IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 August 2022, accepted 10 October 2022, date of publication 19 October 2022, date of current version 26 October 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3215700*

RESEARCH ARTICLE

# A Selective Many-to-Many Pickup and Delivery Problem With Handling Cost in the Omni-Channel Last-Mile Delivery

**YALI LI**
School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
e-mail: lyl0829@hust.edu.cn

**ABSTRACT** This paper introduces a selective many-to-many pickup and delivery problem with handling cost (SMMPDPH) arising in the omni-channel last-mile delivery. In SMMPDPH, each request is associated with multiple pickup and delivery nodes, which provide or require commodities. A request is to load commodities from the pickup nodes and transport them to the delivery nodes. Since the total supply is greater than the total demand, we need to determine the selected subset of pickup nodes to visit for each request. Based on the loading and unloading policy, we take into account the handling cost incurred by additional operations. SMMPDPH aims to obtain a routing plan with the minimum total cost, including the travel cost and handling cost, for a fleet of homogeneous vehicles with limited capacity and driving mileage to satisfy the requests. We present two mixed integer programming formulations of the problem and propose two algorithms, including an iterated local search (ILS) and a memetic algorithm (MA), with specially designed move operators to solve the problem. Experiments on small instances clearly indicate the effectiveness of the two heuristic methods. With a comparison of efficiency on large-scale instances, we find that MA outperforms ILS in terms of both the best and average solution quality.

**INDEX TERMS** Vehicle routing problem, pickup and delivery, handling cost, mixed integer programming, heuristics.

## I. INTRODUCTION

This research is partially motivated by a new last-mile delivery strategy, the Buy-Online-and-Deliver-from-Store (BODS) mode, in the household appliance industry. With the rapid development of e-commerce, more and more household appliance manufacturers and retailers have developed an online channel. In the past, the online and offline channels have been operated independently. In recent years, the physical stores have suffered a drop in the number of customers and the sales revenue due to the impact of COVID-19, whereas the online sales improve significantly. This accelerates the integration of available channels in household appliance industry so as to improve the last-mile delivery efficiency and customer service level. BODS, under which the commodities of online orders are picked up from the physical stores and then delivered to the customers, is a commonly adopted strategy by manufacturers and retailers engaged in household appliances [1]. For example, Midea Group, a Chinese home appliance giant, makes attempts to improve the online order fulfillment service by using its offline stores for the last-mile delivery.

With BODS, an online order could be fulfilled by loading commodities from multiple pickup locations, including the distribution centers and physical stores of the corresponding manufacturer or retailer. This makes the vehicle routing design of the last-mile delivery more complicated, since the logistics service provider should not only determine the visiting order of locations but also select the pickup locations from which the commodities are loaded.

In practice, the vehicles used for the transportation of large household appliances are usually rear-loaded with a single stack operating in a last-in-first-out (LIFO) fashion. The LIFO policy means that only the last loaded commodity is accessible for delivery. In this case, the pickup commodities

The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh.

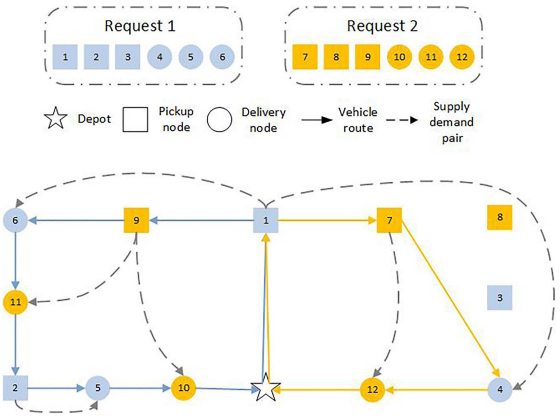



**FIGURE 1.** A schematic example of the investigated problem.

may obstruct the delivery commodities. To be more specific, at a delivery location, if the commodity to deliver is the one picked up last, it could be obtained without handling operations on other commodities; otherwise, all the commodities in the vehicle loaded after it would be unloaded before the delivery and reloaded afterwards due to the obstruction. The unloading and reloading operations are called additional operations according to Battarra et al. [2]. Since the rearrangement of large household appliances takes a lot of effort [3], we associate a handling cost with each additional operation on these commodities as a penalty.

To the best of our knowledge, the pickup location selection and the handling cost incurred by additional operations have not been considered simultaneously. This paper addresses this research gap in the literature by introducing the selective many-to-many pickup and delivery problem with handling cost (SMMPDPH). In SMMPDPH, each request is associated with a set of pickup nodes and delivery nodes. An unlimited fleet of homogeneous vehicles with limited capacity and driving mileage is available to operate the routes. The SMMPDPH aims to make a routing plan for the vehicles by selecting a pickup node for each delivery node and designing the node sequences in order to satisfy the requests with a minimum total of travel cost and handling cost. Fig. 1 provides a schematic example of the problem. The combined decisions about the selection of pickup nodes and the routing of vehicles as well as the trade-off between travel cost and handling cost make the problem interesting and challenging.

The remainder of the paper is organized as follows. In Section II, the related work is presented. Section III gives the problem formulations. Section IV includes the heuristic methods for SMMPDPH. Computational results are reported in Section V. Section VI is the conclusion.

## II. LITERATURE REVIEW

The vehicle routing problem has been extensively studied since it is introduced by Dantzig and Ramser [4]. As a variant of VRP, pickup and delivery problem (PDP) refers to a problem in which a group of transportation requests needs to be satisfied by a set of vehicle routes while each request specifies the origins and destinations of the commodity being transported [5]. PDPs have got much attention among researchers. Lokin [6] is the first to introduce the precedence constraint. Savelsbergh and Sol [5] discuss the characteristics of PDPs, and provide an overview of the problem types as well as solution methods. Mitrovic-Minic [7] conducts an early survey on the PDPs with time windows. Recent reviews on PDPs could be found in [8], [9], [10], [11], [12], and [13].

According to Berbeglia et al. [8], based on the number of origins and destinations of the commodity, PDPs could be classified into three categories, including one-to-one (1-1), one-to-many-to-one (1-M-1) and many-to-many (M-M). In the 1-1 PDPs, the commodity is loaded from a specific pickup location and transported to a specific delivery location; in 1-M-1 PDPs, the commodity at the depot is delivered to the locations and the commodity at the locations is picked up to be transported to the depot; in M-M PDPs, a commodity could be picked up from more than one location and delivered to any location that demands it. In SMMPDPH, each request consists of multiple pickup and delivery locations, and the commodity loaded from a pickup location could be transported to any delivery location of the request. Therefore, it is a M-M PDP.

Hernández-Pérez and Salazar-González [14] introduce the one-commodity M-M pickup and delivery traveling salesman problem, in which each location has a known supply or demand quantity for the commodity and the commodity could be collected from or delivered to multiple locations. The problem aims to minimize the travel distance of the vehicle with limited capacity to satisfy the pickup and delivery demand by visiting each location exactly once. Two heuristic methods are proposed to solve the problem. Hernández-Pérez and Salazar-González [15] propose new inequalities for the problem. Hernández-Pérez and Salazar-González [16], Hernández-Pérez et al. [17], Lu et al. [18] address an extension of the problem by considering multi-commodity. Hernández-Pérez and Salazar-González [19] introduce the one-commodity M-M pickup and delivery traveling salesman problem with split demand and propose a branch-and-cut algorithm. The M-M PDPs have many applications in the bike sharing system and the inventory reallocation context. Dell'Amico et al. [20] investigate the bike sharing rebalancing problem and formulate it as a one-commodity M-M PDP. They propose mixed integer linear programming models and branch-and-cut algorithms for the problem. Dell'Amico et al. [21] develop a destroy and repair algorithm for the one-commodity M-M PDP. Dell'Amico et al. [22] study a variant of the problem by considering stochastic demands and design five exact algorithms as well as a heuristic method based on variable neighborhood decent to solve it. Li et al. [23] study an extension of the one-commodity M-M pickup and delivery traveling salesman problem by considering multiple vehicles faced by a fast fashion retailer for inventory relocation. The problem could also be used to formulate the bike sharing rebalancing problem studied by Lu et al. [24].

In some other M-M PDPs, the pickup or delivery quantities at some locations are not known beforehand and not all the locations would be visited, since the total supply quantity is not equal to the total demand quantity. In this paper, such problems are referred to as selective M-M PDPs so as to be distinguished from the above M-M PDPs. Ho and Szeto [25] study the static bike sharing repositioning problem, which determines the subset of locations to visit and the quantities of bikes to load and unload at the locations. An iterated tabu search method is designed for the problem. Li et al. [26] study an extension of the problem by considering multiple types of bikes and propose a hybrid genetic algorithm to solve the problem. Chen et al. [27] study the inventory reallocation problem in the fashion retailer chains, which is an extension of the problem in Li et al. [26] by considering split loads, and present a variable neighborhood search for the problem. Xu et al. [28], [29] study a similar problem faced by a major tobacco company in central China for the raw materials reallocation between plants. Different from the way of modeling in Ho and Szeto [25], Chen et al. [27], and Li et al. [26], in which the pickup and delivery quantities at the locations are decision variables, Xu et al. [28], [29] take the pairing of supply and demand as decision variables. In SMMPDPH, the total supply quantity at the pickup locations is larger than the total demand quantity at the delivery locations. Thus, it is a selective M-M PDP, and we select from the total supply with pickup location selection by pairing the supply and demand.

The PDP with handling cost is first introduced by Battarra et al. [2], which incorporate the handling cost in the objective of a multi-commodity 1-M-1 PDP. And integer linear programming formulations under three different reloading policies as well as exact methods based on branch-and-cut algorithm are proposed for the problem. Hornstra et al. [31] study an extension of the problem proposed by Battarra et al. [2] by considering multiple vehicles and solve this problem with an adaptive large neighborhood search. Veenstra et al. [30] investigate a 1-1 PDP with handling cost and develop a large neighborhood search for the problem. In this paper, the selective M-M PDP with handling cost is investigated. The interaction between the pickup location selection, the routing and handling components makes the problem more complicated.

Contributions of our study lie in the following. We introduce and formulate a new variant of the pickup and delivery problem called SMMPDPH. It makes combined decisions about the selection of pickup nodes and the determination of node sequences to make a routing plan for a fleet of vehicles with a minimum total of travel cost and handling cost. Then we present two mixed integer programming formulations to solve the problem optimally. Furthermore, two algorithms with newly proposed move operators are developed to solve the problem heuristically. And the effectiveness of these approaches is demonstrated with performance tests on different instances.

## III. MATHEMATICAL FORMULATIONS

The investigated problem considers a third-party logistics service provider which performs the last-mile delivery service with BODS strategy for multiple manufacturers and retailers engaged in household appliances. For simplification, when a customer orders commodities from different manufacturers and retailers, we assume that the commodity from each manufacturer or retailer could be delivered separately by duplicating the customer. This assumption guarantees the minimum of the total cost. If a customer orders commodities from different manufacturers and retailers, enforcing a single delivery may increase the travel cost since all the selected pickup locations would be visited before the customer by the same vehicle. When the total cost of the resulting routing plan is a minimum, the algorithm will automatically assign a single delivery.

Given is a set $R$ of manufacturers and retailers. Each manufacturer or retailer $r \in R$ owns a set $P_r$ of distribution centers and physical stores (pickup locations), and services a set $D_r$ of customers (delivery locations). The request of each manufacturer or retailer is to load commodities from the distribution centers and physical stores, and then deliver them to the customers to satisfy the demand. The problem may be defined on a graph $G = (V, A)$, where $V = 0 \cup 0' \cup P \cup D$ is the set of nodes and $A = \{(i, j)|i \in V, j \in V, i \neq j\}$ is the set of arcs. Subsets $P$ and $D$ contain the pickup nodes and delivery nodes, respectively, while 0 corresponds to the origin depot and $0'$ the destination depot. Let $V' = P \cup D$ denote the set of pickup and delivery nodes, and $A' = \{(i, j)|i \in V', j \in V', i \neq j\}$ the set of arcs whose ending nodes are both in $V'$.

A travel cost $c_{ij}$ and a travel distance $d_{ij}$ are associated with each arc $(i, j) \in A$ and they satisfy the triangle inequality. The demand of each delivery node $i \in D$ is associated with a handling cost $h_i$ and a weight $w_i$. An unlimited fleet $K$ of homogeneous vehicles with limited capacity $Q$ and driving mileage $L$ is available at the origin depot to perform the transportation service. The vehicles are rear-loaded with a single stack operating in the LIFO fashion. SMMPDPH aims to obtain a routing plan for the vehicles to satisfy the requests with the minimum total of travel cost and handling cost. The problem could be mathematically formulated with the following decision variables:

- $x_{ijk}$ equals to 1 if vehicle $k \in K$ traverses arc $(i, j) \in A$, and 0 otherwise
- $q_{ijk}$ equals to 1 if vehicle $k \in K$ loads commodity from pickup node $i \in P$ and transports the commodity to delivery node $j \in D$, and 0 otherwise
- $e_{ijk}$ equals to 1 if on vehicle $k \in K$ the commodity transported to delivery node $i \in D$ is obstructed by the commodity transported to delivery node $j \in D$, and 0 otherwise
- $v_{ik}$ the remaining load of vehicle $k \in K$ when it leaves node $i \in V$

and we propose two mixed integer programming formulations for the problem.

### A. FORMULATION F1

The objective of SMMPDPH is to minimize the total cost, including the travel cost and handling cost:

$$\min f = \sum_{k\in K}\sum_{(i,j)\in A} c_{ij}x_{ijk} + \sum_{k\in K}\sum_{i\in D}\sum_{j\in D} h_j e_{ijk} \tag{1}$$

It is subject to a series of constraints.

- Vehicle route definition The vehicles start the journey from the origin depot, and arrive at the destination depot when the journey is finished.

$$\sum_{i:(0,i)\in A} x_{0ik} = 1 \quad \forall k \in K \tag{2}$$

$$\sum_{i:(i,0)\in A} x_{i0k} = 0 \quad \forall k \in K \tag{3}$$

$$\sum_{i:(0',i)\in A} x_{0'ik} = 0 \quad \forall k \in K \tag{4}$$

$$\sum_{i:(i,0')\in A} x_{i0'k} = 1 \quad \forall k \in K \tag{5}$$

A pickup node could be visited by multiple vehicles, but each vehicle must visit it at most once. A delivery node must be visited exactly once by exactly one vehicle.

$$\sum_{j:(i,j)\in A} x_{ijk} \le 1 \quad \forall i \in P, k \in K \tag{6}$$

$$\sum_{k\in K}\sum_{j:(i,j)\in A} x_{ijk} = 1 \quad \forall i \in D \tag{7}$$

- Flow conservation constraints

$$\sum_{j:(j,i)\in A} x_{jik} = \sum_{j:(i,j)\in A} x_{ijk} \quad \forall i \in V', k \in K \tag{8}$$

- Mileage constraints

$$\sum_{(i,j)\in A} d_{ij}x_{ijk} \le L \quad \forall k \in K \tag{9}$$

- Pickup and delivery operation constraints We use constraints (10) to define the dependence between variables $x_{ijk}$ and $q_{ijk}$.

$$2q_{ijk} \le \sum_{l:(i,l)\in A} x_{ilk} + \sum_{l:(j,l)\in A} x_{jlk} \quad \forall i \in P, j \in D, k \in K \tag{10}$$

The demand of a delivery node should be satisfied by exactly one pickup node from the same request.

$$\sum_{i\in P_r} q_{ijk} = 1 \quad \forall r \in R, j \in D_r \tag{11}$$

- Capacity constraints Constraints (12) specify that the vehicle is empty when it leaves from the origin depot and arrives at the destination depot. The total weight of commodities on a vehicle never exceeds capacity $Q$, and it is ensured by constraints (13). Constraints (14)–(17) record a vehicle's remaining load based on node sequences and type.

$$v_{ik} = 0 \quad \forall i \in \{0, 0'\}, k \in K \tag{12}$$

$$v_{ik} \le Q \quad \forall i \in V, k \in K \tag{13}$$

$$v_{jk} + Q(1 - x_{ijk}) \ge v_{ik} + \sum_{l\in D} q_{jlk}w_l \quad \forall i \in V, j \in P, (i,j) \in A, k \in K \tag{14}$$

$$v_{jk} - Q(1 - x_{ijk}) \le v_{ik} + \sum_{l\in D} q_{jlk}w_l \quad \forall i \in V, j \in P, (i,j) \in A, k \in K \tag{15}$$

$$v_{jk} - Q(1 - x_{ijk}) \le v_{ik} - w_j \quad \forall i \in V, j \in D, (i,j) \in A, k \in K \tag{16}$$

$$v_{jk} + Q(1 - x_{ijk}) \ge v_{ik} - w_j \quad \forall i \in V, j \in D, (i,j) \in A, k \in K \tag{17}$$

- Additional operation constraints We define $u_{ik}$ ($u_{ik} \ge 0$) to indicate the position of node $i$ in the node sequences visited by vehicle $k$. Let $z_{ijk} = 1$ if vehicle $k$ visits node $i$ before node $j$, and $z_{ijk} = 0$, otherwise. Constraints (18) are used to set an initial value to $u_{ik}$ when vehicle $k$ departs from the origin depot.

$$u_{0k} = 0 \quad \forall k \in K \tag{18}$$

Constraints (19) define the dependence between variables $u_{ik}$ and $z_{ijk}$, where $\lambda$ is a large constant.

$$u_{jk} - u_{ik} \le \lambda z_{ijk} \quad \forall i, j \in V, k \in K \tag{19}$$

Constraints (20) ensure that the values of $u_{ik}$ are correctly set along the route.

$$u_{jk} + \lambda(1 - x_{ijk}) \ge u_{ik} + 1 \quad \forall i, j \in V, (i,j) \in A, k \in K \tag{20}$$

Constraints (21) are the precedence constraints.

$$u_{jk} + \lambda(1 - q_{ijk}) \ge u_{ik} + 1 \quad \forall i \in P, j \in D, (i,j) \in A, k \in K \tag{21}$$

For constraints (22), when vehicle $k$ loads commodity from pickup node $m$ for delivery node $i$ ($q_{mik} = 1$) and from pickup node $n$ for delivery node $j$ ($q_{njk} = 1$), if vehicle $k$ visits node $m$ before node $n$, node $i$ before node $j$ and node $n$ before node $i$, then the commodity transported to node $j$ would obstruct the delivery of the commodity transported to node $i$. For SMMPDPH, the obstruction between commodities from the same pickup node or transported to the same delivery node is not counted.

$$z_{mnk} + z_{nik} + z_{ijk} + q_{mik} + q_{njk} \le e_{ijk} + 4 \quad \forall r \in R, r' \in R, m \in P_r, i \in D_r, n \in P_{r'}, j \in D_{r'}, k \in K \tag{22}$$

- Variable type constraints

$$x_{ijk} \in \{0, 1\} \quad \forall (i,j) \in A, k \in K \tag{23}$$

$$q_{ijk} \in \{0,1\} \quad \forall i \in P, j \in D, k \in K \tag{24}$$

$$e_{ijk} \in \{0,1\} \quad \forall r \in R, r' \in R, i \in D_r, j \in D_{r'}, k \in K \tag{25}$$

$$v_{ik} \geq 0 \quad \forall i \in V, k \in K \tag{26}$$

$$u_{ik} \geq 0 \quad \forall i \in V, k \in K \tag{27}$$

$$z_{ijk} \in \{0,1\} \quad \forall i, j \in V, k \in K \tag{28}$$

### B. FORMULATION F2

In formulation F2, to model the visiting order of nodes in the routes, binary variables $y^1_{oijk}$, $y^2_{oijk}$, $y^3_{oijk}$ ($\forall r \in R$, $o \in D_r$, $(i,j) \in A$ and $k \in K$) are defined. For each delivery node $o \in D$, let $p(o)$ denote the selected pickup node of o. Then, $y^1_{oijk}$ equals to 1 if vehicle $k$ traverses arc $(i,j)$ on the partial path $0 \sim p(o)$, and 0 otherwise. Likely, $y^2_{oijk}$ equals to 1 if vehicle $k$ traverses arc $(i,j)$ on the partial path $p(o) \sim o$, and 0 otherwise; $y^3_{oijk}$ equals to 1 if vehicle $k$ traverses arc $(i,j)$ on the partial path $o \sim 0'$, and 0 otherwise. We use constraints (29)–(33) to define the dependence between variables $y^1_{oijk}$, $y^2_{oijk}$, $y^3_{oijk}$ and $x_{ijk}$, $q_{ijk}$,

$$y^f_{oijk} \leq x_{ijk} \quad \forall f \in \{1,2,3\}, r \in R, o \in D_r, (i,j) \in A, k \in K \tag{29}$$

$$y^f_{oijk} \leq \sum_{l \in P} q_{lok} \quad \forall f \in \{1,2,3\}, r \in R, o \in D_r, (i,j) \in A, k \in K \tag{30}$$

$$y^1_{oijk} + y^2_{oijk} + y^3_{oijk} \leq 1 \quad \forall r \in R, o \in D_r, (i,j) \in A, k \in K \tag{31}$$

$$y^1_{oijk} + y^2_{oijk} + y^3_{oijk} \geq x_{ijk} + \sum_{l \in P} q_{lok} - 1 \quad \forall r \in R, o \in D_r, (i,j) \in A', k \in K \tag{32}$$

$$y^1_{oijk} + y^3_{oijk} \geq x_{ijk} + \sum_{l \in P} q_{lok} - 1 \quad \forall r \in R, o \in D_r, (i,j) \in A \backslash A', k \in K \tag{33}$$

Constraints (34) ensure that each delivery node is visited after the corresponding selected pickup node.

$$\sum_{j:(i,j) \in A} y^3_{oijk} + q_{iok} \leq 1 \quad \forall r \in R, o \in D_r, i \in P_r, k \in K \tag{34}$$

Constraints (35)–(37), as shown at the bottom of the next page, are used to define the route for partial paths $0 \sim p(o)$, $p(o) \sim o$ and $o \sim 0'$. For constraints (35), if vehicle $k$ delivers commodity for node $o \in D_r$ ($r \in R$), then $\lambda(1 - \sum_{l \in P_r} q_{lok}) = 0$. For node $i$ visited by vehicle $k$, when node $i$ is the origin depot, then path $0 \sim p(o)$ of vehicle $k$ has an arc leaving node $i$ but none arcs entering node $i$, which is ensured by $\sum_{j:(i,j) \in A} y^1_{oijk} - \sum_{j:(j,i) \in A} y^1_{ojik} \geq 1$; when node $i$ is pickup node $p(o)$ ($q_{iok} = 1$), then path $0 \sim p(o)$ of vehicle $k$ has an arc entering node $i$ but none arcs leaving node $i$, which is ensured by $\sum_{j:(i,j) \in A} y^1_{oijk} - \sum_{j:(j,i) \in A} y^1_{ojik} = -1$; when node $i$ is neither the origin depot nor pickup node $p(o)$, then path $0 \sim p(o)$ has both arcs leaving and entering node $i$, which is ensured by $\sum_{j:(i,j) \in A} y^1_{oijk} - \sum_{j:(j,i) \in A} y^1_{ojik} = 0$.

Constraints (36) and (37) work similarly with constraints (35) for partial paths $p(o) \sim o$ and $o \sim 0'$ respectively.

For constraints (38), as shown at the bottom of the next page, if vehicle $k$ delivers commodity for node $o$ and node $v$, when $k$ visits pickup node $p(v)$ but not delivery node $v$ on the partial path $p(o) \sim o$, then the commodity delivered to node $v$ would obstruct the commodity delivered to node $o$.

Constraints (39) are the variable type constraints.

$$y^1_{oijk}, y^2_{oijk}, y^3_{oijk} \in \{0,1\} \quad \forall r \in R, o \in D_r, (i,j) \in A, k \in K \tag{39}$$

Formulation F2 of the SMMPDPH is thus given by (1)–(17), (23)–(26) and (29)–(39).

## IV. HEURISTIC METHODS

The local search algorithm is used to solve the problem, and we adopt two different strategies to increase the search diversification. An iterated local search is proposed by adding a perturbation phase to the local search, and a memetic algorithm is designed by including the local search in the evolutionary framework. In this section, we first define some basic terms used in the presentation. Then, the initial solution construction procedures are presented. After that, we show the details of the iterated local search and the memetic algorithm.

### A. DEFINITIONS

In SMMPDPH, the total supply quantity of the pickup nodes is greater than the total demand of delivery nodes, and we select from the supply by assigning a pickup node to each delivery node so as to pair the supply and demand. We use P-D pair $i - j$ to match the supply at the selected pickup node $i$ and the demand by delivery node $j$. In the problem, each request is associated with a set of pickup nodes and delivery nodes. A pickup node could be considered as the supply for any delivery node of the request and visited by multiple vehicles. Given a route of a solution to SMMPDPH, if a pickup node is selected as the supply for exactly one delivery node in the route, then the pair of the pickup node and the corresponding delivery node is called a couple; if the pickup node is the selected pickup node of multiple delivery nodes in the route, then the group of this pickup node and these delivery nodes is termed component.

### B. INITIAL SOLUTION CONSTRUCTION

SMMPDPH jointly makes decisions on the selection of pickup nodes and the routing of vehicles to satisfy the demand of the delivery nodes. Considering this, we propose a

simple randomized greedy construction algorithm to generate an initial solution for SMMPDPH. The procedures are as follows:

- Step 1 (Pickup node selection) For each delivery node $o \in D_r$ of request $r \in R$, randomly select a pickup node from $P_r$ and put the P-D pair (i.e., the selected pickup node and the corresponding delivery node) into set *SET_PD*.
- Step 2 (Vehicle routing) Open an empty route as the current route. At each iteration, randomly select an uninserted P-D pair from *SET_PD* and insert it at the current route. The insertion of each P-D pair consists of a position for the pickup node and a position for the delivery node (the pickup node is inserted at a position before the delivery node). Each P-D pair is inserted at the best position which results in the minimum total cost increase in the current route. When the insertion is infeasible, the algorithm continues with another empty route as the current route and inserts the P-D pair. Repeat the procedures until all the pairs in *SET_PD* have been inserted and put the routes into *SET_ROUTE*.

Then, *SET_ROUTE* and *SET_PD* contain the routes and the associated pickup node selection strategy of the initial solution, respectively.

### C. ITERATED LOCAL SEARCH

Iterated local search (ILS) is first introduced by Lourenço et al. [32] and has got many applications in combinatorial optimization problems with competitive performance in comparison with many metaheuristic methods [33], [34]. The framework of the proposed ILS for SMMPDPH is summarized as Algorithm 1. Starting from an initial solution $s_0$, the search of ILS alternates between an improvement phase and a perturbation phase. The new solution $s'$ would replace the current solution $s$ if the solution acceptance criterion is met. And the best-found solution $s^*$ is updated with $s'$ if $f(s') < f(s^*)$. ILS terminates when it reaches the predefined time limit and returns the best-found solution $s^*$.

**Algorithm 1** Iterated Local Search

**Require:** SMMPDPH instance, number of P-D pairs to remove ($\rho$), probability of pickup node change ($\alpha$), threshold parameter ($\beta$)
**Ensure:** the best-found solution ($s^*$)

```
s_0 ← initial_solution_construction
s ← s_0
s* ← s_0
while stop condition is not met do
    s' ← s
    s' ← improvement_phase(s')
    s' ← perturbation_phase(s')
    if f(s') < f(s*) * (1 + β) then
        s ← s'
    end if
    if f(s') < f(s*) then
        s* ← s'
    end if
end while
```

#### 1) IMPROVEMENT PHASE

The search of improvement phase is based on local search operators and explores neighborhoods $N_1$–$N_6$ in a random order to attain a local optimum. For each neighborhood $\epsilon \in \{N_1, N_2, N_3, N_4, N_5, N_6\}$, we examine the possible moves that could be applied to the current solution $s$ with $\epsilon$ in a random order. A neighboring solution $s'$ is accepted to replace solution $s$ if $f(s') < f(s)$. The improvement phase is based on the following six move operators:

- node exchange ($N_1$) exchange the positions for two nodes from the same route (see Fig. 2(a));
- couple exchange ($N_2$) exchange the positions for two couples (see Fig. 2(b));

$$\sum_{j:(i,j)\in A} y^1_{oijk} - \sum_{j:(j,i)\in A} y^1_{ojik} \begin{cases} \geq 1-\lambda(1-\sum_{l\in P_r} q_{lok}) & \text{if } i=0 \\ =-q_{iok} & \text{if } i\in P \\ =0 & \text{if } i\in D \end{cases} \quad \forall r\in R, o\in D_r, i\in V, k\in K \tag{35}$$

$$\sum_{j:(i,j)\in A} y^2_{oijk} - \sum_{j:(j,i)\in A} y^2_{ojik} \begin{cases} =q_{iok} & \text{if } i\in P \\ \leq \lambda(1-\sum_{l\in P_r} q_{lok})-1 & \text{if } i=o \\ =0 & \text{if } i\in D\backslash\{o\} \end{cases} \quad \forall r\in R, o\in D_r, i\in V, k\in K \tag{36}$$

$$\sum_{j:(i,j)\in A} y^3_{oijk} - \sum_{j:(j,i)\in A} y^3_{ojik} \begin{cases} \geq 1-\lambda(1-\sum_{l\in P_r} q_{lok}) & \text{if } i=o \\ \leq \lambda(1-\sum_{l\in P_r} q_{lok})-1 & \text{if } i=0' \\ =0 & \text{if } i\in V'\backslash\{o\} \end{cases} \quad \forall r\in R, o\in D_r, i\in V, k\in K \tag{37}$$

$$\sum_{i:(i,l)\in A} y^2_{oilk} - \sum_{i:(i,v)\in A} y^2_{oivk} + q_{lvk} - q_{lok} \leq e_{ovk} + 1 \quad \forall r\in R, r'\in R, o\in D_r, v\in D_{r'}: v\neq o, l\in P_{r'}, k\in K \tag{38}$$

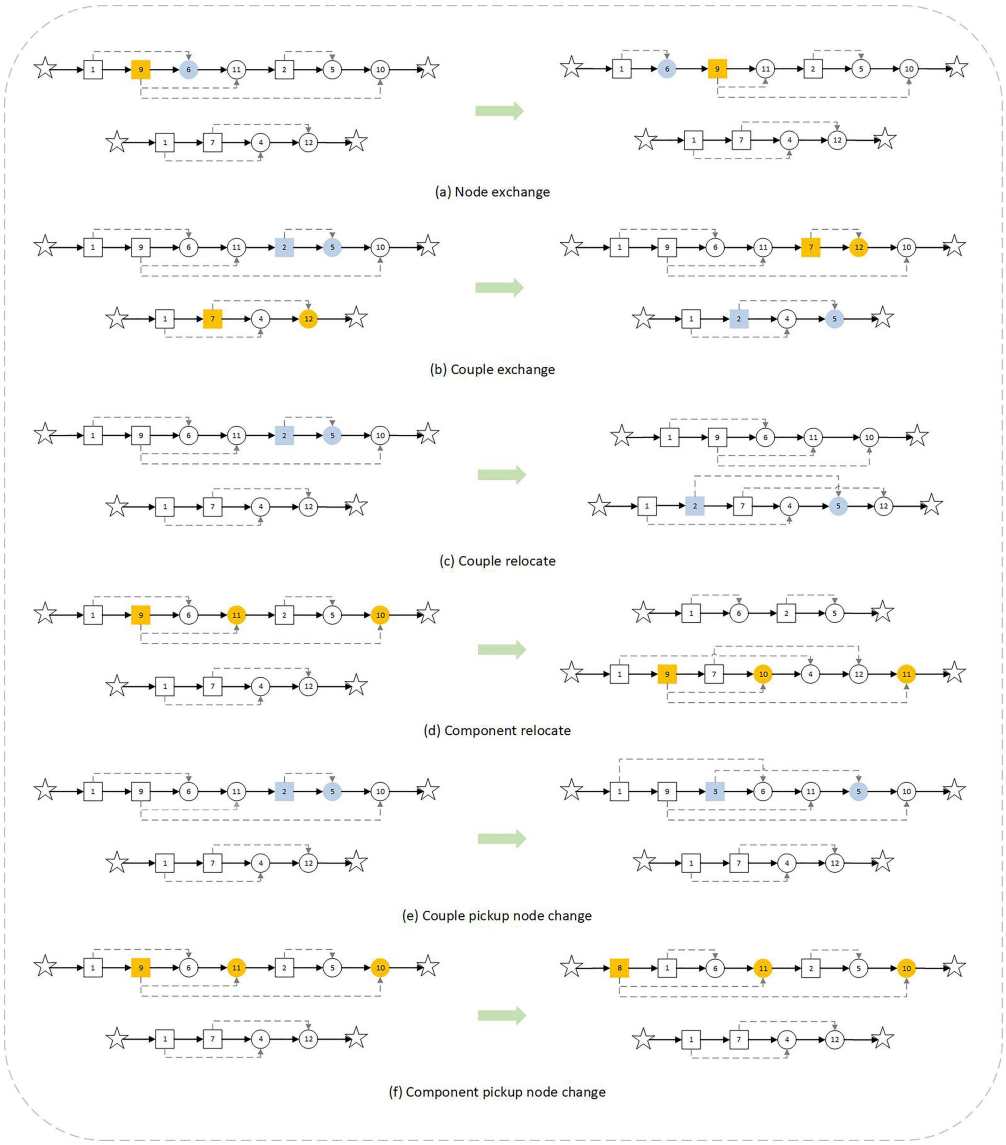



**FIGURE 2.** Examples of local search neighborhood structures.

- couple relocate ($N_3$) remove a couple and reinsert it at the best possible position (see Fig. 2(c));
- component relocate ($N_4$) remove a component and reinsert it (see Fig. 2(d));
- couple pickup node change ($N_5$) replace the pickup node of a couple and insert the new pickup node at the best possible position in the route (see Fig. 2(e));
- component pickup node change ($N_6$) replace the pickup node of a component and insert the new pickup node at the best possible position in the route (see Fig. 2(f));

Node exchange swaps nodes within a route, whereas couple exchange is used both in intra- and inter-route fashions. Relocate operators move couples and components within a route or between routes. When inserting a component into a route, the nodes are inserted in a random order and each node is inserted at the feasibly best position. The component is finally inserted in the route corresponding to the minimum total cost increase. For the relocate operators, the pickup node of the couple or component may be already in the route since a pickup node could be visited by multiple vehicles. In this case, the pickup node would be removed from the route before the insertion and reinserted in the route along with the insertion of the couple or component. Similarly, for the pickup node change operators, if the new pickup node is in the route, it would also be removed and then reinserted in the route.

### 2) PERTURBATION PHASE

The perturbation phase is based on large neighborhood search (LNS) operators. LNS is first introduced by Shaw [35] and has got applications in many pickup and delivery problems. The LNS heuristic improves the solution by removing some nodes and then reinserting them into better positions.

According to Ropke and Pisinger [36], these procedures help the search move from one promising area to another since a large number of nodes could be rearranged at one iteration. The LNS for SMMPDPH consists of removing several P-D pairs from the solution, applying the random pickup node change operator to the removed P-D pairs and then reinserting them in the left routes.

#### *a: REMOVAL OPERATORS*

At each iteration, $\rho$ P-D pairs are removed from the solution by using a randomly selected removal operator.

- Worst removal ($R_1$) Worst removal is used to remove the P-D pairs with high total cost in the current solution. Calculate $c_p(s) = f(s) - f_{-p}(s)$, where $f(s)$ is the total cost of the current solution $s$, and $f_{-p}(s)$ is the one if P-D pair $p$ is removed from $s$. Rank all the P-D pairs by descending $c_p(s)$ and the first $\rho$ P-D pairs are the ones to remove;
- Worst handling removal ($R_2$) The worst handling removal is to remove P-D pairs with high handling cost. It works similarly to worst removal. The difference lies in that worst handling removal is based on the handling cost of a P-D pair;
- Random removal ($R_3$) This operator randomly selects $\rho$ P-D pairs and removes them. In the current solution, some P-D pairs may not have high cost, but it may reduce much cost if they are inserted in better positions. The random removal could be used for such conditions so as to diversify the search;

#### *b: RANDOM PICKUP NODE CHANGE*

Let $\alpha$ denote the probability of pickup node change. For each removed P-D pair, a random number $n$ is generated. If $n \leq \alpha$, then the pickup node in the P-D pair is replaced by a randomly selected one from the corresponding alternative pickup nodes.

#### *c: INSERTION OPERATORS*

The insertion operators are used for repairing by inserting the removed P-D pairs into the destroyed solution. In our algorithm, the greedy insertion, regret-2 insertion and random insertion are used. Initially, put all the P-D pairs to be inserted into set *SET_INS* and randomly select an insertion operator. The selected operator performs $\rho$ iterations as it inserts one P-D pair at each iteration. For P-D pair $p$, let $FI(p)$ denote the set of its feasible insertions in the current solution and $\bigtriangleup f_{pi}$ denote the increase of objective value for insertion $i$. At each iteration, the selected P-D pair $p$ is inserted with the insertion that minimizes $\bigtriangleup f_{pi}$ if $FI(p)$ is not empty.

- Greedy insertion ($I_1$) Define $f_p = min_{i \in FI(p)} \bigtriangleup f_{pi}$. Then P-D pair $p$ that minimizes $\min_{p \in SET\_INS} f_p$ is chosen. It means that the P-D pair with the lowest cost increase is always to be inserted at each iteration;
- Regret-2 insertion ($I_2$) Define $f_p^j$ as the $j$th lowest cost increase of P-D pair $p$, that is, $f_p^j \leq f_p^{j'}$, for $j \leq j'$ $(j, j' \in N^+)$. For each P-D pair $p$, calculate $f_p' = f_p^2 - f_p^1$. Then P-D pair $p$ that maximizes $\max_{p \in SET\_INS} f_p'$ is the one to choose. It means that the P-D pair with the maximal difference of objective value between its second-lowest cost insertion and lowest cost insertion is selected at each iteration;
- Random insertion ($I_3$) At each iteration, the P-D pair to insert is randomly selected from *SET_INS*;

#### 3) ACCEPTANCE CRITERION

In order to further increase the search diversification, the algorithm accepts not only the improving solutions, but also those non-improving solutions bellow a certain threshold. Given a parameter $\beta$, which is used for threshold calculation and randomly selected from a given interval $I_\beta$, a new solution $s'$ is accepted to replace the current solution if $f(s') < f(s^*)(1 + \beta)$, where $s^*$ is the best-found solution.

#### 4) ACCELERATION STRATEGIES

A list is maintained to record the load of vehicles at each node in the current solution routes. Therefore, the load of vehicles in the new solution needs not be fully computed since each insertion or removal involves only one route.

At each iteration of the improvement phase, the neighborhood is defined by a reduced set of moves that are selected with a random greedy mechanism. To be more specific, given a solution $s$ and operator $\epsilon \in \{N_1, N_2, N_3, N_4, N_5, N_6\}$, a weight $\omega_m^s$, which denotes the cost of the move object $b_m$ in the solution, is associated with each possible move $m$ that can be applied to $s$ with $\epsilon$. $\omega_m^s = f(s) - f_{-b_m}(s)$, where $f(s)$ is the total cost of solution $s$ and $f_{-b_m}$ is the one when $b_m$ is removed from $s$. We select 60% of all the possible moves with the mechanism showed in Algorithm 2. The parameter $\gamma$ $(\gamma \geq 1)$ is used for randomization.

**Algorithm 2** Random Greedy Selection

**Require:** solution to SMMPDPH ($s$), list of moves that could be applied to $s$ with operator $\epsilon$ ($L(s, \epsilon)$)
**Ensure:** set of selected moves ($MOVE\_SET(s, \epsilon)$)
1: calculate $\omega_m^s$ for each move $m \in L(s, \epsilon)$
2: sort all moves by descending $\omega_m^s$
3: **while** stop condition is not met **do**
4: generate a random number $n \in [0, 1)$
5: select move $m$ with $m = L[n^\gamma |L|]$
6: remove $m$ from $L(s, \epsilon)$
7: put $m$ into $MOVE\_SET(s, \epsilon)$
8: **end while**

### *D. MEMETIC ALGORITHM*

Memetic algorithm (MA) is a powerful optimization method due to its combination of population based algorithms for search diversification and local search based improvement for intensification, and it has got successful applications on various vehicle routing problems [36]. The general architecture of MA for SMMPDPH is summarized as Algorithm 3. MA starts

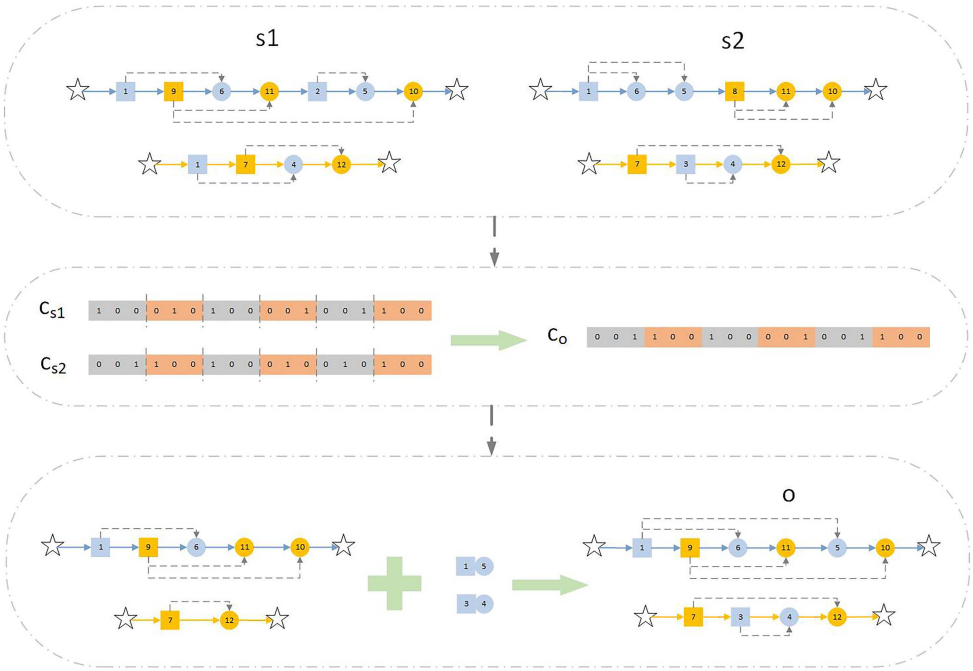



**FIGURE 3.** Illustration of the crossover operator.

with an initial population $S$ of solutions. At each iteration, it iterates between a crossover phase which recombines the best-found solution and a randomly selected solution from the population to obtain an offspring solution, and a local search phase to improve the offspring solution. The new solution $s'$ is accepted as the best-found solution $s^*$ if $f(s') < f(s^*)$. Furthermore, the population is updated with $s'$. The algorithm terminates when the predefined time limit is reached and returns the best-found solution $s^*$.

**Algorithm 3** Memetic Algorithm

**Require:** SMMPDPH instance, population size (*pop_size*)
**Ensure:** the best-found solution ($s^*$)
1: $S \leftarrow$ population_initilization
2: **while** stop condition is not met **do**
3: $s \leftarrow$ random_selection($S$)
4: $o \leftarrow$ crossover_phase($s^*$, $s$)
5: $s' \leftarrow$ local_search_phase($o$)
6: **if** $f(s') < f(s^*)$ **then**
7: $s^* \leftarrow s'$
8: $S \leftarrow$ population_update($S$, $s'$)
9: **end if**
10: **end while**

### 1) POPULATION INITIALIZATION AND UPDATING RULE

The solutions in the initial population are based on different selected pickup node configurations. Two configurations are different when they have different selected pickup nodes for at least one delivery node. Given a SMMPDPH instance, a selected pickup node configuration is determined by randomly selecting a pickup node that could satisfy the demand for each delivery node. The procedure is repeated until *pop_size* different configurations are obtained. For each configuration, a solution is generated with the procedures of initial solution construction.

The population is updated each time a new solution $s'$ is accepted as the best-found solution in the following way. Replace the solution $s_c$ from the population if $s_c$ has the same configuration with $s'$ and $f(s') < f(s_c)$. If there is no solution in the population that has the same configuration with $s'$, replace the worst solution $s_w$ from the population if $f(s') < f(s_w)$.

### 2) CROSSOVER

At each iteration of MA, the crossover operator generates an offspring solution by recombining the best-found solution and a randomly selected solution from the population. For an instance with $N$ delivery nodes, we use a chromosome that contains $N$ fragments of binary strings to represent a pickup node selection configuration. Each string fragment corresponds to one delivery node and the length of it is equal to the number of pickup nodes that belong to the same request as this delivery node. The value 1 indicates that the corresponding pickup node is selected, and 0 means that the pickup node is not selected.

Given parent solutions $s1$ (the best-found solution) and $s2$ (a solution randomly selected from the population), as well as the corresponding pickup node selection configurations $c_{s1}$ and $c_{s2}$, the offspring solution $o$ is generated with the following procedures. For delivery node $i$, let $c_i^1$ denote the corresponding string fragment in $c_{s1}$ and $c_i^2$ the corresponding string fragment in $c_{s2}$. Then, for each delivery node $i$, we randomly select a string from $\{c_i^1, c_i^2\}$ as the pickup node

selection decision of the delivery node in the offspring pickup node selection configuration $c_o$. Then, we transform the routes of solution $s1$ into a solution with $c_o$ in the following way. The P-D pairs selected with $c_{s1}$ but not with $c_o$ are removed from solution $s1$. Then, the P-D pairs selected with $c_o$ but not with $c_{s1}$ are inserted into the left routes of $s1$ with random insertion to obtain the routes of the offspring solution $o$. Fig. 3 illustrates the crossover operation on two solutions to the SMMPDPH instance depicted in Fig. 1.

## V. COMPUTATIONAL RESULTS

In this section, we present the computational experiments to evaluate the performance of the mathematical formulations and the heuristic methods. The mathematical models are solved using IBM ILP solver CPLEX 12.6 and the heuristic methods are implemented in JAVA. All the tests are executed on a Dell XPS-PC with an Intel Core i5-4460, 3.20 GHz processor, and 8.00 GB of Ram.

### A. TEST INSTANCES AND PARAMETER SETTING

The computational experiments are performed on two sets of SMMPDPH instances generated from six TSP instances including brd14051, d15112, d18512, fnl4461, nrw1379, pr1002 [37]. The procedures of generating a SMMPDPH instance from a TSP instance are as follows. First, determine the number of requests of the instance. Then, for each request, generate two random integer values in [1,5] as the number of pickup nodes and delivery nodes, respectively. After that, calculate the total number of nodes (i.e., the depot, pickup nodes and delivery nodes) of this SMMPDPH instance and randomly select the same number of different vertices from the TSP instance. The weight associated with each delivery node is a random value in $\{10, 20, 30, 40, 50\}$. Let $c_{max} = \lceil max_{r\in R}\{c(0, v^+(r)) + c(v^+(r), v^-(r)) + c(v^-(r), 0')\}\rceil$ be the maximum travel cost of a vehicle to satisfy the demand of a single delivery node, where $v^+(r)$ and $v^-(r)$ represent a pickup node and a delivery node of request $r$ respectively. The handling cost associated with each delivery node $i \in D$ is set to a random number in the interval $[0, 0.001c_{max}]$ to produce appropriate handling cost with respect to the travel cost.

For the first set, SMMPDPH instances with 2, 3, 4 requests are generated from each TSP instance; for the second set, SMMPDPH instances with 5, 10, 15, 20, 25 requests are generated. For each arc $(i, j) \in A$ in the instances, the travel cost $c_{ij}$ and travel distance $d_{ij}$ are equal to the Euclidean distance between the two nodes. Let $d_{max}$ denote the maximum travel distance of a vehicle to satisfy the demand of a single delivery node ($d_{max} = c_{max}$), the driving mileage is set to $L = d_{max}$ for instances in the first set and $L = 2 \times d_{max}$ for instances in the second set. The vehicle capacity is set to $Q = 200$ for all the instances. Each instance in the two sets is characterized by the TSP instance from which it is generated and the number of requests. For example, brd_ R2 is a SMMPDPH instance generated from TSP instance brd14051 and with 2 requests.

The parameter tuning for the two algorithms is conducted on a sample of 10 instances randomly selected from the second set. We set the initial values of the parameters based on preliminary experiments and then sequentially modify each of them individually. The number of removed P-D pairs at each iteration is set to $\rho = \min\{0.2|D| + 1, 5\}$, where $|D|$ is the number of delivery nodes of the respective instance. We put an upper bound 5 on $\rho$ to keep the computing time at a reasonable level, since the insertion procedures would take too much time when lots of P-D pairs are involved for large sized instances. The pickup node change probability is set to $\alpha = 0.5$. The threshold ratio $\beta$ is randomly selected from interval $I_\beta = [0.05, 0.1]$ and the parameter $\gamma$ for randomization is set to $\gamma = 2$. The population size of the MA is set to $pop\_size = 5$.

### B. PERFORMANCE EVALUATION

In this section, the performance of the approaches is evaluated with computational results on the two sets of SMMPDPH instances. Beginning with presenting the comparative results of the mathematical formulations, ILS and MA on small instances from the first set, we then show the comparative results of ILS and MA on medium and large instances from the second set. This section ends with the discussion of the effect of the parent solutions selection strategy in the crossover operator in MA.

#### 1) COMPARATIVE RESULTS OF FORMULATIONS F1, F2, AND THE TWO HEURISTIC METHODS ON INSTANCES FROM THE FIRST SET

Table 1 shows the computational results of the formulations and the two algorithms on instances from the first set. We impose a time limit of 10,800 s (3h) to CPLEX, which is also the time limit used by many other studies [38, 39, 40]. The results of the algorithms are based on 10 runs and the best one is reported. In each row, we indicate in bold the best value computed for that instance. The percentage deviation (i.e., the gap) is defined as (corresponding objective value of the solution−objective value of the solution by the formulation) / objective value of the solution by the formulation. $\Delta 1$ and $\Delta 2$ correspond to the percentage deviation of the algorithm from the objective value of the solution by formulation F1 and F2, respectively. Column 1 contains the instance and column 2 is the number of nodes (the pickup nodes and the delivery nodes) of the instance. Columns 3–4 and columns 5–6 provide the solution objective value and computing time in seconds by formulation F1 and F2, respectively. Column 7 is the time limit in seconds of the algorithms for the instance. Columns 8–10 and 11–13 present the best solution, the gaps by ILS and MA respectively.

As shown in Table 1, the 6 instances with 2 requests are optimally solved by the formulations. And for 5 of the 6 instances, formulation F1 takes slightly less time. For the 12 instances with 3 and 4 requests, formulation F1 obtains the optimal solution for 2 instances, whereas formulation F2 finds the optimal solution for 5 instances. Furthermore, the

**TABLE 1.** Comparison of results of the heuristic methods to CPLEX on instances from the first set.

| Instance | Node | F1 | | F2 | | Time | ILS | | | MA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $f$ | $t$ | $f$ | $t$ | | $f_b$ | $\Delta 1$ | $\Delta 2$ | $f_b$ | $\Delta 1$ | $\Delta 2$ |
| brd_R2 | 8 | **13943.53** | 3.21 | **13,943.53** | 3.30 | 1 | **13,943.53** | 0.00 | 0.00 | **13,943.53** | 0.00 | 0.00 |
| brd_R3 | 14 | **21,063.76** | 10,800.00 | **21,063.76** | 1,645.17 | 5 | **21,063.76** | 0.00 | 0.00 | **21,063.76** | 0.00 | 0.00 |
| brd_R4 | 20 | # | 10,800.00 | # | 10,800.00 | 10 | 39,695.84 | - | - | **39,636.15** | - | - |
| d15_R2 | 10 | **63,365.75** | 15.58 | **63,365.75** | 32.71 | 1 | **63,365.75** | 0.00 | 0.00 | **63,365.75** | 0.00 | 0.00 |
| d15_R3 | 15 | 71,237.54 | 10,800.00 | **60,053.86** | 3,504.74 | 5 | **60,053.86** | −15.70 | 0.00 | **60,053.86** | −15.70 | 0.00 |
| d15_R4 | 20 | # | 10,800.00 | 126,992.25 | 10,800.00 | 10 | 93,936.52 | - | −26.03 | **93,032.54** | - | −26.74 |
| d18_R2 | 9 | **13,352.21** | 1.84 | **13,352.21** | 5.16 | 1 | **13,352.21** | 0.00 | 0.00 | **13,352.21** | 0.00 | 0.00 |
| d18_R3 | 13 | **13,743.96** | 98.49 | **13,743.96** | 73.29 | 5 | **13,743.96** | 0.00 | 0.00 | **13,743.96** | 0.00 | 0.00 |
| d18_R4 | 18 | # | 10,800.00 | 57,532.90 | 10,800.00 | 10 | **42,191.38** | - | −26.67 | **42,191.38** | - | −26.67 |
| fnl_R2 | 10 | **6,498.16** | 3.01 | **6,498.16** | 15.42 | 1 | **6,498.16** | 0.00 | 0.00 | **6,498.16** | 0.00 | 0.00 |
| fnl_R3 | 17 | 13,922.48 | 10,800.00 | 12,671.44 | 10,800.00 | 5 | **12,358.15** | −11.24 | −2.47 | **12,358.15** | −11.24 | −2.47 |
| fnl_R4 | 19 | # | 10,800.00 | 14,723.79 | 10,800.00 | 10 | 14,710.69 | - | −0.09 | **14,563.21** | - | −1.09 |
| nrw_R2 | 10 | **6,549.69** | 160.05 | **6,549.69** | 23.16 | 1 | **6,549.69** | 0.00 | 0.00 | **6,549.69** | 0.00 | 0.00 |
| nrw_R3 | 15 | # | 10,800.00 | 8,458.80 | 10,800.00 | 5 | **8,453.60** | - | −0.06 | **8,453.60** | - | −0.06 |
| nrw_R4 | 16 | # | 10,800.00 | 7,973.42 | 10,800.00 | 10 | **7,915.43** | - | −0.73 | **7,915.43** | - | −0.73 |
| pr1_R2 | 11 | **34,311.81** | 30.38 | **34,311.81** | 51.11 | 1 | **34,311.81** | 0.00 | 0.00 | **34,311.81** | 0.00 | 0.00 |
| pr1_R3 | 12 | 56,076.80 | 10,800.00 | **53,648.50** | 588.28 | 5 | **53,648.50** | −4.33 | 0.00 | **53,648.50** | −4.33 | 0.00 |
| pr1_R4 | 24 | 52,173.81 | 10,800.00 | **48,493.31** | 2,662.80 | 10 | **48,493.31** | −7.05 | 0.00 | **48,493.31** | −7.05 | 0.00 |

**TABLE 2.** Comparison of results of the heuristic methods on instances from the second set.

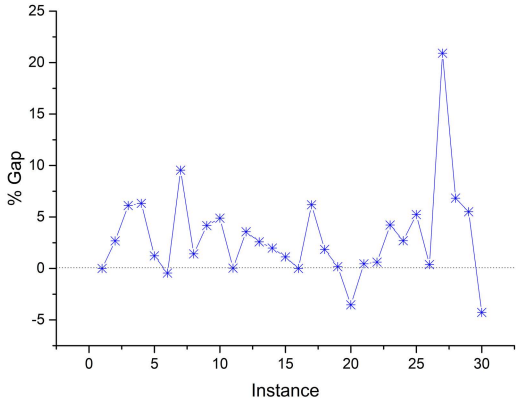
| Instance | Node | Time | ILS | | MA | |
|---|---|---|---|---|---|---|
| | | | $f_b$ | $f_a$ | $f_b$ | $f_a$ |
| brd_R5 | 20 | 10 | **20156.41** | 22176.61 | **20156.41** | 20261.07 |
| brd_R10 | 52 | 60 | 42376.34 | 47004.74 | **38414.65** | 41548.08 |
| brd_R15 | 91 | 300 | 69667.01 | 75876.42 | **59450.04** | 66777.04 |
| brd_R20 | 125 | 600 | **112814.01** | 121423.06 | 138720.37 | 156030.24 |
| brd_R25 | 139 | 1800 | 111264.83 | 116283.42 | **98048.31** | 101209.32 |
| d15_R5 | 28 | 10 | 74264.72 | 85425.69 | **69983.13** | 74269.71 |
| d15_R10 | 65 | 60 | 144427.96 | 182434.26 | **141107.90** | 157402.19 |
| d15_R15 | 91 | 300 | 240003.22 | 257774.10 | **188271.09** | 208368.46 |
| d15_R20 | 115 | 600 | 329398.74 | 345394.02 | **276597.84** | 306766.09 |
| d15_R25 | 143 | 1800 | 397571.58 | 428179.69 | **330428.07** | 357075.12 |
| d18_R5 | 30 | 10 | 41783.29 | 46746.59 | **39626.39** | 40108.96 |
| d18_R10 | 61 | 60 | 51443.69 | 56866.16 | **44869.16** | 50084.43 |
| d18_R15 | 83 | 300 | 93835.22 | 100497.58 | **81079.58** | 83459.33 |
| d18_R20 | 109 | 600 | 107463.78 | 120098.49 | **93637.80** | 100260.16 |
| d18_R25 | 131 | 1800 | 140219.90 | 151252.51 | **114343.09** | 122722.10 |
| fnl_R5 | 23 | 10 | 12747.93 | 14126.72 | **12681.25** | 13347.73 |
| fnl_R10 | 69 | 60 | 36529.78 | 39131.28 | **32209.12** | 36751.07 |
| fnl_R15 | 93 | 300 | 55751.19 | 59106.35 | **48326.01** | 50571.78 |
| fnl_R20 | 122 | 600 | 63049.63 | 67124.17 | **59341.40** | 66243.81 |
| fnl_R25 | 151 | 1800 | 79688.96 | 83268.12 | **69731.02** | 73554.67 |
| nrw_R5 | 23 | 10 | 6881.84 | 7289.35 | **6470.84** | 6824.19 |
| nrw_R10 | 66 | 60 | 20788.15 | 22600.66 | **19059.48** | 20407.33 |
| nrw_R15 | 101 | 300 | 29141.87 | 31883.02 | **28452.68** | 30679.58 |
| nrw_R20 | 107 | 600 | 34581.60 | 37127.69 | **29206.03** | 30950.93 |
| nrw_R25 | 143 | 1800 | 43637.29 | 47211.11 | **36735.07** | 38865.54 |
| pr1_R5 | 24 | 10 | 46927.76 | 57158.87 | **46469.30** | 47925.48 |
| pr1_R10 | 61 | 60 | 132740.58 | 151381.45 | **96697.77** | 118003.52 |
| pr1_R15 | 95 | 300 | 184994.59 | 200134.95 | **157519.64** | 176162.59 |
| pr1_R20 | 113 | 600 | 244562.91 | 262772.10 | **194893.86** | 209062.81 |
| pr1_R25 | 159 | 1800 | **325813.98** | 340671.50 | 327345.62 | 348513.24 |

computing time of F1 is much longer than F2. For 6 out of the 18 instances, formulation F1 fails to find a feasible solution within the time limit, whereas formulation F2 fails to find a feasible solution for only 1 instance. In summary, formulation F2 performs much better than F1.

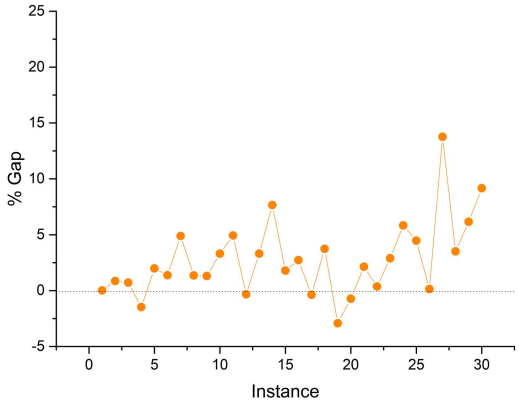
For all the instances, the two proposed heuristic methods find the same good or better solutions than those by the two formulations. Furthermore, the computing time of ILS and MA is much shorter than the formulations, especially for instances with 3 and 4 requests. On average, the ability of ILS and MA to solve SMMPDPH within a shorter computing time is much better.

### 2) COMPARATIVE RESULTS OF THE TWO HEURISTIC METHODS ON INSTANCES FROM THE SECOND SET

Table 2 shows the computational results of the two algorithms on instances from the second set. The results are based on 10 runs. For each instance, the better result is shown in bold.




(a) $\Delta_b$




(b) $\Delta_a$

**FIGURE 4. Comparative results of MA and MA' on instances from the second set.**

Column 1 contains the instance and column 2 includes the number of nodes in the instance. Column 3 is the predefined time limit. Columns 4–5 and columns 6-7 present the best ($f_b$) and the average ($f_a$) objective value by ILS and MA, respectively.

As shown in Table 2, for instance brd_R5, the two heuristic methods report the same good solution. For 2 out of 30 instances, ILS gives better results, whereas MA obtains better solutions for the other 27 instances. Furthermore, the average solution quality of MA is much better than ILS for most of the instances (29 out of 30). Although both the two heuristic methods are based on the same local search phase, MA is much better than ILS in terms of solution quality. It could be concluded that including the local search in the evolutionary framework brings much more diversification than adding a perturbation phase to the local search.

### 3) EFFECT OF THE PARENT SOLUTIONS SELECTION STRATEGY IN THE CROSSOVER OPERATOR

In MA, we obtain the offspring solution by recombining the best-found solution and a solution randomly selected from the population, rather than two solutions both from the population. In order to investigate the effect of the proposed strategy, we compare MA with MA', in which the crossover operator considers two solutions randomly selected from the population as parent solutions. The experiments are performed on all the instances from the second set. For each instance, we compute the difference between the objective values of the best solutions obtained by the two algorithms and the difference between the objective values of the average solutions. Let $\Delta_b$ and $\Delta_a$ denote the percentage gap between the best solutions and average solutions, respectively, then the gaps are computed with the following relations:

$$\Delta_b = (f_b^{MA'} - f_b^{MA})/f_b^{MA} \times 100$$
$$\Delta_a = (f_a^{MA'} - f_a^{MA})/f_a^{MA} \times 100$$

where $f_b^{MA}, f_b^{MA'}$ are the objective values of the best solutions by MA, MA' respectively, and $f_a^{MA}$, $f_a^{MA'}$ are the objective values of the corresponding average solutions.

The results are plotted in Fig.4. We could observe that MA performs better than MA' on most instances in terms of both the best and average solution quality. This proves the benefit of the proposed parents selection strategy in the crossover operator in MA.

## VI. CONCLUSION

This paper addresses the vehicle routing problem arising in the omni-channel last-mile delivery by formulating it as a new variant of the pickup and delivery problem called SMMPDPH. In SMMPDPH, each request is associated with a set of pickup and delivery locations. A request is to load commodities from the selected pickup locations and transport them to the delivery locations. Based on the loading and unloading policy, the handling cost is incorporated in the objective function. The problem aims to design the routes for a fleet of homogeneous vehicles with limited capacity and driving mileage with a minimum total cost in order to satisfy the requests. We propose two mathematical formulations to solve the problem optimally and develop two algorithms for the problem by considering two different ways for search diversification. ILS alternates between a local search phase for improvement and a large neighborhood search phase for perturbation. MA bases the search on a population of solutions and diversifies the search with a crossover operator, which generates an offspring solution by recombining the best-found solution and a randomly selected solution from the population. Then, the offspring solution is improved by the local search phase. Two sets of instances are generated to test the formulations and algorithms. In the computational experiments, we compare the performance of the formulations and algorithms on instances from the first set. The results show that formulation F2 performs much better and that the algorithms can obtain the same good or better solutions within shorter computing time. With a comparison of efficiency on large-scale instances, we find that MA is relatively better in terms of solution quality.

The following research directions could be considered in future work. The proposed algorithms could be further improved by introducing new operators for pickup node selection. It is meaningful to extend the SMMPDPH by considering the customer service time windows, in which case the additional operation is penalized by being associated with a handling time. The problem with multi-commodity or heterogeneous vehicles is also worth studying.

## REFERENCES

[1] P. He, Y. He, and H. Xu, "Buy-online-and-deliver-from-store strategy for a dual-channel supply chain considering retailer's location advantage," *Transp. Res. E, Logistics Transp. Rev.*, vol. 144, Dec. 2020, Art. no. 102127, doi: 10.1016/j.tre.2020.102127.
[2] M. Battarra, G. Erdoğan, G. Laporte, and D. Vigo, "The traveling salesman problem with pickups, deliveries, and handling costs," *Transp. Sci.*, vol. 44, no. 3, pp. 383–399, Aug. 2010.
[3] M. Ruan, C. Shen, J. Tang, C. Qi, and S. Qiu, "A double traveling salesman problem with three-dimensional loading constraints for bulky item delivery," *IEEE Access*, vol. 9, pp. 13052–13063, 2021.
[4] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, Oct. 1959.
[5] M. W. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transp. Sci.*, vol. 29, no. 1, pp. 17–29, Feb. 1995.
[6] F. Lokin, "Procedures for traveling salesman problems with additional constraints," *Eur. J. Oper. Res.*, vol. 3, no. 2, pp. 135–141, May 1979.
[7] S. Mitrovic-Minic, "Pickup and delivery problem with time windows: A survey," School Comput. Sci., Simon Fraser Univ., Burnaby, BC, Canada, Tech. Rep. 1998-12, May 1998.
[8] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: A classification scheme and survey," *Top*, vol. 15, pp. 1–31, Jul. 2007.
[9] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems: Part I: Transportation between customers and depot," *J. Für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, Apr. 2008.
[10] S. N. Parragh, K. F. Doerner, and R. F. Hart, "A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations," *J. Für Betriebswirtschaft*, vol. 58, no. 2, pp. 81–117, Apr. 2008.
[11] G. Berbeglia, J. Cordeau, and G. Laporte, "Dynamic pickup and delivery problems," *Eur. J. Oper. Res.*, vol. 202, no. 1, pp. 8–15, Apr. 2010.
[12] N. Wassan and G. Nagy, "Vehicle routing problem with deliveries and pickups: Modelling issues and meta-heuristics solution approaches," *Int. J. Transp.*, vol. 2, no. 1, pp. 95–110, Apr. 2014.
[13] Ç. Koç, G. Laporte, and I. Tükenmez, "A review of vehicle routing with simultaneous pickup and delivery," *Comput. Oper. Res.*, vol. 122, Oct. 2020, Art. no. 104987, doi: 10.1016/j.cor.2020.104987.
[14] H. Hernández-Pérez and J. J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Appl. Math.*, vol. 145, pp. 126–139, Dec. 2004.
[15] H. Hernández-Pérez and J.-J. Salazar-González, "The one-commodity pickup-and-delivery traveling salesman problem: Inequal. and algorithms," *Networks*, vol. 50, no. 4, pp. 258–272, Sep. 2007.
[16] H. Hernández-Pérez and J.-J. Salazar-González, "The multi-commodity pickup-and-delivery traveling salesman problem," *Networks*, vol. 63, no. 1, pp. 46–59, Oct. 2013.
[17] H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González, "A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem," *Eur. J. Oper. Res.*, vol. 251, no. 1, pp. 44–52, May 2016.
[18] Y. Lu, U. Benlic, and Q. Wu, "A population algorithm based on randomized Tabu thresholding for the multi-commodity pickup-and-delivery traveling salesman problem," *Comput. Oper. Res.*, vol. 101, pp. 285–297, Jan. 2019.
[19] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for the split-demand one-commodity pickup-and-delivery travelling salesman problem," *Eur. J. Oper. Res.*, vol. 297, no. 2, pp. 467–483, Mar. 2022.
[20] M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," *Omega*, vol. 45, pp. 7–19, Jun. 2014.
[21] M. Dell'Amico, M. Iori, S. Novellani, and T. Stützle, "A destroy and repair algorithm for the bike sharing rebalancing problem," *Comput. Oper. Res.*, vol. 71, pp. 149–162, Jul. 2016.
[22] M. Dell'Amico, M. Iori, S. Novellani, and A. Subramanian, "The bike sharing rebalancing problem with stochastic demands," *Transp. Res. B, Methodol.*, vol. 118, pp. 362–380, Dec. 2018.
[23] C. Li, L. Gong, Z. Luo, and A. Lim, "A branch-and-price-and-cut algorithm for a pickup and delivery problem in retailing," *Omega*, vol. 89, pp. 71–91, Dec. 2019.
[24] Y. Lu, U. Benlic, and Q. Wu, "An effective memetic algorithm for the generalized bike-sharing rebalancing problem," *Eng. Appl. Artif. Intell.*, vol. 95, Oct. 2020, Art. no. 103890, doi: 10.1016/j.engappai.2020.103890.
[25] S. C. Ho and W. Y. Szeto, "Solving a static repositioning problem in bike-sharing systems using iterated Tabu search," *Transp. Res. E, Logistics Transp. Rev.*, vol. 69, pp. 180–198, Sep. 2014.
[26] Y. Li, W. Szeto, J. Long, and C. Shui, "A multiple type bike repositioning problem," *Transp. Res. B*, vol. 90, pp. 263–278, Aug. 2016.
[27] Q. Chen, K. Li, and Z. Liu, "Model and algorithm for an unpaired pickup and delivery vehicle routing problem with split loads," *Transp. Res. E, Logistics Transp. Rev.*, vol. 69, pp. 218–235, Sep. 2014.
[28] D. Xu, K. Li, X. Zou, and L. Liu, "An unpaired pickup and delivery vehicle routing problem with multi-visit," *Transp. Res. E, Logistics Transp. Rev.*, vol. 103, pp. 218–247, Jul. 2017.
[29] X. Dongyang, L. Kunpeng, Y. Jiehui, and C. Ligang, "A multicommodity unpaired pickup and delivery vehicle routing problem with split loads and unloads," *Ind. Manage. Data Syst.*, vol. 120, no. 8, pp. 1565–1584, Jul. 2020.
[30] M. Veenstra, K. J. Roodbergen, I. F. A. Vis, and L. C. Coelho, "The pickup and delivery traveling salesman problem with handling costs," *Eur. J. Oper. Res.*, vol. 257, no. 1, pp. 118–132, Feb. 2017.
[31] R. P. Hornstra, A. Silva, K. J. Roodbergen, and L. C. Coelho, "The vehicle routing problem with simultaneous pickup and delivery and handling costs," *Comput. Oper. Res.*, vol. 115, Mar. 2020, Art. no. 104858, doi: 10.1016/j.cor.2019.104858.
[32] H. R. Lourenco, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of Metaheuristics*. Boston, MA, USA: Springer, 2003, pp. 320–353.
[33] M. R. Nakkala, A. Singh, and A. Rossi, "Multi-start iterated local search, exact and matheuristic approaches for minimum capacitated dominating set problem," *Appl. Soft Comput.*, vol. 108, Sep. 2021, Art. no. 107437, doi: 10.1016/j.asoc.2021.107437.
[34] L. Jiang, H. Chang, S. Zhao, J. Dong, and W. Lu, "A travelling salesman problem with carbon emission reduction in the last mile delivery," *IEEE Access*, vol. 7, pp. 61620–61627, 2019.
[35] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *Proc. CP 4th Internat. Conf. Princ. Pract. Constraint Program.* Berlin, Germany: Springer, 1998, pp. 417–431.
[36] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, 2006.
[37] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, Nov. 1991.
[38] M. Veenstra, M. Cherkesly, G. Desaulniers, and G. Laporte, "The pickup and delivery problem with time windows and handling operations," *Comput. Operations Res.*, vol. 77, pp. 127–140, Jan. 2017.
[39] Z. Huang, W. Huang, and F. Guo, "Integrated sustainable planning of self-pickup and door-to-door delivery service with multi-type stations," *Comput. Ind. Eng.*, vol. 135, pp. 412–425, Sep. 2019.
[40] J. Yang, F. Guo, and M. Zhang, "Optimal planning of swapping/charging station network with customer satisfaction," *Transp. Res. E, Logistics Transp. Rev.*, vol. 103, pp. 174–197, Jul. 2017.

**YALI LI** is currently pursuing the Ph.D. degree in management science and engineering with the Huazhong University of Science and Technology, Wuhan, China. Her research interests include the vehicle routing problems and heuristic algorithms.