Rs}^*(S^*)$ under the storage assignment S^* in terms of the objective value $\phi(\cdot)$. The optimal solution of RS-MTMS is composed of the optimal storage assignment set S^* and the optimal retrieval sequence Rs^* under this set. Therefore, optimality can be guaranteed in the BiJSR model. That is, RS-MTMS and BiJSR have the same optimal solution.

In the proposed BiJSR, storage assignment set S can be represented as a list $\{s_1, s_2, \ldots, s_m\}$, where each s_i is contained in Γ, and they are different from each other. Note that s_i only represents an element in S, rather than the corresponding storage item of r_i. For example, in Figure 2(b), os_1 and os_2 are selected as the storage assignment set $S = \{\text{os}_1, \text{os}_2\}$, while the storage locations of the r_1 and r_2 are os_2 and os_1, respectively. The rack can also be stored in its own retrieval location or other empty retrieval locations, such as in Figures 2(c) and (d), $S = \{r_1, r_2\}$ and $S = \{r_1, \text{os}_1\}$. A solution Rs is considered to be a feasible solution if the storage location of each retrieval rack comes from a unique location in S. Suppose there are two retrieval racks, two open storage locations, and two picking stations. Then, the retrieval sequences are $l_0 \to r_2 \to p_2 \to \text{os}_1 \to r_1 \to p_1 \to \text{os}_2 \to l_0'$ in Figure 2(b), $l_0 \to r_1 \to p_1 \to r_1 \to r_2 \to p_2 \to r_2 \to l_0'$ in Figure 2(c), and $l_0 \to r_1 \to p_1 \to \text{os}_1 \to r_2 \to p_2 \to \text{os}_2 \to l_0'$ in Figure 2(d).

5 Lower-level optimization

Based on (31) and (32), the fitness function of a storage assignment S can be set to the objective value of the optimal solution $\text{Rs}^*(S)$ under S. That is

$$\text{eval}(S) = \phi(\text{Rs}^*(S)) = \min_{\text{Rs} \in \Omega_{\text{Rs}}(S)} \phi(\text{Rs}). \tag{35}$$

Algorithm 1 The framework of lower-level optimization

Input: A storage assignment set S, the retrieval rack set R.
Output: The best retrieval sequence $\mathrm{Rs}^*(S)$.
1: **if** $S \cap R \neq \emptyset$ **then**
2: **if** $S \subseteq R$ **then**
3: Execute Case I to obtain the $\mathrm{Rs}^*(S)$;
4: **else**
5: Execute Case III to obtain the $\mathrm{Rs}^*(S)$;
6: **end if**
7: **else**
8: Execute Case II to obtain the $\mathrm{Rs}^*(S)$;
9: **end if**
10: **return** $\mathrm{Rs}^*(S)$.

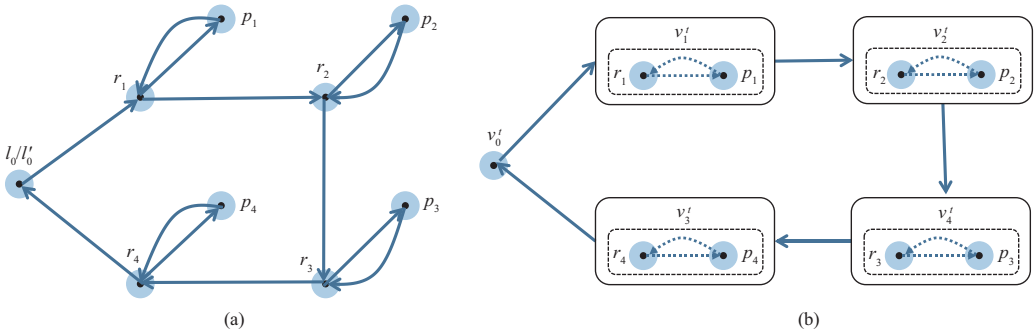

Figure 3 (Color online) Schematic description of Case I. (a) An example in the lower-level optimization; (b) the identified TSP transformed from (a).

To find $\mathrm{Rs}^*(S)$, we develop the transformation heuristic to transform the lower-level optimization problem to TSP, asymmetric TSP (ATSP), and ATSP with precedence constraints (ATSP-PC) according to the variable S in the following three special cases. The framework of the lower-level optimization strategy is given in Algorithm 1.

5.1 Case I: identifying the TSP

In this subsection, we discuss the first case where $S = \{r_1, r_2, \ldots, r_m\}$. As shown in Figure 3(a), once all products are retrieved from the retrieval racks, these racks are all stored in their initial retrieval locations.

Theorem 1. If each retrieval rack is stored in its own retrieval location, the lower-level optimization problem is formulated as a TSP problem.

Proof. In our problem, each retrieval rack is first moved to the picking station and then stored in storage. When the storage location is selected as its initial location, this travel time from the sequence segment $r_i \to p_i \to r_i$ is a value that is independent of the decision of the retrieval sequence. Hence, the sequence segment corresponding to r_i can be regarded as a retrieval node v_i^t with a fixed cost $2\mathrm{Tp}_{i,i}$. In addition, an extra node v_0^t can be used to represent the starting point l_0 and ending point l_0' because their locations are the same. The cost matrix can be defined as an $(m+1) \times (m+1)$ matrix:

$$D_t = \begin{bmatrix} d_{00}^t & d_{01}^t & \cdots & d_{0m}^t \\ d_{10}^t & d_{11}^t & \cdots & d_{1m}^t \\ \vdots & \vdots & \ddots & \vdots \\ d_{m0}^t & d_{m1}^t & \cdots & d_{mm}^t \end{bmatrix}, \tag{36}$$

where d_{ij} is the distance from node v_i^t to v_j^t that can be calculated as

$$d_{ij}^t = \begin{cases} M_4, \ i = j, \\ t\left(\zeta_0, \zeta_{r_j}\right), \ i = 0, j \in \{1, 2, \ldots, m\}, \\ t\left(\zeta_{r_i}, \zeta_0\right), \ i \in \{1, 2, \ldots, m\}, j = 0, \\ t\left(\zeta_{r_i}, \zeta_{r_j}\right), \ i, j \in \{1, \ldots, m\}, i \neq j, \end{cases} \tag{37}$$

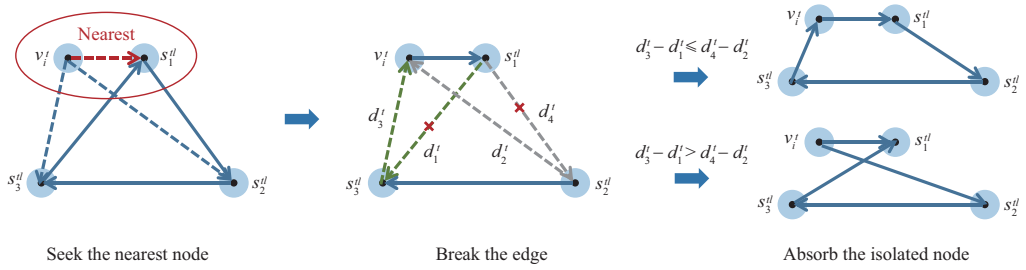

Figure 4 (Color online) The absorption strategy for an isolated node.

where M_4 is a big value.

Obviously, D_t is a symmetric matrix. Hence, as shown in Figure 3(b), we can rephrase the lower-level optimization problem as how to choose a tour in TSP, where every node is visited without repetition and the starting and ending nodes are the same. Let $G_t = (\upsilon^t, \varepsilon^t)$ represent an instance of the identified TSP, where $\upsilon^t = \{{v_0}^t, {v_1}^t, \ldots, {v_m}^t\}$. The objective function $\phi\,(\mathrm{Rs})$ is defined as the all fixed cost and the sum of the cost of G_t divided by robot speed:

$$\phi\,(\mathrm{Rs}) = \phi\,(G_t) + 2\sum_{i=1}^{m} \mathrm{Tp}_{i,i}. \tag{38}$$

In this situation, we extend our previous study [42] and propose a loop-based heuristic (LBH) strategy to address the identified TSP. The key steps are as follows.

(a) Initialize the solution to an empty set.

(b) Find the shortest edge among the ε_t and add the two endpoints of this edge to the initial solution.

(c) Add the nearest neighbor of these two nodes as the third node to the current solution. Now the three selected nodes in the initial solution form a small loop.

(d) Absorb a "best" isolated node into the initial solution. In order to extend the solution, the current loop needs to absorb an isolated node from all nodes that have not been added. For an isolated node v_i^t and a loop s^{tl} with α nodes, we first seek the node in s^{tl} that is closest to v_i^t,

$$v_s^{tl} = \arg\min\left\{d^t_{v_i^t, s_1^{tl}}, d^t_{v_i^t, s_2^{tl}}, \ldots, d^t_{v_i^t, s_\alpha^{tl}}\right\}, \tag{39}$$

where $d^t_{v_i^t, s_j^{tl}}$ is the distance between node v_i^t and the jth node in the s^{tl}.

We then evaluate the absorption cost for each node-edge combination for the two edges connected to node v_s^{tl}. As shown in Figure 4, the loop can absorb the node successfully by breaking the edge to expose the two endpoints and connecting the isolated node with the endpoints. The minimum absorption cost of the two node-edge combinations is used as the evaluation value for absorbing v_i^t. Here the absorption cost refers to the change of the cost value in the loop after absorbing the isolated node. We consider all the situations and greedily select the "best" isolated node with the smallest evaluation value.

(e) Repeat steps (d) until all isolated nodes are added to the initial solution.

(f) After the initial tour is constructed, a 2-opt operator [43] is applied to it to obtain a higher-quality solution. Note that to obtain the lower-level solution Rs, each node in the obtained tour is restored to the starting (or ending) point or the sequence segment of the retrieval rack.

Remark 1. In our problem, the robot starts from the starting point and returns to the ending point after performing all of the delivery tasks. Therefore, the first and last positions in Rs should be adjusted to l_0 and l_0' if v_0^t is not visited first in the tour. For example, if the obtained tour is $\{v_1^t, v_0^t, v_2^t\}$, the transformed Rs should be $l_0 \to r_2 \to p_2 \to r_2 \to r_1 \to p_1 \to r_1 \to l_0'$. This operation does not change the objective value because the connection between nodes is not varied.

5.2 Case II: identifying the ATSP

This subsection discusses the second case where $S = \{s_i \,|s_i \in \mathrm{Os},\ s_i \neq s_j, \forall i, j \in \{1, \ldots, m\}\}$. As shown in Figure 5, four racks are stored in different open storage locations after leaving the picking stations.

Theorem 2. If each retrieval rack is stored in the open storage location, the lower-level optimization problem is formulated as an ATSP problem.

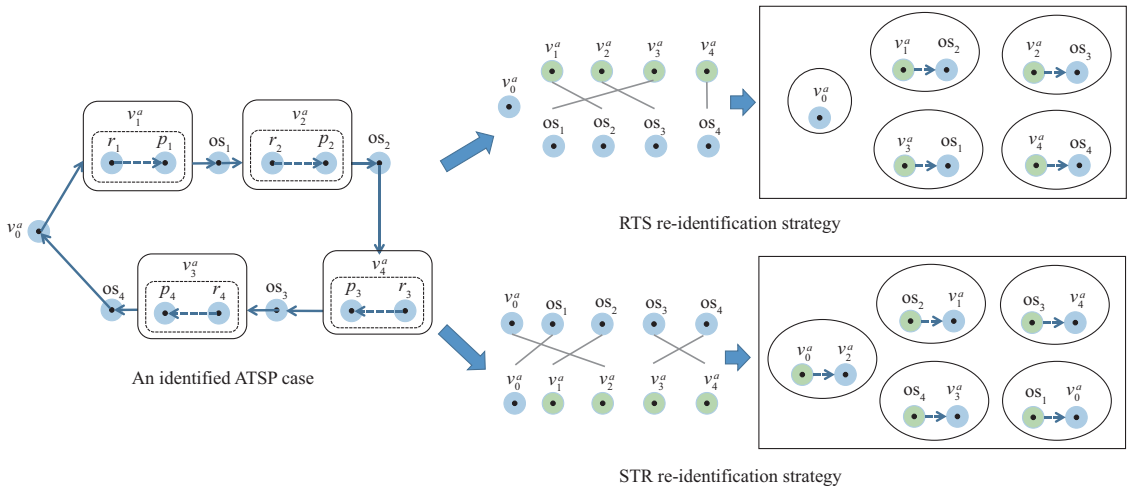

Figure 5 (Color online) Schematic description of Case II.

Proof. In the TSP-case, a sequence segment from the retrieval location to the storage location can be regarded as a node because each rack's storage area is its own retrieval location. Although the rack returns to the open storage location in this case, the connection between the rack and the picking station is still fixed. Therefore, the sequence segment $r_i \to p_i$ corresponding to r_i can also be regarded as a retrieval node v_i^a with a fixed cost $\mathrm{Tp}_{i,i}$. Similarly, the starting and ending points are regarded as one node v_0^a.

In an RMFS, a robot needs to move one storage location after leaving the picking station, and then it moves to the retrieval location of the next delivery task. The connections $v_i^a \to v_j^a$ and $s_i \to s_j$ for $i, j \in \{1, \ldots, m\}$ are infeasible because these two steps are consecutive and they cannot be interrupted. In addition, we can see that a robot can only move from the starting point to the retrieval location and return to the ending point from the storage position. Hence, the connections $v_0^a \to s_i$ and $v_i^a \to v_0^a$ for $i \in \{1, \ldots, m\}$ are also infeasible. We set the cost of those infeasible connections to the large values and give the following $(2m+1) \times (2m+1)$ cost matrix D_a:

$$d_{ij}^a = \begin{cases} t(\zeta_0, \zeta_{r_j}),\ i = 0, j \in \{1, \ldots, m\}, \\ M_4,\ i = 0, j \in \{0, m+1, \ldots, 2m\}, \\ M_4,\ i \in \{1, \ldots, m\}, j \in \{0, 1, \ldots, m\}, \\ \mathrm{Tp}_{i,j},\ i \in \{1, \ldots, m\}, j \in \{m+1, \ldots, 2m\}, \\ t\left(\zeta_{s_{i-m}}, \zeta_0\right),\ i \in \{m+1, \ldots, 2m\}, j = 0, \\ t\left(\zeta_{s_{i-m}}, \zeta_{r_j}\right),\ i \in \{m+1, \ldots, 2m\},\ j \in \{1, \ldots, m\}, \\ M_4,\ i, j \in \{m+1, \ldots, 2m\}, \end{cases} \tag{40}$$

where d_{ij}^a represents the distance from v_i^a to v_j^a.

The proof of identified ATSP is intuitive according to D_a. Hence, the lower-level optimization problem is also reformulated as one that shows how to solve an ATSP problem. Let $G_a = (\upsilon^a, \varepsilon^a)$ represent an instance of the identified ATSP, where $\upsilon^a = \{{v_0}^a, {v_1}^a, \ldots, v_{2m}^a\}$. The $\left\{v_{m+1}^a, \ldots, v_{2m}^a\right\}$ denotes the storage set $\{s_1, s_2, \ldots, s_m\}$. According to the cost matrix D_a, a solution is feasible ATSP-tour if the following conditions are satisfied:

Condition 1. Every node must be carried out exactly once and must be involved in a tour.

Condition 2. In a tour, the node pointed from v_0^a must be a node from $\{v_i^a | i = 1, 2, \ldots, m\}$, and the v_0^a node can only be pointed to by a node from $\left\{v_j^a | j = m+1, m+2, \ldots, 2m\right\}$.

Condition 3. The node from $\{v_i^a | i = 1, 2, \ldots, m\}$ and the node from $\left\{v_j^a | j = m+1, m+2, \ldots, 2m\right\}$ are visited alternately in a tour.

The objective function $\phi(\mathrm{Rs})$ is defined as all fixed cost and the sum of the cost of G_a divided by robot speed.

$$\phi(\mathrm{Rs}) = \phi(G_a) + \sum_{i=1}^{m} \mathrm{Tp}_{i,i}. \tag{41}$$

In order to embed the above conditions in the LBH strategy to improve the solving efficiency, we need to add a re-identification strategy before constructing the solution. This additional strategy can transform the ATSP problem heuristically into a simplified ATSP problem. We propose two node-assignment modes to build the re-identification strategy.

5.2.1 *Retrieval to storage (RTS)*

In a feasible tour, the node pointed from each retrieval node is one unique storage node. Hence, we can allocate one storage node to each retrieval node without duplication. The assignment scheme with the minimum cost will be selected, where the cost is defined as $\sum_{i=1}^{m} d^a_{v^a_i, v^a_{i+m}}$ for an assignment combination $\{(v^a_i, v^a_{i+m}) | i = 1, 2, \ldots, m\}$. Because the number of retrieval nodes is the same as that of storage nodes, this problem can be approximated as a classical quadratic assignment problem, and thus solved by the Hungarian algorithm [44]. For the obtained assignment scheme, we connect each pair $(v^a_{\gamma_i}, v^a_{\chi_i}), \forall \gamma_i \in \{1, \ldots, m\}, \chi_i \in \{m+1, \ldots, 2m\}$ and regard this connection as a new node v^{ar}_i. The cost of each v^{ar}_i is the distance from node v_{γ_i} to v_{χ_i} in D_a. Letting v^{ar}_0 represent node v^a_0, the cost matrix is represented as an $(m+1) \times (m+1)$ matrix D_{ar}:

$$d^{\mathrm{ar}}_{ij} = \begin{cases} M_4, \ i = j, \\ t\left(\zeta_0, \zeta_{r_j}\right), \ i = 0, j \in \{1, \ldots, m\}, \\ t\left(\zeta_{s_{\chi_i - m}}, \zeta_0\right), \ i \in \{1, \ldots, m\}, j = 0, \\ t\left(\zeta_{s_{\chi_i - m}}, \zeta_{r_j}\right), \ i, j \in \{1, \ldots, m\}, i \neq j, \end{cases} \tag{42}$$

where d^{ar}_{ij} is an item of D_{ar}, and represents the distance from node v^{ar}_i to v^{ar}_j. Obviously the re-identified problem using D_{ar} is also an ATSP problem. Letting $G_{ar} = (v^{\mathrm{ar}}, \varepsilon^{\mathrm{ar}})$ represent an instance, $\phi(G_a)$ is equivalent to the sum of the objective value of G_{ar} and the all assignment cost

$$\phi(G_a) = \phi(G_{ar}) + \sum_{i=1}^{m} d^a_{v^a_{\gamma_i}, v^a_{\chi_i}}. \tag{43}$$

5.2.2 *Storage to retrieval (STR)*

Analogous to the RTS mode, the other is a re-identification strategy from the storage perspective. Consider another feasible case in Figure 5. If we disregard its concatenation at the starting and ending nodes, its remaining part can be considered as a concatenation of $m-1$ pairs from the storage node to the retrieval node. In other words, except for the last storage node and the first retrieval node, the node pointed from each storage node is one unique retrieval node. According to Condition 2, the starting and ending points must be connected with a retrieval node and a storage node, respectively. Hence, we can establish an allocation problem between $\{v^a_0, v^a_1, v^a_2, \ldots, v^a_m\}$ and $\{v^a_0, v^a_{m+1}, \ldots, v^a_{2m}\}$. Obviously, it can also be solved by the Hungarian algorithm. In order to unify the representation, we also use v^{ar}_i to represent the connection $v^a_{\chi_i} \to v^a_{\gamma_i}$ for $i = 1, 2, \ldots, m-1$, and v^{ar}_0 and v^{ar}_m represent the connection $v^a_0 \to v^a_{\gamma_m}$ and $v^a_{\chi_m} \to v^a_0$. Each item of the cost matrix D_{ar} can be represented as

$$d^{\mathrm{ar}}_{ij} = \begin{cases} M_4, \ i = j, \\ t(\mathrm{TP}_{\gamma_m, \chi_j}), \ i = 0, j \in \{1, \ldots, m-1\}, \\ M_4, \ i = 0, j = m, \\ M_4, \ i \in \{1, \ldots, m-1\}, j = 0, \\ t(\mathrm{TP}_{\gamma_m, \chi_j}), \ i, j \in \{1, \ldots, m-1\}, i \neq j, \\ t(\mathrm{TP}_{\gamma_m, \chi_j}), \ i \in \{1, \ldots, m-1\}, j = m, \\ 0, \ i = m, j = 0, \\ M_4, \ i = m, j \in \{1, \ldots, m-1\}. \end{cases} \tag{44}$$

The transformation from the objective value $\phi(G_{ar})$ to $\phi(G_a)$ can be defined as

$$\phi(G_a) = \phi(G_{ar}) + \sum_{i=1}^{m-1} d^a_{v_{\chi_i}, v_{\gamma_i}} + d^a_{0, v_{\gamma_m}} + d^a_{v_{\chi_m}, 0}. \tag{45}$$

Algorithm 2 LBH-A for identified ATSP-case

Input: Identified ATSP graph $G_{ar} = (v^{\mathrm{ar}}, \varepsilon^{\mathrm{ar}})$, the cost matrix D_{ar}.
Output: The lower-level optimization solution Rs.
Transform an identified ATSP instance G_a into a re-identified instance G_{ar} according to (42) and (43) for RTS or (44) and (45) for STR;
Set cL $\leftarrow \emptyset$;
for each node $v_i^{\mathrm{ar}} \in v^{\mathrm{ar}}, i = 1, 2, \ldots, m+1$ **do**
  for each node $v_j^{\mathrm{ar}} \in v^{\mathrm{ar}}, j = i+1, \ldots, m+1$ **do**
    Calculate the cost of the loop containing nodes $(v_i^{\mathrm{ar}}, v_j^{\mathrm{ar}})$ and add it to cL;
  end for
end for
Select the smallest loop s^{al} in cL as an initial loop;
Delete the nodes in s^{al} from v^{ar} to obtain an isolated node set Isn;
repeat
  Set cA $\leftarrow \emptyset$;
  for each node $v_k^{\mathrm{ar}} \in$ Isn, $k = 1, 2, \ldots, |\mathrm{Isn}|$ **do**
    Calculate the absorption cost between s^{al} and v_k^{ar} according to (39) and (40) and add the cost to cA;
  end for
  Select the smallest item in cA as an absorbed node v_{s*}^{ar}, delete v_{s*}^{ar} from Isn, and add it to s^{al};
until All the isolated node is absorbed Isn $= \emptyset$;
Re-transform the s^{al} of G_{ar} into a tour of G_a, and execute the 2-opt operator with the feasibility check on the tour of G_a;
Convert the tour of G_a to the lower-level optimization solution Rs;
return Rs.

5.2.3 *Extension of LBH*

We solve this re-identified problem on G_{ar} with the LBH but change the evaluation function and absorption strategy. Because of the asymmetric properties of the cost matrix, two connected nodes $(v_i^{\mathrm{ar}}, v_j^{\mathrm{ar}})$ can be regarded as a loop whose cost value is the sum of the cost of edge $v_i^{\mathrm{ar}} \to v_j^{\mathrm{ar}}$ and the cost of $v_j^{\mathrm{ar}} \to v_i^{\mathrm{ar}}$. In this case, we first find the smallest loop instead of an edge in step (b). Then, steps (c) and (d) will be combined to form a new absorption strategy. For a formed loop s^{al} with β edges and an isolated node v_k^{ar}, the absorption cost is defined as

$$\mathrm{cA}_k^l = \min\left\{\mathrm{dis}\left(v_k^{\mathrm{ar}}, s_j^{\mathrm{al}}\right) | j = 1, 2, \ldots, \beta\right\}, \tag{46}$$

where $\mathrm{dis}(v_k^{\mathrm{ar}}, s_j^{\mathrm{al}})$ represents the exchange cost between v_k^{ar} and the jth edge in s^{al}. Here the exchange cost refers to the change of the cost value of the loop after breaking the edge s_j^{al} and connecting the isolated node with the exposed two endpoints:

$$\mathrm{dis}\left(v_k^{\mathrm{ar}}, s_j^{\mathrm{al}}\right) = d^{\mathrm{ar}}_{v_k^{\mathrm{ar}}, \mathrm{vl}_j^{\mathrm{al}}} + d^{\mathrm{ar}}_{v_k^{\mathrm{ar}}, \mathrm{vr}_j^{\mathrm{al}}} - d^{\mathrm{ar}}_{\mathrm{vr}_j^{\mathrm{al}}, \mathrm{vl}_j^{\mathrm{al}}}. \tag{47}$$

The $\mathrm{vr}_j^{\mathrm{al}}, \mathrm{vl}_j^{\mathrm{al}}$ are two endpoints of the edge s_j^{al} with the connection $\mathrm{vr}_j^{\mathrm{al}} \to \mathrm{vl}_j^{\mathrm{al}}$. All of the node-loop combinations are considered, and the item with the smallest absorption cost will be selected.

Repeat the absorption strategy until all isolated nodes are absorbed and an initial tour of G_{ar} is obtained. In step (f), we first restore each v_i^{ar} in the initial tour to the corresponding sequence pair, thus obtaining the tour in G_a. Then, a 2-opt operator is applied to improve the quality of the solution. Note that only solutions satisfying the above three feasibility conditions will be accepted during the local search process. Finally, the improved solution on G_a is converted into the lower-level optimization solution Rs as in the TSP case. Algorithm 2 describes an algorithmic description of the LBH applied to the identified ATSP case (LBH-A).

5.3 Case III: identifying the ATSP-PC

This subsection discusses the third case where the retrieval racks are allowed to be stored in storage locations, including open storage locations and retrieval locations, namely, $S = S_{\mathrm{rl}} \cup S_{\mathrm{os}}$ where $S_{\mathrm{rl}} = \{s_i | s_i \in R,\ s_i \neq s_j, \forall i \in \{1, \ldots, m_1\}\}$, $S_{\mathrm{os}} = \{s_i | s_i \in \mathrm{Os},\ s_i \neq s_j, \forall i \in \{m_1 + 1, \ldots, m\}\}$, and m_1 is the number of items belonging to the retrieval locations in S. Specifically, as shown in Figure 6(a), when all products are retrieved from a rack, this rack can be stored in one of the open storage locations, its own retrieval location, and other empty retrieval locations.

In this case, as shown in Figure 6(b), let a retrieval node v_i^c denote a sequence segment $r_i \to p_i$ because the connection between the rack and the picking station is still fixed. However, some additional precedence constraints exist in the connection between retrieval nodes and storage nodes owing to the

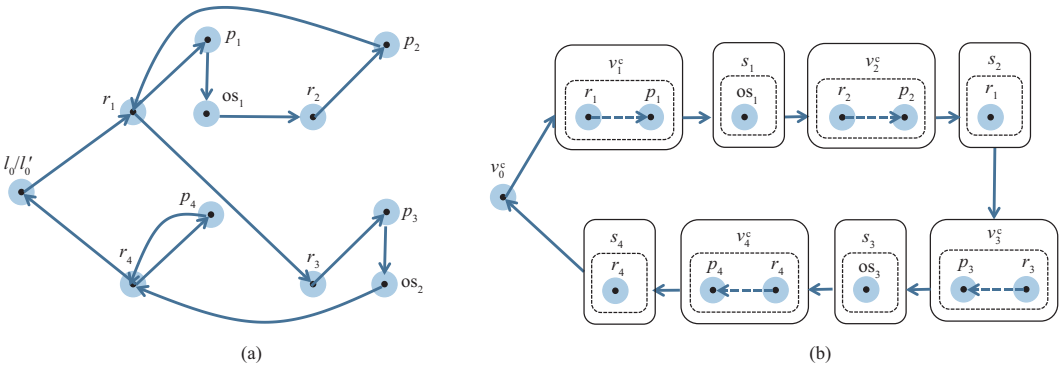

Figure 6 (Color online) Schematic description of Case III. (a) An example in the lower-level optimization; (b) the identified ATSP-PC transformed from (a).

possible selection of other empty retrieval locations. If rack r_i can return to the retrieval location of r_j, then $r_j, \forall j \in \{1, 2, \ldots, i-1\}$ must have been picked, and all racks picked prior to r_i did not return to the retrieval location of r_j. Let $G_c = (\upsilon^c, \varepsilon^c)$ be a graph, where υ^c and ε^c are the same set of nodes and edges in the graph G_a, and the cost matrix D_c is also established according to (40). The ATSP induced by the graph G_c is unambiguous. Let us apply the following constraints to the ATSP induced by G_c.

Definition 1. A feasible ATSP tour $v_0^c \to v_1^c \to v_{m+1}^c \to v_2^c \to v_{m+2}^c \to \cdots \to v_m^c \to v_{2m}^c \to v_0^c$ satisfying the above Conditions 1–3 and additional Condition 4 is an eligible ATSP tour of the graph G_c.

Condition 4. If the storage node represented by $v_i^c, \forall i \in \{m+1, m+2, \ldots, 2m\}$ is the retrieval location of r_j, then v_j^c corresponding to r_j must be visited earlier than v_i^c (we consider v_0^c to be the first node visited).

Because the ATSP-PC is a variant of ATSP, we still use the "first re-identification then LBH" strategy to solve it. Analogous to the ATSP case, we first obtain an assignment scheme in the RTS or STR mode. However, the mutual assignment may occur in the obtained assignment scheme, thus leading to the subtours and an infeasible tour. As shown in Figure 7, there exist assignment pairs $\{(v_{\gamma_i}^c, v_{\chi_i}^c) | i \in [1, 7]\}$ in the obtained scheme of the RTS mode, where $v_{\gamma_1}^c$–$v_{\gamma_7}^c$ represent the storage nodes s_1–s_7, which are selected as the retrieval locations of r_1, r_2, r_5–r_7, os_1, and os_2, respectively. When we apply the LBH to the case with mutual assignment, the feasible solution cannot be constructed because of the violation of Condition 4.

In order to enable the LBH, we present a repair strategy in Algorithm 3 to eliminate these mutual assignments. We find the mutual assignment combinations ma in the infeasible assignment scheme Φ, and the remaining feasible assignment pairs are considered an assignment combination fa. Then, as shown in lines 4–8, we select a pair with the worst assignment cost in fa and each item of ma, respectively. After that, the connection among these pairs will be broken, and each destroyed retrieval node will select a destroyed storage node with the smallest assignment cost in turn (lines 9–17). Note that the pair that has been broken will no longer be reconnected. Finally, these reconnected assignments will be merged with the remaining pairs of ma and fa into a new repaired assignment scheme. The proposed repair strategy can also be applied to the STR mode because the mutual assignments are generated only in pairs between storage and retrieval nodes.

After obtaining the repaired assignment scheme, we extend the LBH-A to generate a tour on G_c. This extension is called LBH-C. Unlike LBH-A, which finds the shortest loop among all edges, it first selects the smallest loop containing v_0^{cr} in ε^c. Specifically, the closest node to v_0^{cr} will be selected to construct the initial smallest loop together with v_0^{cr}. In a tour, the v_0^{cr} should be visited first because it represents the sequence segment pointed from the starting point. Therefore, the operation that adds the v_0^{cr} to the initial loop will clarify the visiting order of the subsequently added nodes.

In addition, an extra checking strategy is introduced in the absorbing process to ensure that precedence constraints are satisfied. For the jth edge s_j^{cl} in a formed loop s^{cl} and an isolated node v_k^{cr}, the key steps are as follows:

(1) Identify the storage node of the sequence segment represented by v_k^{cr}. If the storage node is the retrieval location of r_s, continue to the next step; otherwise, go to step (3).

(2) Check the priority order of visits. In LBH-C, the isolated node that is preferentially absorbed is visited earlier because the v_0^{cr} is firstly added to the initial loop, and each isolated node is sequentially

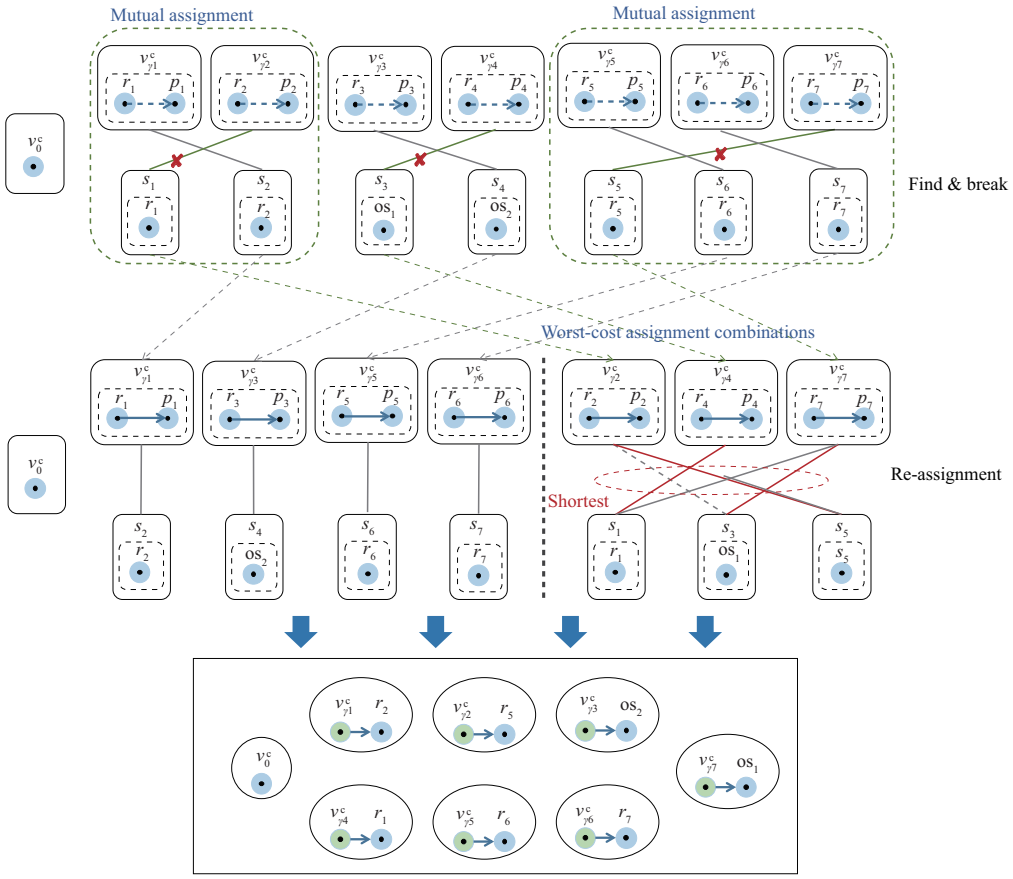

Figure 7 (Color online) The repair strategy of the RTS mode for the ATSP-PC.

Algorithm 3 Repair method for identified ATSP-case

Input: The infeasible assignment scheme Φ, the cost matrix D_c.
Output: The repaired assignment scheme Φ'.

Set $\Phi' \leftarrow \emptyset$, nR $\leftarrow \emptyset$;
Find the mutual assignment combinations ma in Φ, and obtain remaining assignment combination fa $= \Phi \setminus$ ma;
Merge ma and fa into Ma;
for each combination $\text{Ma}_i \in$ Ma **do**
    Rank the pairs in Ma_i according to the increasing assignment cost, and select the pair $(v^c_{\gamma_i}, v^c_{\chi_i})$ with the largest cost;
    Break the assignment connection between this selected pair, and add the destroyed retrieval and storage nodes to nR;
    Add the remaining unselected pairs to Φ';
end for
for each retrieval node $v^c_{\gamma_j}$ in nR **do**
    Set cR $\leftarrow \emptyset$;
    for each storage node $v^c_{\chi_j}$ in nR **do**
        if the assignment pair $(v^c_{\gamma_j}, v^c_{\chi_j}) \notin$ Ma **then**
            Calculate the assignment cost of this pair and add it to cR;
        end if
    end for
    Select the storage node $v^c_{\chi_s}$ with the smallest cost in the cR and delete it from the nR;
    Add the assignment pair $(v^c_{\gamma_j}, v^c_{\chi_s})$ to Φ';
end for
return Φ'.

absorbed into the loop. In other words, if the node representing the sequence segment of r_s has not been added to the loop, v^{cr}_k will not be considered for absorption into the current loop.

(3) Execute (46) to calculate the absorption cost.

This checking strategy is also applied to the 2-opt operator in LBH-C. Only the improved solutions satisfy conditions 1–3 and this strategy will be accepted.

Algorithm 4 VNS for the upper-level optimization

Input: The set of all locations Γ, the locations of m picking stations $\{\zeta_p^1, \zeta_p^2, \ldots, \zeta_p^m\}$.
Output: The best storage location set S_{best}.
1: Set $S_{\text{ini}} \leftarrow \emptyset$, $\Omega \leftarrow \emptyset$, $\Gamma \leftarrow \Gamma'$;
2: Calculate the generalized distance matrix D_g according to (48);
3: **for** each retrieval rack $r_i \in R$ **do**
4: Select the location nearest to the r_i from Γ';
5: Add the selected location to S_{ini} and remove it from Γ';
6: **end for**
7: Set $S \leftarrow S_{\text{ini}}$, $S_{\text{best}} \leftarrow S_{\text{ini}}$, $k \leftarrow 1$;
8: **repeat**
9: $S' \leftarrow \text{Shake}(S, \delta_k, \Gamma)$;
10: $S'' \leftarrow \text{Improve}(S', k, \Gamma)$;
11: **if** $\text{eval}(S'') < \text{eval}(S_{\text{best}})$ **then**
12: $S \leftarrow S''$, $S_{\text{best}} \leftarrow S''$, $k \leftarrow 1$;
13: **else**
14: $k \leftarrow k + 1$;
15: **end if**
16: **until** $k = k_{\max}$;
17: **return** S_{best}.

6 Upper-level optimization

Upper-level optimization aims to optimize the storage assignment to minimize the total travel time to complete all delivery tasks. Meta-heuristics are a class of algorithms that can find high-quality solutions, as they can follow the rules to explore in detail the most promising regions of the solution space. Its applications have increased in number and pertain to many fields [11,45,46] including scheduling, portfolio, and artificial intelligence. In this situation, we design a meta-heuristic algorithm VNS with problem-specific knowledge and effective candidate modes to solve this problem, where m storage locations are selected from Γ.

The framework of our strategy is given in Algorithm 4. It starts from an initial feasible solution, which can be obtained greedily, such as selecting a location $\zeta_j, j = 1, 2, \ldots, m+n$ with the shortest cost for each r_i. In the retrieval sequence, the selection of the storage position of r_i is mainly closely relative to the location of p_i and that of the retrieval rack in the next delivery task. All retrieval racks except r_i may appear in the next delivery task because the sequencing optimization problem is solved in the lower-level optimization. In order to evaluate the selection cost of the storage location more reasonably, we define a generalized distance, which reflects not only the cost of storing the current rack but also the potential cost of executing the next delivery task. The generalized distance matrix D_g is defined as follows:

$$D_g(i,j) = \text{Tp}_{i,j} + \frac{1}{m-1} \sum_{r_k \in R, k \neq i} t(r_k, \zeta_j), \tag{48}$$

where $D_g(i,j)$ is an item of D_g and it represents the generalized distance between r_i and ζ_j. On the right side of the formula, the first item represents the distance between the picking station p_i for r_i picking and the location ζ_j, and the second item measures the average cost of arriving at other retrieval racks from ζ_j.

At each local search, a new perturbation solution S' is generated based on existing solutions S by the shaking stage (line 9), and then the solution S' is further improved to S'' by local search. Each solution is evaluated by $\text{eval}(\cdot)$ in the lower-level optimization. In line 11, if there has been an improvement, the search returns to the first neighborhood. Otherwise, the next shaking stage will be switched. The procedure terminates if the stop condition is reached.

Specifically, in the kth shaking stage $\text{Shake}(S, \delta_k, \Gamma)$, we first randomly select δ_k locations for perturbation. For each perturbed location, we remove an item from the current solution and insert it. It can be seen that there are m possible removal-insertion perturbation schemes. The scheme with the smallest swap cost $\sum_{i=1}^{m} \text{Tp}_{i,s_i^*}$ is used to the perturbation, where $\zeta_{s_i^*}$ indicates the location nearest to p_i. Note that the location that has been added to the solution will not be removed in this shaking stage.

The detail of the kth improving stage is shown in Algorithm 5. Unlike the shaking stage that selects the insert location from all locations, the local search only specifies the location from the promising locations to perform the insert operation. These promising locations, called candidate location sets, consist of the locations nearest to the retrieval racks. More specifically, we find the $2k$ nearest locations to each r_i and then merge these locations to form a candidate set. Our strategy explores a neighborhood in search of

Algorithm 5 Local search strategy Improve (S', k, Γ)

Input: The set of all locations Γ, the generalized distance matrix D_g, the sequence number of the neighborhood k, and the perturbation solution S'.
Output: The obtained solution S''.

```
Set Cand ← ∅;
for each retrieval rack r_i ∈ R do
    Rank the locations in Γ according to the increasing D_g(i, ·);
    Obtain the Cand_i by selecting the top 2k locations from Γ and then Cand ← Cand ∪ Cand_i;
end for
Set S'' ← S', S_cur ← S';
repeat
    for each s_i in S_cur do
        for each Γ_j in Cand/S_cur do
            Set S_new ← S_cur;
            Remove s_i from S_new and add Γ_j to S_new;
            Skip S_new if it has been evaluated in the previous steps;
            if eval(S_new) < eval(S'') then
                S'' ← S_new;
            end if
        end for
    end for
    S_cur ← S'';
until there is no improvement;
return S''.
```

Table 2 Varying factor values for simulation instances

No.	m	n	Instance size
1–25	10	10, 30, 50, 70, 90	Small-sized
51–75	20	20, 60, 100, 140, 180	Medium-sized
26–50	15	15, 45, 75, 105, 135	
76–100	25	25, 75, 125, 175, 225	Large-sized
101–125	30	30, 90, 150, 210, 270	

the best-improving move at each iteration, and it ends until no further improvement is found, and returns the best solution.

7 Computational experiments and analysis

This section describes the extensive experiments on RS-MTMS instances to evaluate the performance of the proposed algorithm. All experiments were implemented by using the MATLAB_R2019a environment on a PC with an Intel Xeon E5 2.60GHz CPU and 64 GB internal memory.

7.1 Instance data

To verify the efficacy of the proposed algorithm, the instance data for the RS-MTMS problem are randomly generated according to [11, 20]. In our scenario, there are 36 zones divided by five aisles and five cross aisles. On the northernmost row and the southernmost row, each zone has five racks. For the 24 zones on the middle part, each zone has 10 racks. The four picking stations are located in the middle of the four directions of east, west, north, and south. The robot starts and ends its operations located at the lower left part of the scenario, and delivers racks to picking stations according to the preset visit assignments. The robot speed is set to 1.3. Given the number of retrieval racks m and open storage locations n, 25 sets of varying sizes are generated in Table 2 for performance validation and comparison. Each instance set contains five different instances, and a total of 125 random instances consist of 25 small-scale instances, 50 medium-scale instances, and 50 large-scale instances. For each instance, n is 1, 3, 5, 7, and 9 times m, and all locations are divided into multiple zones. The locations of retrieval racks and open storage locations are randomly generated using a uniform distribution.

7.2 Comparison algorithms and parameter setting

(1) Adaptive large neighborhood search (ALNS) algorithm. The adaptive large neighborhood search proposed by Gharehgozli et al. [11] represents the state-of-the-art for the simplified RS-MTMS problem,

where the rack can only be stored in the open storage locations after it visits a picking station. If the storage assignment set belongs only to Os, it can find a retrieval sequence which minimizes the total travel time to complete all delivery tasks. As the optimality of the solution in the special case can be guaranteed, this algorithm is employed in our computational experiments. We extend the types of storage locations, and introduce Condition 4 into the construction of the initial solution and removal and insertion operators to avoid generating infeasible solutions.

(2) Discrete water cycle algorithm (DWCA). The discrete water cycle algorithm is a promising approximation algorithm [43] used for solving the family of TSP and ATSP. Our RS-MTMS problem can be transformed as the sequence decision problem, which is a particular variant of TSP or ATSP, because the storage assignment problem can be reflected in the encoding of the solution. The following adjustment should be applied to the original algorithm to suit our situation.

Firstly, we encode the solution as $x = \{x_1, x_2, \ldots, x_{2m+1}\}$, where x_1 is set to 0 (representing the starting/ending point) and $x_{2i} \in [1, m]$ and $x_{2i+1} \in [1, m+n]$ for $i = 1, 2, \ldots, m$. This encoding mode not only includes the selection information of the storage location but also satisfies Conditions 2 and 3. To satisfy Conditions 1 and 4 when constructing the initial solutions, we generate a random permutation of 1 to m as the retrieval sequence of the racks. Then, for each x_{2i+1} generation, we select an item from $[1, m+n]$ (satisfies priority constraints if it belongs to $[1, m]$) without duplication. During the update of the solution, we apply our identification strategy to each individual before using the movement and insertion operators. If this individual is identified as TSP or ATSP, the original operators are used directly to obtain the improved solution; otherwise, Condition 4 will be incorporated into these operators to obtain the high-quality feasible solution.

(3) Practice heuristics algorithm (PHA). To solve the storage assignment and retrieval sequence decision problems, several decision rules [10] are used in a realistic scenario such as nearest neighborhood, farthest neighborhood, and greedy-random. In the nearest neighborhood heuristic, the robot travels to the nearest retrieval rack, and each rack returns to the nearest storage location after it visits a picking station. The farthest neighborhood heuristic is the same as the nearest neighborhood with the difference that the item with the farthest distance is always selected. The greedy-random heuristic each time selects a retrieval rack or storage location from the list. In each selection, the greedy strategy is applied with probability ρ. This algorithm selects the best results among the three heuristics as the final solution.

To achieve a good trade-off between the solution quality and the runtime of the proposed BiJSR, we set the maximum number of neighborhoods $k_{\max}$ to 3, and the list of the number of perturbation locations δ to $[0.1n, 0.2n, 0.5n]$. Because the decision schemes of BiJSR and its three comparison algorithms may be unstable owing to the randomness of the solution exploration mechanism, the four algorithms will run 20 times for each instance. To ensure that two meta-heuristic comparison algorithms can fully discover a better solution, their iterations are set large enough, and they will terminate when they run out of the allowable time. For a fair comparison, at least $10000 \times (m+n)$ greedy-random heuristic in PHA is executed for each instance to ensure that it can fully find a higher-quality solution. The value of ρ is used iteratively over the range [0, 0.9] to increase by 0.1 in each run. The greedy strategy is randomly selected as the nearest or farthest. For each instance, the allowable runtime for any algorithm is set to 1800 s, and that of the Gurobi is 3600 s.

7.3 Experimental results and analysis

In this subsection, we analyze the running results of the experiments from the perspectives of solution quality and running time. Some data are highlighted in bold to highlight the best computational results obtained by all algorithms. In addition, several practical extensions of RS-MTMS are discussed.

7.3.1 *Preliminary experiment*

In the preliminary experiment, we test the performance of the transformation heuristic in the lower-level optimization of BiJSR. To analyze the solution quality and runtime on the identified ATSP or ATSP-PC, m storage locations are preselected from Os or $\text{Os} \cup R$ and at least one storage location should be selected from the retrieval location for the ATSP-PC. We randomly generate 20 cases in each instance with different storage locations for the identified ATSP or ATSP-PC. For the identified TSP, there exists one case in each instance because all storage locations are selected as retrieval locations.

Table 3 displays the comparison results between our lower-level algorithm and Gurobi solver for all instances. The positive or zero percentage deviation indicates that our algorithm has an optimality gap or

Table 3 Comparison results with respect to Gurobi for the lower-level optimization problem

		Ave (%)	Best (%)	Worst (%)	Count (%)	T-opt (s)	T-ours (s)
LBH		**0.00**	**0.00**	0.06	97.60	17.92	**0.94**
LBH-A	RTS	0.28	0.11	1.56	57.44	1.25	**0.03**
	STR	**0.01**	**0.00**	**0.33**	**97.52**		**0.03**
LBH-C	RTS	−4.12	−9.78	−1.84	96.40	3232.64	**0.09**
	STR	**−5.96**	**−12.35**	**−3.15**	**100.00**		0.10

reaches the optimal solution, and the negative deviation indicates that our algorithm can explore a better result than the solver within the time limit imposed. T-opt and T-ours represent the runtime of the solver and our algorithms. For the identified TSP problem, Ave (%), Best (%), and Worst (%) respectively denote the average, best, and worst deviation from the optimal solution on $\{25 \times 1 \times 5\}$ cases. Count (%) is the ratio of the number of cases reaching the optimal solution divided by the total number of cases. As shown in Table 3, we find that our LBH can discover the optimal result with an average of 19.01 times less time cost on 122 instances. For the identified ATSP and ATSP-PC problems, the number of cases is $\{25 \times 20 \times 5\}$. In terms of the solution quality, the performance of LBH-A using STR mode is better than that of RTS mode, especially in medium- and large-sized cases. Both re-identification modes have the same average running time, i.e., about 41.73 times less than Gurobi.

The Count (%) of LBH-C represents the ratio of cases whose deviation is zero or a negative value. We can see that both modes have similar runtimes, while the solution quality of STR has a significant advantage over RTS. The STR can find the same or better results in all cases, and the best solution average obtained is 12.35% better than Gurobi. However, there is a deviation in some instances and the best deviation is 9.78% for the RTS. More detail comparison results are shown in Appendix B. In summary, our proposed lower-level algorithms have excellent performance, and the STR mode is more effective than RTS in both LBH-A and LBH-C.

7.3.2 *Comparison with Gurobi solver*

Table 4 displays the comparison results between our proposed BiJSR and Gurobi solver on small-, medium-, and large-sized instances, respectively. In Table 4, UB is the upper bound as reported by the Gurobi solver for (1)–(12), and LB is a larger value between the lower bound reported by the Gurobi solver and that solved by our proposed relaxation formulation. The LB values are shown as UB if the optimal problem is found. If the relaxation model cannot directly obtain the optimal solution within the time limit, its lower bound will be used as LB. Ave-U and Ave-L denote the average deviation from the upper and lower bound, respectively, where the negative value indicates that the solution obtained by our method is better than the solver within the time limit.

From this table, we find that BiJSR can discover the optimal solutions for all small-sized instances and better results for all medium- and large-sized instances. Specifically, the objective values obtained by our algorithm average 6.08% for medium-sized instances and 13.88% for large-sized instances, which is better than that obtained by the Gurobi solver within its time limit. In terms of the runtime performance, BiJSR can solve the RS-MTMS problem much more efficiently, especially in larger-sized instances. It can be seen that the computation time of BiJSR averaged about 7.14, 20.16, and 6.29 times less than that of Gurobi for small-, medium-, and large-sized instances. Moreover, the average Ave-L is about 16.68% for large-sized instances. The result implies that the average gap of BiJSR to the optimal solution is moderate, which is no more than 16.68%.

7.3.3 *Comparison with other algorithms*

From Table 4, the proposed BiJSR performs significantly better than comparison algorithms in most instances, and the advantage is further expanded when the size of the instance is larger. Specifically, the objective values obtained by BiJSR are on average 10.58%, 1.33%, and 9.31% for small-sized, 10.26%, 6.31%, and 12.58% for medium-sized, and 12.98%, 7.17%, and 11.56% for large-sized instances better than that of PHA, ALNS, and DWCA, separately. To prove the stability of each algorithm's solution results, we used the Wilcoxon rank-sum test to evaluate the statistical performance. The score in Table 4 is calculated as the total result of each algorithm compared with other algorithms. It is the sum of the rank-sum test results on all runs for every five instances with the same size. In terms of runtime, ALNS and DWCA will terminate when they run out of the allowable 1800 s, and the computation time of PHA

Table 4 Comparison results of the Gurobi solver, the proposed BiJSR, and three comparison algorithms (The time limitation is set to 3600 s for Gurobi solver and 1800 s for the rest algorithms)

No.	Gurobi solver			BiJSR			PHA			ALNS			DWCA		
	UB	LB	T (s)	Ave-U (%)	Ave-L (%)	T (s)	Ave-U (%)	Ave-L (%)	T (s)	Ave-U (%)	Ave-L (%)	T (s)	Ave-U (%)	Ave-L (%)	T (s)
1–5	333.38	UB	150.73	**0.00**	**0.00**	**13.27**	7.29	7.29	301.52	2.46	2.46	1800.01	4.71	4.71	1800.01
6–10	321.20	UB	100.91	**0.00**	**0.00**	**26.87**	7.88	7.88	601.71	1.78	1.78	1800.01	9.64	9.64	1800.01
11–15	332.03	UB	221.83	**0.00**	**0.00**	**33.06**	13.65	13.65	901.97	0.65	0.65	1800.00	12.18	12.18	1800.00
16–20	338.80	UB	246.22	**0.00**	**0.00**	**28.78**	11.54	11.54	1200.62	1.02	1.02	1800.01	11.59	11.59	1800.01
21–25	344.22	UB	141.72	**0.00**	**0.00**	**26.77**	12.52	12.52	1534.47	0.75	0.75	1800.01	8.42	8.42	1800.00
26–30	494.83	487.38	3600.00	**−0.49**	**1.00**	**100.49**	5.40	6.97	1800.14	2.04	3.56	1800.00	7.16	8.77	1800.01
31–35	490.09	480.62	3600.00	**−0.30**	**1.68**	**163.95**	9.85	12.03	1800.15	4.28	6.33	1800.01	10.92	13.11	1800.01
36–40	480.78	471.31	3600.00	**−0.12**	**1.92**	**149.80**	8.82	11.04	1800.14	2.35	4.44	1800.01	8.18	10.39	1800.01
41–45	502.62	493.66	3600.00	**−0.54**	**1.22**	**149.98**	4.54	6.39	1800.12	3.15	4.96	1800.00	9.91	11.81	1800.01
46–50	514.63	506.51	3600.00	**−0.08**	**1.54**	**154.99**	4.80	6.49	1800.12	3.00	4.67	1800.05	8.28	10.02	1800.01
51–55	648.15	524.62	3600.00	**−14.66**	**5.99**	**208.80**	1.27	27.06	1800.09	−4.62	19.30	1800.00	2.28	27.95	1800.03
56–60	666.43	539.68	3600.00	**−14.23**	**5.48**	**266.77**	−3.57	18.73	1800.06	−4.33	17.90	1800.00	−3.70	18.90	1800.02
61–65	683.35	556.09	3600.00	**−13.96**	**4.81**	**236.75**	−1.51	20.09	1800.44	−5.70	14.90	1800.21	2.56	24.95	1800.42
66–70	668.12	572.34	3600.00	**−9.33**	**6.17**	**254.90**	−0.89	16.43	1800.68	−4.68	11.80	1800.46	0.31	17.48	1800.46
71–75	663.05	587.74	3600.00	**−7.05**	**4.72**	**287.33**	3.07	16.36	1800.78	−0.24	12.60	1800.31	7.95	22.05	1800.25
76–80	864.77	707.72	3600.00	**−11.88**	**7.60**	**407.37**	−0.33	21.83	1800.14	−4.41	16.79	1800.01	−1.91	19.81	1800.04
81–85	865.11	693.00	3600.00	**−12.11**	**10.17**	**442.58**	−0.81	24.44	1800.08	-4.97	19.56	1800.62	2.10	27.79	1800.04
86–90	820.77	655.77	3600.00	**−7.10**	**16.68**	**411.27**	2.24	28.31	1800.07	−2.33	22.62	1800.42	6.42	33.86	1800.04
91–95	857.66	678.62	3600.00	**−12.44**	**10.31**	**499.68**	−2.22	23.19	1800.26	−4.44	20.46	1800.24	−8.15	15.79	1800.37
96–100	849.20	686.57	3600.00	**−7.61**	**14.35**	**512.98**	2.29	26.64	1800.90	−1.49	21.91	1800.55	2.65	27.13	1800.62
101–105	1195.78	826.86	3600.00	**−19.73**	**15.79**	**606.47**	−8.67	31.74	1800.09	−14.57	23.18	1800.27	−8.80	31.49	1800.05
106–110	1175.14	856.98	3600.00	**−17.08**	**13.83**	**713.90**	−4.87	30.57	1800.09	−11.40	21.65	1800.61	−6.72	28.05	1800.07
111–115	1159.23	810.45	3600.00	**−19.75**	**14.16**	**876.25**	−7.32	31.76	1800.08	−14.72	21.27	1800.32	−11.45	26.12	1800.10
116–120	1088.83	803.85	3600.00	**−14.02**	**16.49**	**886.35**	−2.47	32.13	1800.09	−6.98	26.03	1800.11	−4.77	29.06	1800.07
121–125	1131.14	819.75	3600.00	**−17.06**	**14.32**	**935.52**	−5.61	30.11	1800.09	−12.02	21.17	1800.59	−8.68	25.76	1800.08
Score		–			**370**			66			224			71	

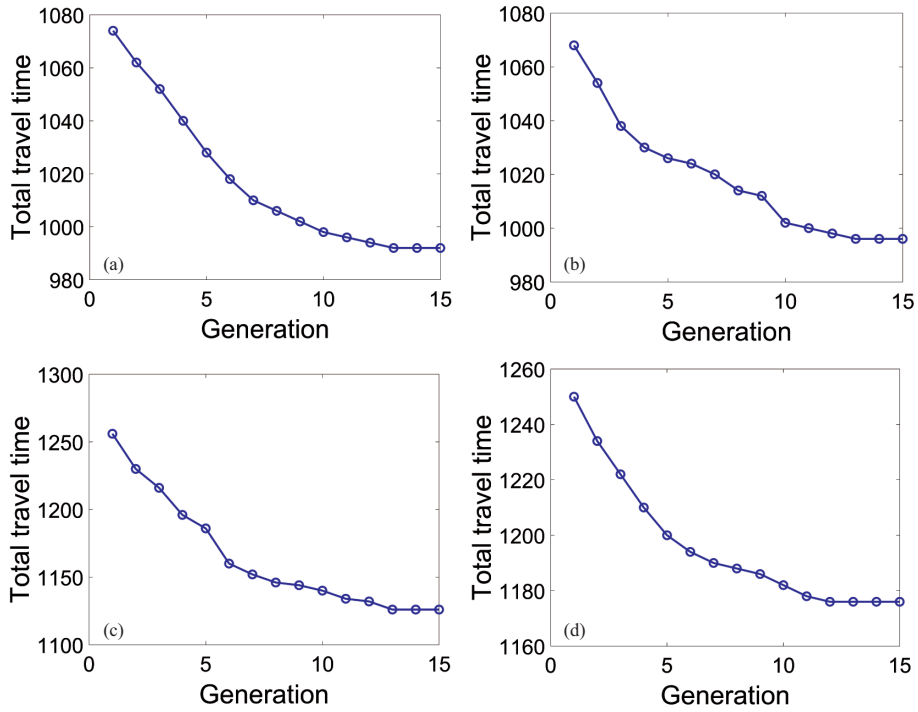

Figure 8 (Color online) Convergence curve of the BiJSR for four random large-sized instances. (a) $m = 25$, $n = 25$; (b) $m = 25$, $n = 225$; (c) $m = 30$, $n = 30$; (d) $m = 30$, $n = 270$.

is much more than that of BiJSR for the same instance. Specifically, the average computation time of BiJSR for all instances is about 12.15 times less than that of PHA and 20.77 times less than that of ALNS and DWCA. More detailed comparison results are shown in Appendix C.

The competitiveness of the proposed BiJSR is due to the transformation and loop-based strategies obtained by the lower-level optimization. Table 3 shows that they can efficiently construct an optimal sequence in most cases. On the other hand, the proposed generalized distance and candidate mode can more efficiently obtain the storage location of the retrieval racks, which are very helpful for the neighborhood search during the search process of BiJSR to find high-quality solutions. This fact can be observed from Figure 8, which shows the convergence curve of BiJSR on four random large-sized instances (2 each for $m = 25$ and $m = 30$).

7.4 Practical extensions of the problem

7.4.1 *Impact of multiple types of storage locations*

Table 5 shows how the allowed types of storage locations affect the total travel time of the robot. We consider the instances with $m = 10$ because the optimal solution can be obtained from them. Compared with only considering open storage locations for the rack storage, the multiple types of storage locations in our case experience shorter travel time because their own or empty retrieval locations are closer to the visited picking stations and racks to be retrieved. Although some potentially empty retrieval locations can be replaced by adjacent open storage locations, as the number of open storage locations increases, the case of multiple types of storage locations still have better performance. The results in Table 5 show that the total travel time of our RS-MTMS case is on average about 2.42% better than that of single types of storage locations.

7.4.2 *Impact of multiple robots*

In this subsection, we show the impact of multiple robots completing delivery tasks. The experiment is carried out on the five large-sized instances with 25 cases (101–125 in Table 2). The number of robots is increased from 1 to 5. All tasks are randomly assigned to each robot, with 15 tasks per robot for 2 robots, 10 for 3 robots, 7 or 8 for 4 robots, and 6 for 5 robots. To improve the order picking efficiency, all robots simultaneously and independently complete assigned delivery tasks. Figure 9 shows the average total

Table 5 Comparison results between the single type and multiple types of storage locations

Ins.	Multitype		Single type	
	Gurobi	BiJSR	Gurobi	BiJSR
$m = 10, n = 10$	**333.38**	**333.38**	359.45	359.45
$m = 10, n = 30$	**321.20**	**321.20**	328.31	328.31
$m = 10, n = 50$	**332.03**	**332.03**	336.09	336.09
$m = 10, n = 70$	**338.80**	**338.80**	341.17	341.17
$m = 10, n = 90$	**344.22**	**344.22**	346.92	346.92

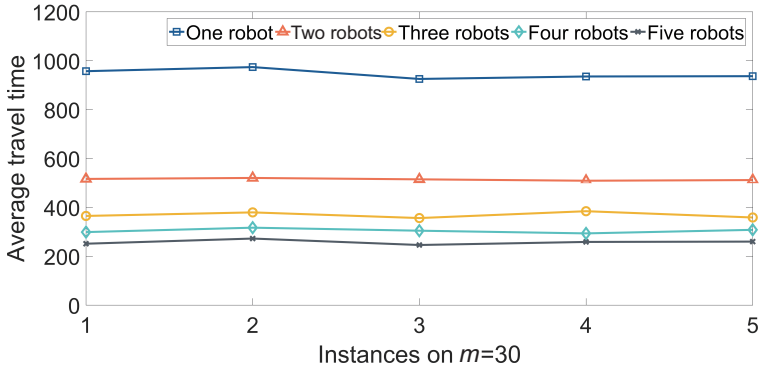

Figure 9 (Color online) Impact of multiple robots on travel time.

travel time of each robot for different numbers of robots. It can be seen that the multi-robot scenario has more time savings as the number of robots increases because each robot needs to complete fewer delivery tasks. However, the time saving is lower in the real world. The conflicts and deadlocks in the travel path of robots will increase the travel time to complete the delivery tasks. There is a need for developing a joint optimization of multi-robot scheduling and multi-robot path planning to meet the real-world needs.

8 Conclusion

In this paper, we focus on the RS-MTMS problem, which jointly decides the retrieval sequence of racks and assigns each rack a storage location after it visits a picking station. We present a bi-level optimization model BiJSR. In the upper-level optimization, a VNS strategy with an effective candidate mode is designed to address the storage assignment problem. The retrieval sequence optimization is solved by the transformation strategy and LBH with sufficient problem-specific knowledge in the lower-level optimization. The proposed BiJSR can obtain high-quality solutions using less computational time to solve the RS-MTMS problem. The computational results show that our proposed algorithm performs 13.88% better than the solver within the time limit and 7.17% than the most effective comparison algorithm on the large-sized instances. Although the efficacy of the proposed BiJSR has been verified, there remain some investigations that deserve further study in the future.

(1) As a key component of the upper-level optimization, the VNS combined with candidate mode has been shown in this paper to improve the quality of the storage assignment scheme. It is also interesting to design more efficient and effective neighborhood operators or consider other excellent meta-heuristic solution algorithms to further improve the solution.

(2) Apply the different forms of BiJSR to other related scenarios, such as the multi-tier shuttle, container terminals, and material transportation, which will be very helpful for solving more problems in social systems.

(3) For practicality, the proposed algorithm will be extended to more challenging multi-robot systems where it is necessary to consider further task assignment and path planning for each robot. In addition, we will alleviate the assumptions of the proposed scenario to verify the performance of algorithms with real-world problems.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant No. 61933002), National Science Fund for Distinguished Young Scholars (Grant No. 62025301), and National Natural Science Foundation of China Basic Science Center Program (Grant No. 62088101).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Boysen N, de Koster R, Weidinger F. Warehousing in the e-commerce era: a survey. Eur J Oper Res, 2019, 277: 396–411
2 Azadeh K, de Koster R, Roy D. Robotized and automated warehouse systems: review and recent developments. Transpation Sci, 2019, 53: 917–945
3 da Costa Barros Í R, Nascimento T P. Robotic mobile fulfillment systems: a survey on recent developments and research opportunities. Robot Autonom Syst, 2021, 137: 103729
4 Rimélé A, Gamache M, Gendreau M, et al. Robotic mobile fulfillment systems: a mathematical modelling framework for e-commerce applications. Int J Production Res, 2022, 60: 3589–3605
5 Shi X, Deng F, Fan Y, et al. A two-stage hybrid heuristic algorithm for simultaneous order and rack assignment problems. IEEE Trans Automat Sci Eng, 2022, 19: 2955–2967
6 Kim H J, Pais C, Shen Z J M. Item assignment problem in a robotic mobile fulfillment system. IEEE Trans Automat Sci Eng, 2020, 17: 1854–1867
7 Valle C A, Beasley J E. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. Comput Oper Res, 2021, 125: 105090
8 Zou B, Gong Y Y, Xu X, et al. Assignment rules in robotic mobile fulfilment systems for online retailers. Int J Production Res, 2017, 55: 6175–6192
9 Yue L, Fan H. Dynamic scheduling and path planning of automated guided vehicles in automatic container terminal. IEEE CAA J Autom Sin, 2022, 9: 2005–2019
10 Merschformann M, Lamballais T, de Koster M B M, et al. Decision rules for robotic mobile fulfillment systems. Oper Res Perspect, 2019, 6: 100128
11 Gharehgozli A, Zaerpour N. Robot scheduling for pod retrieval in a robotic mobile fulfillment system. Transpation Res Part E-Logistics Transpation Rev, 2020, 142: 102087
12 Weidinger F, Boysen N, Briskorn D. Storage assignment with rack-moving mobile robots in KIVA warehouses. Transpation Sci, 2018, 52: 1479–1495
13 Boysen N, Briskorn D, Emde S. Parts-to-picker based order processing in a rack-moving mobile robots environment. Eur J Oper Res, 2017, 262: 550–562
14 Yang X, Hua G, Hu L, et al. Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system. Comput Oper Res, 2021, 135: 105467
15 Xie J, Mei Y, Ernst A T, et al. A bi-level optimization model for grouping constrained storage location assignment problems. IEEE Trans Cybern, 2016, 48: 385–398
16 Hanson R, Medbo L, Johansson M I. Performance characteristics of robotic mobile fulfilment systems in order picking applications. IFAC-PapersOnLine, 2018, 51: 1493–1498
17 de Koster R, Le-Duc T, Roodbergen K J. Design and control of warehouse order picking: a literature review. Eur J Oper Res, 2007, 182: 481–501
18 Martinez-Carranza J, Rojas-Perez L O. Warehouse inspection with an autonomous micro air vehicle. Unmanned Sys, 2022, 10: 329–342
19 Chu W J, Zhang W, Zhao H Y, et al. Massive self-organized shape formation in grid environments. Sci China Inf Sci, 2022, 65: 164101
20 Lamballais T, Roy D, de Koster M B M. Estimating performance in a robotic mobile fulfillment system. Eur J Oper Res, 2017, 256: 976–990
21 Duan G, Zhang C, Gonzalez P, et al. Performance evaluation for robotic mobile fulfillment systems with time-varying arrivals. Comput Industrial Eng, 2021, 158: 107365
22 Jaghbeer Y, Hanson R, Johansson M I. Automated order picking systems and the links between design and performance: a systematic literature review. Int J Production Res, 2020, 58: 4489–4505
23 Lienert T, Staab T, Ludwig C F, et al. Simulation-based performance analysis in robotic mobile fulfilment systems. In: Proceedings of the 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, 2018
24 Bozer Y A, Aldarondo F J. A simulation-based comparison of two goods-to-person order picking systems in an online retail setting. Int J Production Res, 2018, 56: 3838–3858
25 Yang X, Liu X, Feng L, et al. Non-traditional layout design for robotic mobile fulfillment system with multiple workstations. Algorithms, 2021, 14: 203
26 Yuan R, Cezik T, Graves S C. Stowage decisions in multi-zone storage systems. Int J Production Res, 2018, 56: 333–343
27 Gong Y, Jin M, Yuan Z. Robotic mobile fulfilment systems considering customer classes. Int J Production Res, 2021, 59: 5032–5049
28 Zhuang Y, Zhou Y, Yuan Y, et al. Order picking optimization with rack-moving mobile robots and multiple workstations. Eur J Opera Res, 2022, 300: 527–544
29 Xie L, Thieme N, Krenzler R, et al. Introducing split orders and optimizing operational policies in robotic mobile fulfillment systems. Eur J Opera Res, 2021, 288: 80–97
30 Shahriari M, Biglarbegian M. A new conflict resolution method for multiple mobile robots in cluttered environments with motion-liveness. IEEE Trans Cybern, 2018, 48: 300–311
31 Cai J, Li X, Liang Y, et al. Collaborative optimization of storage location assignment and path planning in robotic mobile fulfillment systems. Sustainability, 2021, 13: 5644
32 Li X P, Pan D Y, Wang Y D, et al. Scheduling multi-tenant cloud workflow tasks with resource reliability. Sci China Inf Sci, 2022, 65: 192106
33 Cao Z C, Lin C R, Zhou M C, et al. Scheduling semiconductor testing facility by using Cuckoo search algorithm with reinforcement learning and surrogate modeling. IEEE Trans Automat Sci Eng, 2018, 16: 825–837
34 Li S L, Zhai D S, Du P F, et al. Energy-efficient task offloading, load balancing, and resource allocation in mobile edge computing enabled IoT networks. Sci China Inf Sci, 2019, 62: 029307

35 Yuan H, Zhou M C, Liu Q, et al. Fine-grained and arbitrary task scheduling for heterogeneous applications in distributed green clouds. IEEE CAA J Autom Sin, 2020, 7: 1380–1393
36 Merschformann M. Active repositioning of storage units in robotic mobile fulfillment systems. In: Proceedings of Operations Research Proceedings 2017, 2018. 379–385
37 Yuan R, Graves S C, Cezik T. Velocity-based storage assignment in semi-automated storage systems. Prod Oper Manag, 2019, 28: 354–373
38 Ji T, Zhang K, Dong Y. Model-based optimization of pod point matching decision in robotic mobile fulfillment system. In: Proceedings of IEEE 7th International Conference on Industrial Engineering and Applications (ICIEA), 2020. 216–223
39 Ouzidan A, Pardo E G, Sevaux M, et al. BVNS approach for the order processing in parallel picking workstations. In: Proceedings of the 8th International Conference on Variable Neighborhood Search, 2021. 176–190
40 Wang Z, Sheu J-B, Teo C-P, et al. Robot scheduling for mobile-rack warehouses: human-robot coordinated order picking systems. Production Oper Manag, 2022, 31: 98–116
41 Yuan W, Sun H. A task scheduling problem in mobile robot fulfillment systems. In: Proceedings of the 12th International Conference on Advanced Computational Intelligence (ICACI), 2020. 391–396
42 Lu S, Xin B, Zhang H, et al. Agent-based self-organized constructive heuristics for travelling salesman problem. In: Proceedings of the 59th IEEE Conference on Decision and Control (CDC), 2020. 1164–1169
43 Osaba E, Ser J D, Sadollah A, et al. A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl Soft Computing, 2018, 71: 277–290
44 Kuhn H W. The Hungarian method for the assignment problem. Naval Res Logistics, 1955, 2: 83–97
45 Zhou Z, Liu Z T, Su H Y, et al. Multi-objective optimization for 10-kW rated power dynamic wireless charging systems of electric vehicles. Sci China Inf Sci, 2022, 65: 202201
46 Chen Y X, Shi Z K, Xu B, et al. Optimal design of a scaled-up PRO system using swarm intelligence approach. Sci China Inf Sci, 2021, 64: 222203