algorithms

MDPI

Article

Efficient Mathematical Lower Bounds for City Logistics Distribution Network with Intra-Echelon Connection of Facilities: Bridging the Gap from Theoretical Model Formulations to Practical Solutions

Zhiqiang Niu [1], Shengnan Wu [2,*] and Xuesong (Simon) Zhou [3,*]

[1] Research Center of Logistics, Ministry of Transport Research Institute of Highways, Beijing 100088, China; zq.niu@rioh.cn
[2] Technology and Data Science Department, JD Logistics, Beijing 100176, China
[3] School of Sustainable Engineering and the Built Environment, Arizona State University, Tempe, AZ 85281, USA
* Correspondence: wushengnan1@jd.com (S.W.); xzhou74@asu.edu (X.Z.)

Abstract: Focusing on the dynamic improvement of the underlying service network configuration, this paper aims to address a specific challenge of redesigning a multi-echelon city logistics distribution network. By considering the intra-echelon connection of facilities within the same layer of echelon, we propose a new distribution network design model by reformulating the classical quadratic assignment problem (QAP). To minimize the overall transportation costs, the proposed model jointly optimizes two types of decisions to enable agile distribution with dynamic "shortcuts": (i) the allocation of warehouses to supply the corresponding distribution centers (DCs), and (ii) the demand coverage decision from distribution centers to delivery stations. Furthermore, a customized branch-and-bound algorithm is developed, where the lower bound is obtained by adopting Gilmore and Lawler lower Bound (GLB) for QAP. We conduct extensive computational experiments, highlighting the significant contribution of GLB-oriented lower bound, to obtain practical solutions; this type of efficient mathematical lower bounds offers a powerful tool for balancing theoretical research ideas with practical and industrial applicability.

Keywords: multi-echelon distribution network; lateral-transshipment; quadratic assignment problem; branch-and-bound; matheuristics

Citation: Niu, Z.; Wu, S.; Zhou, X. Efficient Mathematical Lower Bounds for City Logistics Distribution Network with Intra-Echelon Connection of Facilities: Bridging the Gap from Theoretical Model Formulations to Practical Solutions. *Algorithms* **2023**, *16*, 252. https://doi.org/10.3390/a16050252

Academic Editor: Angel A. Juan

Received: 16 March 2023
Revised: 3 May 2023
Accepted: 8 May 2023
Published: 12 May 2023

1. Introduction

The rise of e-commerce purchases and the growing reliance on the underlying city logistics distribution system have led to rapid development of timely deliveries. In particular, the recent pandemic creates a wide range of challenges for the global system of logistics, especially in mitigating the shortages of many commodities, such as masks, vaccine vials, and semiconductors, as e-commerce logistics providers start to shift their focus from expanding product availability to ensure time-sensitive product accessibility. An efficient distribution network is a building block to offer rapid fulfillment services within budget constraints. In essence, by achieving the economy of scale, service providers aim to continuously optimize the commodity flows (or raw materials) using the multi-echelon distribution system. In general, the distribution network design problems can be classified into two categories [1], i.e., (i) keep the existing distribution network and optimize the transportation of commodity flows, (ii) improve the existing distribution network and optimize the network configuration. Main decisions include the locations of facilities in different echelons, the assignment between facilities, and the allocation of commodity flows [2].

The shipping system underpinning globalization with distributed production and consumers across different cities, states, and countries, seems to be unable to absorb highly dynamic demand and supply disruptions, e.g., from COVID-19. More precisely, the traditional supply chain with standard layers cannot adapt to rapidly changing environments associated with economy and customer demands. By designing lateral transshipment policies (that is, "shortcuts") and enabling agile supply chain networks to respond to such issues immediately, supply chain systems can successfully operate in dynamic environments. From a long term operating perspective, the decision-makers aim to recover quickly from other potential unexpected disruptions by enabling agile supply chain networks. Particularly, our study is interested in lateral-transshipments and intra-echelon transfers, which are used to improve the performance of an inventory system and supply chain [3]; this structure can potentially lead to dramatic cost reductions throughout the course of manufacturing and shipping.

Lateral-transshipments involved in previous studies are conducted within the start or end echelons of a distribution network, i.e., warehouses (inventory system) [4] or retailers [5,6]. In their models, flow-based decision variables for intra-echelon links are used to capture the transshipment feature; the impact of lateral transport mainly occurs at the single echelon in the distribution network, as shown in Figure 1. This study will consider a distribution network design problem following the single sourcing or single route strategy, where lateral-transshipments in DCs are allowed. In this situation, the allocation decisions between each entity in different echelons are tightly coupled. Rabbani [6] simply divided the original problem into three sub-problems: plant-DCs, DCs-retailers, and retailers-customers, and then solved them sequentially. However, this sequential solution framework cannot be directly applied in our problem with the strongly coupled decision variables across different layers.

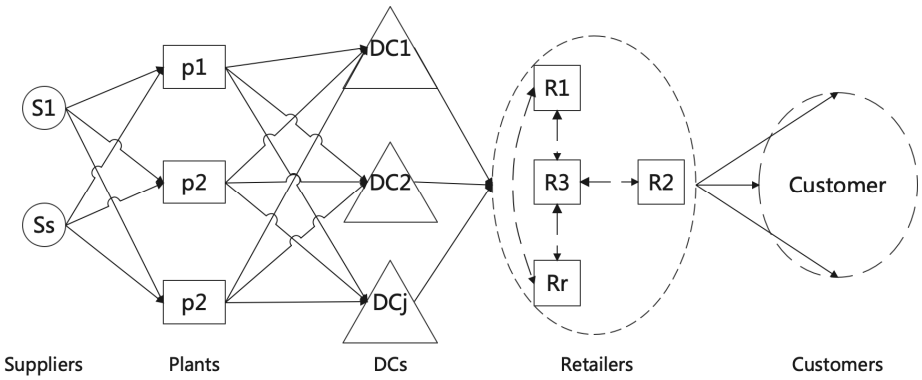


Figure 1. A five-echelon supply chain network with lateral-transshipments in retailers [6].

As lateral-transshipments within DCs require that the objective function must include pairwise allocation costs, the proposed model in this paper is a special case of the generalized quadratic assignment problem (GQAP). GQAP has been a very important modelling framework and has been adopted widely to solve real-world problems [7,8]. A recent and great application of GQAP is the work of [9], where they formulated an optimization model for the school time selection problem (STSP) as GQAP which led to $5 million in yearly savings, maintaining service quality for students despite a 50-bus fleet reduction. As [9] pointed out, even small instances of GQAP could be computationally intractable and typically simple local improvement heuristics are used in practice. Recently, a novel research line to overcome this challenge has been proposed by [10,11], through reformulating QAP as the Quadratic Unconstrained Binary Optimization modelling framework (QUBO), which is currently widely used to bridge classical combinatory optimization and emerging quantum computing areas. We refer the reader to the excellent tutorials for QUBO by [11] and optimization problems that QUBO can handle [12]. As QUBO models lie at the heart of experimentation carried out with quantum computers developed by D-Wave Systems and

neuromorphic computers developed by IBM, we mainly highlight this important research direction for future research to further reformulate the proposed model to the QUBO with efficient quantum computing implementation.

This paper hopes to offer the following potential contributions: First, we study and develop a new type of multi-echelon distribution network design models, where lateral-transshipments are allowed within DCs. Second, we further reformulate a nonlinear 0–1 integer programming model from the classical quadratic assignment modeling perspective by enhancing the network relationships between facility entities, to fully utilize the effective approximate bounding rules such as GLB. Third, both exact and heuristic algorithms combined with effective heuristic lower bound estimates are systematically developed and tested to demonstrate the effectiveness of proposed algorithms.

We would like to clarify that our paper presents a mathheuristic approach, which are problem agnostic optimization algorithms that make use of mathematical programming (MP) techniques in order to obtain heuristic solutions. Problem-dependent elements such as GLB bound are included within the lower-level mathematic programming-based components. The current research frontiers for solving QAP are still mainly heuristic algorithms such as Tabu search [13], artificial bee colony algorithm [14,15], or hybrid heuristic algorithm [16]. Although heuristic or metaheuristic algorithms can find feasible solutions in an acceptable computation time, they cannot guarantee to find the best solution and they may become stuck in a cycle or fail to escape a suboptimal solution. In addition, the behavior of a heuristic algorithm can be highly dependent on the values of its parameters and choosing optimal parameter settings can be a time-consuming and difficult task. The main advantages of our method are related to the ability to decompose a complex problem into smaller subproblems, and to use both exact mathematical methods and heuristic techniques to solve them, which can balance feasibility and optimality. The proposed algorithm with decomposition schemes has at least the following specific advantages:

(1) Increased efficiency: By breaking down the original problem into smaller subproblems, the algorithm can reduce the overall computational complexity of the optimization problem. This can lead to faster solution times and increased efficiency compared to traditional optimization algorithms.
(2) Improved scalability: The algorithm can handle large-scale optimization problems that would be intractable using other methods. Large problems can be decomposed into smaller subproblems that can be solved independently, which makes it easier to solve the overall problem.
(3) Better quality solutions: Since the algorithms combine both exact and heuristic methods, they are able to find high-quality solutions that balance feasibility and optimality. Some subproblems may be solved exactly while others may use heuristic methods to find good approximate solutions.
(4) Robustness: Since the algorithms are designed to handle complex real-world problems, they are often more robust than traditional optimization algorithms. This means that they can handle uncertain or variable inputs and can adapt to changing conditions or constraints during the optimization process.

The remainder of this paper is arranged as follows: The next section provides a literature review on the multi-echelon distribution network design problem. Section 3 provides the problem description and formulation. A customized heuristic solution approach based on the branch-and-bound algorithm is developed in Section 4. Computational results and analysis of different instances are presented in Section 5. Lastly, Section 6 delivers the conclusions and future research directions.

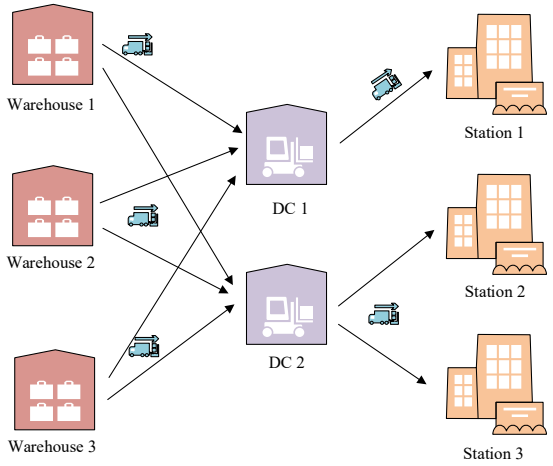
2. Literature Review

2.1. Basic Types of City Logistics Distribution Networks

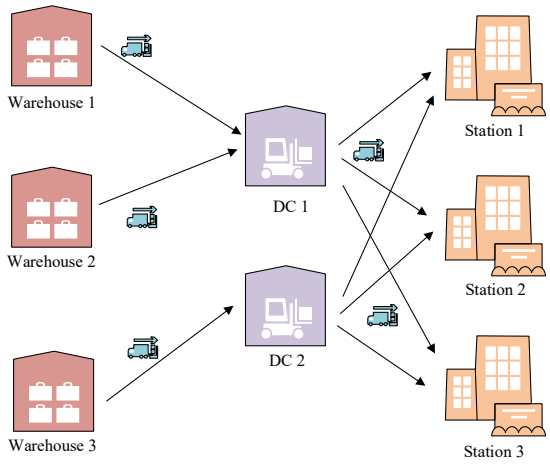
A simple distribution network usually consists of three types of facilities/nodes, i.e., warehouses/plants, DCs, and delivery stations/customers [1]. City logistics, as warehouses and DCs occupy relatively large areas, are usually located in suburban areas. Meanwhile,

delivery stations are widely distributed across the city to guarantee the fast delivery of orders.

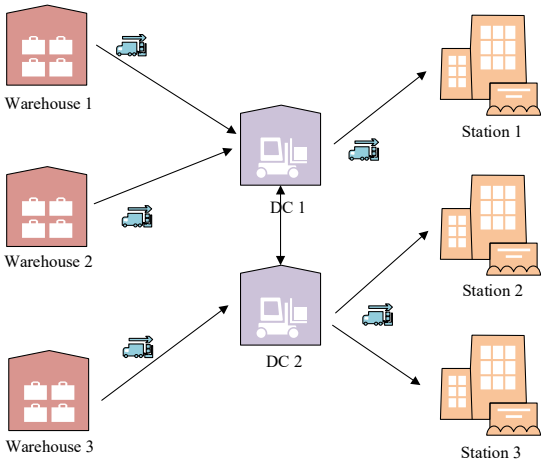
According to the volumes of goods, fulfillment services, and types of goods, different distribution network structures should be designed and continuously improved. Generally, there are three basic types of network structures, as shown in Figure 2. For the distribution network with regional DCs in Figure 2a [17,18], warehouses and DCs are fully connected and each delivery station can only be served by one DC. In the case of the distribution network shown in Figure 2b with fully covered DCs, warehouses are allowed to connect specific DCs, while each DC can connect to all delivery stations. The last type of structure shown in Figure 2c implements the idea of the consolidation of goods to the maximum extent. In this situation, warehouses and DCs do not need to supply all their downstream facilities, but they are required to ensure that demand can always be met by transfer transportation between DCs. It should be noted that there are no extra connections (in comparison to the basic skeleton model) in most analytical studies, while multiple coverages are allowed in practice [19–21].



(a) Distribution network with regional DCs



(b) Distribution network with fully-covered DCs



(c) Distribution network with Lateral-Transshipments in DCs

Figure 2. Different types of distribution networks.

2.2. Distribution Network Design Problem and Related Models

The distribution network design (DND) problem is concerned with the decisions regarding the number of facilities and their optimal locations and facility capacity allocation [22–27]. It can also be viewed as a variation of the facility location-allocation or the production-distribution problem, which are reviewed in this subsection.

The coordination of distribution network design and planning has been addressed by some studies, which aim to integrate a number of different layers of supply chain (SC). For example, [28] designed an integrated model considering production and distribution functions in a two-echelon system on a just-in-time (JIT) basis. Focusing on the production process in supply chain, [29] constructed a continuous flexible process network model to maximize the operating profit. Ref. [30] set up a bi-objective model to minimize costs and the delay for the JIT delivery in a three-echelon supply chain. To provide a system-level optimized supply network, [19] attempted to solve the problems jointly in the entire supply network, including network design, production quota assignment, production planning, capacity planning for various facilities, and distribution planning. Ref. [31] investigated the effectiveness of production and distribution integration by a computational study under different logistic environments. In [32], the configuration of a production and distribution network needs to be optimized; there are two types of constraints (namely operational and financial constraints) in the model. Considering process uncertainty and robustness, [33] studied the design and planning of supply chain networks involving multi-echelon, multi-product, and multi-period situations. Ref. [17] presented mathematical models to optimize inventory control and facility locations for a four-echelon supply chain network. With environmental considerations, some studies designed more realistic production-inventory models [34,35]. Ref. [21] developed a nonlinear mixed-integer programming model to optimize multi-echelon sustainable production-distribution supply networks under carbon emission policies. Aiming to increase the total value of a company by configuring and controlling all parts of a supply chain, [36] proposed a three-echelon, multi-commodity, and multi-period model for tactical and strategic decision-making. To ensure a responsive and resilient supply chain network, [37] addresses a multi-period supply chain (SC) network design problem where customer demands depend on the delivery lead-times of the facilities serving them. Interested readers are further referred to a number of excellent review papers on integrating production and distribution in supply chain management [23,38–42].

2.3. Solution Algorithms

In multi-echelon distribution network design and planning problems, the combined multi-layer decisions could dramatically increase the size of the search space, especially when additional real-world factors (the capacity, costs, uncertainty, etc.) are included. As a result, the design of algorithms for real-life instances has been challenging for practical applications and theoretical research. Given its inherent multi-layer structure, it is beneficial to adopt decomposition methods that can simplify the original problems and solve them efficiently. The Lagrangian relaxation-based method [43] is one of the most frequently used decomposition techniques. For instance, [44] used Lagrangian relaxation to decompose a location-inventory problem and provide a lower bound rule, then the branch-and-bound algorithm was used to obtain feasible solutions. To solve a distribution planning problem, [45] presented a Lagrangian substitution-based solution approach to transform the original nonlinear model into a mixed-integer linear programming model with univariate (solvable) concave models. Ref. [46] also developed a Lagrangian relaxation solution framework to decompose the network design problem into closely related knapsack and time-dependent least cost path problems. Although there are other decomposition-oriented applications in the field of supply chain management [18,19,28,47], theoretically rigorous and computationally reliable decomposition techniques are much needed for different complex scenarios [21]. Comparatively, commonly used heuristic methods aim to find a close-to-optimal solution rapidly in integrated production-distribution problems. To name a few: adapted imperialist competitive algorithm (AICA), variable neighborhood search (VNS) algorithm [22], genetic algorithm (GA) [28,30], and ant colony (AC) algorithm [24]. Ref. [35] utilized the VNS, Tabu Search (TS), Keshtel Algorithm (KA), Water Wave Optimization (WWO), and Particle Swarm Optimization (PSO) to solve a tri-level location-allocation model for forwarding/reverse supply chain systems. Furthermore, dif-

ferent commercial software packages are also developed to solve production-distribution planning problems [2,17,32,48].

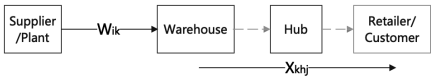
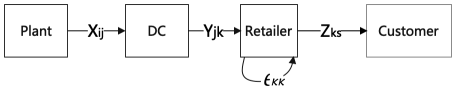
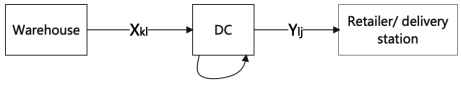
Table 1 compares key modeling components in some closely related literature. Most multi-echelon DND models in the existing literature follow a linear structure in which lateral-transshipments within the same echelon is not considered [22,33] or there is no single sourcing strategy [6]. In the work of [17], they used a general linearization technique in quadratic assignment problems to address the quadratic term in the objective function. Our proposed approach is developed from the perspective of quadratic assignment problems, with more emphasis on the problem decomposition scheme and branch-and-bound algorithms with domain-specific lower bound rules. Table 2 describes the key differences between this study and two highly related papers [6,17] in terms of model structure and solution methods.

Table 1. Comparison of key modeling components in some closely related literature.

Publication	Number of Echelons	Model	Problem Decomposition Schemes	Solution Algorithms
[28]	2	MIP, linear	Lagrangian relaxation	LR
[19]	7	MIP, linear	Sequential; Lagrangian relaxation	LR; GA
[31]	2	MIP, linear	Two-phase heuristic	LS
[30]	3	MIP, linear	-	GA
[32]	3	MIP, linear	-	CPLEX solver
[18]	3	0–1 IP, linear	Lagrangian relaxation	LR
[17]	4	CQMIP, nonlinear	-	CPLEX solver
[22]	9	MIP, linear	-	AICA; VNS
[33]	3	MIP, linear	-	CPLEX solver; Simulation
[35]	4	MIP, nonlinear	Nested approach	VNS; TS; PSO; KA; WWO
[20]	3	MIP, nonlinear	Sequential	Heuristic
[6]	4	MIP, linear	Sequential	Heuristic; GAMS
This paper	3	0–1 IP, nonlinear	Two-stage decomposition via cost estimation	BB; ALNS

Model: CQMIP—conic quadratic mixed-integer programming; IP—integer programming; MIP—mixed-integer programming. Solution algorithms: LR—lagrangian relaxation; GA—genetic algorithm; AICA—adapted imperialist competitive algorithm; VNS—variable neighborhood search; LS—local search; TS—tabu search; PSO—particle swarm optimization; KA—keshtel algorithm; WWO—water wave optimization; BB—branch-and-bound.

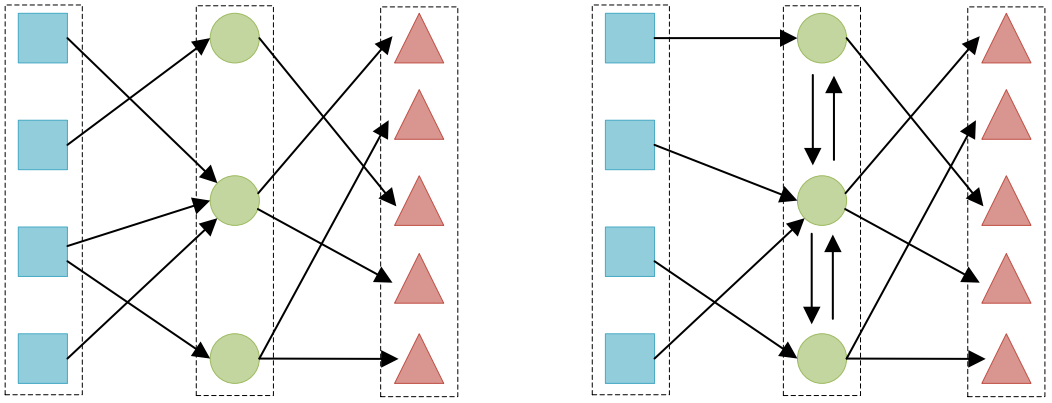
Table 2. Comparison of solution methods in some highly related papers.

Publication	Network Planning Strategy	Decomposition/Linearization Method	Cost Propagation	Illustration of Key Decision Variables
[17]	single sourcing, lateral-transshipments and direct shipment are prohibited	by introducing a new binary variable M_{ikhj} to linearize the binary quadratic terms $W_{ik}X_{khj}$ in the objective function, then solve it using the CPLEX 12 solver	-	
[6]	multiple sourcing, lateral-transshipments and direct shipment are allowed	convert the supply chain network to a bipartite graph and divide the problem into three main sub-problems: plant-DCs (X_{ij}), DCs-retailers (Y_{jk}), and retailers-customers (Z_{ks})	solve three sub-problems sequentially ($X_{ij} \rightarrow Y_{jk} \rightarrow Z_{ks}$), the cost of each edge is calculated with respect to the two different end of it and it propagates forward	
This paper	single sourcing, lateral-transshipments are allowed and direct shipments are prohibited	reconstruct the network relationships within DC echelons from the classical quadratic assignment modeling and divide the problem into two stage sub-problems: warehouses-DCs (X_{kl}) and DCs-retailers (Y_{lj})	for each fixed X_{kl}, solve sub-problems of DCs-retailers ($X_{kl} \rightarrow Y_{lj}$); solve sub-problems of warehouses-DCs based the solution of DCs-retailers ($Y_{lj} \rightarrow X_{kl}$) It has both forward calculation and cost backward propagation	

3. Problem Description and Formulation

3.1. Problem Description

This paper aims to design a three-echelon distribution network with "transfer shortcuts" by making allocation decisions between different facilities. For the standard form of multi-echelon SCND, interested readers can refer to the studies by [20,34,47]. In their research, materials or commodities are processed through a range of echelons and only one node in each echelon can be chosen for a process chain (as shown in Figure 3a); the decisions typically are associated with the facility locations and the allocation between different echelons. In this paper, we consider a three-echelon city logistic distribution network, as shown in Figure 3b, with facilities at fixed locations, i.e., warehouses, DCs, and delivery terminals/stations. Commodities are stored in warehouses (origins) and transported to stations (destinations) via DCs. We need to decide on two pairs of assignment relationship (i.e., warehouses-to-DCs and DCs-to-delivery stations) in the city logistics distribution network, which will greatly affect the overall transportation costs, since we further consider transferring options in the same echelon (DCs) in our problem, which could lead to very difficult assignment relationship (warehouses-to-DCs and DCs-to-delivery stations), and is not covered in the standard multi-echelon SCND reformulation.



(a) A traditional multi-echelon network

(b) A city logistic distribution network with transfer shortcuts

Figure 3. Different types of distribution networks.

Without loss of generality, the following assumptions are adopted in the underlining mathematical model:

(1) A single sourcing/single path strategy [17] assumes that each warehouse can only be assigned to a single DC and one station can only be covered by one DC.
(2) Each delivery station has deterministic demand following a known pattern [6,17,49].
(3) Transportation cost is expressed as a piecewise linear function of material flow [50] and it is proportional to the volume of freight flows (i.e., transportation costs equal to the product of distance and freight flows in our problem).

Then, we can state the overall problem as follows: The major input data include the number of warehouses, DCs, stations, locations of facilities, and the connections between DCs. Moreover, other additional given parameters include distances between different facilities, the capacities of DCs, and the demand. The decision variables are related to the assignment of warehouses to DCs and the assignment of DCs to stations. The goal is to minimize the total costs subject to the demand satisfactory constraints under available resources. Table 3 lists the notations used throughout this paper, and the definitions of the decision variables are provided in Table 4.

Table 3. Sets, indices, and parameters used in models.

Symbol	Definition
Sets	
N	Set of nodes in the physical network
E	Set of transportation links in the physical network
E^B	Set of transportation links to be built in the physical network, $E^B \in E$
E^E	Set of transportation links existing in the physical network, $E^E \in E$
K	Set of warehouses, $K \in N$
L, L'	Set of DCs, $L \in N, L' \in N$
J	Set of stations, $J \in N$
Indices	
k	Index of set K, $k \in K$
l	Index of set L, $l \in L$
l'	Index of the set L', $l' \in L'$
j	Index of set J,$j \in J$
Parameters	
d_{kl}	Distance from warehouse k to DC l
c_{kl}	Estimated cost from warehouse k to DC l
$d_{ll'}$	Distance from DC l to DC l'
$d_{l'j}$	Distance from DC l' to station j
f_{kj}	Demand for station j at warehouse k
$cap_{(l,l')}$	Capacity of the DC l

Table 4. Decision variables used in models.

x_{kl}	=1 if assign warehouse k to distribution center l, = 0 otherwise
$y_{l'j}$	=1 if assign station j to the distribution center l', = 0 otherwise

3.2. Network Model Construction for Logistics Network with Shortcuts

Consider a city logistics distribution network, which includes a set of warehouses denoted by K, a set of DCs denoted by L, a set of delivery stations denoted by J, a set of existing transportation links (inter DC links) denoted by E^E, and a set of transportation links to be built denoted by E^B. Let a physical network $G = (N, L)$ be a collection of possible physical network elements, where $N = K \cup L \cup J$, and all available and existing links are denoted by the set $E = E^B \cup E^E$. For the logistics distribution service, the physical network also consists of many origin and destination pairs. An origin is a warehouse with an index of $k \in K$ where commodities originate from, and a destination is a station with an index of $j \in J$ where commodities terminate.

Figure 4 shows an illustrative network in our problem. To address the challenge of cross-layer connections within the same echelon, we apply a network decomposition strategy, i.e., splitting the original echelon L into two new echelons, namely L and L', as shown in Figure 4b. Each pair of nodes in the sets (L, L') with the same index represents a physical location node in the set L. Specifically, the combination of (l_1, l_1') indicates the actual node l_1. Table 5 provides detailed information about the mapping between the original network and the extended network.

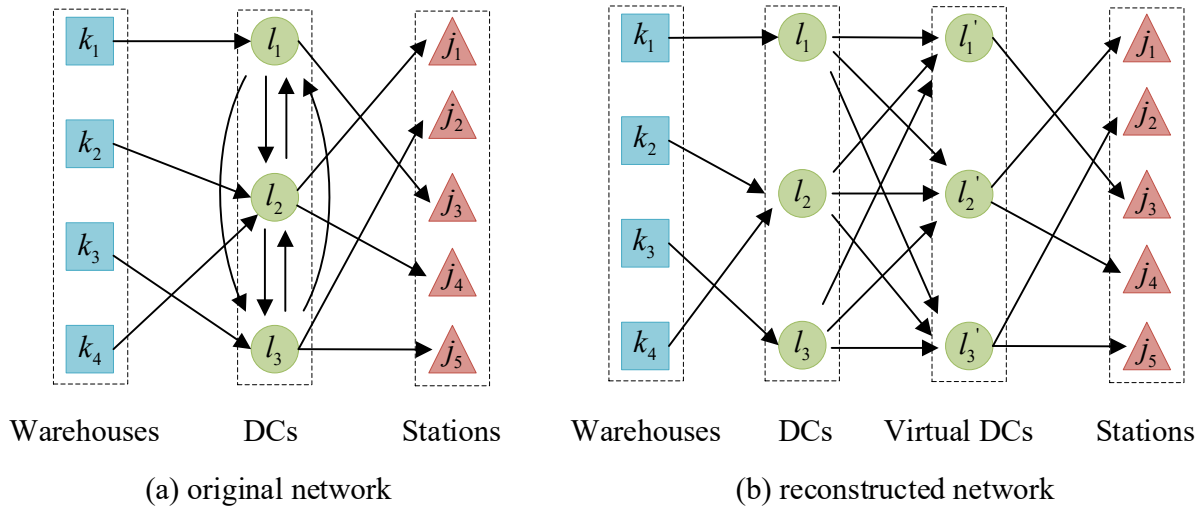


Figure 4. Network reconstruction.

Table 5. Decision variables used in models.

Elements	Original Network	Extended Network	Distance
node	l_1	l_1 and l_1'	$d_{l_1 l_1'} = 0$
node	l_2	l_2 and l_2'	$d_{l_2 l_2'} = 0$
link	(l_1, l_2)	(l_1, l_2')	$d_{l_1 l_2} = d_{l_1 l_2'}$
link	(l_2, l_1)	(l_2, l_1')	$d_{l_2 l_1} = d_{l_2 l_1'}$
path	$k_1 - l_1 - j_1$	$k_1 - l_1 - l_1' - j_1$	$d_{k_1 l_1} + d_{l_1 j_1} = d_{k_1 l_1} + d_{l_1 l_1'} + d_{l_1' j_1}$
path	$k_1 - l_1 - l_2 - j_1$	$k_1 - l_1 - l_2' - j_1$	$d_{k_1 l_1} + d_{l_1 l_2} + d_{l_2 j_1} = d_{k_1 l_1} + d_{l_1 l_2'} + d_{l_2' j_1}$

3.3. Quadratic Semi-Assignment Model M1 with Capacity Constraints

Based on the above expanded network, our problem can be formulated as the following quadratic semi-assignment model with capacity constraints (QSAP-C, M1):

$$\textbf{Min} \sum_{k\in K}\sum_{l\in L}\sum_{j\in J} f_{kj} d_{kl} x_{kl} + \sum_{l\in L}\sum_{l'\in L'}\sum_{k\in K}\sum_{j\in J} f_{kj} d_{ll'} x_{kl} y_{l'j} + \sum_{l'\in L'}\sum_{j\in J}\sum_{k\in K} f_{kj} d_{l'j} y_{l'j} \tag{1}$$

$$\sum_{l\in L} x_{kl} = 1 \;\forall\, k \in K \tag{2}$$

$$\sum_{l'\in L} y_{l'j} = 1 \;\forall\, j \in J \tag{3}$$

$$\sum_{k\in K}\sum_{j\in J} x_{kl} f_{kj} + \sum_{k\in K}\sum_{j\in J} y_{l'j} f_{kj} - \sum_{k\in K}\sum_{j\in J} x_{kl} y_{l'j} f_{kj} \le Cap_{(l,l')} \;\forall\, (l, l') \in (L, L') \tag{4}$$

$$x_{kl} \in \{0,1\} \;\forall\, k \in K, l \in L \tag{5}$$

$$y_{l'j} \in \{0,1\} \;\forall\, l' \in L',\; \forall\, j \in J \tag{6}$$

The objective function in Equation (1) aims to minimize the sum of transportation costs across multiple layers (i.e., from warehouses to DCs and from DCs to delivery stations) and the transfer transportation costs between DCs. As indicated in Assumption (6), transportation costs in this problem are considered as the product of distance and flow volume. Essentially, the flow volume of transfer activities is jointly determined by the two types of allocation decisions x_{kl} and $y_{l'j}$, which clearly leads to a quadratic term $\sum_{l\in L}\sum_{l'\in L'}\sum_{k\in K}\sum_{j\in J} f_{kj} d_{ll'} x_{kl} y_{l'j}$ in the objective function. Constraint (2) then ensures that

one warehouse can only be covered by a single DC. Constraint (3) requires that one station is assigned to exactly one DC. Constraint (4) specifies that the capacity of DCs cannot be exceeded. Constraints (5) and (6) describe two sets of binary variables.

The left-hand-side of constraint (4) is interpreted as the incoming (or outgoing) flow summation for the combination of (l, l'), corresponding to the actual DC l. To be specific, Figure 5 shows a potential feasible solution for the illustrative example shown in Figure 4, which can be further checked to know whether the capacity is sufficient. For DC l_1 (combination of (l_1, l'_1)), the incoming bottleneck should consider the flow on link (k_1, l_1), (l_2, l'_1), and (l_3, l'_1), as shown in Figure 6a. Figure 6b shows the outgoing bottleneck links, i.e., link (l_1, l'_2), (l_1, l'_3), and (l'_1, j_3). As the flow balance for each node in layer L (Figure 7a) and L' (Figure 7b), bottleneck links are links (l_1, l'_1), (l_1, l'_2), (l_1, l'_3), (l_2, l'_1), and (l_3, l'_1) (Figure 8). The term of $\sum_{k\in K}\sum_{j\in J} x_{kl} f_{kj} + \sum_{k\in K}\sum_{j\in J} y_{l'j} f_{kj}$ in Equation (4) covers all flow on bottleneck links but calculates the internal transfer link (link (l_1, l'_1)) twice, where the recalculated flow can be expressed as $\sum\limits_{k\in K}\sum_{j\in J} x_{kl} y_{l'j} f_{kj}$ and we finally obtain constraint (4) by removing this part.

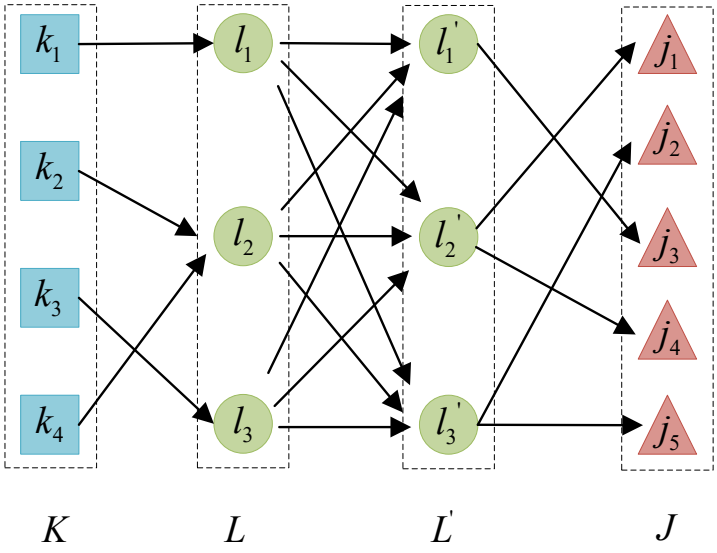


Figure 5. A potential feasible solution for QSAP-C.

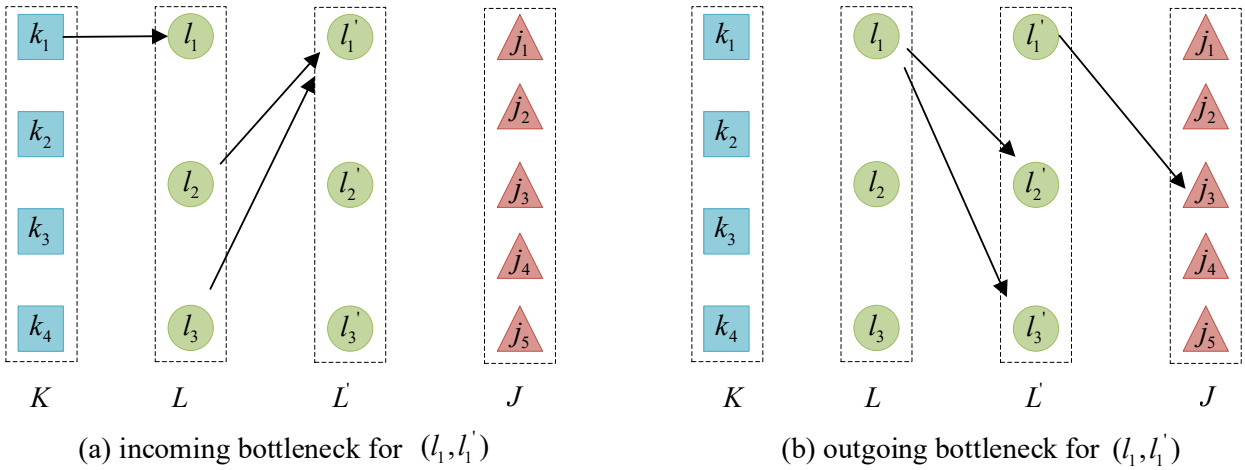


Figure 6. DC l_1 can be a bottleneck as all incoming flow (direct transport + transfers) could exceed outgoing handling capacity.

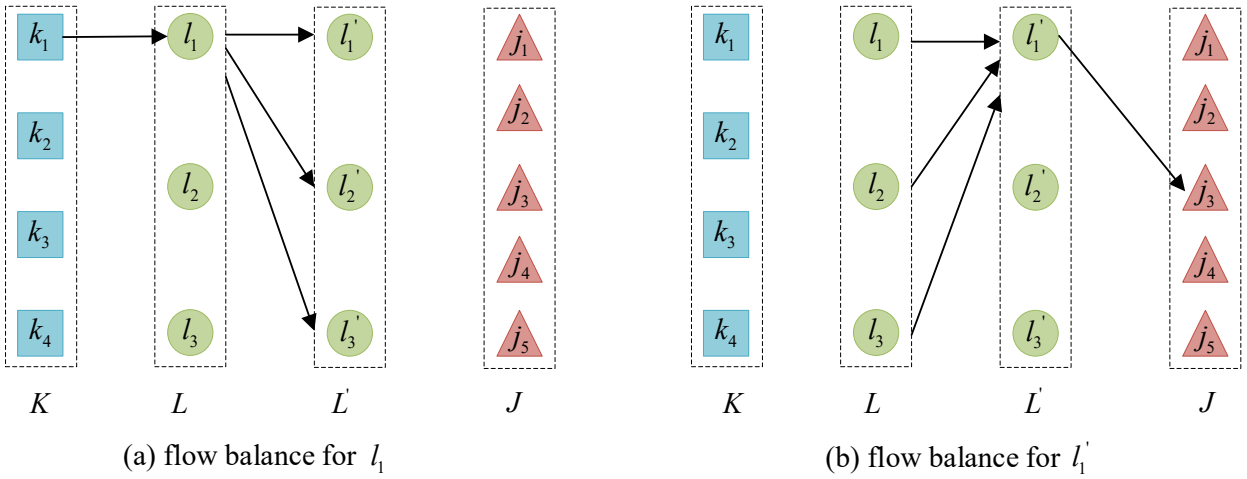


Figure 7. Flow balance for a pair of DC nodes l_1 and l_1'.

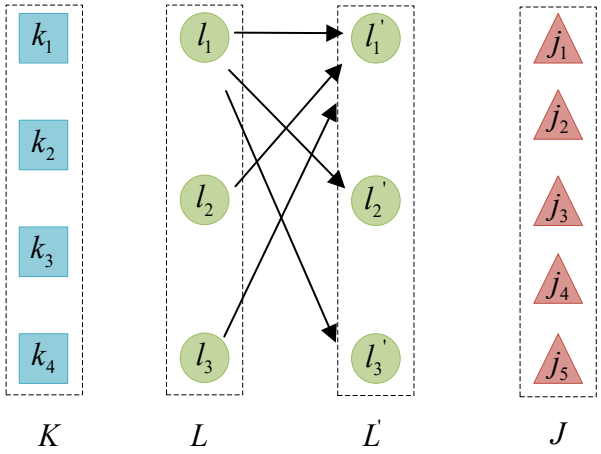


Figure 8. Checking flow balance constraints for incoming/outgoing links for DC l_1.

The proposed formulation differs from the standard quadratic assignment problem (QAP) (e.g., [51]), in terms of its sets of constraints. In the QAP, facilities need to be allocated to the locations such that each facility is allocated to exactly one location, and every location can only be assigned to one facility to minimize the total transportation costs, which are equal to the sum of the products of flows and distances. The connection between standard QAP and QSAP-C formulation is given in Appendix A.1.

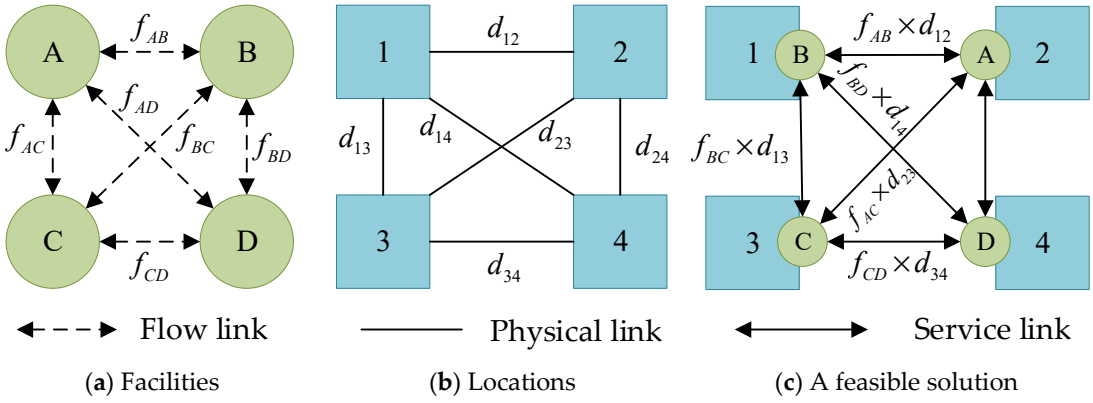
4. Solution Algorithms

Many methods have been developed for solving standard QAP, including a wide range of exact, heuristic, and metaheuristic approaches, and interested readers are referred to the review paper by [52]. Since the standard QAP is NP-hard, high-quality lower bounds are of great importance and typically obtained through implicit enumeration procedures such as branch-and-bound. The Gilmore and Lawler lower bound (GLB) is one of the best-known standard QAP lower bounding rules, due to its simplicity and low computational cost. Aiming to introduce or adapt this classical approach from the standard QAP to the specific problem under consideration, we design a matheuristic approach to obtain "approximate" lower bound estimates through a decomposition scheme with built-in GLB rules. Accordingly, a solution-finding method is developed to convert the obtained lower bound solution to feasible ones as tighter upper bounds. To improve the quality of solutions, we next propose both exact and heuristic algorithms with the embedded GLB lower bound estimator.

4.1. Stage-Wide Problem Decomposition Schemes of GLB

A good lower bound for a combinatorial optimization problem should be tight and easy to compute compared with solving the problem itself [53]. The GLB can easily provide relatively tight lower bounds. The request for decomposition for QAP or other problems such as coupling decisions has the following stated objectives: (1) decouple the decisions and the original problem can be divided into simple subproblems; (2) provide the basis for aggregation techniques and form the backbone of the procedures; (3) the problem can be solved finally by coordinating the solution of linear subprograms. These objectives need to be achieved in absolute terms of course, but also in a pragmatic way if the results of decomposition are expected to have a significant impact on real-world planning and decision-making. This means that the tools and methodology developed must be sufficiently accurate, reliable, and consistent to meet decision making needs.

In this section, we first briefly introduce the solution principle of GLB, of which details can be found in [54]. Figure 9 considers an illustrative example with $|N| = 4$ (where N is defined as set of facilities, index $i, j \in N$, and M are defined as the set of locations, and index $k, l \in M$, $|M| = |N|$), which is adapted from the classical QAP instance of the NEOS (https://neos-guide.org/case-studies/sc/la/qap/qap-of-size-4/, accessed on 7 May 2023). For this example, there are four facilities (Figure 9a) that need to be allocated to four locations (Figure 9b) following the one-to-one assignment constraints. Figure 9c provides a feasible solution, i.e., permutation (A-2, B-1, C-3, D-4). In Figure 10, we would like to highlight the symmetrical two-stage decision feature of this instance, where the first stage aims to determine the locations of the flow-sending facilities and where the second stage is concerned about the locations of flow-receiving facilities.



(**a**) Facilities (**b**) Locations (**c**) A feasible solution

Figure 9. An instance of standard QAP.

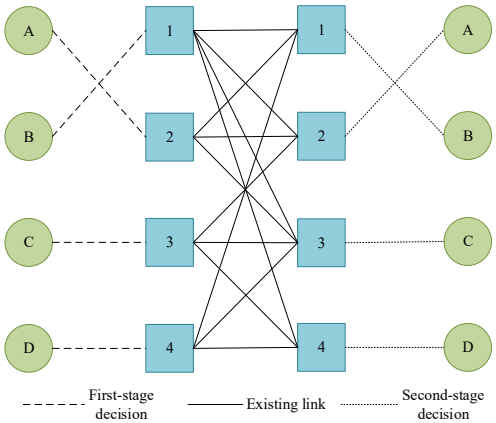
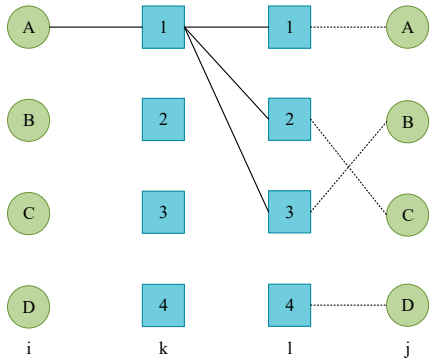
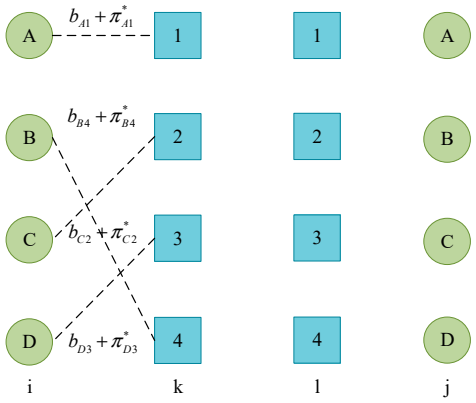


Figure 10. The two-stage decision form of the instance.

From a dynamic programming-oriented perspective, we describe the process of obtaining GLB as three steps: (1) the decisions at the second stage are decomposed into a collection of simpler linear assignment problems (LAPs); (2) sequential solving of each LAP; (3) their solutions are used as the backward cost estimate (BCE) when making first-stage decisions. This three-step process is further illustrated in Table 6.

Table 6. The three-step process of obtaining GLB.

Step	Formula	Illustration
Estimate the second-stage cost by fixing the decision variable x_{ik} in the first stage	Let $x_{ik} = 1$, the optimal solution of following linear assignment problem is defined as π^*_{ik} for each (i,k) $\pi^*_{ik} = \text{Min} \sum_{l=1}^{N} \sum_{j=1}^{N} f_{ij} d_{kl} x_{lj}$ $\sum_{j=1}^{N} x_{lj} = 1 \; \forall \; l = 1, 2, \ldots N$ $\sum_{l=1}^{N} x_{lj} = 1 \; \forall \; j = 1, 2, \ldots N$	
Calculate the total cost c_{ik} for each (i,k) as BCE	$c_{ik} = b_{ik} + \pi^*_{ik}$	-
Solve the first-stage problem based on BCE of c_{ik}	$\text{GLB} = \text{Min} \sum_{i=1}^{N} \sum_{k=1}^{N} (c_{ik} + \pi^*_{ik}) x_{ik}$ $\sum_{i=1}^{N} x_{ik} = 1 \; \forall \; k = 1, 2, \ldots N$ $\sum_{k=1}^{N} x_{ik} = 1 \; \forall \; i = 1, 2, \ldots N$	

As presented in Tables 7 and 8, for the given input data (distance matrix and flows), we detail the calculation steps as shown below.

Table 7. Distance matrix of locations.

	1	**2**	**3**	**4**
1	0	22	53	53
2	22	0	40	62
3	53	40	0	55
4	53	62	55	0

Table 8. Flows of different source-sink pairs.

	A	**B**	**C**	**D**
A	0	3	0	2
B	3	0	0	1
C	0	0	0	4
D	2	1	4	0

Step 1: Calculate the π^*_{ik} for each (i,k) pair.

For example, let $x_{A1} = 1$ then we need to assign three flows (f_{AB}, f_{AC}, f_{AD}) to three distances (d_{12}, d_{13}, d_{14}), as shown in Figure 11. The objective is to find the minimum value of the sum of products of flows and distances. The LAP can be simply solved using sorting algorithms [55] to match the minimum flows with the maximum distances. Accordingly, the optimal solution of the LAP for $x_{A1} = 1$ is $\pi^*_{kl} = f_{AB}d_{12} + f_{AC}d_{14} + f_{AD}d_{13} = 22 \times 3 + 53 \times 2 + 53 \times 0 = 172$.

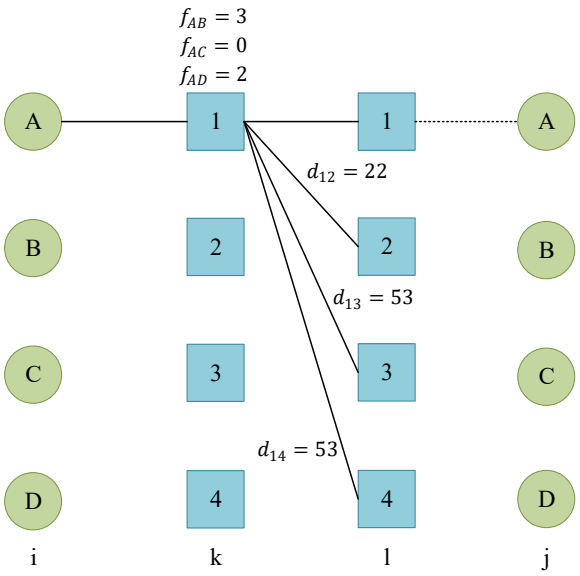


Figure 11. Subproblem of LAP for node pair $(A, 1)$..

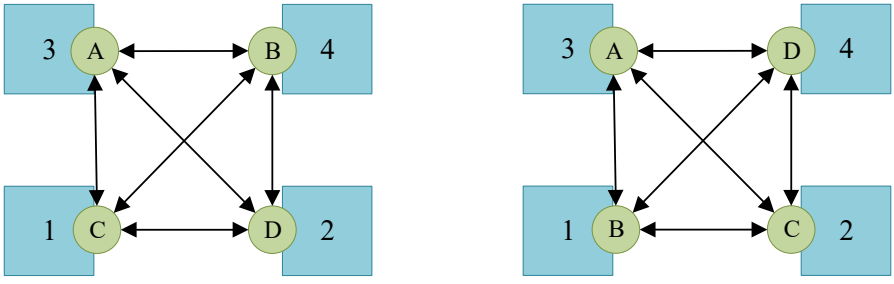
Step 2: For each (i,k) pair, we calculate the total cost c_{ik} for each (i,k).

By using Table 9 with BCE π^*_{ik}, we calculate $c_{ik} = b_{ik} + \pi^*_{ik}$, where $b_{ik} = 0$ in this instance and $c_{ik} = \pi^*_{ik}$.

Table 9. Values of backward cost estimate π^*_{ik} for the first stage.

	1	**2**	**3**	**4**
A	$\pi^*_{A1} = 172$	$\pi^*_{A2} = 22$	$\pi^*_{A3} = 88$	$\pi^*_{A4} = 0$
B	$\pi^*_{B1} = 146$	$\pi^*_{B2} = 22$	$\pi^*_{B3} = 88$	$\pi^*_{B4} = 0$
C	$\pi^*_{C1} = 226$	$\pi^*_{C2} = 40$	$\pi^*_{C3} = 160$	$\pi^*_{C4} = 0$
D	$\pi^*_{D1} = 269$	$\pi^*_{D2} = 53$	$\pi^*_{D3} = 212$	$\pi^*_{D4} = 0$

Step 3: Solve the first-stage problem based on c_{ik} Solve the LAP with cost $c_{ik} = \pi^*_{ik}$, we then obtain the permutation (A-3, B-1, C-2, D-4) with optimum value GLB = 274 and the corresponding objective function value $z = 570$, as shown in Figure 12. For this instance, the optimal solution is (A-3, B-4, C-1, D-2) with value $z^* = 395$.



(**a**) Facilities (**b**) Locations

Figure 12. Illustration of solutions.

4.2. Model Decomposition Scheme for Utilizing Efficient Lower Bound Rules for QSAP-C

We reformulate the primal problem as a number of subproblems of LAP with computationally efficient algorithms by relaxing the capacity constraints of DCs (constraint (4)). The corresponding relaxed model (M2) of the original problem can be stated, as seen below, with its optimal solution defined as Z_R.

M2:

$$Z_R = Min\ Z \tag{7}$$

subject to Equations (2) and (3) and Equations (5) and (6)

Similar to the problem decomposition scheme used in GLB, model M2 can be further decomposed into a sequence of generalized linear assignment problem (GLAP) with an optimal solution π^*_{kl} for each $(k,l) \in E^B$. The main idea of the decomposition is to fix the variable x_{kl} (assume that $x_{kl} = 1$) and then solve the problem related to $y_{l'j}$ to obtain the second-stage cost. Thus, we can establish the following equations for each $(k,l) \in E^B$.

M3:

$$\pi^*_{kl} = \text{Min} \sum_{l' \in L'} \sum_{j \in J} f_{kj} d_{ll'} y_{l'j} + \sum_{l' \in L'} \sum_{j \in J} f_{kj} d_{l'j} y_{l'j} = \text{Min} \sum_{l' \in L'} \sum_{j \in J} f_{kj} \left(d_{ll'} + d_{l'j} \right) y_{l'j} \tag{8}$$

subject to Equations (3) and (6)

These equations constitute a generalized assignment model M3, where the cost for each $(l', j) \in E^B$ is $f_{kj}\left(d_{ll'} + d_{l'j}\right)$. Figure 13 illustrates a feasible solution with $x_{k1l1} = 1$ in the network as in Figure 4. When $k = k1$ and $l = l1$, a subnetwork can be obtained and the links $(k1, l1)$, $(l1, l'1)$, $(l1, l'2)$, $(l1, l'3)$ are the existing links, the delivery stations J need to be assigned to exactly one distribution center. By solving these GLAPs (M3), the second-stage cost π^*_{kl} for each $(k,l) \in E^B$ is now obtained.

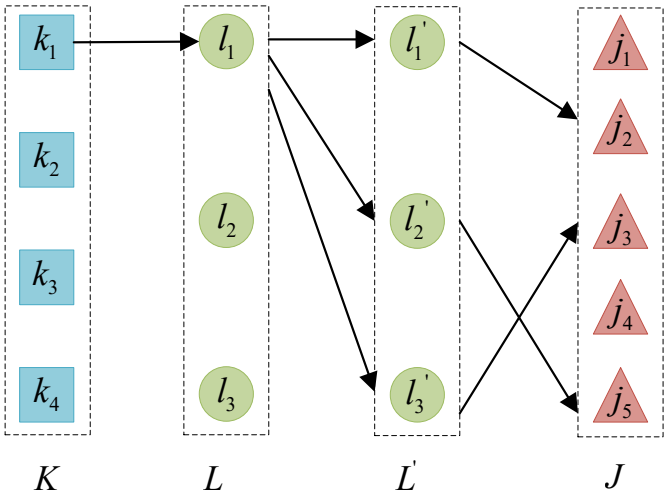


Figure 13. A feasible solution for $x_{k1l1} = 1$.

Based on the transportation cost $\sum_{j \in J} f_{kj} d_{kl}$ on the link (k,l) (the first-stage cost) and second-stage cost π^*_{kl}, the estimated cost for links (k,l) becomes $c_{kl} = \sum_{j \in J} f_{kj} d_{kl} + \pi^*_{kl}$. Then, we can solve the following generalized assignment model (M4) (Figure 14) to retrieve a lower bound of the relaxed model M2. The overall solution algorithm for obtaining this lower bound estimate is described in Algorithm 1.

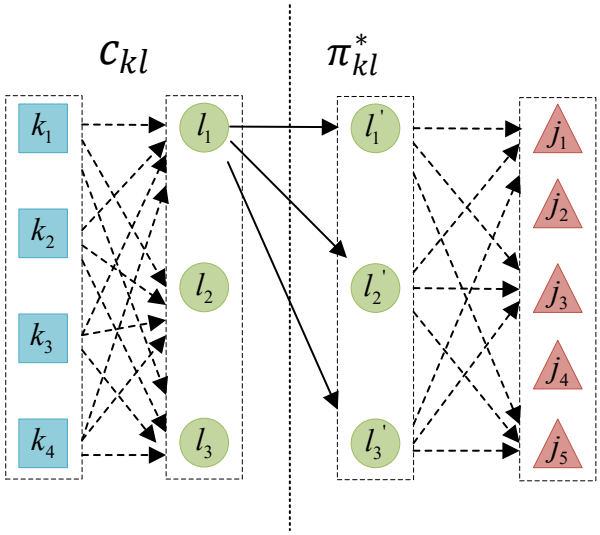


Figure 14. Illustration for the generalized assignment model (M4).

M4:

$$Z_{LB} = \text{Min} \sum_{k \in K} \sum_{l \in L} \left(\sum_{j \in J} f_{kj} d_{kl} + \pi^*_{kl} \right) x_{kl} = \text{Min} \sum_{k \in K} \sum_{l \in L} c_{kl} x_{kl} \tag{9}$$

subject to Equations (2) and (5)

Algorithm 1. GLB-based algorithm for obtaining the lower bound of the simplified problem(M2) without enforcing the capacity constraints of DCs.

Step 1: Initialization
Read the input data, including nodes, links, and agents (flow)
Step 2: Calculate the subsequent cost given the decision of x_{kl}
For each link$(k, l) \in E^B$
solve the generalized assignment problem(M3) by calling the Gurobi solver and obtain the value of π^*_{kl}
Step 3: Calculate the estimated cost c_{kl} on the link $(k, l) \in E^B$
For each link$(k, l) \in E^B$

$$c_{kl} = \sum_{j \in J} f_{kj} d_{kl} + \pi^*_{kl}$$

Step 4: Solving the generalized assignment problem by optimizing x_{kl}
According to the estimated cost c_{kl}, solve the generalized assignment problem (M4) by calling the Gurobi solver and retrieve the lower bound estimate

4.3. Connection between Traditional GLB for Standard QAP and Proposed Lower Bound Estimation Process for QSAP-C

Compared with the GLB for standard QAP, we use the same cost-estimating framework to obtain the lower bound for QSAP-C. In the QAP, problems at two stages are LAPs and the cost is fixed as constant. In the QSAP-C, we need to solve GLAPs at two different stages and the flow-dependent cost is the product of distances and commodity flows. For the cost backward propagation or cost estimating process, optimal solutions of second stage subproblems (LAPs or GLAPs) are used to guide the decision at the first stage. We further use Table 10 to list the mapping between GLB for standard QAP and the proposed lower bound estimates for QSAP-C.

Table 10. A comparison of obtaining lower bounds in QAP and QSAP-C.

		GLB for Standard QAP	**Lower Bound Estimation in QSAP-C**
First stage decision	problem type	LAPs	GLAPs
	cost	fixed cost independent of flows	the product of distances and flows
Second stage decision	problem type	LAPs	GLAPs
	cost	fixed cost independent of flows	the product of distances and flows
Cost backward propagation		Optimal solutions of the LAPs in second stage decision	Optimal solutions of the GLAPs in second stage decision

4.4. Computation of Upper Bound Solutions

By solving the relaxation problem (M2) using Algorithm 1, we can obtain the lower bound which gives the values of the decision variables x_{kl}, denoted as x^*_{kl}. To obtain the upper bound of the original problem (M1), let $x_{kl} = x^*_{kl}$ for each $(k,l) \in E^B$ and we can convert the original problem to the following GLAPs (M5) with capacity constraints. By solving the model M5, one can fetch a feasible solution to the original problem.

M5:

$$\sum_{k\in K}\sum_{l\in L}\sum_{j\in J} f_{kj}d_{kl}x^*_{kl} + \sum_{l\in L}\sum_{l'\in L'}\sum_{k\in K}\sum_{j\in J} f_{kj}d_{ll'}x^*_{kl}y_{l'j} + \sum_{l'\in L'}\sum_{j\in J}\sum_{k\in K} f_{kj}d_{l'j}y_{l'j} \tag{10}$$

subject to Equations (3) and (6)

$$\sum_{k\in K}\sum_{j\in J} x^*_{kl}f_{kj} + \sum_{k\in \mathrm{K}}\sum_{j\in J} y_{l'j}f_{kj} - \sum_{k\in \mathrm{K}}\sum_{j\in J} x^*_{kl}y_{l'j}f_{kj} \leq Cap_{(l,l')} \;\forall\; (l,l') \in (L,L') \tag{11}$$

4.5. A Branch-and-Bound Algorithm Combined with Enhanced Lower Bound Rules

In this subsection, we present a customized branch-and-bound algorithm for the model M1; the flowchart of the algorithm is shown in Figure 15.

The algorithm first generates the initial temporary lower bound lb by calling Algorithm 1 and obtains the initial upper bound ub using the rule described in Section 3.2, then the initial solution can be marked as the one in the root node. A best-lower-bound-first search strategy is adopted in the algorithm, that is, when executing the branching process on the pending node of the enumeration tree, we always choose the node with the smallest lower bound value. To implement this strategy, we use a priority queue to store the search nodes and they will be sorted by the lower bound value. In the initialization stage of the algorithm, the root node is placed in an empty priority queue. For large-scale instances, it is difficult to enumerate all nodes in a limited computational time. Therefore, we also define the solution gap tolerance η and the maximum number of enumerating iterations n as the terminal conditions.

The algorithm starts selecting nodes from the priority queue and executes the branch-and-bound procedure until the queue becomes empty. We choose the top one in the priority queue each iteration, update the best global lower bound LB, and calculate the gap between LB and UB. Then, the algorithm checks if the termination conditions are met, i.e., whether the gap is less than or equal to η or the number of iterations exceeds n. In the branching process, we assign the active station j of the current node to the DC and accordingly create child nodes for every feasible assignment of station j to each DC. In this process, the proposed Algorithm 1 is called to calculate the temporary lower bound lb and the temporary upper bound ub for each child node. For children nodes, if their temporary lower bounds are greater than UB they will be pruned, otherwise, they will be pushed into the priority queue. The detailed algorithm execution process is summarized in Algorithm 2, and Figure 15 provides the flow chart of the customized branch-and-bound algorithm.

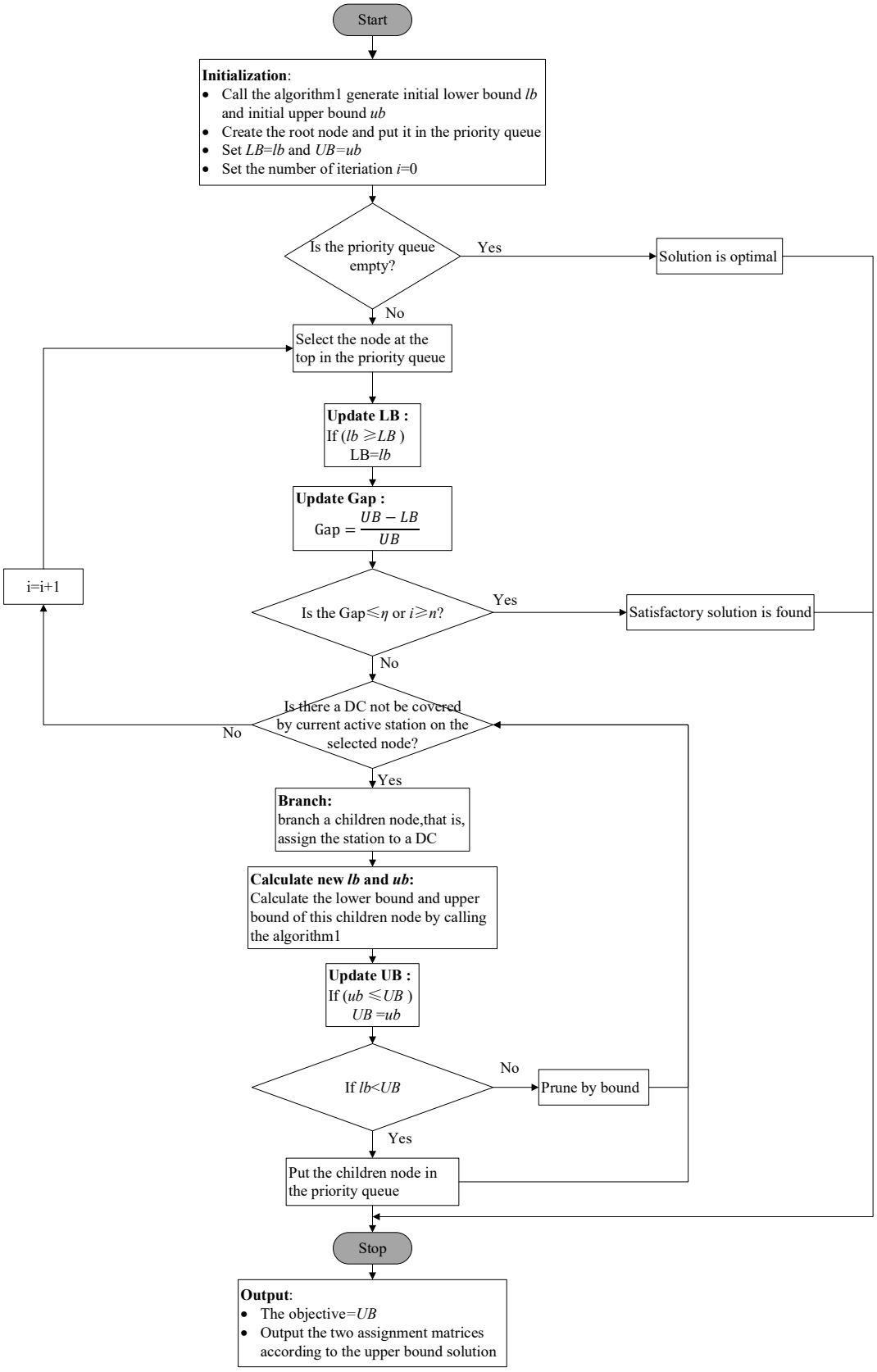


Figure 15. Flowchart of the customized branch-and-bound algorithm.

Algorithm 2. A branch-and-bound algorithm combined with heuristic lower bound.

Step 1: Initialization
Step 1.1: Obtain the initial solution
Call Algorithm 1 to obtain the initial lower bound lb and initial upper bound ub
Set the $LB = lb$, $UB = ub$, $i = 0$
Step 1.2: Initialization for the root node and priority queue
Create an empty priority queue pq, initialize a root node according to the lower bound, push the root node into the priority queue pq
Step 2: Branch-and-bound process
While the pq is not empty:
Step 2.1 Select the top node from the priority queue pq and update the lower bound
Pull the top node from the priority queue pq, defined as active_node
If active_node.$node_{lb}$>LB: # update the best global lower bound and the Gap
LB = active_node.$node_{lb}$
Gap = $\frac{UB-LB}{UB}$
If Gap $\leq \eta$ or $i \leq n$: # check the termination conditions
Break
If active_node.$node_{active\_station} + 1 < |J|$: # update the active station
active_node.$node_{active\_station}$ = active_node.$node_{active\_station} + 1$
Else: # all delivery stations have been assigned
Continue
Step 2.2 Branch on the live node
Set temporary matrices
$temp\_node_{assign\_matrix}$ = active_node.$node_{assign\_matrix}$
$temp\_node_{budget\_matrix}$ = active_node.$node_{budget\_matrix}$
try to assign the $active\_station$ to all possible DCs
For each DC $l \in L$:
Create the children node and inherit the properties of the live node
Step 2.3 Calculate the lower bound and upper bound of the children node
Call Algorithm 1 to obtain the lower bound lb and upper bound ub
update the best global upper bound
If children_node.$node_{ub}$<UB:
UB = children_node.$node_{lb}$
If children_node.$node_{lb} < UB$:
Push the children node into the priority queue
otherwise this children node will be pruned

4.6. An Adaptive Large Neighborhood Search Algorithm with Efficient Initial Solution

It is well known that exact methods are often able to find good solutions at early stages of the search (in this paper, initial solutions' average GAP obtained by branch-and-bound algorithm is less than 10% and the acquisition time is shorter than 2 s) but employ a lot of time to marginally improve that solution or prove its optimality. This behavior suggests a classical way to design a heuristic algorithm: adopt the initial good solution generated by the branch-and-bound algorithm; then, explore the neighbors to improve the solution quickly. [55] pointed out that even using local search to obtain a feasible solution of classical quadratic assignment problems is difficult, and the time complexity is exponential for simple neighborhood structures. Since the adaptive large neighborhood search (ALNS) algorithm can explore larger searching space [56], this heuristic algorithm is integrated to improve the initial solution.

The large neighborhood search algorithm includes two types of operators: destroy operator and repair operator. The solution is destructed and reconstructed using these operators, which are selected by a roulette-wheel method according to their previous performance in each iteration of ALNS algorithm. For QSAP-C model, some warehouses or stations are removed from its current assigned DC in the destruction step, leading to a partial solution. This partial solution is repaired in the reconstruction step, which focuses on the unassigned warehouses or stations. After the reconstruction phase, a feasible solution

is obtained. The roulette statistics are updated and iteration continues until the termination condition is met. The pseudocode is displayed in Algorithm 3.

Algorithm 3. An adaptive large neighborhood search algorithm with efficient initial solution.

Step 1: Initialization
Step 1.1: Obtain the initial solution
Call Algorithm 2 to obtain the initial feasible solution in the time limit T
Step 1.2: Initialization for ALNS
Given the total iteration num Q, initial weight of operators $w(h)$, reaction factor ρ
Step 2: ALNS
For i = 1 . . . Q do
(1) select a destroy operator d by the roulette-wheel procedure, destroy current solution and obtain the set of removed nodes
(2) select a repair operator r by the roulette-wheel procedure, repair and obtain the new_solution
(3) update the number of times the operator has been used, $u(d) += 1, u(r) += 1$
(4) calculate the objective function of new_solution, if it is infeasible a large constant value will be given.
If new_solution_obj <= current_solution_obj
update current solution, current_solution_obj = new_solution_obj
If current_solution_obj <= best_solution_obj
update the best solution, best_solution = current_solution
update the success of operators, $s(d) += \delta_1, u(r) += \delta_1$
Else
update the success of operators, $s(d) += \delta_2, u(r) += \delta_2$
update the weight of selected operators,
$$w(d) = (1-\rho)w(d) + \rho\frac{s(d)}{u(d)}, w(r) = (1-\rho)w(r) + \rho\frac{s(r)}{u(r)}$$
If current solution keeps the same for M consecutive iterations
break
Return the best solution

4.6.1. Initial Solution

We prefer to adopt the GLB-based initial solution in the proposed customized branch-and-bound algorithm. Although, in some instances, it is hard to obtain a feasible initial solution or the problem is infeasible because of the unreasonable parameters of DCs' capacity. We also set a time limit for obtaining the initial feasible solution; if the running time is exceeded, a random assignment will be generated.

4.6.2. Solution Destruction

In the destruction phase, we put forward four different removal methods to maintain the diversity during the searching process. Each removal method aims to remove a predefined number of warehouses or stations from current assignment. We denote the removed nodes as unassigned nodes.

(a) Random-remove

This operator is very simple but important for exploring larger search space [57], the main idea is to randomly choose warehouses or stations as target nodes, then remove them. In this paper, we remove N/2 target nodes, where N means the total number of warehouses and stations.

(b) Worst-remove

This worst-remove is adapted from the worst-remove proposed by [56]. The main idea is to remove the nodes that have a larger impact on the solution quality. Specifically, the Equation (1) is calculated before and after removal, the larger difference it is, the higher probability that considered nodes (denoted as worst nodes) are removed.

(c) Break-remove

Inspired by the work of [58], where they design break-remove operator in the ALNS to solve the gate assignment problem. This operator is carried out in two steps, firstly, we sort all the DCs by the average cost $c_l, l \in L$ in descending order. c_l is calculated as the sum of transportation cost related to DC l. Secondly, we select the DC with maximum average cost and all the nodes (warehouses or stations) assigned to the selected DC are removed.

(d) Exchange

This operator can be regarded as a special type of destroy operator, the solution can be updated without using repair operator. In this paper, we adopt the two exchange and can be further divided into three situations: exchange for warehouses, exchange for stations, and both. In the algorithm, they are three types of separate destroy operators. Take the exchange for warehouses as an example, two warehouses are selected, if they have different DCs then exchange them. Otherwise, reassign them to other DCs.

4.6.3. Solution Reconstruction

In the reconstruction phase, we propose three reconstruction operators to insert the unassigned nodes.

(a) Random-insert

Similar to the random-remove, this operator means inserting the unassigned nodes to DCs randomly to generate new solution.

(b) Greedy-insert

This greedy-insert operator is adopted from the greedy heuristic proposed by [56]. The idea is to calculate the lowest insertion cost $c_{kl}/c_{l'j}$ for all removed nodes, and then insert the one with smallest $c_{kl}/c_{l'j}$. This process continues until all unassigned nodes have been repaired or none of them can be inserted.

(c) Regret-insert

Based on basic greedy insertion, this operator incorporates look-ahead information by considering not only the unassigned node with lowest insertion cost, but also some other promising nodes. To illustrate this strategy, let $l = 3$ (there are three DCs) and the insertion cost of warehouse k to three DCs in ascending order as c_{k1}, c_{k2}, c_{k3}. Then, the regret cost of $(k, 1)$ is calculated as $\sum_{l=1,2,3} c_{kl-}c_{k1}$. In each iteration, we calculate the regret cost for all remaining nodes and insert the nodes with the highest regret cost. This process repeats until all unassigned nodes have been repaired or none of them can be inserted.

4.6.4. Weight Adjustment

The selection of operators in each iteration is based on their weights then a roulette-wheel procedure is applied to select the operator to generate the neighborhood. Let $D = \{d_i | i = 1, \ldots, m\}$ be the set of m destroy operators and let $R = \{r_j | j = 1, \ldots, n\}$ be the set of n repair operators. The initially equal weights of the heuristics are denoted by $w(d_i)$ and $w(r_j)$, so that the probabilities to select the operator are $p(d_i) = \frac{w(d_i)}{\sum_{i=1}^{m} w(d_i)}$ and $p(r_j) = \frac{w(r_j)}{\sum_{j=1}^{n} w(r_j)}$.

For the weight adjustment, we adopt the strategy in [57]. Where $w(h)$ denotes the weight of a heuristic operator h, $u(h)$ denotes the number of times the heuristic h has been used, and $s(h)$ denotes the success of h. The success $s(h)$ of h is initialized with zero and is increased by δ_i if the resulting solution is of the corresponding quality: δ_1- the new solution is the best one found thus far; δ_2- the new solution improves the current solution. We need to ensure the inequality $\delta_1 > \delta_2$ and use a reaction factor $0 \leq \rho \leq 1$ controls the influence of the recent success of a heuristic on its weight. Finally, we calculate the weights in each iteration by $w(h) = (1 - \rho)w(h) + \rho \frac{s(h)}{u(h)}$ when $u(h) > 0$.

5. Numerical Experiments

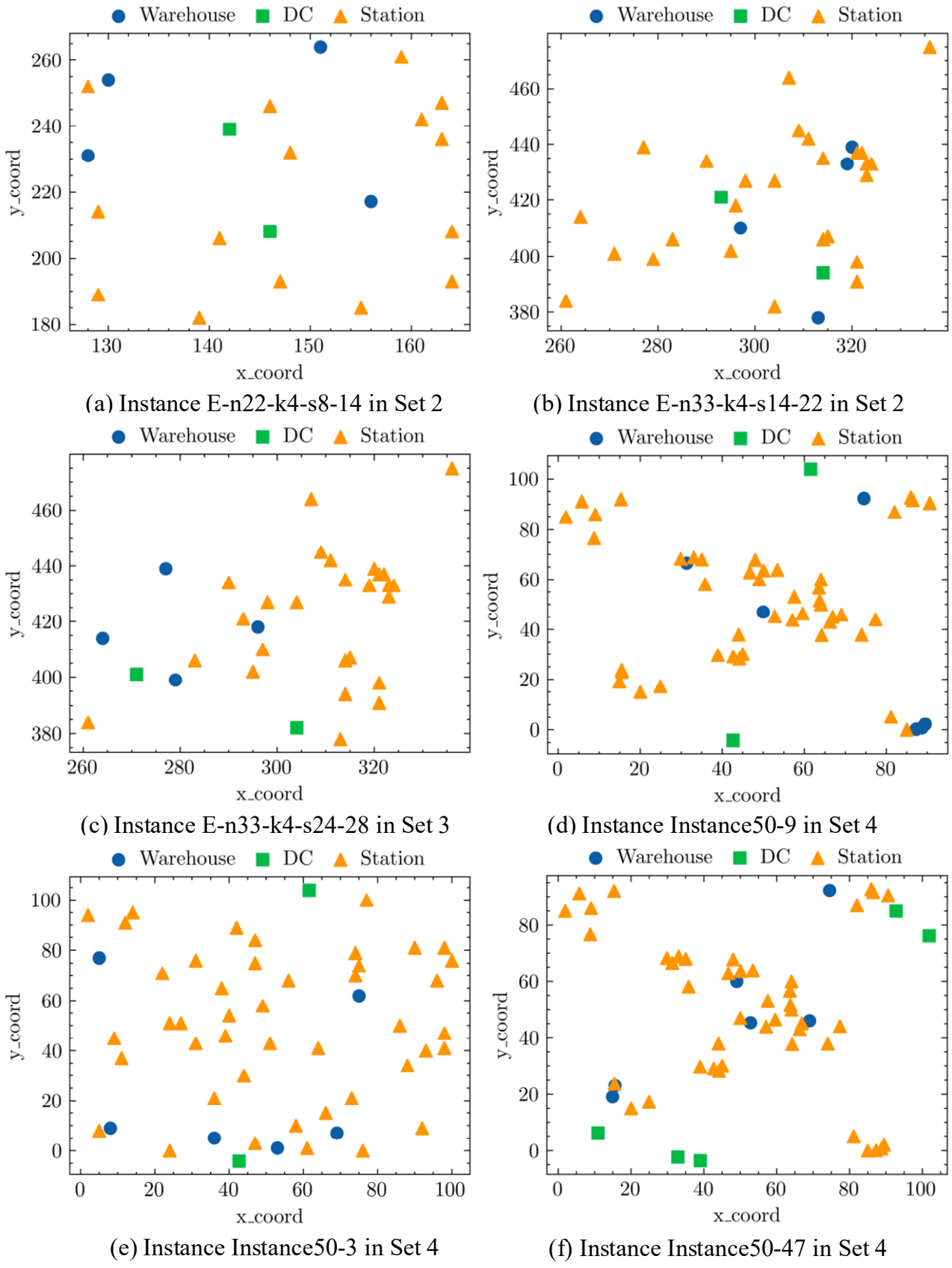
To examine the behavior of our proposed models and algorithms, numerical experiments of the QSAP-C model are reported in this section. In Section 5.1, small- and medium-sized instances are generated according to benchmarks for the 2E-CVRP in the literature [59]. Section 5.2 aims to present the computational results on a wide set of test instances and analysis of the results to identify the instance characteristics. Section 5.3 shows the converge process and sensitive analysis. Then, Section 5.4 provides main insights on algorithm analysis for the classical dataset. Section 5.5 tests a large-scale case using the real-world city logistics distribution network of JD logistics and we evaluate the effectiveness of the proposed algorithm. Finally, Section 5.6 provides the managerial implications for dynamical evaluation of cross-layer service synchronization. The algorithms have been implemented in Python and run on an Intel (R) Core (TM) i7-8550U, 1.8 GHz, and 8 GB of RAM. The github site for this paper can be found at https://github.com/zqNiu/qap, accessed on 7 May 2023.

5.1. Base Instance Sets Generation and Description

In this section, we introduce different instance sets for the QSAP-C model. There are 93 instances in total and are grouped in three sets. Although [59] developed four sets of instances for the 2E-CVRP, coordinates of nodes in the first set are not provided (given distances matrices) and they are not allowed to generate corresponding instances for our problem because of the lateral transshipment. Therefore, following three dataset names notation in [59], the three instance sets are Set 2, Set 3, and Set 4. In the original dataset, Set 2 comprises 12 small-sized instances with 21 or 32 customers and nine medium-sized instances with 50 customers, small-sized instances have two satellites and medium-sized instances have four satellites. Set 3 is similar to Set 2 but with more realistic distributed satellites [59]. There are 54 instances in Set 4 with 50 customers and five satellites are generated from [60], which can represent different scenarios in city logistics. Next, we create the corresponding instance of QSAP-C for each instance of 2E-CVRP.

5.1.1. Locations of Warehouses, DCs, and Delivery Stations

The locations of DCs keep the same with satellites in original instances and we randomly select nodes from customer nodes as warehouses. For Set 2 and Set 3, we locate four warehouses, randomly in all instances, and regard unselected customer nodes as delivery stations. Therefore, there are four warehouses, two DCs and 17 or 28 delivery stations in small-size instances, and four DCs and 46 delivery stations in medium-size instances; Set 4 has 50 customer nodes and we randomly place six warehouses instead of two. Figure 16 shows some instance configurations in different sets and the name of instance keeps the same with [59].



(a) Instance E-n22-k4-s8-14 in Set 2

(b) Instance E-n33-k4-s14-22 in Set 2

(c) Instance E-n33-k4-s24-28 in Set 3

(d) Instance Instance50-9 in Set 4

(e) Instance Instance50-3 in Set 4

(f) Instance Instance50-47 in Set 4

Figure 16. Some instance configurations.

5.1.2. Demand between Warehouses and Delivery Stations

Demands of the customer nodes are given in the 2E-CVRP, which are combined in origin (warehouses)–destination (delivery stations) pairs for our problem. In this subsection, three rules corresponding to different scenarios are provided to generate demand between warehouses and delivery stations. Random Nodes rule (RN), the demand of each delivery station is fully supplied by only one warehouse and the warehouse is chosen randomly. Equal Proportion rule (EP), the demand of each delivery station is supplied by all warehouses with equal proportions. Random Proportion rule (RP), the demand of each delivery station is supplied by all warehouses with random proportions.

5.1.3. Capacity of DC

The total capacity of all DCs can be calculated as $Cap_{DCs} = m_2K_2$ according to parameters in the 2E-CVRP [59], where m_2 indicates number of the 2^{nd}-level vehicles and K_2 indicates capacity of the vehicles for the 2^{nd} level. For baseline cases, the capacity of each DC is $\frac{Cap_{DCs}}{Numberof DCs}$.

We summarize the main features of the different sets for baseline cases in Table 11. The first column shows the instance set and the number of instances is reported in Column 2. Column 3, Column 4, and Column 5 specify the number of warehouses, DCs, and delivery stations, respectively. Column 6 provides the capacity of each DC. Demand for all baseline instances is generated by adopting the EP rule.

Table 11. A summary of the main features of the different sets for baseline cases.

Set	Number of Instances	Number of Warehouses	Number of DCs	Number of Stations	Capacity of Each DC
2	6	4	2	15	12,000
2	6	4	2	26	16,000
2	3	4	4	42	500
2	6	4	2	44	1000
3	6	4	2	15	12,000
3	6	4	2	26	16,000
3	6	4	2	44	1000
4	18	6	2	44	15,000
4	18	6	3	44	20,000
4	18	6	5	44	24,000

5.2. Overall Computational Results

In this section, we present the results of the tests in Set 2, Set 3, and Set 4. The results of the model on each set are summarized in Tables 12–14. Each table contains the instance name and the four selected warehouses in columns 1 and 2. For Set 2 and Set 3, the optimal solution OPT by Gurobi is reported in the third column and Column 4 contains the time in seconds needed to solve the instances, while Columns 5, 6, and 7 present the final solution BEST_SOL1 by BB, the gap between BEST_SOL1 and OPT, and the running time of obtained BEST_SOL1. We also provide the initial solution INIT_SOL by BB, the corresponding gap and running time in Columns 8, 9, and 10. Finally, Columns 11, 12, and 13 present the final solution BEST_SOL2 by ALNS, the corresponding gap and running time. Moreover, "-" indicates the instance is infeasible and cannot report related information, "*" means the result is showed by keeping two decimal places.

Table 12. Results of dataset Set 2.

Gurobi						BB				ALNS		
Instance	**Warehouses ID**	**OPT**	**T_G (s)**	**BEST_SOL1**	**GAP1**	**T_B (s)**	**INIT_SOL**	**GAP2**	**T_B_I (s)**	**BEST_SOL2**	**GAP3**	**T_A (s)**
Set 2												
E-n22-k4-s10-14	3, 11, 13, 20	-	0.44	-	-	142.24	-	-	1.07	-	-	1.95
E-n22-k4-s11-12	2, 8, 15, 17	-	0.27	-	-	141.90	-	-	0.02	-	-	1.91
E-n22-k4-s12-16	2, 4, 9, 17	1,034,673.25	0.17	1,034,673.25	0	97.48	1,073,847.25	0.04	0.36	1,034,673.25	0	1.14
E-n22-k4-s6-17	7, 16, 19, 21	1,015,276.50	0.31	1,015,276.5	0	168.92	1,248,893.50	0.23	33.21	1,015,276.50	0	2.06
E-n22-k4-s8-14	1, 3, 11, 12	-	0.28	-	-	144.06	-	-	0.02	-	-	1.92
E-n22-k4-s9-19	2, 8, 13, 15	-	0.28	-	-	139.90	-	-	0.02	-	-	1.97
E-n33-k4-s1-9	3, 5, 18, 19	-	0.30	-	-	281.21	-	-	0.02	-	-	2.5
E-n33-k4-s14-22	5, 17, 23, 32	-	0.34	-	-	261.08	-	-	0.03	-	-	2.52
E-n33-k4-s2-13	8, 12, 19, 21	-	0.41	-	-	260.54	-	-	0.03	-	-	2.48
E-n33-k4-s3-17	14, 20, 25, 27	-	0.42	-	-	318.96	-	-	0.05	-	-	2.66
E-n33-k4-s4-5	13, 15, 27, 31	-	0.42	-	-	304.99	-	-	0.03	-	-	2.5
E-n33-k4-s7-25	5, 11, 12, 20	-	0.27	-	-	260.56	-	-	0.03	-	-	2.41
E-n51-k5-s11-19-27-47	20, 26, 35, 49	31,225.74	8.86	33,026.46	0.06	2772.59	36,200.85	0.16	0.17	35,362.62	0.13	9.86
E-n51-k5-s11-19	2, 13, 23, 46	38,424.49	0.16	38,424.49	0	374.50	41,906.84	0.09	0.11	39,643.09	0.03	3.26
E-n51-k5-s2-17	5, 12, 15, 42	38,123.45	0.16	38,123.45	0	3.71	40,924.43	0.07	0.06	38,158.67	0	3.16
E-n51-k5-s2-4-17-46	3, 5, 32, 33	32,179.54	2.00	34,922.4	0.09	2481.72	35,839.62	0.11	0.13	35,278.47	0.1	13.88
E-n51-k5-s27-47	5, 31, 42, 48	29,960.37	0.14	29,960.37	0	3.46	33,184.19	0.11	0.09	30,060.14	0	3.1
E-n51-k5-s32-37	15, 16, 18, 30	55,265.48	0.14	55,265.48	0	4.04	59,857.78	0.08	0.05	55,571.24	0.01	3.94
E-n51-k5-s4-46	17, 23, 34, 47	45,900.66	0.16	49,731.34	0.08	425.31	50,887.61	0.11	0.04	46,930.39	0.02	2.52
E-n51-k5-s6-12-32-37	9, 20, 27, 35	35,067.50	1.78	36,834.41	0.05	2771.15	37,583.28	0.07	0.16	36,850.91	0.05	8.93
E-n51-k5-s6-12	3, 15, 21, 48	29,194.02	0.19	29,744.63	0.02	425.92	29,962.32	0.03	0.11	29,918.82	0.02	2.41
Average			0.83		0.02	561.15		0.1	1.71/0.55		0.03	3.67

Table 13. Results of dataset Set 3.

Gurobi						BB				ALNS		
Instance	**Warehouses ID**	**OPT**	**T_G (s)**	**BEST_SOL1**	**GAP1**	**T_B (s)**	**INIT_SOL**	**GAP2**	**T_B_I (s)**	**BEST_SOL2**	**GAP3**	**T_A (s)**
Set 3												
E-n22-k4-s13-14	2, 7, 12, 19	1,094,569	0.38	1,107,942	0.01	150.92	1,107,942	0.01	0.09	1,094,569	0	2.2
E-n22-k4-s13-16	6, 7, 10, 17	-	0.28	-	-	147.93	-	-	0.02	-	-	1.84
E-n22-k4-s13-17	9, 10, 15, 18	-	0.26	-	-	151.17	-	-	0.02	-	-	1.88
E-n22-k4-s14-19	2, 10, 18, 21	-	0.33	-	-	143.38	-	-	0.02	-	-	1.96
E-n22-k4-s17-19	7, 9, 10, 13	1,430,406.5	0.53	-	-	144.55	-	-	0.02	-	-	1.94
E-n22-k4-s19-21	14, 16, 18, 20	1,145,212	0.2	1,145,212	0	173.45	1,145,212	0	0.03	1,145,212	0	1.02

Table 13. *Cont.*

Instance	Gurobi			BB						ALNS		
	Warehouses ID	OPT	T_G (s)	BEST_SOL1	GAP1	T_B (s)	INIT_SOL	GAP2	T_B_I (s)	BEST_SOL2	GAP3	T_A (s)
						Set 3						
E-n33-k4-s16-22	4, 11, 20, 28	-	0.35	-	-	265.54	-	-	0.03	-	-	2.55
E-n33-k4-s16-24	5, 14, 19, 25	-	0.42	-	-	263.49	-	-	0.03	-	-	2.64
E-n33-k4-s19-26	11, 17, 22, 28	1,268,675.12	0.62	1,268,675.12	0	294.66	1,268,675.12	0	0.04	1,268,675.12	0	1.7
E-n33-k4-s22-26	2, 16, 18, 20	-	0.38	-	-	281.88	-	-	0.02	-	-	2.5
E-n33-k4-s24-28	15, 27, 29, 30	-	0.33	-	-	230.3	-	-	0.03	-	-	2.77
E-n33-k4-s25-28	6, 13, 27, 29	-	0.38	-	-	257.84	-	-	0.03	-	-	2.59
E-n51-k5-13-19	12, 16, 25, 26	29,506.77	0.29	29,508.19	0 *	461.45	29,578.3	0 *	0.06	29,536.02	0	4.21
E-n51-k5-13-42	10, 15, 20, 29	34,614.3	0.18	34,614.3	0	29.75	35,484.11	0.03	0.07	34,614.3	0	2.78
E-n51-k5-13-44	11, 31, 43, 49	33,231.61	0.18	33,231.61	0	2.55	42,098.91	0.27	0.05	33,231.61	0	2.78
E-n51-k5-40-42	17, 19, 33, 35	45,554.18	0.2	49,147.94	0.08	461.9	50,786.8	0.11	0.04	45,554.18	0	3.14
E-n51-k5-41-42	14, 24, 44, 47	48,968.88	0.15	48,968.88	0	0.11	48,968.88	0	0.07	48,968.88	0	2.97
E-n51-k5-41-44	20, 21, 23, 46	63,776.34	0.19	69,598.45	0.09	460.5	69,653.59	0.09	0.04	64,706.92	0.01	3.2
Average			0.31		0.02	217.85		0.06	0.04		0 *	2.48

OPT—optimal solution obtained by Gurobi; T_G—solving time of Gurobi; BEST_SOL1—final solution obtained by BB; GAP1—GAP between BEST_SOL1 and OPT; T_B—solving time of obtaining final solution by BB; INIT_SOL—initial solution obtained by BB; GAP2—GAP between INIT_SOL and OPT; T_B_I—solving time of obtaining initial solution by BB; BEST_SOL2—final solution obtained by ALNS; GAP3—GAP between BEST_SOL2 and OPT; T_A—solving time of obtaining final solution by ALNS."*" means the solution is optimal.

Table 14. Results of dataset Set 4.

Instance	Gurobi			BB						ALNS		
	Warehouses ID	OPT	T_G (s)	BEST_SOL1	GAP1	T_B (s)	INIT_SOL	GAP2	T_B_I (s)	BEST_SOL2	GAP3	T_A (s)
						Set 4						
Instance50-1	8, 11, 23, 28, 32, 40	-	0.69	-	-	441.68	-	-	0.09	-	-	45.39
Instance50-2	4, 13, 22, 27, 29, 34	2,353,038.69	1.17	2,354,456.53	0	600.14	2,354,456.53	0	0.07	2,354,456.03	0	41.89
Instance50-3	10, 11, 29, 36, 45, 50	-	0.72	-	-	438.56	-	-	0.06	-	-	45.58
Instance50-4	5, 12, 34, 37, 41, 46	2,052,019.54	2.21	2,065,752.61	0.01	600.09	2,073,126.27	0.01	0.06	2,066,013.86	0.01	43.32
Instance50-5	10, 17, 24, 26, 46, 52	-	0.81	-	-	487.83	-	-	0.06	-	-	46.22
Instance50-6	10, 18, 23, 24, 37, 42	2,531,530.61	3.04	2,558,792.49	0.01	600.06	2,558,792.49	0.01	0.08	2,558,077.13	0.01	39.77
Instance50-7	8, 21, 24, 26, 42, 49	-	0.69	-	-	518.98	-	-	0.07	-	-	45.75
Instance50-8	4, 9, 28, 34, 41, 47	2,334,898.41	4.61	2,334,918.51	0	600.1	2,334,918.51	0	0.07	2,334,898.41	0	28.87
Instance50-9	3, 26, 36, 49, 50, 51	-	0.91	-	-	448.86	-	-	0.06	-	-	47.08
Instance50-10	8, 15, 17, 25, 38, 44	2,414,882.45	13.5	2,414,976.65	0	469.67	2,414,976.65	0	0.06	2,414,882.45	0	47.57
Instance50-11	12, 23, 24, 33, 41, 43	-	0.52	-	-	411.43	-	-	0.06	-	-	44.67
Instance50-12	7, 18, 19, 20, 31, 35	2,352,275.41	1.33	2,432,949.17	0.03	600.1	2,432,949.17	0.03	0.06	2,403,460.87	0.02	34.59

Table 14. *Cont.*

Gurobi						BB				ALNS		
Instance	**Warehouses ID**	**OPT**	**T_G (s)**	**BEST_SOL1**	**GAP1**	**T_B (s)**	**INIT_SOL**	**GAP2**	**T_B_I (s)**	**BEST_SOL2**	**GAP3**	**T_A (s)**
						Set 4						
Instance50-13	9, 28, 31, 35, 43, 44	-	0.66	-	-	553.71	-	-	0.05	-	-	46.38
Instance50-14	23, 24, 27, 28, 38, 45	2,660,451.97	1.22	2,660,451.97	0	533.23	2,664,045.12	0	0.07	2,664,045.12	0	28.48
Instance50-15	13, 27, 31, 32, 43, 50	-	1.12	-	-	501.76	-	-	0.05	-	-	45.65
Instance50-16	3, 5, 19, 27, 41, 45	2,671,580.82	3.02	2,671,580.82	0	600.09	2,671,580.82	0	0.07	2,671,580.82	0	32.57
Instance50-17	8, 20, 22, 30, 34, 48	-	1.13	-	-	594.8	-	-	0.05	-	-	44.77
Instance50-18	12, 26, 34, 35, 40, 52	2,185,894.49	1.83	2,271,866.64	0.04	600.05	2,350,154.92	0.08	0.08	2,350,154.92	0.08	26.28
Instance50-19	4, 21, 34, 35, 37, 38	3,095,738.16	33.97	3,833,050.17	0.24	600.03	-	-	0.15	3,160,491.46	0.02	120.5
Instance50-20	19, 21, 22, 23, 24, 46	2,006,202.56	12.03	2,362,593.33	0.18	600.19	2,449,572.61	0.22	0.12	2,449,572.61	0.22	40.61
Instance50-21	4, 24, 25, 32, 41, 47	-	596.02	-	-	600.19	-	-	0.09	-	-	59.56
Instance50-22	10, 12, 19, 40, 42, 45	2,193,589.3	16.38	2,375,657.27	0.08	600.23	2,387,727.99	0.09	0.12	2,386,309.46	0.09	88.34
Instance50-23	6, 16, 29, 30, 34, 41	3,010,242.22	38.41	3,016,038.26	0	600.04	3,027,444.15	0.01	0.15	3,022,099.76	0	56.93
Instance50-24	10, 25, 34, 38, 45, 52	2,042,939.28	4.14	2,500,959.97	0.22	600.07	2,502,531.35	0.22	0.13	2,087,650.13	0.02	113.4
Instance50-25	4, 26, 34, 42, 47, 50	-	26.98	-	-	600.22	-	-	0.12	-	-	57.86
Instance50-26	15, 30, 37, 39, 41, 42	2,503,005.74	600.06	-	-	600.3	-	-	0.12	2,517,170.74	0.01	123.27
Instance50-27	11, 12, 13, 31, 39, 44	3,578,475.15	600.16	-	-	600.14	-	-	0.1	3,968,476.71	0.11	95.59
Instance50-28	9, 16, 34, 42, 46, 49	2,279,228.62	600.07	2,458,660.18	0.08	600.21	2,458,660.18	0.08	0.13	2,458,660.18	0.08	37.22
Instance50-30	5, 16, 17, 21, 49, 52	2,202,457.04	14.19	2,564,826.69	0.16	600.17	2,772,129.95	0.26	0.15	2,244,602.5	0.02	117.82
Instance50-31	10, 25, 30, 33, 34, 36	3,538,293.13	600.17	-	-	600.09	-	-	0.12	-	-	60.54
Instance50-32	21, 25, 32, 33, 44, 46	1,878,546.46	30.07	2,029,022.3	0.08	600.08	2,191,026.53	0.17	0.12	2,191,026.53	0.17	43.87
Instance50-33	8, 9, 30, 31, 39, 51	3,291,317.32	43.08	-	-	600.29	-	-	0.11	3,398,041.71	0.03	95.7
Instance50-34	5, 17, 18, 21, 38, 48	2,333,071.7	600.09	2,724,473.28	0.17	600.37	2,724,473.28	0.17	0.13	2,724,473.28	0.17	38.86
Instance50-35	5, 16, 27, 35, 38, 40	3,036,353.76	41.89	3,456,148.13	0.14	600.38	-	-	0.11	3,241,485.67	0.07	84.8
Instance50-36	14, 22, 26, 41, 48, 52	2,440,068.72	600.1	2,805,978.48	0.15	600.41	3,086,533.95	0.26	0.13	2,490,056.29	0.02	113.62
Instance50-37	7, 14, 24, 29, 39, 51	3,360,291.88	29.09	-	-	600.03	-	-	0.28	-	-	107.01
Instance50-38	30, 34, 39, 44, 47, 48	2,207,816.8	79.89	-	-	600.91	-	-	0.29	2,458,399.73	0.11	223.98
Instance50-39	16, 23, 26, 27, 39, 47	3,780,946.1	102.42	-	-	600.51	-	-	0.3	-	-	108.85
Instance50-40	13, 31, 39, 42, 43, 49	2,044,887.42	117.12	-	-	601.65	-	-	0.31	2,108,119.16	0.03	212.68
Instance50-41	21, 25, 26, 28, 40, 55	3,350,595.12	64.1	3,769,016.92	0.12	601.75	3,769,016.92	0.12	2.26	3,769,016.92	0.12	92.32
Instance50-42	20, 30, 38, 51, 52, 55	2,124,265.87	112.77	2,262,700.15	0.07	600.87	2,262,700.15	0.07	0.33	2,259,353.63	0.06	149.69
Instance50-43	8, 9, 24, 36, 39, 54	3,367,863.47	186.06	-	-	600.72	-	-	0.29	-	-	106.19
Instance50-44	6, 12, 13, 31, 36, 41	2,492,396.07	230.35	-	-	600.31	-	-	0.31	2,598,240.84	0.04	244
Instance50-45	10, 15, 24, 26, 36, 50	3,194,804.7	55.74	-	-	600.95	-	-	0.33	-	-	105.42
Instance50-46	28, 38, 42, 46, 48, 51	2,017,974.23	53.99	-	-	601.43	-	-	0.28	2,232,209.02	0.11	159.64

Table 14. *Cont.*

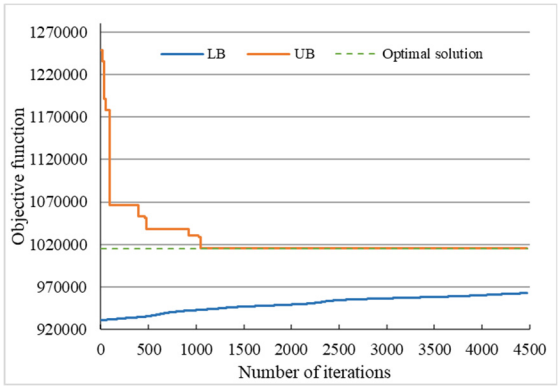
Instance	Gurobi					BB				ALNS		
	Warehouses ID	OPT	T_G (s)	BEST_SOL1	GAP1	T_B (s)	INIT_SOL	GAP2	T_B_I (s)	BEST_SOL2	GAP3	T_A (s)
						Set 4						
Instance50-47	10, 16, 31, 39, 47, 50	3,165,681.32	71.95	-	-	600.76	-	-	0.31	-	-	108.09
Instance50-48	6, 24, 32, 37, 45, 47	1,937,052.99	62.7	2,207,363.15	0.14	600.32	2,240,577.27	0.16	0.58	2,192,519.1	0.13	183.77
Instance50-49	7, 24, 43, 44, 45, 54	3,317,037.06	36.24	-	-	600.87	-	-	0.3	-	-	106.45
Instance50-50	8, 9, 23, 25, 43, 50	1,990,073.91	93.06	2,006,195.21	0.01	600.35	2,016,004.83	0.01	0.34	2,016,004.83	0.01	79.03
Instance50-51	8, 18, 22, 26, 46, 55	2,852,775.41	144.84	-	-	600.67	-	-	0.33	-	-	108
Instance50-52	6, 16, 17, 31, 36, 44	2,201,921.31	31.97	-	-	601.44	-	-	0.34	2,466,307.6	0.12	197.75
Instance50-53	13, 24, 25, 27, 39, 43	3,528,043.81	42.56	-	-	600.25	-	-	0.48	-	-	103.91
Instance50-54	24, 31, 35, 40, 41, 43	1,968,231.5	22.17	-	-	600.58	-	-	0.28	2,308,093.02	0.17	168.76
Average			112.12		0.08	578.13		0.09			0.06	87.21

According to the results of Set 2 and Set 3, branch-and-bound algorithm can obtain optimal solutions for most instances, the gap is less than 9% for some hard instances, and the mean gap is 2%. In the computational time, Gurobi solver (561.15 s for Set 2 and 217.85 s for Set 3) is much more efficient than the branch-and-bound (0.83 s for Set 2 and 0.31 s for Set 3). However, we noticed that the initial solution of branch-and-bound is tight, the mean gaps are 10% and 6%, and average running times are 0.55 s and 0.04 s. For some instances, even initial solutions are optimal (E-n22-k4-s19-21, E-n33-k4-s19-26, E-n51-k5-13-19, and E-n51-k5-41-42). Results indicate the heuristic lower bound is relative loose, even though the initial solution has good quality, branch-and-bound algorithm needs more branches to converge. Based on this behavior, we design the ALNS with the efficient initial solution and results also show the benefits of heuristics from the computational point of view, presenting mean values of 3.67 s and 2.48 s, and average gap of 3% and 0%. For instances E-n51-k5-s11-19-27-47, E-n51-k5-s2-4-17-46, and E-n51-k5-s6-12-32-37 in Set 2, both the solver and branch-and-bound need spend more time than other instances, which indicates the locations of warehouses and DCs can affect the complexity of the model. Moreover, some cases cannot obtain a feasible solution because of unreasonable supply and demand relationship; we explore the influence of demand in the section of sensitive analysis.

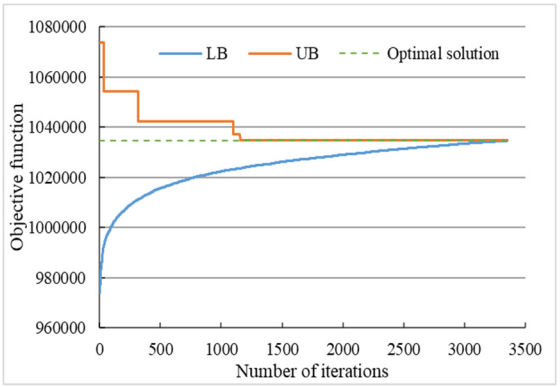
Table 14 presents results of larger instances of Set 4 with six warehouses and 44 stations, where the meaning of the columns is the same as in Table 13, these results show different realistic distribution of both warehouses and stations. As it is difficult to obtain exact solutions for some instances and the global time limit (the solver and branch-and-bound) is set to 600 s. Without losing generality, we also set a limit of 4000 iterations for ALNS. From the results, we can see that Gurobi solver is strong enough to explore exact solutions for most instances and algorithms' mean gaps are larger but still limited to 8% (branch-and-bound) and 6% (ALNS). In terms of computational time, the branch-and-bound algorithm shows time deterioration but ALNS only spends 80% of the solver. Comparatively, the quality of the initial solution obtained by adopting the heuristic lower bound is quite satisfactory (the average gap is equal to 9% and the computational time of all instances is less than 1 s). Moreover, for instances 50-43, 50-45, and 50-47, both branch-and-bound and ALNS cannot even obtain feasible solutions, which confirms points in [54] that even using local search to obtain a feasible solution of classical quadratic assignment problems is difficult, and the heuristic lower bound is critical for problem solving.

5.3. Convergence and Sensitivity Analysis

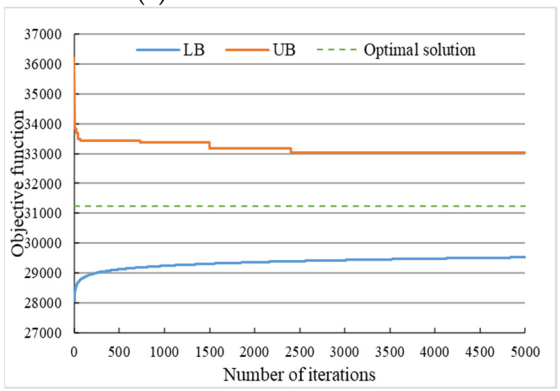
Although a good upper bound can be obtained by adopting the proposed heuristic lower bound in branch-and-bound algorithm, the convergence needs to be judged by the GAP between the upper bound and lower bound. In this section, we analyze the performance of the heuristic lower bound according to the results of Set 2 solved by branch-and-bound algorithm. Figure 17 shows the convergence of some representative cases where the blue solid line represents the lower bound, the yellow solid line represents the upper bound, and the green dotted line represents the optimal solution. For instances E-n22-k4-s6-17 and E-n22-k4-s12-16, the upper bound converges to optimal solution after nearly 1000 iterations, and the lower bound seems relatively loose and converges slowly. The lower bound is tight in instances E-n51-k5-s27-47 and E-n51-k5-s32-37, which equates to optimal solution at the beginning of iterations. For other instances, the lower bound convergence is close to the upper bound. In conclusion, we cannot simply say whether the lower bound is loose or tight, it is loose from the lower bound construction point of view because the capacity constraint is relaxed, but it has different performances in different instances.



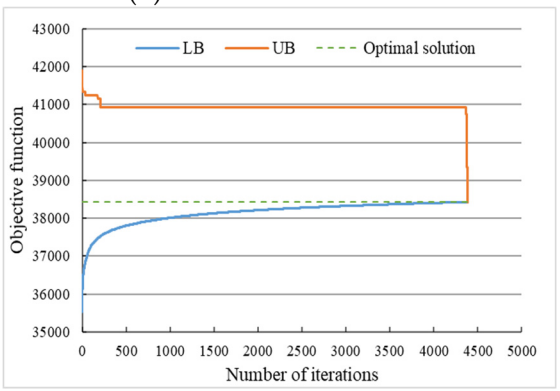
(**a**) Instance E-n22-k4-s6-17



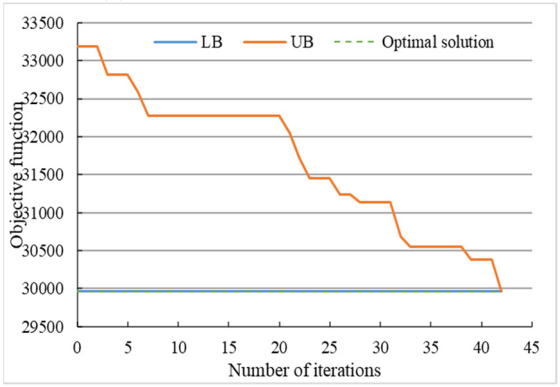
(**b**) Instance E-n22-k4-s12-16



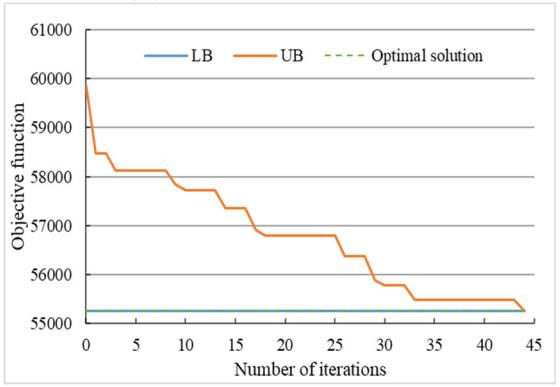
(**c**) Instance E-n51-k5-s11-19-27-47



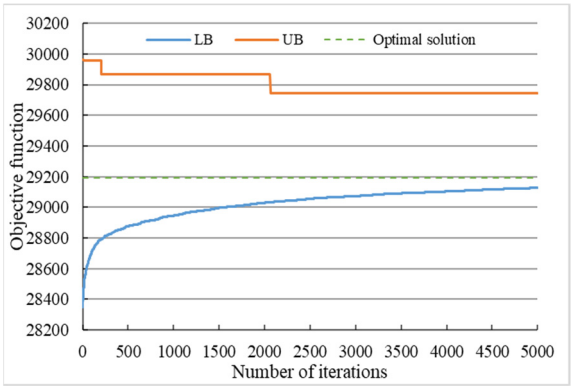
(**d**) Instance E-n51-k5-s11-19



(**e**) Instance E-n51-k5-s27-47



(**f**) E-n51-k5-s32-37



(**g**) E-n51-k5-s6-12

Figure 17. Convergence process of branch-and-bound algorithm.

To analyze the influence of demand pattern, we solve the 54 instances in Set 4 with different demand input. As mentioned in Section 5.1.2, we design three demand generating rules corresponding to different scenarios, which are the Random Nodes rule (RN), The Equal Proportion rule (EP), and the Random Proportion rule (RP). The instances generated by RN are all feasible, and there are some infeasible cases when using EP and RP rules. Figure 18 compares the results of the feasible instances using three generating rules where we focus on the total transportation cost (the histogram) and the proportion of lateral-transshipments volume (the line chart). According to Figure 18, we can see that the lateral-transshipments volume is equal to zero for all instances except for instance 50-50 in RN, which indicates there is no need for lateral-transshipments when the demands of a station all originate from one warehouse. In the real-world scenario, if the station has massive demand for a warehouse, the decision maker will arrange a direct "warehouse-station" route, that is, to directly transport from the warehouse to the station without going through DCs. Figure 15 also shows that lateral-transshipments volume will make total transportation costs higher, and that the transportation costs under the RN rule are always the lowest, followed by RP, and the cost of EP is the highest for most instances. Moreover, lateral-transshipments are effective strategies to reduce transportation costs in certain demand pattern. For instances 50-46, proportions of lateral-transshipments volume under the EP and RP rules are more than 70%, and the value under EP even exceeds 80%.

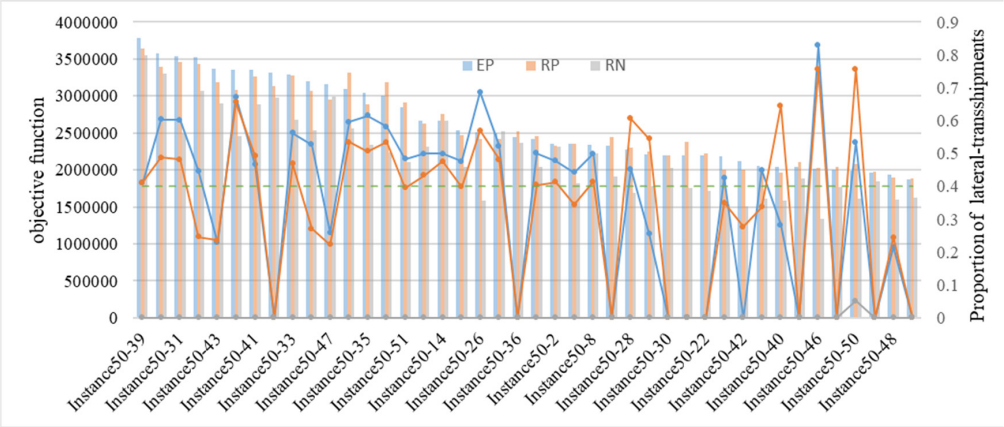


Figure 18. Convergence process of branch-and-bound algorithm.

5.4. Managerial Insights on Analysis of Algorithms

Based on the results of the classical dataset and relevant analysis, the main insights can be summarized as follows:

(1) **By adopting the GLB-oriented lower bound, the initial solution with high quality can be obtained efficiently.** The average GAP between the initial solution and the optimal solution is within 10%, and the computational time is less than 1 s. The GLB-oriented lower bound estimation process comprehensively considers the cost of the previous decision and the subsequent decision, and the computational time is shortened by decomposing the original problem with quadratic terms in both constraints and objective functions into a series of linear generalized assignment problems. Moreover, this method is sensitive to the locations of different types of nodes, and it is difficult to obtain the initial feasible solution for some instances.

(2) **The GLB-oriented lower bound is relatively loose for the original problem. Although the GAP between the initial solution and the optimal solution is usually small, the convergence of branch- and-bound algorithm is slow due to the poor lower bound**. For Set 3, the initial solution of several instances has been the optimal solution, and the branch-and-bound process is mainly to improve the lower bound. However, the convergence process analysis shows that the lower bound is not always loose and needs a case-by-case evaluation.

(3) **The branch-and-bound algorithm combined with the GLB-oriented lower bound is suitable for solving small-scale examples, while the adaptive large neighborhood search algorithm with efficient initial solution can almost achieve the same effect as the solver**. For Set 3, the heuristic algorithm obtains optimal solutions in a relatively short time, and for Set 4, the solution with an average gap of 6% can be obtained within 80% of computational time of the solver. The GLB-oriented lower bound is the main reason to ensure an efficient solution. In some instances, even large neighborhood operations cannot improve the initial solution.

(4) **Although lateral-transshipments will make transportation costs higher, it is still the reasonable choice under specific demand pattern**. In the sensitive analysis of demand, the proportion of lateral-transshipments volume for some instances up to 70%, and the overall proportion of EP and RP strategies is 40%~50%, which indicates the necessity of lateral-transshipments.

5.5. Real-World Instances

We also carry out the real-world case study based on the JD Logistics' distribution network in Beijing. As shown in Figure 19, we choose a subset of facilities from the real-world network as the input data, where 40 warehouses store commodities and 10 DCs provide urban distribution services for 100 delivery stations. The demand of stations is 1000 times of the volume of goods in the original dataset. The demand between warehouses and stations is generated according to the EP principle. The capacity of each DC is evenly configured according to the total demand. The distances of links are specified using to the real-world road network.

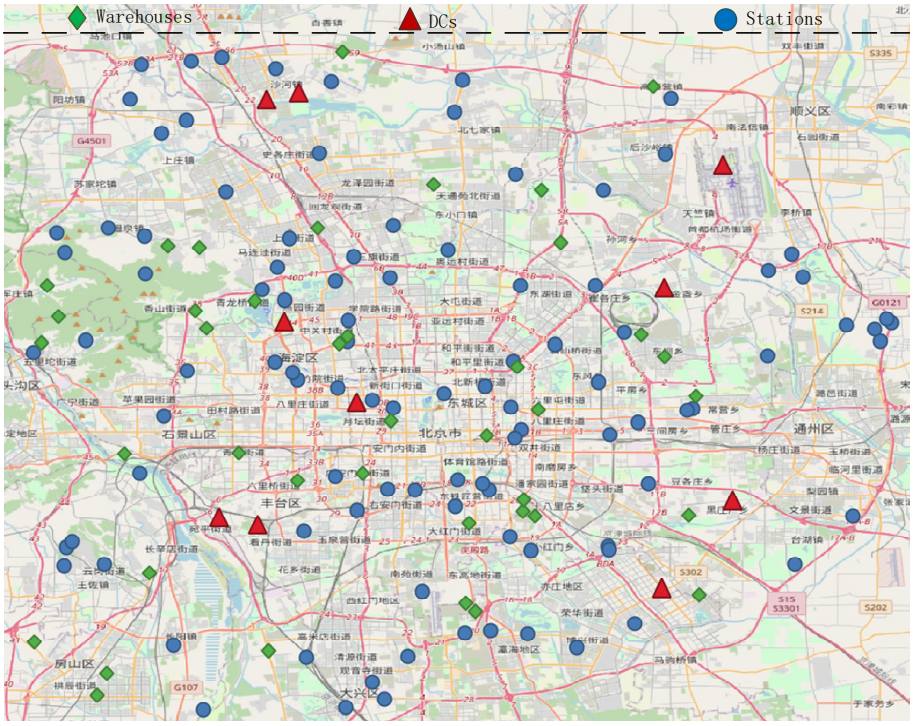


Figure 19. Warehouses, DCs, and delivery stations in the network.

For this real-world case, the Gruobi solver could not obtain a feasible solution within a time limit of 5 h. ALNS with an efficient initial solution finally converged in 18 min. Figure 20 shows the convergence curve of the solution. As the initial solution of the algorithm is obtained by adopting the GLB-oriented lower bound, the heuristic algorithm proposed in this paper can roughly estimate the quality of the solution. In this case, the GLB-oriented lower bound is 11,120.16, the initial upper bound is 142,226.97, and the initial GAP is 21%. The final solution obtained when the algorithm terminates is 13,402.43. If

still compared with the initial lower bound, the GAP of the final solution should be less than 17%.

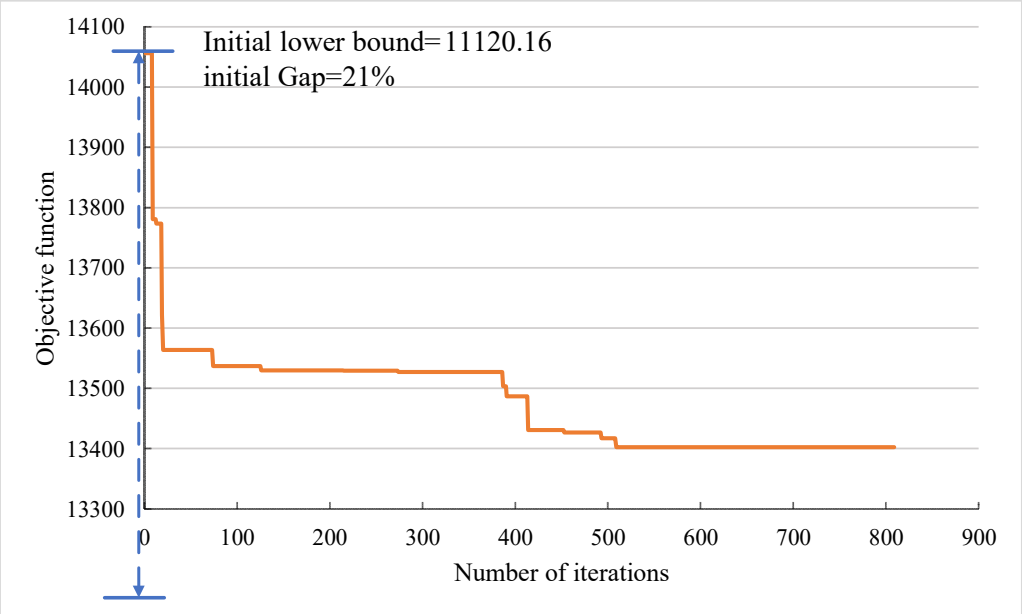


Figure 20. Convergence curve of the lower bound and upper bound of the real-world case.

Table 15 presents the details of the final solution, that is, the allocation of warehouses to supply the corresponding DCs and the demand coverage decision from DCs to stations. Figure 21 shows the spatial visualization of the optimization results. According to the final solution, DCs 145 and 146 assume the main distribution function, covering 40% of the total number of warehouses and 59% of the total number of stations. Affected by the spatial distribution of nodes, DC 143 only receives the goods transferred from other DCs and sends them to the station, while DC 148 is only responsible for handling the goods from warehouses and transferring them to other DCs. The flow in the logistics network is shown in Figure 22. In the figure, the red line represents lateral-transshipments flow between DCs, the blue line represents the non-transfer flow, and the width of the line reflects the size of the flow. In this real-world case, the total demand is 35,610 and the lateral-transshipments volume is 6126.

Table 15. Details of the final solution.

DC	Warehouses Assigned to DC	Stations Covered by DC
141	101; 102; 107; 108; 115; 129	4; 8; 37; 43; 57; 92; 94
142	110; 117; 121; 128; 132; 138	5; 34; 55; 68; 69; 87
143	-	12; 39; 51; 77
144	124;	73
145	105; 119; 127; 134; 136; 137	2; 6; 7; 9; 10; 11; 17; 18; 25; 27; 28; 29; 30; 32; 33; 36; 40; 41; 45; 46; 54; 58; 64; 81; 82; 85; 93; 97; 98
146	106; 109; 111; 112; 116; 118; 122; 130; 139; 140	1; 3; 13; 14; 16; 19; 20; 21; 22; 31; 35; 48; 49; 50; 52; 56; 60; 61; 62; 67; 70; 72; 75; 76; 83; 89; 90; 96; 99; 100
147	104; 133	26; 65
148	114; 125	-
149	120; 135	15; 24; 38; 79; 80; 84; 95
150	103; 113; 123; 126; 131	23; 42; 44; 47; 53; 59; 63; 66; 71; 74; 78; 86; 88; 91

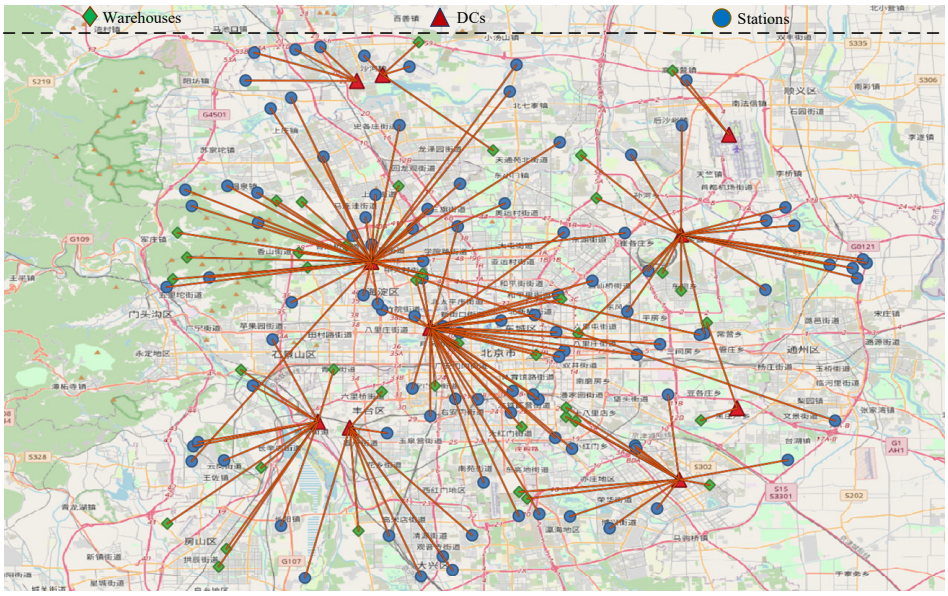


Figure 21. Optimized multi-echelon distribution network in the best upper bound solution.

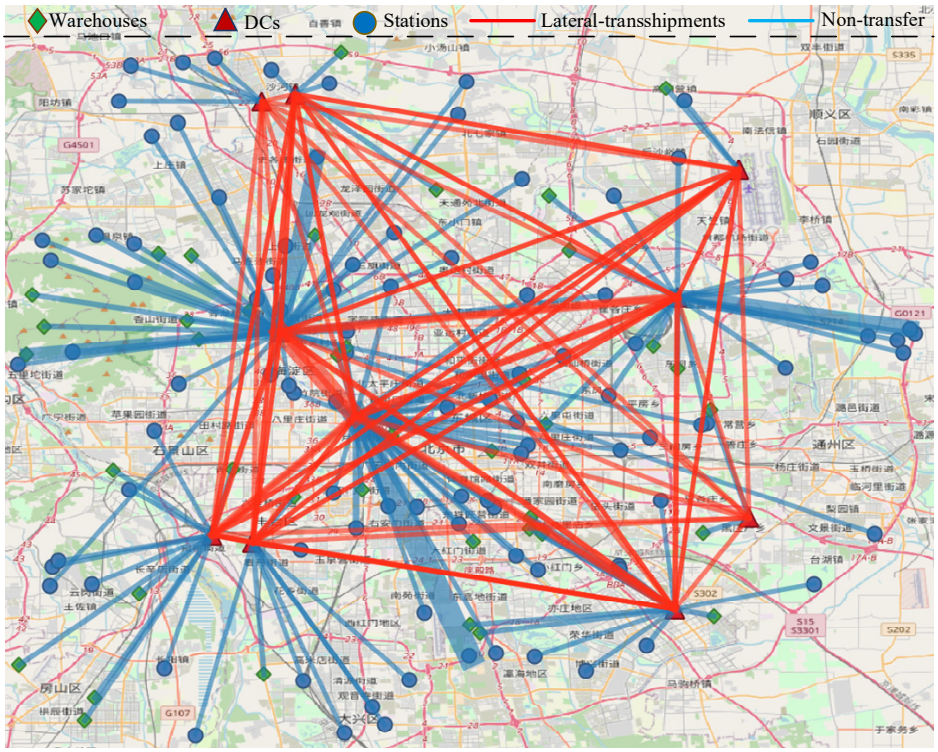


Figure 22. The flow in the optimized multi-echelon distribution network.

5.6. Managerial Implications for Dynamical Evaluation of Cross-Layer Service Synchronization

One of the core purposes to conduct cross-layer service synchronization evaluation is to increase operation efficiency and controllability of the urban logistics system at different levels of OD demand distribution. From the perspective of the economies of scale and long-term operation, single sourcing strategy should be adopted due to its less administrative effort and easier coordination. Single sourcing is necessary for time-dependent transportation services provided by third-party logistics companies, and suppliers need to choose appropriate delivery times at warehouses so that customers can pick up the goods at their booked time [59]. The existence of lateral-transshipment enables agile distribution

but leads to a more complex planning structure. To provide effective delivery services for muti-echelon logistics networks with lateral-transshipment, decision-makers need to know: (1) how lateral-transshipment improves the performance of the distribution network, and (2) to what extent we can proactively adjust the matching relationship within warehouse-DCs and DCs-delivery stations at different levels of OD demand distribution as part of controllability quantification tasks. The proposed two-stage cost-estimating framework can quickly obtain a solution by estimating the cost across different layers. Specifically, the application scenarios of the control measures for highly dynamic demand and supply impacts can be categorized as follows: (1) excessive quantity of goods need by customers at a delivery station (i.e., on some shopping days); (2) extensive delays in delivery services and planner need to quickly synchronize suppliers' fulfillment and customers' pick-up time.

5.7. QUBO Formulation of the QSAP-C

In this section, we present the QUBO formulation of the proposed model inspired by [11,60] and show the potential for acceleration through quantum computing. QUBO problems are unconstrained, quadratic, and of binary form generally defined as follows:

$$E(x) = x^T Q x + q \tag{12}$$

where Q represents a $m \times m$ matrix, q is a constant term, a solution $x = (x_1, \ldots, x_m)$, $x_i \in \{0, 1\}$ is an m-dimensional binary vector, and $E(x)$ is the energy (or fitness) of x.

The QSAP-C can be formulated as QUBO using Equation (14) where the cost function $c(x)$ and the constraint function $g(x)$ are presented in Equations (15) and (A1), respectively. The penalty weight is denoted by α and set according to the method described in [61,62].

$$E(x) = c(x) + \alpha g(x) \tag{13}$$

$$c(x) = \sum_{k\in K}\sum_{l\in L}\sum_{j\in J} f_{kj} d_{kl} x_{kl} + \sum_{l\in L}\sum_{l'\in L'}\sum_{k\in K}\sum_{j\in J} f_{kj} d_{ll'} x_{kl} y_{l'j} + \sum_{l'\in L'}\sum_{j\in J}\sum_{k\in K} f_{kj} d_{l'j} y_{l'j} \tag{14}$$

$$\begin{aligned} g(x) = {} & \sum_{k\in K}\left(1 - \sum_{l\in L} x_{kl}\right)^2 + \sum_{j\in J}\left(1 - \sum_{l'\in L} y_{l'j}\right)^2 + \sum_{(l,l')\in(L,L')} (Cap_{(l,l')} - \\ & \sum_{k\in K}\sum_{j\in J} x_{kl} f_{kj} - \sum_{k\in K}\sum_{j\in J} y_{l'j} f_{kj} + \sum_{k\in K}\sum_{j\in J} x_{kl} y_{l'j} f_{kj} - \sum_{i=0}^{n} 2^i x_{ll'}^i)^2 \end{aligned} \tag{15}$$

According to the preceding procedure, Transformation # 1 proposed by [11] that transforms the general problem into an equivalent QUBO model, which requires all constraints to be equations rather than inequalities, we convert constraints (4) to equations by including slack variables s_4 via a binary expansion. To accomplish this, we first estimate upper bounds on the constraints, that is $0 \le s_4 \le \sum_{k\in K}\sum_{j\in J} f_{kj}$, which can be regarded as a basis for determining how many binary variables will be required to represent the slack variables in the binary expansions. Assume that we need n binary variables and $n = \max_n \left\{2^n < \sum_{k\in K}\sum_{j\in J} f_{kj}\right\}$, then s_4 can be expressed as $s_4 = \sum_{i=0}^{n} 2^i x_{ll'}^i$.

6. Conclusions and Future Research

Focusing on real-world e-commerce logistics applications, this paper is motivated by the need to solve multi-echelon distribution network design problems with intra-echelon connections, where it is possible to select paths containing "shortcuts" for minimizing the overall generalized cost of travel. To capture the additional complexity of multi-echelon network configuration with intra-echelon connections, we propose a QSAP-C model with two types of binary variables for deciding the allocation of warehouses to DCs and the allocation of DCs to delivery stations. Furthermore, this paper designs a matheuristic

approximation method to efficiently obtain the GLB-oriented lower bound estimate. The lower bound estimation procedure is an extension and adaptation to the general assignment case of GLB for solving the standard QAP. The proposed solution approach reformulates the original problem as two-stage decisions and solves them sequentially to obtain the estimated cost. To improve the quality of solutions, both exact and heuristic algorithms with the embedded GLB-oriented lower bound estimator are proposed.

By using the parametric sensitivity analysis and OD distribution information, the proposed model can effectively evaluate different types of distribution network configuration. The solution framework is also tested using real-world distribution networks of JD logistics in Beijing. The experimental results indicate that OD demand patterns can significantly influence the underlying city logistics distribution and transfer transportation, which further possibly reduces the overall transportation cost to a certain degree. The results also suggest that both the proposed branch-and-bound framework combined approximation schemes and ALNS algorithm can provide high-quality solutions in all instances and work effectively in the real-world case.

In terms of future works, we will consider more realistic factors to further address the limitations associated with simplistic assumptions in this paper: (1) the transportation cost should also reflect economies of scale, e.g., the unit transportation cost is a nonincreasing function of the rate of flow [50]; (2) reactive transshipment should occur after observing demand [3]; and (3) we will further integrate our proposed optimization model and the time-dependent travel time model in rich arc routing problem developed by [63] to provide more reasonable solutions, where the time-dependent travel time and traffic congestion will be included in a more comprehensive manner.

Author Contributions: Conceptualization, S.W. and X.Z.; methodology, X.Z.; software, Z.N.; validation, Z.N., S.W., and X.Z.; formal analysis, Z.N.; investigation, Z.N. and X.Z.; writing—original draft preparation, Z.N., S.W., and X.Z.; writing—review and editing, S.W. and X.Z.; visualization, Z.N.; supervision, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

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

Appendix A

Appendix A.1 The Relationship between Standard QAP and QSAP-C

Given three $n \times n$ input matrices with real elements $F = (f_{ij})$, $D = (d_{kl})$, and $B = (b_{ik})$, where f_{ij} is the flow between the facility i and facility j, d_{kl} is the distance between the location k and location l, and b_{ik} is the cost of placing facility i at location k. The Koopmans-Beckmann version of the QAP can be formulated as follows:

$$\text{Min} \sum_{i=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} \sum_{j=1}^{N} f_{ij} d_{kl} x_{ik} x_{lj} + \sum_{i=1}^{N} \sum_{k=1}^{N} b_{ik} x_{ik} \tag{A1}$$

$$\sum_{i=1}^{N} x_{ik} = 1 \; \forall \; k = 1, 2, \ldots N \tag{A2}$$

$$\sum_{k=1}^{N} x_{ik} = 1 \; \forall \; i = 1, 2, \ldots N \tag{A3}$$

Ref. [59] generalize traditional QAP as the network-based QAP (NET-QAP). Their generic QAP network depicts a QAP model with N origins defined as N_S, N candidate locations for origins defined as N_L, M candidate locations for destinations defined as M_S,

and M destinations defined as M_L. Using the same indices, let $i \in N_S$, $k \in N_L$, $l \in M_L$, $j \in M_s$, and the NET-QAP model can be presented as follows:

$$\text{Min}\sum_{i=1}^{N_S}\sum_{k=1}^{N_L}\sum_{l=1}^{M_S}\sum_{j=1}^{M_L} f_{ij}d_{kl}x_{ik}x_{lj} + \sum_{i=1}^{N_S}\sum_{k=1}^{N_L} b_{ik}x_{ik} + \sum_{l=1}^{M_S}\sum_{j=1}^{M_L} b_{lj}x_{lj} \tag{A4}$$

$$\sum_{i=1}^{N_S} x_{ik} = 1 \;\forall\, k = 1, 2, \ldots N \tag{A5}$$

$$\sum_{k=1}^{N_L} x_{ik} = 1 \;\forall\, i = 1, 2, \ldots N \tag{A6}$$

$$\sum_{l=1}^{M_S} x_{lj} = 1 \;\forall\, j = 1, 2, \ldots M \tag{A7}$$

$$\sum_{j=1}^{M_L} x_{lj} = 1 \;\forall\, l = 1, 2, \ldots M \tag{A8}$$

In this paper, we further generalize the NET-QAP model: (1) the number of nodes is different in four echelons (that is $|N_S| \neq |N_L|, |M_L| \neq |M_s|$); (2) the assignment between $N_S(M_s)$ and $N_L(M_L)$ is general assignment; (3) capacities of $N_L(M_L)$ are considered; (4) the cost on links (i, k) and (l, j) is not fixed and it is also the product of distance and freight flow. It is immediate that we can write QSAP-C as the NET-QAP of the form.

$$\text{Min}\sum_{i=1}^{N_S}\sum_{k=1}^{N_L}\sum_{l=1}^{M_S}\sum_{j=1}^{M_L} f_{ij}d_{kl}x_{ik}x_{lj} + \sum_{i=1}^{N_S}\sum_{k=1}^{N_L}\sum_{j=1}^{M_L} f_{ij}d_{ik}x_{ik} + \sum_{l=1}^{M_S}\sum_{j=1}^{M_L}\sum_{i=1}^{N_S} f_{ij}d_{lj}x_{lj} \tag{A9}$$

$$\sum_{k=1}^{N_L} x_{ik} = 1 \;\forall\, i = 1, 2, \ldots N_S \tag{A10}$$

$$\sum_{l=1}^{M_L} x_{lj} = 1 \;\forall\, j = 1, 2, \ldots M_S \tag{A11}$$

$$\sum_{i\in I}\sum_{j\in J} x_{ik}f_{ij} + \sum_{i\in I}\sum_{j\in J} x_{lj}f_{ij} - \sum_{i\in I}\sum_{j\in J} x_{ik}x_{lj}f_{kj} \leq Cap_{(k,l)} \;\forall\, (k,l) \in (N_L, M_L) \tag{A12}$$

Appendix A.2 The Quality of Different Lower Bounds

Table A1. Comparison of lower bounds for standard QAP.

Instance	GLB62	RRD95	HG98	KCCEB99	AB01	RRRP02	RS03	R04	BV04	HH01
Had16	9.7%	-	4.4%	4.5%	3.4%	-	0.6%	0	1.3%	0
Had18	10.9%	-	5.1%	5.2%	4.0%	-	0.8%	0.04%	1.1%	0
Had20	10.9%	-	5.1%	5.1%	3.5%	-	0.5%	0.03%	1.6%	0
Kra30a	23.1%	14.5%	14.7%	15.0%	22.9%	-	12.7%	-	2.5%	3.0%
Kra30b	24.5%	16.0%	16.3%	16.6%	24.5%	-	11.2%	-	4.1%	4.7%
Nug12	14.7%	9.5%	9.5%	9.9%	13.8%	0	3.6%	1.9%	1.7%	0
Nug15	16.3%	9.5%	9.7%	10.2%	13.0%	-	2.4%	1.0%	0.8%	0
Nug18	-	-	-	-	-	-	-	-	-	-
Nug20	20.0%	15.1%	15.2%	15.4%	10.9%	-	4.6%	3.0%	2.5%	2.4%
Nug22	-	-	-	-	-	-	-	-	-	2.4%
Nug30	25.9%	21.5%	21.7%	21.9%	12.4%	-	5.2%	-	3.1%	5.8%
Rou15	15.7%	8.3%	8.5%	8.6%	14.2%	-	5.9%	1.5%	1.1%	0

Table A1. *Cont.*

Instance	GLB62	RRD95	HG98	KCCEB99	AB01	RRRP02	RS03	R04	BV04	HH01
Rou20	22.8%	11.3%	11.5%	11.6%	16.2%	-	8.5%	4.7%	4.2%	3.6%
Tai20a	17.5%	-	12.3%	12.3%	16.8%	-	9.4%	-	4.5%	3.9%
Tai25a	17.5%	-	13.8%	13.8%	15.7%	-	10.8%	-	4.7%	6.5%
Tai30a	17.2%	-	13.9%	13.9%	16.5%	-	9.1%	-	6.1%	7.3%
Tho30	39.6%	32.8%	33.3%	33.4%	16.8%	-	9.3%	-	4.8%	8.8%

Lower bounds: GLB62—Gilmore-Lawler bound from [64]; RRD95—interior-point bound from [65]; HG98—dual ascent bound from [66]; KCCEB99—dual-based bound from [67]; AB01—quadratic programming bound from [68]; RRRP02—interior point bound from [69]; RS03—semi-definite programming bound from [70]; R04—semi-definite programming (SDP) bound [71]; BV04—lift-and-project SDP bound from [72]; HH01—Hahn-Hightower dual ascent bound from [73].

References

1. Ambrosino, D.; Scutella, M.G. Distribution network design: New problems and related models. *Eur. J. Oper. Res.* **2005**, *165*, 610–624. [CrossRef]
2. Puga, M.S.; Minner, S.; Tancrez, J.S. Two-stage supply chain design with safety stock placement decisions. *Int. J. Prod. Econ.* **2019**, *209*, 183–193. [CrossRef]
3. Dehghani, M.; Abbasi, B. An age-based lateral-transshipment policy for perishable items. *Int. J. Prod. Econ.* **2018**, *198*, 93–103. [CrossRef]
4. Paterson, C.; GKiesmüller Teunter, R.; Glazebrook, K. Inventory models with lateral transshipments: A review. *Eur. J. Oper. Res.* **2011**, *210*, 125–136. [CrossRef]
5. Jovan, G. Amiya Chakravarty. Sharing and Lateral Transshipment of Inventory in a Supply Chain with Expensive Low-Demand Items. *Manag. Sci.* **2001**, *47*, 579–594.
6. Rabbani, M.; Sabbaghnia, A.; Mobini, M.; Razmi, J. A graph theory-based algorithm for a multi-echelon multi-period responsive supply chain network design with lateral-transshipments. *Oper. Res.* **2020**, *20*, 2497–2517. [CrossRef]
7. Domschke, W. Schedule synchronization for public transit networks. *Oper.-Res.-Spektrum* **1989**, *11*, 17–24. [CrossRef]
8. Hahn, P.M.; Kim, B.J.; Guignard, M.; Smith, J.M.; Zhu, Y.R. An algorithm for the generalized quadratic assignment problem. *Comput. Optim. Appl.* **2008**, *40*, 351–372. [CrossRef]
9. Bertsimas, D.; Delarue, A.; Martin, S. Optimizing schools' start time and bus routes. *Proc. Natl. Acad. Sci. USA* **2019**, *116*, 5943–5948. [CrossRef]
10. Glover, F.; Lewis, M.; Kochenberger, G. Logical and inequality implications for reducing the size and difficulty of quadratic unconstrained binary optimization problems. *Eur. J. Oper. Res.* **2018**, *265*, 829–842. [CrossRef]
11. Glover, F.; Kochenberger, G.; Hennig, R.; Du, Y. Quantum Bridge Analytics I: A tutorial on formulating and using QUBO models. *Ann. Oper. Res.* **2022**, *314*, 141–183. [CrossRef]
12. Kochenberger, G.A.; Glover, F. A unified framework for modeling and solving combinatorial optimization problems: A tutorial. In *Multiscale Optimization Methods and Applications*; Springer: Boston, MA, USA, 2006; pp. 101–124.
13. Zhang, H.; Liu, F.; Zhou, Y.; Zhang, Z. A hybrid method integrating an elite genetic algorithm with tabu search for the quadratic assignment problem. *Inf. Sci.* **2020**, *539*, 347–374. [CrossRef]
14. Dokeroglu, T.; Sevinc, E.; Cosar, A. Artificial bee colony optimization for the quadratic assignment problem. *Appl. Soft Comput.* **2019**, *76*, 595–606. [CrossRef]
15. Peng, Z.Y.; Huang, Y.J.; Zhong, Y.B. A discrete artificial bee colony algorithm for quadratic assignment problem. *J. High Speed Netw.* **2022**, *28*, 131–141. [CrossRef]
16. Wang, H.; Alidaee, B. A New Hybrid-heuristic for Large-scale Combinatorial Optimization: A Case of Quadratic Assignment Problem. *Comput. Ind. Eng.* **2023**, *179*, 109220. [CrossRef]
17. Shahabi, M.; Akbarinasaji, S.; Unnikrishnan, A.; James, R. Integrated inventory control and facility location decisions in a multi-echelon supply chain network with hubs. *Netw. Spat. Econ.* **2013**, *13*, 497–514. [CrossRef]
18. Shi, J.; Zhang, G.; Sha, J. A Lagrangian based solution algorithm for a build-to-order supply chain network design problem. *Adv. Eng. Softw.* **2012**, *49*, 21–28. [CrossRef]
19. Jang, Y.J.; Jang, S.Y.; Chang, B.M.; Park, J. A combined model of network design and production/distribution planning for a supply network. *Comput. Ind. Eng.* **2002**, *43*, 263–281. [CrossRef]
20. Manupati, V.K.; Jedidah, S.J.; Gupta, S.; Bhandari, A.; Ramkumar, M. Optimization of a multi-echelon sustainable production-distribution supply chain system with lead time consideration under carbon emission policies. *Comput. Ind. Eng.* **2019**, *135*, 1312–1323. [CrossRef]
21. Melo, M.T.; Nickel, S.; Saldanha-Da-Gama, F. Facility location and supply chain management–A review. *Eur. J. Oper. Res.* **2009**, *196*, 401–412. [CrossRef]
22. Devika, K.; Jafarian, A.; Nourbakhsh, V. Designing a sustainable closed-loop supply chain network based on triple bottom line approach: A comparison of metaheuristics hybridization techniques. *Eur. J. Oper. Res.* **2014**, *235*, 594–615. [CrossRef]

23. Eskandarpour, M.; Dejax, P.; Miemczyk, J.; Péton, O. Sustainable supply chain network design: An optimization-oriented review. *Omega* **2015**, *54*, 11–32. [CrossRef]
24. Wang, H.S. A two-phase ant colony algorithm for multi-echelon defective supply chain network design. *Eur. J. Oper. Res.* **2009**, *192*, 243–252. [CrossRef]
25. Wang, K.J.; Makond, B.; Liu, S.Y. Location and allocation decisions in a two-echelon supply chain with stochastic demand–A genetic-algorithm based solution. *Expert Syst. Appl.* **2011**, *38*, 6125–6131. [CrossRef]
26. Park, S.; Lee, T.E.; Sung, C.S. A three-level supply chain network design model with risk-pooling and lead times. *Transp. Res. Part E Logist. Transp. Rev.* **2010**, *46*, 563–581. [CrossRef]
27. Mogale, D.G.; Kumar, M.; Kumar, S.K.; Tiwari, M.K. Grain silo location-allocation problem with dwell time for optimization of food grain supply chain network. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *111*, 40–69. [CrossRef]
28. Barbarosoğlu, G.; Özgür, D. Hierarchical design of an integrated production and 2-echelon distribution system. *Eur. J. Oper. Res.* **1999**, *118*, 464–484. [CrossRef]
29. Chen, P.; Pinto, J.M. Lagrangean-based techniques for the supply chain management of flexible process networks. *Comput. Chem. Eng.* **2008**, *32*, 2505–2528. [CrossRef]
30. Farahani, R.Z.; Elahipanah, M. A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain. *Int. J. Prod. Econ.* **2008**, *111*, 229–243. [CrossRef]
31. Park, Y.B. An integrated approach for production and distribution planning in supply chain management. *Int. J. Prod. Res.* **2005**, *43*, 1205–1224. [CrossRef]
32. Tsiakis, P.; Papageorgiou, L.G. Optimal production allocation and distribution supply chain networks. *Int. J. Prod. Econ.* **2008**, *111*, 468–483. [CrossRef]
33. Akbari, A.A.; Karimi, B. A new robust optimization approach for integrated multi-echelon, multi-product, multi-period supply chain network design under process uncertainty. *Int. J. Adv. Manuf. Technol.* **2015**, *79*, 229–244. [CrossRef]
34. Fard AM, F.; Hajiaghaei-Keshteli, M. A bi-objective partial interdiction problem considering different defensive systems with capacity expansion of facilities under imminent attacks. *Appl. Soft Comput.* **2018**, *68*, 343–359. [CrossRef]
35. Fard AM, F.; Hajaghaei-Keshteli, M. A tri-level location-allocation model for forward/reverse supply chain. *Appl. Soft Comput.* **2018**, *62*, 328–346. [CrossRef]
36. Badri, H.; Ghomi, S.F.; Hejazi, T.H. A two-stage stochastic programming approach for value-based closed-loop supply chain network design. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *105*, 1–17. [CrossRef]
37. Fattahi, M.; Govindan, K.; Keyvanshokooh, E. Responsive and resilient supply chain network design under operational and disruption risks with delivery lead-time sensitive customers. *Transp. Res. Part E Logist. Transp. Rev.* **2017**, *101*, 176–200. [CrossRef]
38. Aikens, C.H. Facility location models for distribution planning. *Eur. J. Oper. Res.* **1985**, *22*, 263–279. [CrossRef]
39. Vidal, C.J.; Goetschalckx, M. Strategic production-distribution models: A critical review with emphasis on global supply chain models. *Eur. J. Oper. Res.* **1997**, *98*, 1–18. [CrossRef]
40. Fahimnia, B.; Farahani, R.Z.; Marian, R.; Luong, L. A review and critique on integrated production–distribution planning models and techniques. *J. Manuf. Syst.* **2013**, *32*, 1–19. [CrossRef]
41. Goetschalckx, M.; Vidal, C.J.; Dogan, K. Modeling and design of global logistics systems: A review of integrated strategic and tactical models and design algorithms. *Eur. J. Oper. Res.* **2002**, *143*, 1–18. [CrossRef]
42. Chen, Z.L. Integrated production and outbound distribution scheduling: Review and extensions. *Oper. Res.* **2010**, *58*, 130–148. [CrossRef]
43. Fisher, M.L.; Jörnsten, K.O.; Madsen, O.B. Vehicle routing with time windows: Two optimization algorithms. *Oper. Res.* **1997**, *45*, 488–492. [CrossRef]
44. Shen ZJ, M.; Coullard, C.; Daskin, M.S. A joint location-inventory model. *Transp. Sci.* **2003**, *37*, 40–55. [CrossRef]
45. Pan, F.; Nagi, R. Multi-echelon supply chain network design in agile manufacturing. *Omega* **2013**, *41*, 969–983. [CrossRef]
46. Ben Abid, T.; Ayadi, O.; Masmoudi, F. An integrated production-distribution planning problem under demand and production capacity uncertainties: New formulation and case study. *Math. Probl. Eng.* **2020**, *2020*, 1520764. [CrossRef]
47. Larimi, N.G.; Yaghoubi, S.; Hosseini-Motlagh, S.M. Itemized platelet supply chain with lateral transshipment under uncertainty evaluating inappropriate output in laboratories. *Socio-Econ. Plan. Sci.* **2019**, *68*, 100697. [CrossRef]
48. Tsiakis, P.; Shah, N.; Pantelides, C.C. Design of multi-echelon supply chain networks under demand uncertainty. *Ind. Eng. Chem. Res.* **2001**, *40*, 3585–3604. [CrossRef]
49. Finke, G.; Burkard, R.E.; Rendl, F. Quadratic assignment problems. *North-Holl. Math. Stud.* **1987**, *132*, 61–82.
50. Abdel-Basset, M.; Manogaran, G.; Rashad, H.; Zaied, A.N.H. A comprehensive review of quadratic assignment problem: Variants, hybrids and applications. *J. Ambient. Intell. Humaniz. Comput.* **2018**, 1–24. [CrossRef]
51. Burkard, R.E.; Cela, E.; Pardalos, P.M.; Pitsoulis, L.S. The quadratic assignment problem. In *Handbook of Combinatorial Optimization*; Springer: Boston, MA, USA, 1998; pp. 1713–1809.
52. Burkard, R.; Dell'Amico, M.; Martello, S. *Assignment Problems: Revised Reprint*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2012.
53. Li, Y.; Pardalos, P.M.; Ramakrishnan, K.G.; Resende, M.G. Lower bounds for the quadratic assignment problem. *Ann. Oper. Res.* **1994**, *50*, 387–410. [CrossRef]
54. Pisinger, D.; Ropke, S. A general heuristic for vehicle routing problems. *Comput. Oper. Res.* **2007**, *34*, 2403–2435. [CrossRef]

55. Lutz, R. Adaptive Large Neighborhood Search. Bachelor's Thesis, Ulm University, Ulm, Germany, 2015.
56. Yu, C.; Zhang, D.; Lau, H.Y. An adaptive large neighborhood search heuristic for solving a robust gate assignment problem. *Expert Syst. Appl.* **2017**, *84*, 143–154. [CrossRef]
57. Perboli, G.; Tadei, R.; Vigo, D. The two-echelon capacitated vehicle routing problem: Models and math-based heuristics. *Transp. Sci.* **2011**, *45*, 364–380. [CrossRef]
58. Crainic, T.G.; Perboli, G.; Mancini, S.; Tadei, R. Two-echelon vehicle routing problem: A satellite location analysis. *Procedia-Soc. Behav. Sci.* **2010**, *2*, 5944–5955. [CrossRef]
59. Wu, X.B.; Lu, J.; Wu, S.; Zhou, X.S. Synchronizing time-dependent transportation services: Reformulation and solution algorithm using quadratic assignment problem. *Transp. Res. Part B Methodol.* **2021**, *152*, 140–179. [CrossRef]
60. Ayodele, M.; Allmendinger, R.; López-Ibáñez, M.; Parizy, M. Multi-objective QUBO solver: Bi-objective quadratic assignment problem. In Proceedings of the Genetic and Evolutionary Computation Conference, Boston, MA, USA, 9–13 June 2022.
61. Verma, A.; Lewis, M. Penalty and partitioning techniques to improve performance of QUBO solvers. *Discret. Optim.* **2020**, *44*, 100594. [CrossRef]
62. Mayowa, A. Penalty Weights in QUBO Formulations: Permutation Problems. In Proceedings of the EvoCOP 2022–22nd European Conference on Evolutionary Computation in Combinatorial Optimization, Madrid, Spain, 20–22 April 2022; Leslie, P.C., Sébastien, V., Eds.; Springer: Cham, Switzerland, 2022; pp. 159–174.
63. Lu, J.; Nie, Q.; Mahmoudi, M.; Ou, J.; Li, C.; Zhou, X.S. Rich arc routing problem in city logistics: Models and solution algorithms using a fluid queue-based time-dependent travel time representation. *Transp. Res. Part B Methodol.* **2022**, *166*, 143–182. [CrossRef]
64. Gilmore, P.C. Optimal and suboptimal algorithms for the quadratic assignment problem. *J. Soc. Ind. Appl. Math.* **1962**, *10*, 305–313. [CrossRef]
65. Resende, M.G.; Ramakrishnan, K.G.; Drezner, Z. Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Oper. Res.* **1995**, *43*, 781–791. [CrossRef]
66. Hahn, P.; Grant, T. Lower bounds for the quadratic assignment problem based upon a dual formulation. *Oper. Res.* **1998**, *46*, 912–922. [CrossRef]
67. Karisch, S.E.; Cela, E.; Clausen, J.; Espersen, T. A dual framework for lower bounds of the quadratic assignment problem based on linearization. *Computing* **1999**, *63*, 351–403. [CrossRef]
68. Anstreicher, K.M.; Brixius, N.W. A new bound for the quadratic assignment problem based on convex quadratic programming. *Math. Program.* **2001**, *89*, 341–357. [CrossRef]
69. Ramakrishnan, K.G.; Resende MG, C.; Ramachandran, B.; Pekny, J.F. Tight QAP bounds via linear programming. In *Combinatorial and Global Optimization*; World Scientific: Singapore, 2002; pp. 297–303.
70. Sotirov, R.; Rendl, F. Bounds for the quadratic assignment problem using the bundle method. In *Discrete Optimization: Methods and Applications*; University of Klagenfurt: Klagenfurt, Austria, 2003; pp. 287–305.
71. Roupin, F. From linear to semidefinite programming: An algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *J. Comb. Optim.* **2004**, *8*, 469–493. [CrossRef]
72. Burer, S.; Vandenbussche, D. Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim.* **2006**, *16*, 726–750. [CrossRef]
73. Adams, W.P.; Guignard, M.; Hahn, P.M.; Hightower, W.L. A level-2 reformulation–linearization technique bound for the quadratic assignment problem. *Eur. J. Oper. Res.* **2007**, *180*, 983–996. [CrossRef]