European Journal of Operational Research 334 (2026) 315–335

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Innovative applications of O.R.

Optimizing warehouse product allocation at JD.com: A community detection-driven approach

Hao Hu [a], Yongzhi Qi [b], Ningxuan Kang [b], Zhe Chen [b,*], Ling Wang [a], Jian Chen [c], Zuo-Jun Max Shen [d,e]

[a] Department of Automation, Tsinghua University, Beijing 100084, China
[b] Intelligent Supply Chain Y, JD.com, Beijing, 100176, China
[c] School of Economics and Management, Tsinghua University, Beijing 100084, China
[d] Faculty of Engineering, The University of Hong Kong, Hong Kong, China
[e] Faculty of Business and Economics, The University of Hong Kong, Hong Kong, China

ARTICLE INFO

Keywords:
Order splitting
Community detection
Heuristic algorithm
JD.com
Online retail

ABSTRACT

In e-commerce platforms, multi-item orders often require split shipments due to inventory distribution across geographically dispersed warehouses. This order splitting results in increased shipping costs and excessive packaging material consumption. The associated order splitting optimization problem presents significant complexity arising from the combinatorial nature of the order structure. Through collaboration with JD.com, China's largest online retailer, we analyze the real historical orders and inventory data from one distribution center. To minimize order splits, we develop a novel K-Community Swap Algorithm (KCSA) incorporating community detection techniques. Comparative benchmarks demonstrate that KCSA achieves solutions within 5% of optimality against Gurobi's exact solutions for small-scale instances while delivering superior solution quality in significantly less computational time than existing methods for medium-to-large-scale instances. The implications in JD.com reduce 440,000 order splittings in five distribution centers and save 2.6 million RMB packing material annually. The annual carbon dioxide equivalent emissions saved are approximately 132 tons.

1. Introduction

As China's dominant e-commerce enterprise, JD.com faces unique operational complexities in warehouse management within the world's largest online retail market, which reaches US$ 2.13 trillion and 915 million users in 2023. During the same year, JD.com achieved US$ 152.8 billion in net revenue, serving approximately 600 million active customers with over 10 million SKUs (stock-keeping units). The key to serve such a large customers is the warehousing system that can manage an extensive product range and massive inventory, which is vital in e-commerce (Boysen & De Koster, 2025; Boysen et al., 2019). The operation efficiency within the warehouses is extensively investigated in recent studies (Albán et al., 2024; Wagner & Mönch, 2023; Zhuang et al., 2024). The company developed an order fulfillment distribution system dividing mainland China into seven service regions. Usually, each region operates with one Regional Distribution Center (RDC) located in major cities housing all SKU types, supplemented by several Front Distribution Centers (FDCs) in smaller cities stocking fast-moving products. Due to capacity constraints in warehouse facilities, JD.com operates multiple warehouses within each distribution center. Typically storing one SKU per warehouse for efficient inventory management, this approach may result in order splitting when products originate from different warehouses, requiring separate shipments.

Customers ordering multiple SKUs through JD.com may experience order splitting if those SKUs are stored in separate warehouses. The detailed fulfillment process is illustrated in Fig. 1. Upon order placement, JD.com's fulfillment system checks the inventory status of each SKU. If items originate from different warehouses, the original order is automatically split into multiple suborders, each directed to the relevant warehouse holding the SKU. This splitting increases shipping costs and packing material usage, contributing to environmental impacts. From the customer perspective, receiving multiple packages also degrades the user experience. Consequently, minimizing order splitting is crucial for online retailers, particularly for JD.com.

Prior research in the order splitting field employs heuristic algorithms to address the order splitting problem. As noted in the following section, the primary approach transforms this challenge into a capacitated graph partition formulation and applies neighborhood search strategies for optimization. For instance, Zhu et al. (2021)

* Corresponding author.
E-mail address: chenzhe19@jd.com (Z. Chen).

https://doi.org/10.1016/j.ejor.2026.04.040
Received 7 August 2025; Accepted 24 April 2026
Available online 27 April 2026
0377-2217/© 2026 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

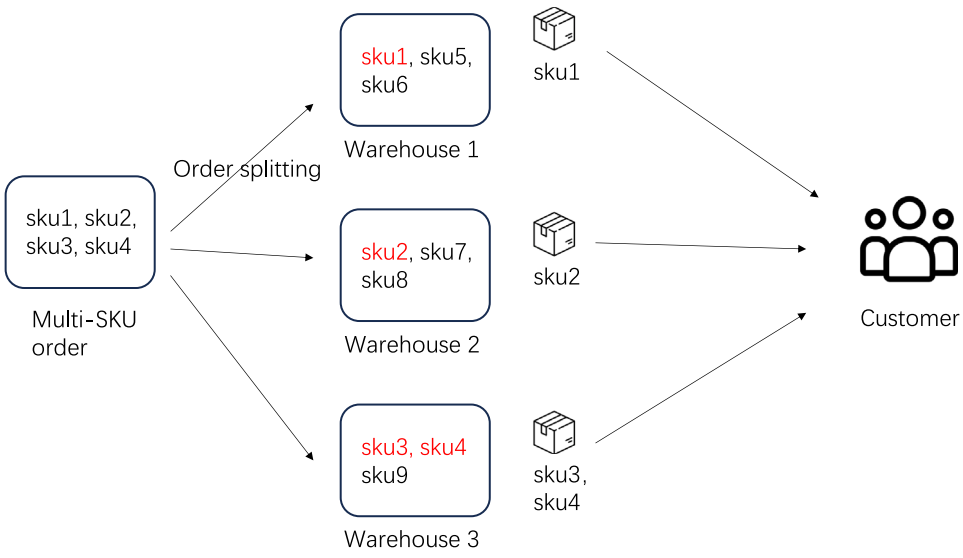


Fig. 1. The process of online order fulfillment.

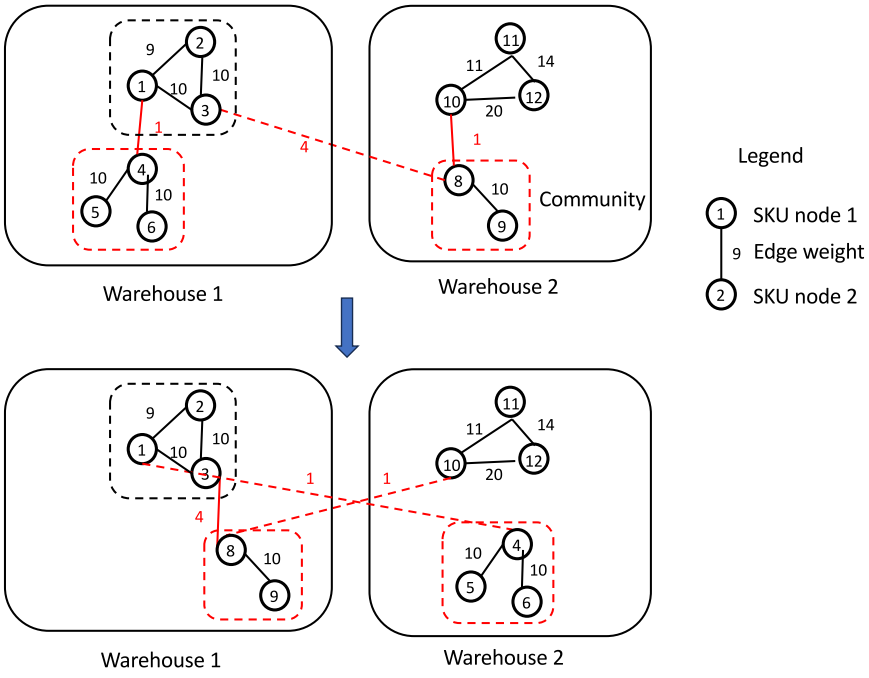


Fig. 2. An example that cannot be improved by simple relocate and switch strategies.

designs relocation and exchange move operations to iteratively enhance solutions: relocation involves removing a SKU from its current warehouse and reassigning it elsewhere, while exchange selects two SKUs from distinct warehouses and swaps their assignments. Both moves execute only when objectively beneficial. However, such strategies risk converging to local optima—exemplified in Fig. 2, where neither relocation nor exchange improves the assignment. Here, SKUs exhibit dense intra-community connectivity but sparse external links. Hence, simultaneously exchanging SKUs {4,5,6} with {8,9} would reduce inter-warehouse edges from four to two, yielding the optimal solution.

To address this problem, we propose a novel community detection-based approach, demonstrating high efficiency and effectiveness in real-world applications at JD.com. Specifically, we frame the order splitting problem as a capacitated hypergraph partition problem where orders (combinations of SKUs) form hyperedges. This problem is formulated as a binary integer program, proven NP-Hard (Lin et al., 2025). To reduce complexity, we follow Catalán and Fisher (2012) and Zhu et al. (2021) in reformulating it into a capacitated graph partition problem with an approximate objective function. This upper-bound approximation yields negligible differences for orders containing four or fewer SKUs, which constitute the majority in online retail. Subsequently, we introduce a novel Community Switch Algorithm leveraging community detection to efficiently solve the capacitated graph partition problem. The algorithm starts with an initial solution and iteratively improves it by selecting two warehouses and optimizing SKU allocation between them. Each iteration solves a capacitated min-2 graph partition sub-problem: SKUs are clustered into communities via community detection, and communities are assigned to warehouses if beneficial for reducing order splits while respecting capacity constraints. The new allocation is adopted if superior to the current solution. Evaluated on

datasets of varying scales, our algorithm produces solutions with less than 5% optimality gaps compared to Gurobi on small-scale problems while achieving significantly faster computation. Applied to large-scale JD.com customer order data, it demonstrates high efficiency and effectiveness in reducing real-world order splitting. Implementation at JD.com has reduced order splits by 440,000 in five distribution centers, enhancing operational efficiency, and saved 2.6 million RMB packing material annually, lowering carbon emissions.

In summary, this research makes the following contributions:

- This study represents the first effort to examine the problem of minimizing order splitting using large-scale, real-world data from JD.com. The size and complexity of this dataset demand highly efficient algorithms that can process tens of thousands of SKUs;
- We propose a systematic framework based on community detection to efficiently and effectively solve the order splitting minimization problem;
- We utilize the real order data from JD.com to verify the proposed algorithm. The efficiency is tested in small scale datasets compared with Gurobi, and the computational time is validated in large scale datasets; and
- Compared with the algorithms in existing literature, our method shows higher efficiency in solving the order splitting minimization problem on a large scale.

The remainder of this paper is structured as follows. Section 2 summarizes the relevant studies in the order splitting, order consolidation, hypergraph/graph partitioning and community detection fields. Section 3 formulates the order splitting minimization problem and reformulates it as a binary integer programming-based graph partition problem. Section 4 first performs complex network analysis to characterize the graph's properties before introducing our proposed K-community swap algorithm. Section 5 analyzes computational experiments across multiple scales using JD.com's real order data, benchmarking our algorithm against both exact solvers and existing methodologies. Finally, Section 6 summarizes key findings and outlines potential directions for future research.

2. Literature review

2.1. Order splitting problem

The order splitting problem has attracted considerable research interest due to its critical importance in online retail operations. Existing literature addressing this challenge can be broadly categorized into three primary streams (Zhu et al., 2021): (1) optimizing SKU allocation across multiple warehouses; (2) optimizing order assignments given fixed SKU allocation; and (3) order consolidation, which aims to reduce split shipments after orders have been split. Our research is aligned with the first stream. The three research streams are summarized in Table 1.

Within the first research stream, studies typically assume sufficient inventory and focus on optimizing SKU allocation across warehouses to minimize order splits for a given set of orders. This problem is highly complex due to its combinatorial nature, being equivalent to the hypergraph partition problem, which is NP-hard. Consequently, researchers often transform it into a graph partition problem using various approximation techniques. Catalán and Fisher (2012) pioneered this approach by intuitively transforming the hypergraph into a primal graph of SKUs, where edge weights represent the number of orders containing each node SKU pair. While this approximation inherently overweights orders containing more than two SKUs, Catalán and Fisher (2012) argues the resulting loss is negligible when most orders comprise four or fewer SKUs. They develop the BESTSELLERS heuristic to solve this primal graph partition problem.

Building on these graph-based transformations, Chockalingam et al. (2019) address a two-warehouse assortment allocation problem, selecting overlapping SKU subsets to minimize split shipments. They define cross-selling factors to characterize SKU coappearance. The model is an extension of the quadratic multiple-knapsack problem accounting for overlapping and load balancing, proposing both greedy and genetic algorithms for real-world application. Later, Zhu et al. (2021) extend this idea to non-overlapping assortment allocation. They refine link calculation and introduce a connection coefficient to mitigate overestimation effects, proposing a K-links clustering algorithm (a type of Variable Neighborhood Search algorithm) to solve the transformed graph partition problem. This algorithm employs relocation and exchange move strategies to iteratively improve assignments.

Most recently, Lin et al. (2025) conduct a theoretical study in a more general multi-warehouse setting with overlapping assortments. Focusing on demand distributions exhibiting second-order dominant Marginal Choice Indexing (MCI), they explore three scenarios: a non-overlapping assortment restriction, demands with nested structures, and two-warehouse networks. Numerical experiments under the two-warehouse scenario demonstrate that their extended MCI policy achieves optimality when demand aligns with the Independent Choice Model. This property implies MCI is based simply on products' marginal choice probabilities, without directly addressing customers' preferences across different subsets. However, such a strong assumption is often unrealistic in practice.

Studies within the second research stream focus on optimizing order assignments to warehouses to minimize split orders and shipping costs, assuming a predetermined SKU allocation across warehouses. Xu et al. (2009) pioneered the analysis of order splitting in online retail. They observe that real-time order assignments are inherently myopic, failing to account for future orders or inventory replenishment, which can increase split orders. They demonstrate benefits from periodically reevaluating assignments and propose a near-optimal heuristic algorithm for large-scale implementation. Other research prioritizes shipping cost reduction. Mahar and Wright (2009) similarly find that accumulating orders before assignment reduces shipping costs, proposing a dynamic allocation policy to maximize cost efficiency across fulfillment centers.

To capture the stochastic nature of future demands, Acimovic and Graves (2015) utilize dual values to approximate the expected future outbound shipping costs associated with inventory depletion. Jasin and Sinha (2015) leverage demand forecasts within a multi-item order fulfillment context, formulating a stochastic control problem and proposing heuristics to minimize expected shipping costs. Torabi et al. (2015) address fulfillment allocation and transshipment among centers in e-tailing, developing a Benders decomposition-based approach for efficient optimal solutions.

Further expanding the scope of order assignments, Ardjmand et al. (2018) integrate order cartonization with fulfillment center assignment, proposing decoupled and integrated integer programming models solved via genetic algorithms. Lei et al. (2018) investigate a joint pricing and fulfillment optimization problem, introducing two heuristic controls with strong performance. Their second heuristic, featuring adaptive adjustment, demonstrates superior theoretical and numerical results. Finally, Ishfaq and Raja (2018) examine omni-channel fulfillment allowing order splits, employing an outer approximation algorithm to maximize profit.

The studies in the third stream focus on merging sub-orders or packages that have already been split. Unlike the upstream strategies to avoid splits in the aforementioned two streams, order consolidation studies accept split orders as a given and seek to recombine them before final delivery. Based on the stage where consolidation occurs, studies in this stream can be further categorized as: (1) Order consolidation at warehouses, and (2) Order consolidation at delivery stations. The first category investigates consolidating sub-orders located

Table 1
Related studies in the three research streams.

Paper	Streams	Method	Data	Main results	Research gaps
SKU allocation					
Catalán and Fisher (2012)	SKU allocation with overlapping SKUs	Hypergraph transformation into a primal SKU graph where edge weights represent order co-occurrences. Solved using the BESTSELLERS heuristic	A real dataset (Yihaodian): 26,356 SKUs, over 100,000 daily orders, 3 warehouses	Pioneered hypergraph-to-graph transformation; Approximation loss is negligible when most orders comprise four or fewer SKUs;	Large optimality gaps in the heuristic method
Zhu et al. (2021)	SKU allocation without overlapping SKUs	Introduced connection coefficients to mitigate overestimation, solved via K-links clustering	Large Chinese online supermarket: 93 categories, 18,757 orders, 4 warehouses	Proposed enhanced approximation method; Developed VNS algorithm with relocation/exchange strategies;	High computational time when solving large problem instances
Lin et al. (2025)	SKU allocation with overlapping SKUs	Examined three scenarios under second-order dominant Marginal Choice Indexing: non-overlapping assortments, nested demands, and dual-warehouse networks	Synthetic experiments: independent choice model and multi-purchase multinomial logit model (6–10 products, 1000 subcases each)	Extended MCI policy achieves optimality under independent choice demand	The assumption of demand distributions is often unrealistic in practice
Order assignments					
Xu et al. (2009)	Order assignments	Near-optimal heuristic for periodic reevaluation in large-scale implementations	Large e-tailer: 800K-900K orders, 400K SKUs, 16K order splits	Periodic reassignment outperforms myopic real-time allocation	Optimizing order assignments given SKU allocation
Jasin and Sinha (2015)	Order assignments	Simulated business environment: Items ranging 10–500	Simulated business environment: Items ranging 10–500	Deterministic LP performs near-optimal across parameters	Optimizing order assignments given SKU allocation
Lei et al. (2018)	Joint pricing and fulfillment optimization	Two heuristics decoupling pricing and fulfillment decisions	Numerical simulation: 5 SKUs, 6 distribution centers	Adaptive control heuristic demonstrates theoretical/numerical superiority	Optimizing order assignments given SKU allocation
Order consolidation					
Zhang et al. (2021)	Order consolidation at warehouses	Logic-based Benders decomposition combining multi-commodity network flow and bin-packing models	Numerical examples: 2–20 SKUs	Extended consolidation across warehouse networks; Generated near-optimal solutions efficiently;	Optimizing post-split order consolidation
Zhang et al. (2023)	Order consolidation at delivery stations	Two-stage stochastic programming with scenario-based algorithm	12 instances with parameters varying by shipment frequency (Poisson distribution 2–4) and time periods (5–8)	Green labeling incentives enable better station-level consolidation	Optimizing post-split order consolidation

across multiple FCs and this is usually achieved through lateral transshipments between warehouses. Hall (1987) provides an early study on consolidation and found that it may greatly reduce transportation costs without compromising service quality. Ramakrishna et al. (2015) propose an optimization algorithm to solve a two-warehouse inventory model, which determines the order policies and consolidation decisions. Zhang et al. (2018) first systematically propose a package consolidation methodology for online supermarkets considering a single consolidation center. Later, Zhang et al. (2021) extend the problem to a network where multiple warehouses can serve as consolidation points. They develop a model integrating multi-commodity network flow and bin-packing problems, which is solved via a logic-based Benders decomposition algorithm. Huang et al. (2024) examine order consolidation from a green logistics perspective, incorporating factors like delivery time windows.

The second category focuses on merging multiple packages which is often by holding packages at the delivery stations to create a larger combined shipment. Theoretically, this line of studies belongs to the time-based shipment consolidation. Çetinkaya and Bookbinder (2003) propose stochastic models in dispatch consolidation. Followed by Stenius et al. (2016), they provide exact analysis for divergent systems to consolidate time-based shipments. Hanbazazah et al. (2019) study freight consolidation with more complex constraints. Zhang et al. (2019) explicitly address last-mile split delivery by proposing a time-based order consolidation strategy at delivery stations, formulating an integer programming model with piecewise transportation costs, and designing a three-stage heuristic algorithm. Zhang et al. (2023) propose a two-stage stochastic programming model and a scenario-based order consolidation algorithm to solve the last-mile split delivery problem. The method uses green labels to incentivize customers to choose delayed deliveries, enabling better order consolidation by consolidating shipments at stations.

The three streams represent sequential stages in the e-commerce fulfillment process. Specifically, SKU allocation is the foundational decision that shapes the structure of the entire fulfillment network. The way SKUs are distributed across warehouses directly impacts the solution space for downstream order assignment (Stream 2). A suboptimal allocation inherently restricts the feasibility of order assignments, leading to order splits regardless of how advanced the assignment algorithm might be. Consequently, the outcomes of order assignments affect the performance of order consolidation (Stream 3). Downstream order consolidation acts as the final stage to mitigate order splits, though it often incurs significant delays or extra transshipment costs. Therefore, optimizing SKU allocation, which is the focus of our research, addresses the root cause of the split-order problem.

2.2. Hypergraph and graph partitioning

Based on the two survey papers (Buluç et al., 2016; Çatalyürek et al., 2023), we briefly summarize the recent advances in the graph and hypergraph partitioning fields. Methodologies for graph partitioning have evolved from simple geometric methods to multi-level metaheuristics capable of handling massive complex networks. Early algorithms, including spectral bisection (Fiedler, 1975), the KL algorithm (Kernighan & Lin, 1970), and the FM algorithm (Fiduccia & Mattheyses, 1988), primarily relied on spectral properties such as the eigenvectors of the Laplacian matrix.

While these methods possessed a sound theoretical foundation, they were computationally expensive for large graphs. The field was subsequently revolutionized by the introduction of the multilevel paradigm. This approach recursively coarsens the graph, partitions the coarsest approximation, and projects the solution back to the original graph through a refinement phase, thereby dramatically enhancing efficiency. The key algorithms included METIS (Karypis & Kumar, 1998), CHACO (Hendrickson & Leland, 1993), and PT-Scotch (Chevalier & Pellegrini, 2008).

Recent studies focuses on integrating evolutionary algorithms such as Tabu Search (Glover, 1989), Adaptive large neighborhood search (Mara et al., 2022), and Simulated annealing (Van Laarhoven & Aarts, 1987) for higher quality and shared-memory parallelism (Meyerhenke et al., 2017; Sanders & Schulz, 2011). For instance, MITS integrates a powerful refinement procedure based on Tabu search with periodic perturbations (Benlic & Hao, 2011). The experimental evaluations show that MITS competes very favorably with METIS and CHACO. Similarly, JABEJA integrates the simulated annealing, and the algorithm is massively parallel (Rahimian et al., 2013). The experimental analysis shows that JABEJA outperforms METIS on large social networks. KaHIP uses flow-based refinement and local search to achieve better quality than Metis (Sanders & Schulz, 2011). ParHIP uses label propagation for parallel partitioning (Meyerhenke et al., 2017). Additionally, the n-level partitioning removes only one vertex or a small set of vertices at a time to obtain high quality but incurs significant computational costs (Gottesbüren et al., 2021; Osipov & Sanders, 2010). As for hypergraph partitioning, it followed a similar trajectory to graph partition but it has higher complexity because hyperedges connect an arbitrary number of nodes. The key studies in this domain include hMetis (Karypis & Kumar, 1999), KaHyPar (Schlag et al., 2023).

2.3. Community detection

Following the two comprehensive survey papers in community detection (Fortunato, 2010; Jin et al., 2021), we provide a overview of the evolution trajectory of community detection. The introduction of edge betweenness and the concept of modularity attracted significant interest from researchers, which marked a new phase for community detection (Fortunato, 2010). Girvan and Newman (2002) introduced a divisive algorithm based on removing edges with high betweenness. Subsequently, Newman and Girvan (2004) introduced a quantitative measure to assess the quality of a partition, which measured the relative density of edges inside communities with respect to edges outside communities. That work inspired the modularity optimization algorithms because optimizing modularity theoretically resulted in the best possible grouping of the nodes. While maximizing modularity is NP-hard, more efficient heuristics were developed (Blondel et al., 2008; Clauset et al., 2004). Notably, Blondel et al. (2008) proposed a greedy optimization method known as the Louvain algorithm, which operates through an iterative two-phase process. In the first phase, nodes are locally reassigned to communities to maximize modularity gain. In the second phase, the graph is reinterpreted so that communities are reduced to a single node. Researchers noticed that nodes often belong to multiple groups, leading to the study of overlapping communities. For instance, Palla et al. (2005) allowed nodes to have multiple memberships. Rosvall and Bergstrom (2008) employed random walks and compression to find communities. However, Fortunato and Barthelemy (2007) identified the resolution limit, demonstrating that the modularity methods failed to detect small clusters in large networks.

As highlighted by Jin et al. (2021), the studies moved toward statistical modeling and focused on generative models that explain how the network was formed. Stochastic Block Models (SBM) were the dominant paradigms, which were introduced by Holland et al. (1983) and extended to handle mixed memberships (Airoldi et al., 2008) and degree corrections (Karrer & Newman, 2011). Meanwhile, nonnegative matrix factorization was also used to discover the community patterns (Wang et al., 2011). More recent studies leveraged the neural networks, such as Auto-Encoder based methods (Wang et al., 2016), Graph convolutional networks (Kipf, 2016) and Generative adversarial networks (Jia et al., 2019).

Recent literature demonstrates the critical usage of community detection algorithms in investigating the topological complexity and enhancing the management of supply chain sectors. Feng et al. (2020) use modularity-based algorithms on maritime shipping networks to

identify core port communities, offering a novel perspective for analyzing trade relationships and resource allocation. Wiedmer and Griffis (2021) empirically confirm the existence of modular community structures within the global supply chains, and find out that identifying these topologies is fundamental for understanding network behavior. Yang et al. (2023) apply these methods to urban heavy truck mobility networks to reveal hierarchical industry clustering for freight policy formulation. Yang et al. (2025) utilize the Louvain algorithm within automotive supplier networks to identify cohesive communities, thereby evaluating network robustness against specific attack scenarios and suggesting hub-like connections to mitigate risk. Similarly, Zhu et al. (2025) employ a multi-layer network community detection approach to project complex plant and warehouse networks, effectively uncovering indirect capacity linkages and production redundancies that aid in managing supply shortages.

3. Problem description and model formulation

In Section 3.1, we present the order splitting problem. In Section 3.2, we model it as a binary integer problem, specifically formulated as a hypergraph partition problem based on established literature. Then, in Section 3.3, following the methods in Catalán and Fisher (2012) and Zhu et al. (2021), we simplify the original model and reformulate it as a graph partition problem. Finally, in Section 3.4, we details the real-world order data employed in this research and summaries the associated statistics.

3.1. Problem description

The order splitting can be reduced by strategically distributing SKUs across warehouses based on an analysis of overall multi-SKU order patterns. Considering warehouse management and operation practices in JD.com, we make the following assumptions:

1. The products are organized in categories, for the convenience of inventory management, that is, SKUs in the same category are stored in the same warehouse.
2. The inventory of each product is sufficient. Stock shortages are not considered.
3. Each category is stored in one at most warehouses. Duplication of categories among warehouses is not considered. In practice, duplication of storage reduces the order split but also increases the difficulty of inventory management: such as the replenishment of inventory in two different sites.
4. Due to the physical limitation of warehouses, there is an inventory capacity limitation in terms of stock quantities. The stock quantities of each category are calculated by the average amount in the past month.

3.2. Model formulation

Under the aforementioned assumptions, the order splitting problem constitutes a type of hypergraph partition problem. It can be formulated as a binary integer program to allocate n product categories across m warehouses, subject to capacity constraints. Since stockouts are not considered, order splitting depends solely on the combination of product categories within an order, irrespective of the quantities ordered. Consequently, for simplicity, we aggregate orders based on their unique product category combinations. Throughout the remainder of this article, the term order refers to such a unique combination. The notation for the binary integer model is provided in Table 2.

As a form of hypergraph partition problem, the order splitting minimization aims to distribute product categories across warehouses while satisfying capacity constraints. In this hypergraph structure, product categories serve as vertices, while combinations of product categories constitute hyperedges. The objective is to determine an optimal partition that minimizes the number of split hyperedges. We formally model this constrained partition problem as the following binary integer program:

Table 2
Model notations in hyper-graph partition model.

Parameters:	
i	Number of product categories;
I	Set of product categories;
j	Number of warehouses;
J	Set of warehouses;
k	Number of order types;
K	Set of order types;
I_k	Set of product categories i in combination k;
s_i	Stock quantity of product category i;
S_j	Stock capacity of warehouse j;
o_k	Number of orders of order type k;
Decision variables:	
$x_{i,j}$	1 if category i is assigned to warehouse j and 0 otherwise;
$y_{k,j}$	1 if order k is assigned to warehouse j, and 0 otherwise;

$$\min \sum_{k\in K}\sum_{j\in J} y_{k,j} o_k \tag{1}$$

$$\text{s.t.} \sum_{j\in J} x_{i,j} = 1 \qquad \forall i \in I \tag{2}$$

$$x_{i,j} \le y_{k,j} \qquad \forall i \in I_k,\ k \in K,\ j \in J \tag{3}$$

$$\sum_{i\in I} x_{i,j} {}^{*} s_i \le S_j \qquad \forall j \in J \tag{4}$$

$$x_{ij} \in \{1,0\} \qquad \forall i \in I,\ j \in J \tag{5}$$

$$y_{kj} \in \{1,0\} \qquad \forall k \in K,\ j \in J \tag{6}$$

The objective function (1) minimizes the number of order splits for each product combination k, equivalent to minimizing the number of warehouses involved in fulfilling each order. Constraint (2) enforces unique assignment by ensuring each product category i is allocated to exactly one warehouse. Constraint (3) establishes consistency: if category i is stored in warehouse j, any order k containing i must be fulfilled by j. Constraint (4) imposes warehouse capacity limits, guaranteeing that the total inventory volume assigned to any warehouse j does not exceed its storage capacity.

3.3. Model approximation and reformulation

To simplify the hypergraph partition model, existing studies often approximate it via graph partitioning. Catalán and Fisher (2012) pioneered a heuristic that projects hypergraphs onto ordinary graphs by constructing a coappearance matrix C, where edge weight $c_{i,j}$ is defined as $\sum_{k\in K} I_k(i) I_k(j) o_k$. Here, $I_k(i) = 1$ indicates product category i belongs to order combination k, and 0 otherwise. However, this approximation overweights SKU pairs from orders containing over two SKUs (Catalán & Fisher, 2012). Addressing this limitation, Zhu et al. (2021) introduced a correlation factor $\theta_k = \frac{2}{NCO_k}$, where NCO_k denotes the number of categories in order o_k. Hence, the edge weight between node i and j is calculated by $\sum_{k\in K} \theta_k I_k(i) I_k(j) o_k$. The revised edge weight mitigates overestimation, though it may still serve as an upper bound for actual order splitting in the hypergraph formulation.

Following the methodology of Zhu et al. (2021), we adopt the edge weight metric as an approximation for order splitting in the objective function. This transforms the original problem into a capacitated graph partition problem (see Table 3). Specifically, we consider a graph $G = (V, E)$, where V represents the set of n SKU categories and E denotes the edge set, with weights defined as $e_{i,j} = \sum_{k\in K} \theta_k I_k(i) I_k(j) o_k$. The reformulated problem requires partitioning SKU categories into m disjoint subsets (subject to warehouse capacity constraints) while minimizing the total weight of edges crossing distinct partitions. Adopting

Table 3
Model notations in graph partition model.

Parameters:	
i	Number of product categories;
I	Set of product categories;
j	Number of warehouses;
J	Set of warehouses;
s_i	Stock quantity of product category i;
S_j	Stock capacity of warehouse j;
e_{i_1,i_2}	Edge weight between category i_1 and i_2;
Decision variables:	
$x_{i,j}$	1 if category i is assigned to warehouse j and 0 otherwise;
z_{i_1,i_2}	1 if category i_1 and i_2 are assigned to same warehouse, 0 otherwise;

a node-based formulation for this capacitated graph partitioning (Ales & Knippel, 2020), we present the binary integer programming model below:

$$\min \sum_{i_1 \in I} \sum_{i_2 \in I} (1 - z_{i_1,i_2}) e_{i_1,i_2} \tag{7}$$

$$\text{s.t.} \sum_{j \in J} x_{i,j} = 1 \qquad \forall i \in I \tag{8}$$

$$\sum_{i \in I} x_{i,j} {}^* s_i \le S_j \qquad \forall j \in J \tag{9}$$

$$x_{i,j} = 0 \qquad \forall i < j \tag{10}$$

$$z_{i_1,i_2} - x_{i1,k} + x_{i2,k} \le 1 \qquad \forall i_1, i_2 \in I, k \in J \tag{11}$$

$$z_{i_1,i_2} + x_{i1,k} - x_{i2,k} \le 1 \qquad \forall i_1, i_2 \in I, k \in J \tag{12}$$

$$-z_{i_1,i_2} + x_{i1,k} + x_{i2,k} \le 1 \qquad \forall i_1, i_2 \in I, k \in J \tag{13}$$

$$x_{ij} \in \{1, 0\} \qquad \forall i \in I,\ j \in J \tag{14}$$

$$z_{i_1,i_2} \in \{1, 0\} \qquad \forall i_1, i_2 \in I \tag{15}$$

Objective function (7) minimizes the total weight of edges crossing partitions, serving as an approximation of the original model's objective. Constraint (8) enforces unique assignment by ensuring each category i is allocated to exactly one warehouse. Constraint (9) imposes warehouse capacity limits. Constraint (10) eliminates solution symmetry by restricting category $i \in I$ from assignment to any warehouse $j \in J$ with index greater than i (Ales & Knippel, 2020). Constraints (11)–(13) collectively enforce the triangle inequalities, guaranteeing $z_{i_1,i_2} = 1$ if categories i_1 and i_2 reside within the same warehouse.

This formulation significantly reduces model complexity. In the original hypergraph partition model, the number of possible order combinations grows exponentially with the number of SKU categories, resulting in $O(2^{I} {}^* J)$ binary variables. After approximation, the graph partition model requires only $O(I^{2} {}^* J)$ binary variables, achieving polynomial complexity.

We discuss the approximation bias for model reformulation. In this study, our primary goal is to minimize the total number of order splits in a hypergraph setting, which is transformed into a graph partition framework via clique expansion. The weight coefficient $\theta_k = \frac{2}{N}$ is assigned to each pairwise edge for an order with N SKUs, which ensures that the total weight of edges scales linearly with its size. There exists a structural bias, though it is negligible for orders with four or fewer SKUs. Without loss of generality, assuming an order with N SKUs is split into K sub-orders of size $\{x_1, \ldots, x_K\}$, the number of cut edges are $\frac{N^2-N}{2} - \sum_{i=1}^{K} \frac{x_i(x_i-1)}{2} = \frac{N^2-\sum_{i=1}^{K} x_i^2}{2}$. The graph cut penalty can be mathematically derived as $Cut = \frac{N^2-\sum_{i=1}^{K} x_i^2}{2} \times \frac{2}{N} = N - \frac{\sum_{i=1}^{K} x_i^2}{N}$. The cost function diverges from the number of order splits K in two primary ways. Firstly, the cost function is size-dependent. For example, consider a balanced K-way split $x_i = \frac{N}{K}$, the penalty function is derived as $N - \frac{N}{K}$. It implies that the penalty increases as the number of SKUs increases for a fixed K. This size dependency bias causes the algorithm to overly protect large orders. Secondly, there exists distribution bias. For a fixed number of order splits K, yet, the graph penalty heavily depends on the SKU distribution variance, which is $\sum_{i=1}^{K} x_i^2$. For example, assume an order with 50 SKUs is split into 2 sub-orders. A balanced split with 25 SKUs each yields a penalty of 25, while a skewed split with 49 SKUs and 1 SKU yields a penalty of about 2. Due to this distribution bias, the algorithm tends to isolate a single SKU from the large orders rather than a balanced splits.

Table 4
Descriptive statistics of the number of categories included in orders.

		No. of orders					Total
		1	2–4	5–9	10–99	≥100	
No. of categories	1	0.02%	0.07%	0.14%	3.04%	66.38%	69.65%
	2	4.62%	2.79%	1.46%	3.59%	3.60%	16.05%
	3	5.28%	0.59%	0.18%	0.29%	0.18%	6.52%
	4	3.03%	0.07%	0.02%	0.04%	0.02%	3.17%
	5–9	3.84%	0.02%	0.01%	0.03%	0.01%	3.91%
	≥10	0.69%	0.00%	0.00%	0.00%	0.00%	0.69%
	Total	17.48%	3.54%	1.81%	6.99%	70.18%	100.00%

Table 5
Descriptive statistics of the number of SKUs included in orders.

		No. of orders					Total
		1	2–4	5–9	10–99	≥100	
No. of SKUs	1	0.25%	1.04%	1.84%	18.31%	40.64%	62.08%
	2	9.56%	2.53%	1.20%	3.07%	2.12%	18.48%
	3	7.11%	0.44%	0.15%	0.38%	0.28%	8.36%
	4	4.09%	0.09%	0.03%	0.08%	0.05%	4.35%
	5–9	5.51%	0.04%	0.01%	0.04%	0.01%	5.61%
	≥10	1.11%	0.00%	0.00%	0.00%	0.00%	1.12%
	Total	27.64%	4.14%	3.24%	21.89%	43.10%	100.00%

3.4. Order data analysis

In this study, we utilize real-world order data from October 2024 and warehouse capacity records of JD.com's one business group at its one distribution center. The dataset comprises 10,139,925 orders encompassing 189,495 unique SKUs, organized hierarchically by product categories. For example, 330 ml Coca-Cola belongs to: first-level category Beverages and Mixes; second-level Beverages; third-level Carbonated Drinks; with vendor Coca-Cola China. Following JD.com's operational practice, products are managed through fourth-level categories (defined as third-level category plus vendor) for storage efficiency, ensuring SKUs within the same fourth-level category reside in a single warehouse. This storage strategy simplifies the optimization problem by shifting focus from SKU allocation to category allocation.

Our analysis classifies orders at the fourth-level category into single-category orders (containing single or multiple SKUs within the same category) and multi-category orders. As summarized in Table 4, 69.65% of orders are single-category and thus immune to splitting under any assignment. Multi-category orders predominantly (99.31%) contain 2–9 categories, representing the primary focus for order-splitting minimization, while orders with no less than 10 categories constitute only 0.69%. From an order volume perspective, 70.18% of orders occur more than 100 times monthly, of which 66.38% are single-category. Orders containing no less than 3 categories exhibit a long-tail distribution, with most combinations appearing only once monthly. Consequently, we prioritize optimizing orders with 2–9 categories. SKU-level analysis in Table 5 mirrors this pattern: 62.08% are single-SKU orders, and orders with no less than 10 SKUs account for 1.12%.

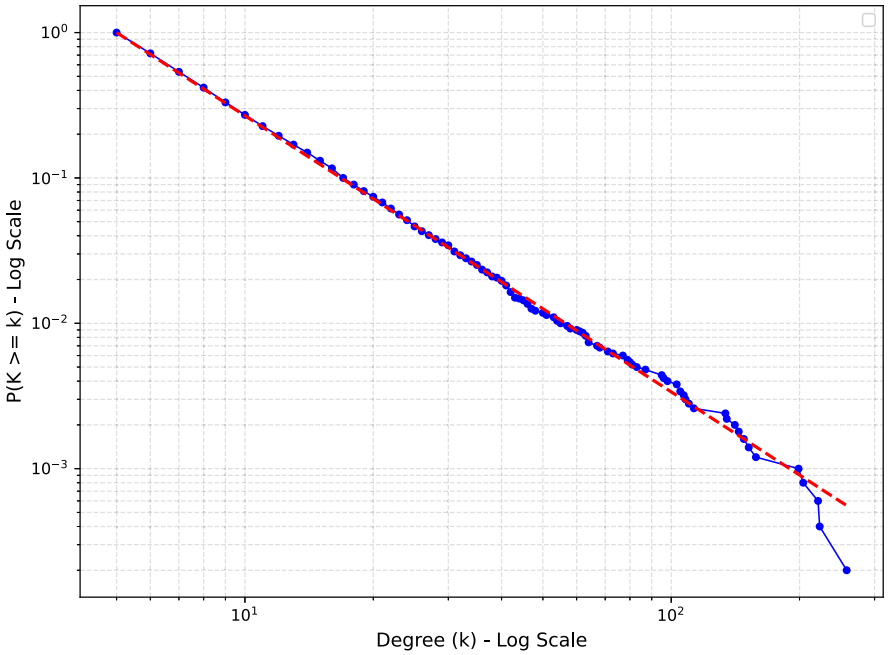


Fig. 3. Product category degree distribution.

4. Solution

4.1. Complex network analysis

We perform complex network analysis to characterize the product correlation graph. As shown in Fig. 3, the degree distribution reveals a linear cumulative probability $P(K > k)$ pattern. The exponential distribution of probabilities — evident from the logarithmic vertical axis — indicates an exponentially distributed network. This structure emerges through sequential growth: each iteration adds a new vertex that randomly connects to existing vertices (Deng et al., 2011). The network exhibits small-world properties with point-to-point product connections (Gong et al., 2018), aligning with the stochastic nature of online retail order structures.

4.2. Community switch algorithm

The order splitting minimization problem is reformulated as a capacitated graph partition problem, aiming to minimize the total edge weight across warehouse boundaries. Since the sum of all edge weights remains constant, this objective equivalently maximizes intra-warehouse edge weight aggregation. Consequently, assigning highly correlated products (those with strong edge connections) to the same partition reduces order splitting. As a computationally challenging NP-hard combinatorial optimization problem, capacitated graph partitioning becomes intractable at scale. To our knowledge, no prior study has addressed order splitting minimization at industrial-scale complexity. We therefore propose the K-Community Swap Algorithm (KCSA) for JD.com's operational context, which iteratively improves randomly initialized solutions until convergence or time constraints are met.

Fig. 4 illustrates the KCSA workflow. First, we construct the category graph $G(V, E)$ using the method from Zhu et al. (2021) to formulate the order splitting minimization problem. Starting from a randomly initialized solution, the algorithm iteratively refines partitions. During each iteration, it selects the two partitions with the highest inter-partition edge weight and applies a 2-Community Swap Algorithm (2-CSA) to reduce cross-partition connections. The procedure terminates when stopping criteria are satisfied, returning the solution achieving the minimal recorded cut value.

Algorithm 1 details the graph generation procedure. We initialize an empty graph where nodes represent product categories with weights corresponding to category stock quantities. Edge weights are computed as $e_{i,j} = \sum_{k \in K} \theta_k I_k(i) I_k(j) o_k$. For each order type k, we iterate through all category pairs (i_1, i_2) contained in k, incrementally updating the edge weight via $e_{i_1,i_2} \leftarrow e_{i_1,i_2} + o_k * \frac{2}{|I_k|}$. This process transforms the order splitting minimization problem into a capacitated graph partition problem.

Algorithm 1 Generate the graph of category relation.

```
Create a new graph G(V, E) and each node is one product category
Let the node weights equal to stock quantities of the corresponding
product category
Initialize e_{i1,i2} ← 0
for k ∈ K do
    for i1 ∈ I_k do
        for i2 ∈ I_k and i2 > i1 do
            e_{i1,i2} ← e_{i1,i2} + o_k * 2/|I_k|
        end for
    end for
end for
```

Our algorithm initializes the solution by distributing categories according to warehouse capacities, then iteratively refines it via the 2-Community Swap Algorithm (2-CSA) until reaching iteration limits or convergence. The initialization employs a first-fit-decreasing bin packing approach: categories are sorted by descending stock quantity and sequentially assigned to the first warehouse with sufficient capacity. At each iteration's start, all warehouse pairs are ranked by decreasing inter-warehouse edge weight sum—a direct indicator of order splitting magnitude, where higher values require prioritization. To achieve high efficiency and effectiveness, we leverage the similarity between min-cut problem and community detection problem and implement 2-CSA for targeted warehouse pair optimization.

Algorithm 2 details the 2-Community Swap Algorithm (2-CSA). Upon selecting a warehouse pair (j_1, j_2), we construct subgraph G' from their constituent nodes. This sub-problem requires partitioning G' into two capacity-constrained warehouses. As Fig. 5 demonstrates, reallocations within the pair preserve the total weight of inter-warehouse links

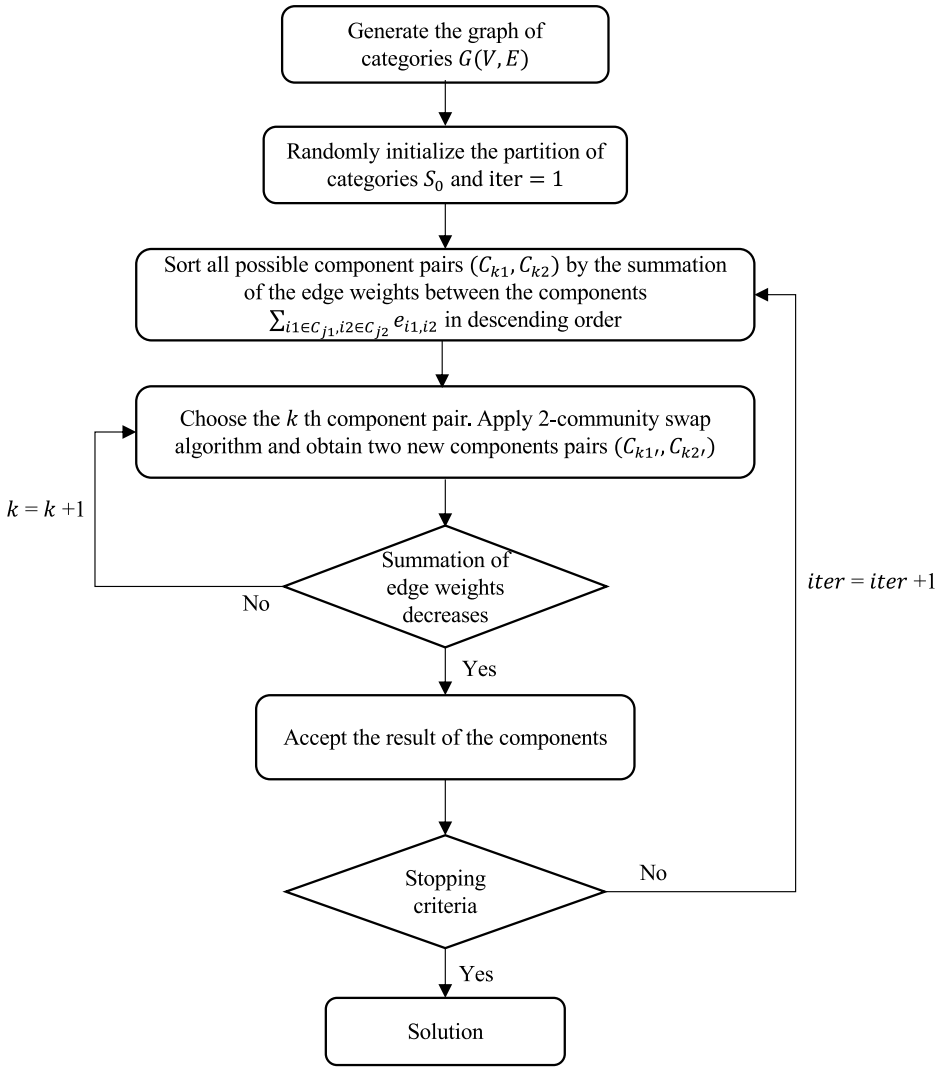


Fig. 4. Flowchart of the K-community swap algorithm.

between (j_1, j_2) and other warehouses. Therefore, any objective value reduction achieved locally in the sub-problem directly translates to a global reduction for the original problem.

To identify non-overlapping communities within the subgraph, we introduce two community detection algorithms based on different measures. Community detection in graphs attracts significant attention across research domains (Girvan & Newman, 2002; Gleiser & Danon, 2003; Guimera et al., 2005; Palla et al., 2005), with numerous algorithms proposed to uncover community structures under varying definitions (Traag et al., 2015). Modularity — a widely adopted measure — quantifies the relative edge density within communities versus external connections (Blondel et al., 2008). The Louvain algorithm efficiently optimizes modularity for large networks but suffers from the resolution limit and underestimates the number of communities. Conversely, surprise is a probability-based measure and outperforms modularity in benchmarks (Aldecoa & Marín, 2013). The asymptotic surprise algorithm (Traag et al., 2015) addresses the resolution limits yet tends to overestimate communities, making it suitable for detecting small communities missed by Louvain. That is, Louvain excels at identifying global structures, it tends to merge smaller communities and underestimates the number of communities. In contrast, Surprise method is effective at detecting smaller communities that are often overlooked by Louvain method. In order to avoid getting trapped in local optima biased by a single structural definition and to maintain robustness across graphs, we therefore integrate both algorithms to comprehensively characterize the community structure in the graph.

We utilize community detection methods to identify densely interconnected node clusters and partition them into communities characterized by high intra-community link density and sparse inter-community connections (Fortunato, 2010). However, community detection results may violate warehouse capacity constraints or yield more communities than available partitions. To address this, the 2-CSA first assigns the highest-stock community to the largest-capacity base warehouse. Subsequent communities are allocated to this base warehouse based on a reward metric computed as $(\sum_{i_1\in C, i_2\in A_2\setminus C} e_{i_1,i_2} - \sum_{i_1\in C, i_2\in A_1} e_{i_1,i_2})/S_C$. This sequential assignment continues until termination conditions are met. Finally, the new partition is adopted only if it reduces the total inter-warehouse edge weight.

To guarantee reproducibility, our implementation uses a globally fixed random seed to initialize the random number generator. This state is passed down to all stochastic sub-processes, including the random selection of Louvain/Surprise methods and the internal community detection methods. The KCSA algorithm employs a maximum iteration limit as the stopping criterion and a hill climbing criterion to accept a new feasible solution with a strictly smaller cut penalty. In our context, K represents the number of available warehouses in the given instances. We utilize a resolution parameter to control the granularity of Louvain algorithm. We summarize all the algorithmic parameters for KCSA, which is shown in Table 6.

Table 6
Algorithmic parameters for KCSA.

Parameter	Value	Description
K	Instance-dependent	Strictly corresponds to the actual number of warehouses.
Global random seed	2345 (Baseline)	Ensures exact reproducibility.
Maximum iteration	20	Maximum iterations to balance solution quality and computational time.
Acceptance criterion	Hill climbing	Greedily accepts strictly improving solutions.
Selection of Louvain/Surprise	Random select	Uniformly chooses between Louvain and Surprise.
Resolution	10	Controls the granularity of Louvain when splitting dense subgraphs.

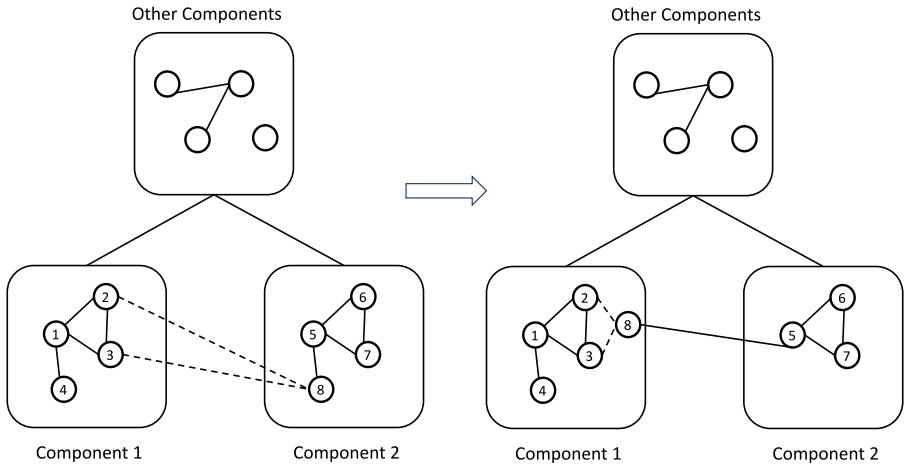


Fig. 5. Community swap between warehouses in 2-community swap algorithm.

An illustrative example comprising 12 nodes is constructed to demonstrate the procedure of the 2-CSA algorithm. As shown in Fig. 6, the warehouse capacity is set to 6, with edge weights between nodes explicitly labeled. The initial partition yields a cut value of 4. This represents a local optimum that cannot be improved through standard node-level swap or reinsertion operators. Within the 2-CSA framework, community detection is first applied to the subgraph, identifying four distinct communities. Subsequently, the largest community (Community 1) is assigned to Warehouse 1. At this stage, the solution becomes infeasible because the remaining eight nodes are assigned to Warehouse 2, resulting in a total cut of 5. The remaining communities are then sorted based on their connectivity to Community 1. Community 4 is selected for reassignment, as this operation reduces the cut from 5 to 2. Following this assignment, feasibility is restored, and the objective value of 2 improves upon the original cut. As Warehouse 1 reaches its capacity limit, the 2-CSA algorithm terminates, yielding an optimized partition.

5. Numerical experiments

Section 5.1 details the numerical experiment settings and summarizes order data from JD.com, China's largest online retailer. Section 5.2 incorporates warehouse stock capacity constraints. We evaluate KCSA against the exact Gurobi solver on small-scale datasets, and verify its effectiveness on large-scale datasets by comparison with the K-Links Clustering Algorithm (KLCA) (Zhu et al., 2021). Section 5.3 introduces cardinality constraints (category count limits), aligning with KLCA and the BESTSELLERS algorithm (Catalán & Fisher, 2012). KCSA outperforms both benchmarks across all instance scales. Section 5.4 analyzes fourth-tier instances (22,285 categories), demonstrating KCSA's superior computational efficiency and solution quality in large-scale scenarios, whereas the existing KLCA algorithm has larger optimality gaps with increasing problem size. Section 5.5 compares the KCSA with graph partitioning algorithms to comprehensively evaluate the efficiency. The evaluation also helps to understand the reason why KCSA is suited for this order splitting problem. Section 5.6 shows the workflow of KCSA in real practice in JD.com. All algorithms were implemented in Python 3.7 on an Intel Xeon 2.6 GHz server with 40 GB RAM. We use 56 threads in Gurobi solver to speed up the solution.

5.1. Experiment setting

In this experiment, we comprehensively evaluate the KCSA algorithm's performance using different category tiers to generate multi-scale instances. Small-scale instances derive from 66 first-tier or 218 second-tier categories, while medium-scale instances utilize 1005 third-tier and large-scale instances include 22,285 fourth-tier categories. The dataset encompasses 10,139,925 orders. Since stock shortages are not considered, category combinations become the primary factor in minimizing split orders. Order type frequencies vary significantly across category levels: small-scale instances contain 29,677 first-tier and 93,512 second-tier order types, whereas medium-scale instances comprise 678,713 third-tier and 1,979,864 fourth-tier order types for large-scale instances.

Stock quantities for each SKU are sourced from JD.com's actual inventory data, with a total inventory volume of 23,618,054 units. Experiments test varying warehouse counts across different-scale datasets, where warehouse capacities are proportionally generated under the assumption of identical warehouse specifications to ensure balanced workload distribution. All experimental parameters are summarized in Table 7.

To assess KCSA efficiency in small-to-medium-scale instances, we benchmark it against the BESTSELLER (Catalán & Fisher, 2012) and K-Links Clustering Algorithm (KLCA) (Zhu et al., 2021) methods in the literature. Existing approaches assume SKU homogeneity and impose cardinality constraints (limiting categories/SKUs per warehouse), disregarding SKU heterogeneity. In Section 5.2, we adapt KLCA to incorporate stock capacity constraints by modifying its exchange and relocation movement strategies: all generated movements must satisfy capacity feasibility. SKU heterogeneity reduces the number of feasible exchange movements, potentially impacting KLCA performance. Consequently, Section 5.3 compares KLCA and BESTSELLER under cardinality constraints. Our comparison focuses on heuristics such as KLCA and BESTSELLER, which are specifically tailored to the order

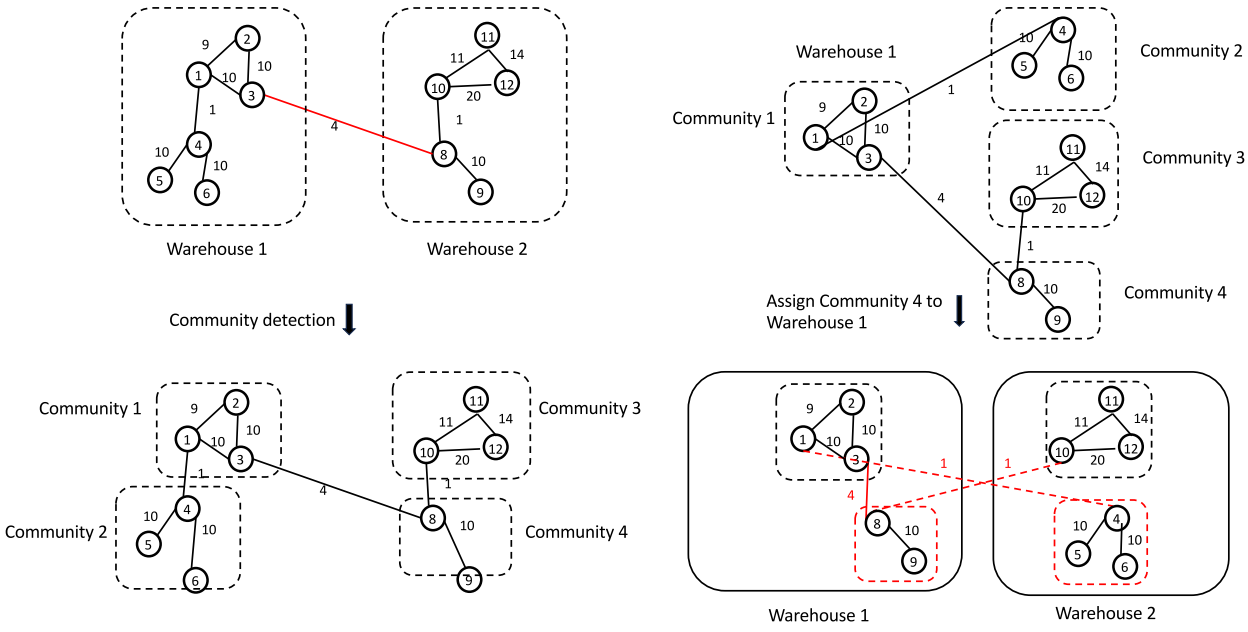


Fig. 6. An illustrative example for 2-CSA.

Table 7
Parameters of the experiment setting.

Parameters	Small-scale		Medium-scale	Large-scale
	First-tier	Second-tier	Third-tier	Fourth-tier
No. of categories	66	218	1005	22,285
No. of warehouses	[2, 6]	[2, 6]	[2, 6]	[2, 6]
No. of orders	10,139,925	10,139,925	10,139,925	10,139,925
No. of multi-category orders	1,364,886	2,126,769	2,760,359	3,077,490
No. of types of orders	29,677	93,512	678,713	1,979,864
No. of types of multi-category orders	29,614	93,304	677,753	1,959,021

splitting problem. To contextualize these baselines within the order-splitting problem setting, we first consider the BESTSELLERS heuristic proposed by Catalán and Fisher (2012). This method initially allocates the $B = \left\lfloor \frac{\sum_d k_d - S}{D-1} \right\rfloor$ top-selling SKUs to all warehouses, where k_d denotes the capacity of warehouse d, S represents the number of categories, as D is the number of warehouses. Given that our study assumes non-overlapping category storage, we adapted the BESTSELLERS heuristic such that the top-selling categories are assigned to a single warehouse, while the remaining categories are allocated to the warehouse with the strongest connectivity. Building on this foundation, Zhu et al. (2021) refined the link calculation by introducing a connection coefficient to mitigate the overestimation effects. Their KLCA, a type of Variable Neighborhood Search, employs iterative relocation and exchange strategies to optimize assignments. Specifically, the relocation strategy shifts a single product category from one warehouse to another to reduce split orders, subject to capacity constraints. However, this approach can get trapped in local optima when warehouses reach maximum capacity. To address this limitation, the exchange strategy is utilized to simultaneously swap two categories between different warehouses.

5.2. Instances with stock capacity constraints

This section evaluates KCSA performance against the exact Gurobi solver on small-scale instances, comparing both total order splits (objective function) and computational time. We test warehouse configurations from 2 to 6 across first-tier and second-tier category instances, with warehouse capacities proportionally allocated based on total stock quantity. Each instance executes KCSA over 20 iterations, retaining the best solution. Gurobi runs with a 40-hour time limit (144,000 s), terminating early if optimality is proven.

Table 8 presents MIP and KCSA algorithm results for small-scale instances, comparing total order splits (objective function) and computation times. Columns 1–3 specify warehouse counts, category quantities, and warehouse capacities. Columns 4–6 and 7–9 display Gurobi results for the original hypergraph partition (HGP) and reduced graph partition (GP) problems, with Column 7 indicating the objective gap between HGP and GP solutions. Columns 10–13 report KCSA results alongside gaps relative to Gurobi's HGP and GP solutions. Solving the original HGP formulation via Gurobi requires several hours due to complexity, with computation time escalating dramatically with increasing warehouses and categories—reaching the 40-hour limit (144,000 s) for the 3-warehouse/218-category instance. Split orders increase proportionally with warehouse counts. Notably, GP solutions achieve less than 0.3% optimality gaps while consuming substantially less time than HGP for solvable instances, demonstrating the approximation's negligible objective impact with radical complexity reduction. However, GP remains NP-Hard, hence Gurobi still hits time limits for larger instances.

Compared to Gurobi solutions, the proposed KCSA algorithm demonstrates outstanding performance with optimality gaps below 3% across most instances—excepting the 3-warehouse/66-category case. Notably, KCSA consistently converges to optimal solutions within 20 s, significantly outperforming both hypergraph (HGP) and graph partition (GP) formulations solved via Gurobi.

For the 3-warehouse/66-category instance, the KCSA solution exhibits an outlier gap of 4.93%. To investigate this anomaly, we perform a case analysis to identify underlying causes. Fig. 7 compares the graph partition results between the KCSA and the exact solution from Gurobi solver. To quantify structural differences, we characterized core

Algorithm 2 2-Community Swap Algorithm (2-CSA)

Require:
Two warehouses j_1, j_2 with $S_{j_1} \geq S_{j_2}$
Node sets N_{j_1}, N_{j_2} and stock capacities S_{j_1}, S_{j_2}
Full graph $G = (V, E)$
Ensure:
New assignments A_1 for j_1, A_2 for j_2

procedure 2-Community-Swap(j_1, j_2, G)
  $G' \leftarrow$ subgraph induced by $N_{j_1} \cup N_{j_2}$ ▷ Step 1: Construct subgraph
  $\mathcal{C} \leftarrow$ CommunityDetection(G') ▷ Step 2: Apply random community detection
  $\mathcal{C} \leftarrow$ Sort($\mathcal{C}$, stock_quantity, descending) ▷ Step 3: Sort communities
  while $s_{C_1} > S_{j_1}$ **do** ▷ Step 4: Handle oversized communities
    if C_1 is indivisible **then**
      Partition C_1 into $\{C_{1a}, C_{1b}\}$ via node removal
      Update $\mathcal{C} \leftarrow \{C_{1a}, C_{1b}\} \cup C_{2:k}$
    else
      $\mathcal{C} \leftarrow$ CommunityDetection(C_1)
    end if
    Resort $\mathcal{C}$
  end while
  $A_1 \leftarrow C_1$, $A_2 \leftarrow \bigcup_{i=2}^{k} C_i$ ▷ Step 5: Initial assignment
  $S_{\text{total}} \leftarrow \sum_{i \in N_{j_1} \cup N_{j_2}} s_i$
  repeat ▷ Step 6: Community reassignment
    for $C \in \{C_2, \ldots, C_k\}$ **do**
      $$R_C \leftarrow \frac{\sum_{\substack{i_1 \in C \\ i_2 \in A_2 \setminus C}} e_{i_1,i_2} - \sum_{\substack{i_1 \in C \\ i_2 \in A_1}} e_{i_1,i_2}}{s_C}$$
    end for
    $C_{\text{res}} \leftarrow$ Sort($\{C_2, \ldots, C_k\}, R_C$, ascending)
    if $S_{A_1} \leq S_{j_1}$ **and** $S_{A_1} \geq S_{\text{total}} - S_{j_2}$ **then**
      break if $R_{C_2} > 0$ **or** $S_{A_1} + s_{C_2} > S_{j_1}$
    end if
    for $C \in C_{\text{res}}$ **do**
      if $S_{A_1} + s_C \leq S_{j_1}$ **then**
        $A_1 \leftarrow A_1 \cup C$, $S_{A_1} \leftarrow S_{A_1} + s_C$
        if $R_C > 0$ **then break**
      end if
    end for
  until no improvement
  $A_2 \leftarrow (N_{j_1} \cup N_{j_2}) \setminus A_1$ ▷ Step 7: Finalize assignment
  $$L_{\text{new}} \leftarrow \sum_{\substack{i_1 \in A_1 \\ i_2 \in A_2}} e_{i_1,i_2}$$
  $$L_{\text{old}} \leftarrow \sum_{\substack{i_1 \in N_{j_1} \\ i_2 \in N_{j_2}}} e_{i_1,i_2}$$
  if $L_{\text{new}} < L_{\text{old}}$ **then**
    return (A_1, A_2)
  else
    return (N_{j_1}, N_{j_2}) ▷ Keep original assignment
  end if
end procedure

nodes and edges in Fig. 7(b), revealing significant assignment discrepancies at nodes 3 and 5. The first subgraph in Fig. 7(b) illustrates the partition backbone generated by KCSA, while the second corresponds to the optimal solution. The third subgraph details partitioning for nodes {0,1,3,4,5,9} in the optimal solution, where identical colors denote shared partition groups. Here, solid red lines designate newly introduced cuts, and dashed blue lines represent cuts reduced in the optimal solution. Analysis reveals KCSA fails to separate node pairs (5,9) and (3,4) due to their strong internal connectivity. This case demonstrates that when warehouse capacity constraints necessitate splitting densely connected clusters, KCSA may yield suboptimal solutions with relatively high gaps. However, as instance scale increases, such capacity-induced partition splits become statistically less probable. Consequently, KCSA's performance limitation diminishes significantly with increasing problem scale, as empirically validated by our later comprehensive benchmarks across partitioned graph instances. As is shown in the above 3-warehouse/66-category instance, KCSA struggles when warehouse capacity constraints necessitate splitting densely connected clusters. From a theoretical perspective, this weakness originates from the KCSA's community detection phase, where nodes are divided into communities with strong internal connectivity. This process is structurally similar to the coarsening phase in classical multi-level graph partitioning algorithms. While the community with strongly connected nodes is treated as one super node in the following stages in KCSA, it inevitably leads to the loss of granularity. That is, when a rigid capacity constraint necessitate splitting the communities to obtain the optimal solution, such as communities (5,9) and (3,4) in the instance, the KCSA framework is trapped. Because the nodes in communities are already locked together in the community detection stage, the algorithm is not able to assign nodes in the same community to different warehouses, thereby yielding a suboptimal gap.

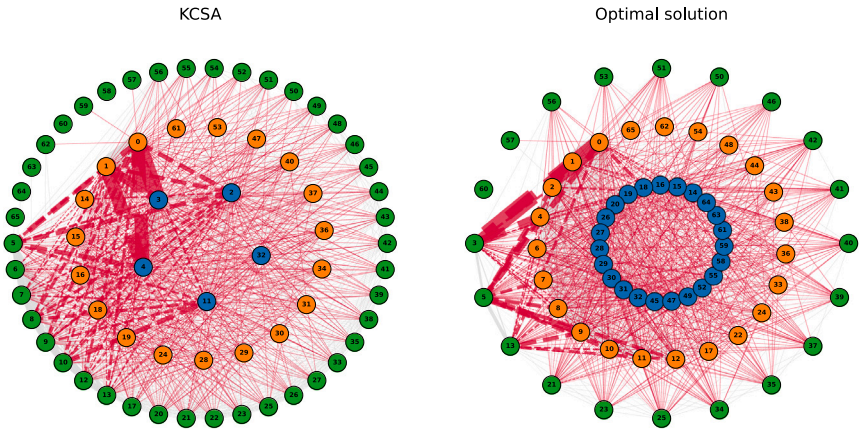
Table 9 presents comparative results. Columns 1–3 indicate warehouse counts, category quantities, and warehouse capacities. Columns 4–5 show KCSA's objective values and computational times. Columns 6–8 and 9–11 detail KCSA variants using Louvain and Surprise community detection algorithms, respectively. Columns 12–14 report modified KLCA results. The best objective value per instance appears in bold. Both community detection algorithms exhibit complementary performance: KCSA with either method typically achieves marginally superior objectives with comparable runtime efficiency. KLCA produces feasible solutions with approximately 10% optimality gaps in first/second-tier instances at runtimes similar to KCSA, but suffers significant computational degradation in third-tier cases (20% gaps with longer runtimes), indicating limited scalability of its exchange/relocation strategies for large-scale problems.

With warehouse capacities calculated proportionally to warehouse counts, overall capacity buffers remain minimal. Table 10 compares Gurobi (solving GP), KCSA, KCSA-Louvain, KCSA-Surprise, and KLCA algorithms on first-tier instances under varying capacity scenarios. Results demonstrate KCSA achieves the smallest optimality gap with significantly reduced computation time relative to Gurobi's GP solutions, confirming its robustness across diverse capacity conditions.

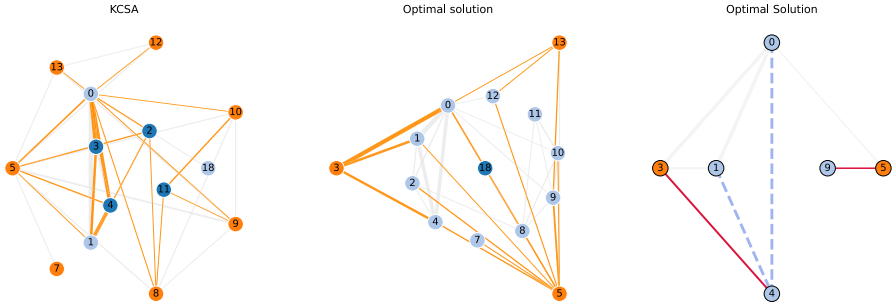
5.3. Instances with number of categories constraints

This subsection evaluates the KCSA algorithm under category-based warehouse capacity constraints (number of categories rather than stock quantities), aligning with Catalán and Fisher (2012) and Zhu et al. (2021). Table 11 details KCSA results for small-scale instances with category-count constraints. Columns 1–3 denote warehouse counts, category quantities, and warehouse capacities. MIP in HGP indicates hypergraph partition results solved by Gurobi, while MIP in GP denotes graph partition solutions. KCSA achieves objective gaps of −0.50% to 3.57% relative to MIP in HGP (which shows marginal superiority in some cases) with computational times under 50 s.

We compare the three different algorithms to validate the performance of proposed KCSA algorithm. The results are presented in Tables 12 and 13. The KCSA algorithm obtains the best solution compared to KLCA proposed by Zhu et al. (2021) and BESTSELLER algorithm proposed by Catalán and Fisher (2012). Generally, the objective gap between KLCA and BESTSELLER algorithms with KCSA algorithm increases as the number of warehouses increases. The KLCA algorithm achieves smaller gaps with relatively longer computational time in most instances, compared to the BESTSELLER algorithm. Table 13 shows the results of the three algorithms under different warehouse



(a) Comparison of the solutions between KCSA and optimal solution.



(b) Core differences between the KCSA and optimal solutions.

Fig. 7. Visualization of the 3-warehouse/66-category instance. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 8
Results of the small-scale instances with stock capacity constraints.

No. of warehouses	No. of categories	Warehouse capacity	MIP in HGP		MIP in GP			KCSA			
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap with HGP	Gap with GP	Time/s
2	66	12,000,000	2,505,932	3703.47	2,505,934	0.00%	1.95	2,505,932	0.00%	0.00%	4.80
3	66	7,900,000	3,106,142	6877.45	3,106,142	0.00%	14.61	3,259,403	4.93%	4.93%	5.93
4	66	6,000,000	3,418,872	28 484.73	3,418,583	−0.01%	41.96	3,428,660	0.29%	0.29%	6.03
5	66	4,800,000	3,734,722	18 813.95	3,735,014	0.01%	34.16	3,739,825	0.14%	0.13%	6.46
6	66	4,000,000	3,964,340	16 446.03	3,962,370	−0.05%	35.93	3,988,393	0.61%	0.66%	6.25
2	218	12,000,000	2,794,855	21 057.03	2,801,184	0.23%	25.18	2,794,856	0.00%	−0.23%	15.88
3	218	7,900,000	3,524,084	144 000	3,393,488	−3.71%	3472.50	3,455,683	−1.94%	1.83%	17.65
4	218	6,000,000	4,090,501	144 000	3,647,381	−10.83%	16 627.54	3,652,692	−10.70%	0.15%	18.10
5	218	4,800,000	4,461,212	144 000	4,130,858	−7.41%	144 000	4,068,873	−8.79%	−1.50%	17.73
6	218	4,000,000	4,601,030	144 000	4,142,068	−9.98%	144 000	4,243,172	−7.78%	2.44%	18.83

capacities. As the warehouse capacity increases, the KLCA algorithm produces large objective gaps in some instances while the gap of BESTSELLER algorithm tends to decrease. That is consistent with our expectation, because that as the warehouse capacity increases, the top selling categories are likely to be assigned to the same warehouse. The KCSA algorithm proposed in this study also obtains the best solution in most instances, verifying the efficiency and effectiveness of this algorithm. We benchmark KCSA with the existing algorithms in the literature, focusing on the optimality gaps of the algorithms. As shown in Table 12, the BESTSELLER algorithm is unstable in solution quality, with optimality gaps reaching nearly 19.6% in instance involving 218 categories and 6 warehouses. KLCA outperforms BESTSELLER in most instances, but it fails to maintain consistency with varying capacity constraints. This limitation is observed in Table 13. However, the proposed KCSA demonstrates superior performance, consistently achieving the lowest objective values across almost all instances and significantly outperforms both the BESTSELLER and the KLCA approaches.

5.4. Fourth-tier instances in JD.com's implications

JD.com mandates fourth-tier categories as the minimum storage granularity, requiring identical fourth-tier products to share warehouse allocation. Consequently, evaluating KCSA and benchmark algorithms in large-scale instances becomes essential. Table 14 compares KCSA, KCSA-Louvain, KCSA-Surprise, and KLCA on fourth-tier instances under varying capacities. KCSA achieves optimal solutions in four of five cases, though its computational time increases significantly in large-scale scenarios due to the Surprise algorithm's higher computational load versus Louvain (which achieves less than 3% optimality gaps in most cases). KLCA exhibits comparable runtime to KCSA-Louvain but suffers from 30%–40% optimality gaps, indicating susceptibility to local optima as problem scale increases. Fig. 8 tracks KLCA's optimization trajectory for a 6-warehouse/4,000,000-capacity fourth-tier instance and the total runtime is 39,063.18 s, revealing a best objective

Table 9
Results of the algorithms with stock capacity constraints.

No. of warehouses	No. of categories	Warehouse capacity	KCSA		KCSA with Louvain			KCSA with Surprise			KLCA		
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
2	66	12,000,000	**2,505,932**	11.97	**2,505,932**	0.00%	12.16	**2,505,932**	0.00%	12.26	2,505,961	0.00%	17.66
3	66	7,900,000	**3,259,403**	13.49	3,260,891	0.05%	13.25	3,261,429	0.06%	14.02	3,462,781	6.24%	18.09
4	66	6,000,000	**3,428,660**	13.89	3,429,508	0.02%	13.46	3,539,998	3.25%	11.02	3,616,174	5.47%	18.84
5	66	4,800,000	**3,739,825**	14.80	3,917,707	4.76%	15.02	3,855,694	3.10%	12.85	3,839,001	2.65%	18.58
6	66	4,000,000	3,988,393	16.14	**3,987,337**	−0.03%	15.25	3,997,942	0.24%	15.34	4,142,771	3.87%	18.03
2	218	12,000,000	**2,794,856**	41.03	2,801,062	0.22%	39.61	**2,794,856**	0.00%	42.55	3,162,651	13.16%	48.84
3	218	7,900,000	**3,455,683**	46.64	3,470,004	0.41%	42.86	3,516,039	1.75%	39.55	3,868,315	11.94%	51.09
4	218	6,000,000	**3,652,692**	47.82	3,712,449	1.64%	45.19	3,704,985	1.43%	46.17	3,990,892	9.26%	53.34
5	218	4,800,000	4,068,873	45.63	**4,037,691**	−0.77%	45.74	4,048,718	−0.50%	49.19	4,315,626	6.06%	52.77
6	218	4,000,000	**4,243,172**	44.92	4,259,889	0.39%	47.12	4,279,207	0.85%	45.27	4,478,802	5.55%	51.56
2	1005	12,000,000	**3,428,374**	400.61	3,434,553	0.18%	401.16	**3,428,374**	0.00%	427.37	3,877,417	13.10%	480.80
3	1005	7,900,000	**4,034,187**	523.52	4,086,513	1.30%	467.22	4,079,571	1.12%	352.36	5,009,347	24.17%	493.18
4	1005	6,000,000	4,405,624	507.08	**4,268,429**	−3.11%	461.19	4,375,544	−0.68%	386.32	5,296,419	20.22%	489.97
5	1005	4,800,000	4,698,901	444.90	**4,636,931**	−1.32%	429.26	4,721,421	0.48%	402.85	5,409,621	15.13%	470.43
6	1005	4,000,000	4,837,177	444.27	5,315,677	9.89%	370.52	**4,787,762**	−1.02%	455.45	5,833,854	20.60%	487.74

Table 10
Results of the algorithms with different capacities under stock capacity constraints.

No. of warehouses	No. of categories	Warehouse capacity	MIP in GP		KCSA			KCSA with Louvain			KCSA with Surprise			KLCA		
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
6	66	4,000,000	3,962,370	37.19	3,988,393	0.66%	14.90	**3,987,337**	0.63%	15.04	3,997,942	0.90%	15.12	4,142,771	4.55%	14.94
6	66	4,400,000	3,867,649	61.54	**3,873,585**	0.15%	15.15	3,940,869	1.89%	12.63	3,938,574	1.83%	15.51	3,964,413	2.50%	14.98
6	66	4,800,000	3,735,014	103.5	**3,739,825**	0.13%	15.02	3,917,707	4.89%	15.33	3,855,694	3.23%	13.09	3,896,246	4.32%	14.92
6	66	5,200,000	3,635,802	177.62	3,679,200	1.19%	14.62	3,725,667	2.47%	14.08	3,707,768	1.98%	14.59	**3,637,269**	0.04%	14.96
6	66	5,600,000	3,559,940	236.91	**3,635,233**	2.12%	15.08	3,676,133	3.26%	14.32	3,705,623	4.09%	15.30	3,836,200	7.76%	14.76
6	66	6,000,000	3,418,583	180.36	**3,428,660**	0.29%	13.99	3,429,508	0.32%	14.06	3,539,998	3.55%	11.15	3,616,204	5.78%	14.69
6	66	6,400,000	3,288,743	182.46	**3,291,152**	0.07%	13.64	3,380,184	2.78%	14.17	3,299,544	0.33%	14.12	3,400,191	3.39%	14.65
6	66	6,800,000	3,262,366	193.83	**3,266,465**	0.13%	13.33	3,313,228	1.56%	13.75	3,305,721	1.33%	14.39	3,267,012	0.14%	14.29
6	66	7,200,000	3,253,390	204.22	3,236,887	−0.51%	14.10	3,260,231	0.21%	13.71	**3,235,242**	−0.56%	14.12	3,235,403	−0.55%	14.50
6	66	7,600,000	3,133,842	113.62	**3,161,263**	0.87%	14.00	3,264,908	4.18%	13.49	3,163,185	0.94%	14.26	3,164,589	0.98%	14.37
6	66	8,000,000	3,101,025	106.16	3,123,912	0.74%	13.40	**3,123,784**	0.73%	13.51	3,259,839	5.12%	9.48	3,384,667	9.15%	14.08

Table 11
Results of the small-scale instances with number of categories constraints.

No. of warehouses	No. of categories	Warehouse capacity	MIP in HGP			MIP in GP			KCSA		
			Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
2	66	34	1,848,820	–	22.83	1,845,695	−0.17%	0.34	1,845,829	−0.16%	12.70
3	66	23	1,871,976	–	50.24	1,871,872	−0.01%	0.21	1,871,669	−0.02%	13.42
4	66	17	2,020,254	–	158.13	2,014,905	−0.26%	2.51	2,013,699	−0.32%	12.64
5	66	14	2,154,705	–	3805.61	2,154,709	0.00%	9.77	2,154,811	0.00%	13.63
6	66	12	2,329,499	–	14 637.27	2,331,437	0.08%	17.62	2,412,640	3.57%	15.49
2	218	110	2,138,905	–	101.89	2,134,162	−0.22%	5.6	2,136,175	−0.13%	40.12
3	218	73	2,147,346	–	386.83	2,147,196	−0.01%	2.35	2,148,727	0.06%	43.45
4	218	55	2,194,663	–	1521.37	2,189,053	−0.26%	47.78	2,193,883	−0.04%	45.72
5	218	44	2,237,673	–	4518.36	2,228,247	−0.42%	225.7	2,230,987	−0.30%	46.74
6	218	37	2,287,260	–	6702.1	2,274,918	−0.54%	401.97	2,275,847	−0.50%	47.11

Table 12
Results of the algorithms with number of categories constraints.

No. of warehouses	No. of categories	Warehouse capacity	KCSA		KCSA with Louvain			KCSA with Surprise			KLCA			BESTSELLER		
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
2	66	34	1,845,829	12.70	1,849,786	0.21%	12.60	1,845,829	0.00%	12.40	**1,845,695**	−0.01%	12.91	1,849,834	0.22%	1.73
3	66	23	1,871,669	13.42	1,947,381	4.05%	13.32	1,871,723	0.00%	13.28	**1,871,637**	0.00%	14.19	1,883,048	0.61%	1.73
4	66	17	**2,013,699**	12.64	2,086,149	3.60%	14.56	2,013,771	0.00%	10.71	2,014,648	0.05%	12.15	2,408,028	19.58%	1.74
5	66	14	2,154,811	13.63	2,534,938	17.64%	14.70	2,154,902	0.00%	11.92	**2,154,709**	0.00%	14.27	2,250,517	4.44%	1.73
6	66	12	2,412,640	15.49	2,564,841	6.31%	15.65	**2,329,782**	−3.43%	14.70	2,564,494	6.29%	14.66	2,545,607	5.51%	1.78
2	218	110	2,136,175	40.12	2,136,175	0.00%	39.73	2,136,175	0.00%	40.12	**2,134,151**	−0.09%	41.24	2,143,422	0.34%	7.10
3	218	73	2,148,727	43.45	2,169,257	0.96%	41.54	2,149,730	0.05%	42.52	**2,147,210**	−0.07%	44.33	2,340,650	8.93%	7.08
4	218	55	2,193,883	45.72	2,226,912	1.51%	43.34	2,194,211	0.01%	43.45	**2,190,107**	−0.17%	43.95	2,458,781	12.07%	6.89
5	218	44	**2,230,987**	46.74	2,285,089	2.43%	46.01	2,237,258	0.28%	44.84	2,235,704	0.21%	44.70	2,582,807	15.77%	6.90
6	218	37	**2,275,847**	47.11	2,381,680	4.65%	46.41	2,295,359	0.86%	34.69	2,285,993	0.45%	43.19	2,721,692	19.59%	7.02
2	1005	510	**2,784,781**	405.16	2,793,434	0.31%	400.40	**2,784,781**	0.00%	414.71	2,785,400	0.02%	451.13	2,794,117	0.34%	62.53
3	1005	340	**2,844,736**	455.93	2,887,346	1.50%	444.21	2,845,182	0.02%	453.19	2,860,041	0.54%	489.56	2,905,857	2.15%	61.41
4	1005	260	**2,930,182**	483.30	3,063,437	4.55%	472.08	2,931,697	0.05%	425.32	2,985,478	1.89%	464.96	3,036,509	3.63%	62.19
5	1005	210	**3,053,479**	498.10	3,285,864	7.61%	442.48	3,067,323	0.45%	479.63	3,191,621	4.52%	460.46	3,184,874	4.30%	62.14
6	1005	170	**3,220,385**	463.92	3,351,014	4.06%	463.81	3,224,969	0.14%	423.91	3,253,502	1.03%	457.75	3,379,641	4.95%	61.56

Table 13
Results of the algorithms with different capacities under number of categories constraints.

No. of warehouses	No. of categories	Warehouse capacity	MIP in GP		KCSA			KCSA with Louvain			KCSA with Surprise			KLCA			BESTSELLER		
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
6	66	12	2,331,437	17.46	2,412,640	3.48%	15.21	2,564,841	6.31%	15.83	**2,329,782**	−3.43%	15.29	2,564,494	10.00%	14.32	2,545,607	5.51%	1.74
6	66	14	2,154,706	14.73	2,154,811	0.00%	13.70	2,534,938	17.64%	14.59	2,154,902	0.00%	11.24	**2,154,708**	0.00%	13.61	2,250,517	4.44%	1.67
6	66	15	2,105,427	5.05	**2,105,427**	0.00%	14.04	2,534,132	20.36%	14.25	2,108,043	0.12%	10.01	2,527,272	20.04%	13.91	2,128,634	1.10%	1.76
6	66	16	2,052,091	8.16	**2,051,986**	−0.01%	14.10	2,558,444	24.68%	14.24	2,052,023	0.00%	13.90	2,525,455	23.07%	13.80	2,097,387	2.21%	1.74
6	66	17	2,015,001	7.64	**2,013,699**	−0.06%	12.20	2,086,149	3.60%	13.82	2,013,771	0.00%	10.20	2,014,648	−0.02%	11.30	2,049,759	1.79%	1.70
6	66	18	1,975,961	4.03	**1,975,879**	0.00%	13.29	2,032,324	2.86%	13.76	1,975,882	0.00%	13.80	1,977,245	0.06%	12.12	2,012,419	1.85%	1.72
6	66	20	1,920,472	0.49	**1,919,461**	−0.05%	13.88	2,510,274	30.78%	14.29	**1,919,461**	0.00%	13.00	2,511,280	30.76%	13.94	1,962,405	2.24%	1.68
6	66	21	1,895,576	0.85	1,895,526	0.00%	13.71	2,512,836	32.57%	13.95	**1,895,507**	0.00%	12.91	2,514,122	32.63%	13.67	1,900,772	0.28%	1.73
6	66	22	1,882,489	0.84	**1,882,119**	−0.02%	13.50	1,898,663	0.88%	13.53	1,882,162	0.00%	12.81	1,882,148	−0.02%	11.20	1,893,998	0.63%	1.81
6	66	23	1,871,871	0.3	**1,871,669**	−0.01%	13.04	1,947,381	4.05%	14.02	1,871,723	0.00%	13.02	1,871,684	−0.01%	12.52	1,883,048	0.61%	1.72
6	66	24	1,862,962	0.62	**1,862,776**	−0.01%	13.44	2,005,165	7.64%	13.72	1,862,886	0.01%	13.01	1,862,797	−0.01%	11.87	1,872,298	0.51%	1.74

Table 14
Results of the algorithms with stock capacity constraints in large-scale instances.

No. of warehouses	No. of categories	Warehouse capacity	KCSA		KCSA with Louvain			KCSA with Surprise			KLCA		
			Objective	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s	Objective	Gap	Time/s
6	22,285	4,000,000	**5,041,513**	41 473.11	5,360,338	6.32%	3748.14	5,543,667	9.96%	7951.11	7,080,603	40.45%	4044.35
6	22,285	4,400,000	**5,143,709**	21 477.46	5,197,642	1.05%	5097.88	5,336,122	3.74%	20 088.72	6,993,003	35.95%	4612.58
6	22,285	4,800,000	5,184,846	24 320.53	**4,921,171**	−5.09%	6330.88	5,021,646	−3.15%	34 218.25	6,723,971	29.69%	3462.82
6	22,285	5,200,000	**4,909,478**	29 303.03	4,986,606	1.57%	2557.05	4,969,497	1.22%	32 367.18	6,661,619	35.69%	3936.16
6	22,285	5,600,000	**4,821,208**	74 226.76	4,965,450	2.99%	3249.40	4,948,804	2.65%	36 253.28	6,557,734	36.02%	4771.87

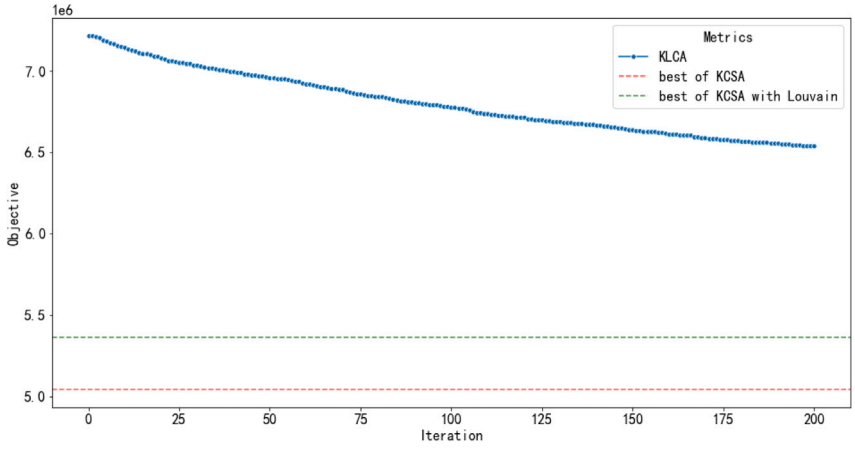


Fig. 8. The objective value of KLCA algorithm at each iteration.

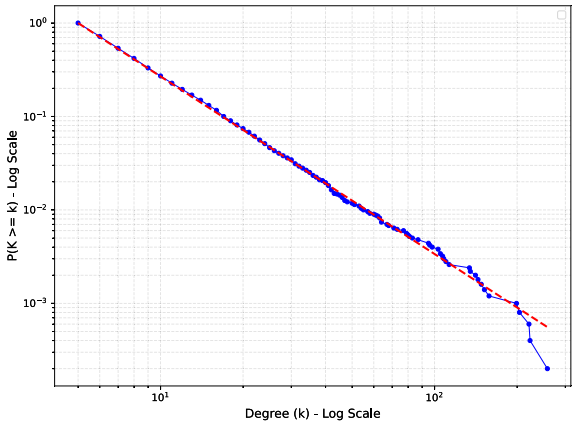
of 6,537,247—about 30% higher than KCSA and 22% higher than KCSA-Louvain.

5.5. Empirical evaluation of KCSA

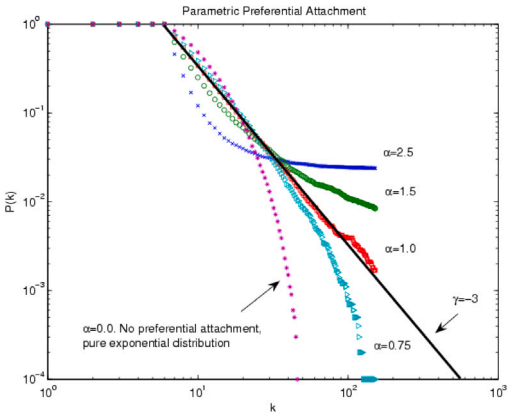
As shown in Fig. 9(b), Pujol et al. (2005) distinguish between three regimes based on cumulative degree distribution patterns. The red straight line ($\alpha = 1$) serves as a benchmark for a power-law network exhibiting the scale-free property. Cases where $\alpha > 1$ present a curve with a thicker tail above this line, suggesting the existence of highly connected super hubs. This reflects a winner-takes-all phenomenon, effectively forming a star-like network where most nodes link to a central hub. Conversely, when $\alpha < 1$, the curve shows a thinner tail below the line, indicating a lack of such super hubs and pointing to an exponential network structure. And for $\alpha < 0.75$, the exponential part clearly dominates the power-law one. As revealed in Fig. 9, the network formed by the coappearance of SKUs possesses small-world characteristics and belongs to the power-law with an exponential influence for having a thinner tail below the line. This indicates that there are fewer high degree nodes in the graph than the power-law graph.

Guided by the survey of random graph models (Drobyshevskiy & Turdakov, 2019), we selected four classical random graph models to empirically evaluate the efficiency of KCSA in solving the graph partition problem: the Erdős–Rényi (ER) model, the fixed-edge random graph with equal probabilities with exactly n nodes and m edges (GNM), the Watts–Strogatz (WS) model, and the Barabási–Albert (BA) model. The ER and GNM models represent foundational classes of random graphs. In the ER model, edges between a set of n nodes appear independently with a constant probability p; conversely, the GNM model is chosen uniformly at random from the set of all possible graphs with exactly n nodes and m edges. Additionally, we incorporate the WS model, which generates local clustering but produces an unrealistic degree distribution. Finally, the BA model is included to represent power-law networks with scale-free properties, but it fails to produce the high levels of clustering, which is not shared by the WS model. Notably, the WS and BA models exhibit topological characteristics most similar to the specific network in this study, which are the scale-free and thinner tail. For the numerical experiments, we generated small-scale and large-scale instances across varying edge densities, as detailed in Table 15. Due to distinct generative mechanisms, the edge densities of the BA and WS instances differ slightly from those of the ER and GNM models. Edge weights follow an exponential distribution with a scale parameter of 1.

To comprehensively evaluate the efficiency of the proposed KCSA, we benchmark it against established high-performance graph partitioning algorithms, including Jabeja (Rahimian et al., 2013), MITS (Benlic & Hao, 2011), and an ALNS variant utilizing cluster coarsening destroy operators. In this specific ALNS implementation, we extend the



(a) Product category degree distribution.



(b) Three different regimes of cumulative degree distribution (Pujol et al., 2005).

Fig. 9. The pattern of the network matches the exponential network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 15
Problem sets for empirical evaluation of KCSA.

	Small-scale	Large-scale
No. of nodes	100, 150, 200	1000
Edge density	0.02, 0.1, 0.2, 0.3, 0.4, 0.6	0.02, 0.1, 0.2, 0.3, 0.4, 0.6

relocation and switch strategies in Zhu et al. (2021) and introduce a cluster destroy operator that removes connected subgraphs via Breadth-First Search (BFS) to modify local structures. Since Jabeja and MITS are originally designed for balanced graph partitioning and may violate strict capacity constraints, we define capacity based on node cardinality and implement a post-hoc feasibility check. Instances where MITS produced infeasible solutions are explicitly marked. Table 16 summarizes the results for small-scale instances across various graph topologies, with the number of partitions set to 4. The best objective values are highlighted in bold, and optimality gaps are calculated relative to the best solution found among the four algorithms. MITS proves highly efficient in these instances, consistently yielding feasible solutions, which demonstrates the effectiveness of its coarsening heuristic. However, while the gaps are around 1% for GNM and ER graphs, they are relatively high for BA and WS graphs, suggesting that MITS is optimized for standard random graph topologies. Jabeja performs better on high-density graphs but yields larger gaps on low-density instances, though its performance remains generally consistent across graph types. ALNS typically identifies the best solutions, with the exception of certain low-density instances. Notably, the proposed KCSA demonstrates superior performance on BA and WS models, that is, power-law and exponential networks, which naturally exhibit stronger community structures. Regarding computational efficiency, MITS is the fastest, requiring less than 0.1 s, followed by KCSA, which completes all instances within 10 s. In summary, KCSA achieves a strong balance between solution quality and speed, excelling particularly in BA and WS models.

Similar patterns persist in the large-scale instances, as summarized in Table 17, where the number of partitions is set to 6. Notably, due to the coarsening and refinement stages in multilevel algorithms, MITS yields infeasible solutions in certain instances, which are explicitly labeled in the final column. It is evident that ALNS produces the best solutions for GNM, ER, and BA models; however, its computational time is significantly higher than that of KCSA. KCSA maintains competitive performance with relatively small optimality gaps in BA and WS models. Crucially, for BA and WS models, KCSA achieves these results with substantially lower computational costs compared to ALNS. Regarding the other baselines, Jabeja consistently maintains small gaps across instances, while MITS retains its high efficiency and superior solution quality in ER and GNM models. To further analyze the robustness of KCSA, we conducted numerical experiments by varying the number of partitions (Table 18). While ALNS identifies the best solutions in most cases, KCSA demonstrates strong robustness, achieving comparable gaps in most of WS and BA instances while requiring much lesser time than ALNS.

In summary, the KCSA algorithm proposed in this study demonstrates superior efficiency on BA and WS graphs, achieving significantly reduced computational times while maintaining relatively small optimality gaps. As noted by Meyerhenke et al. (2014), complex networks are often scale-free (many low-degree nodes, few high-degree nodes) and have the small-world property. These two properties distinguish complex networks from traditional meshes and make finding small cuts difficult. However, these very characteristics facilitate the integration of community detection techniques, which enhance the solution procedure by leveraging the underlying structural relationships between nodes. This integration functions as an aggressive coarsening strategy, which enables us to drastically shrink the size of complex networks.

5.6. Implementation in real operational workflow

We implemented the KCSA algorithm at JD.com to periodically optimize the assignment of fourth-level product categories to warehouses. In the warehouse assignment stage, historically, SKUs within the same first/second category were stocked in the same warehouse, a practice that frequently resulted in order splits. We employ KCSA to derive the optimal assignment for each category, which is then recorded as a key attribute in the inventory management system to guide vendor replenishment. This optimization is performed periodically to adapt to the evolution of the order structure. Specifically, we re-optimize the assignment on a monthly basis and calculate the expected reduction in order splits using data from the preceding week. Given this frequency, computational times of up to tens of hours are operationally acceptable. If the expected reduction exceeds a predefined threshold, the SKU attributes are updated to reflect the new assignments. When an online order is placed, JD.com's fulfillment system utilizes the real-time stock data to determine the fulfilling warehouse minimizing total costs.

The KCSA is implemented for JD.com's Fresh and Frozen Foods division, which requires cold storage in delivery process. The optimized assignments were deployed in real-world operations across five distribution centers based on monthly order data, with performance validated using data from the subsequent month. The resulting statistics are summarized in Table 19. According to JD.com's cold-chain packaging standards, each sub-order requires packs of refrigerant (ice or dry ice bags), weighing approximately 2.15 kg per sub-order. Trapp et al. (2017) report that the carbon dioxide equivalent emissions (CO_2e) for 10 kg of dry ice to be 1.39 kg. Hence, the monthly CO_2e saved are approximately estimated to 11 tons, assuming the exclusive use of dry ice bags.

Table 16
Results of the small-scale instances with different graph types.

Graph type	No. of nodes	No. of edges	Density	Capacity	Jabeja			KCSA			ALNS			MITS		
					Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap
BA	100	99	0.02	28	2.35	0.07	42.4%	**1.65**	0.49	0.0%	3.61	0.39	119.0%	3.89	0.01	135.9%
BA	100	384	0.08	28	43.47	0.53	6.3%	**40.89**	1.60	0.0%	46.37	0.93	13.4%	43.28	0.05	5.8%
BA	100	819	0.17	28	120.12	1.62	1.7%	**118.10**	0.91	0.0%	124.96	1.66	5.8%	119.13	0.09	0.9%
BA	100	1 204	0.24	28	216.21	3.37	4.9%	**206.07**	1.09	0.0%	216.26	2.26	4.9%	213.40	0.08	3.6%
BA	100	1 539	0.31	28	298.13	5.28	1.9%	**292.61**	1.58	0.0%	310.06	2.67	6.0%	300.59	0.10	2.7%
BA	100	2 059	0.42	28	491.68	9.15	2.3%	**480.45**	1.27	0.0%	490.06	3.48	2.0%	494.86	0.17	3.0%
BA	150	149	0.01	42	3.39	0.20	120.9%	**1.53**	1.21	0.0%	2.36	1.02	54.0%	3.24	0.01	111.5%
BA	150	1 001	0.09	42	139.76	3.67	4.0%	**134.41**	1.94	0.0%	135.42	3.69	0.8%	136.06	0.14	1.2%
BA	150	1 904	0.17	42	**332.99**	11.41	0.0%	333.86	2.00	0.3%	338.84	6.08	1.8%	338.21	0.25	1.6%
BA	150	2 816	0.25	42	**599.85**	23.29	0.0%	607.07	2.16	1.2%	602.99	8.29	0.5%	622.20	0.16	3.7%
BA	150	3 509	0.31	42	874.80	36.67	0.1%	875.70	2.82	0.2%	**873.93**	9.84	0.0%	899.88	0.45	3.0%
BA	150	4 664	0.42	42	1483.85	63.05	0.1%	1492.56	3.29	0.7%	**1482.00**	13.21	0.0%	1535.13	0.30	3.6%
BA	200	199	0.01	56	4.71	0.27	84.7%	**2.55**	2.90	0.0%	3.34	1.48	30.9%	3.22	0.02	26.3%
BA	200	1 719	0.09	56	254.57	7.28	3.8%	252.45	4.52	2.9%	**245.22**	6.21	0.0%	257.20	0.32	4.9%
BA	200	3 439	0.17	56	700.48	26.91	2.4%	694.39	3.38	1.5%	**684.03**	11.95	0.0%	720.62	0.31	5.3%
BA	200	4 959	0.25	56	1262.84	52.86	0.9%	1273.46	4.19	1.7%	**1252.18**	17.65	0.0%	1326.17	0.50	5.9%
BA	200	6 279	0.32	56	1989.13	82.96	1.3%	1985.93	4.81	1.1%	**1964.00**	20.28	0.0%	2074.00	0.58	5.6%
BA	200	8 319	0.42	56	3618.99	147.99	1.2%	3633.38	5.37	1.6%	**3576.56**	26.01	0.0%	3730.79	0.64	4.3%
WS	100	100	0.02	28	2.21	0.05	151.3%	**0.88**	0.37	0.0%	1.66	0.38	88.4%	1.24	0.01	41.0%
WS	100	400	0.08	28	19.41	0.27	44.9%	**13.40**	0.50	0.0%	14.48	0.84	8.1%	13.89	0.02	3.7%
WS	100	900	0.18	28	72.50	1.18	7.8%	68.21	0.63	1.4%	**67.24**	1.52	0.0%	68.29	0.04	1.6%
WS	100	1 400	0.28	28	179.98	3.07	1.1%	**178.03**	0.98	0.0%	181.33	2.18	1.9%	180.69	0.08	1.5%
WS	100	1 900	0.38	28	**347.40**	6.23	0.0%	352.83	0.44	1.6%	351.44	2.98	1.2%	348.32	0.17	0.3%
WS	100	2 900	0.59	28	793.22	15.63	0.2%	798.05	2.74	0.9%	808.56	4.33	2.2%	**791.28**	0.19	0.0%
WS	150	150	0.01	42	2.46	0.16	256.9%	**0.69**	1.97	0.0%	1.94	0.95	181.5%	1.54	0.01	123.5%
WS	150	1 050	0.09	42	67.54	2.05	67.7%	41.63	0.76	3.4%	**40.28**	3.29	0.0%	57.29	0.05	42.2%
WS	150	2 100	0.19	42	185.89	6.86	3.8%	181.36	1.67	1.3%	**179.10**	5.77	0.0%	184.19	0.19	2.8%
WS	150	3 300	0.30	42	**525.72**	21.84	0.0%	538.74	0.75	2.5%	529.92	8.63	0.8%	611.02	0.19	16.2%
WS	150	4 350	0.39	42	**1013.61**	42.22	0.0%	1016.39	0.83	0.3%	1031.74	11.04	1.8%	1021.48	0.34	0.8%
WS	150	6 600	0.59	42	2653.06	108.91	0.5%	2683.90	2.27	1.7%	**2640.24**	17.13	0.0%	2661.91	0.52	0.8%
WS	200	200	0.01	56	4.26	0.21	353.9%	**0.94**	1.67	0.0%	2.21	1.30	135.7%	1.21	0.01	29.3%
WS	200	1 800	0.09	56	87.30	3.50	27.0%	**68.75**	1.09	0.0%	69.17	6.12	0.6%	70.87	0.09	3.1%
WS	200	3 800	0.19	56	385.27	16.67	1.5%	386.15	1.65	1.7%	**379.58**	11.87	0.0%	389.95	0.18	2.7%
WS	200	5 800	0.29	56	**1094.38**	49.40	0.0%	1095.26	3.14	0.1%	1099.61	15.09	0.5%	1096.73	0.28	0.2%
WS	200	7 800	0.39	56	**2328.01**	102.54	0.0%	2359.01	2.27	1.3%	2389.79	21.65	2.7%	2358.05	0.77	1.3%
WS	200	11 800	0.59	56	6947.73	262.21	0.8%	7044.75	3.30	2.2%	**6895.45**	32.68	0.0%	6977.18	0.85	1.2%
GNM	100	99	0.02	28	2.81	0.06	44.5%	**1.95**	0.83	0.0%	2.69	0.37	38.4%	2.17	0.01	11.3%
GNM	100	495	0.10	28	54.29	0.57	0.6%	55.68	0.97	3.2%	57.79	0.96	7.1%	**53.94**	0.06	0.0%
GNM	100	990	0.20	28	144.59	2.01	0.5%	**143.80**	0.96	0.0%	149.09	1.69	3.7%	144.68	0.09	0.6%
GNM	100	1 485	0.30	28	264.99	4.38	2.6%	266.11	1.16	3.0%	270.04	2.34	4.5%	**258.32**	0.14	0.0%
GNM	100	1 980	0.40	28	416.63	7.54	0.4%	416.41	1.36	0.3%	422.31	3.01	1.8%	**414.98**	0.12	0.0%
GNM	100	2 970	0.60	28	**824.90**	16.60	0.0%	829.25	1.47	0.5%	846.62	4.14	2.6%	831.84	0.15	0.8%
GNM	150	223	0.02	42	11.53	0.27	11.8%	11.28	1.79	9.4%	11.46	1.19	11.1%	**10.32**	0.02	0.0%
GNM	150	1 117	0.10	42	142.14	3.55	1.8%	145.42	1.78	4.2%	143.05	3.36	2.5%	**139.59**	0.14	0.0%
GNM	150	2 235	0.20	42	393.82	12.82	0.6%	397.83	3.82	1.7%	392.18	6.10	0.2%	**391.37**	0.21	0.0%
GNM	150	3 352	0.30	42	**729.43**	28.08	0.0%	746.09	1.97	2.3%	737.33	8.79	1.1%	737.83	0.38	1.2%
GNM	150	4 470	0.40	42	1219.67	50.16	1.0%	1243.92	1.64	3.0%	**1207.39**	11.32	0.0%	1222.83	0.45	1.3%
GNM	150	6 705	0.60	42	2743.17	115.28	0.6%	2749.66	4.53	0.9%	**2725.53**	16.13	0.0%	2759.17	0.51	1.2%
GNM	200	398	0.02	56	23.82	0.52	2.2%	25.09	2.89	7.7%	**23.30**	2.07	0.0%	23.69	0.05	1.6%
GNM	200	1 990	0.10	56	290.25	8.10	2.5%	297.03	4.29	4.9%	285.00	6.70	0.7%	**283.10**	0.16	0.0%
GNM	200	3 980	0.20	56	806.15	30.49	1.2%	844.14	3.65	5.9%	**796.92**	12.47	0.0%	802.62	0.82	0.7%
GNM	200	5 970	0.30	56	1627.60	68.58	1.1%	1650.26	7.94	2.5%	**1609.52**	17.77	0.0%	1613.57	0.76	0.3%
GNM	200	7 960	0.40	56	2847.53	119.15	1.4%	2883.99	6.23	2.7%	**2808.82**	21.83	0.0%	2831.92	0.99	0.8%
GNM	200	11 940	0.60	56	7171.71	277.29	0.7%	7223.23	5.63	1.4%	**7122.55**	31.31	0.0%	7219.25	1.06	1.4%
ER	100	89	0.02	28	2.34	0.05	22.7%	2.12	0.65	11.3%	2.88	0.41	51.5%	**1.90**	0.01	0.0%
ER	100	474	0.10	28	51.56	0.54	2.1%	**50.48**	1.63	0.0%	53.14	0.96	5.3%	51.20	0.07	1.4%
ER	100	976	0.20	28	145.66	2.01	3.8%	**140.37**	1.85	0.0%	143.89	1.70	2.5%	141.95	0.08	1.1%
ER	100	1 477	0.30	28	262.32	4.43	0.8%	263.45	1.46	1.3%	263.18	2.37	1.2%	**260.13**	0.12	0.0%
ER	100	1 977	0.40	28	**416.51**	7.57	0.0%	420.40	0.93	0.9%	419.28	3.10	0.7%	417.44	0.10	0.2%
ER	100	2 958	0.60	28	**830.94**	16.83	0.0%	831.99	1.71	0.1%	834.00	4.30	0.4%	832.42	0.15	0.2%
ER	150	203	0.02	42	7.88	0.24	17.2%	8.35	1.47	24.1%	7.95	1.11	18.2%	**6.73**	0.02	0.0%
ER	150	1 103	0.10	42	141.25	3.45	2.7%	141.54	2.34	2.9%	139.24	3.56	1.3%	**137.49**	0.13	0.0%
ER	150	2 256	0.20	42	395.35	13.34	1.1%	399.42	4.02	2.2%	394.87	6.51	1.0%	**390.87**	0.30	0.0%
ER	150	3 311	0.30	42	720.72	28.36	1.0%	740.51	4.24	3.8%	**713.33**	9.01	0.0%	720.43	0.35	1.0%
ER	150	4 478	0.40	42	1234.38	51.51	0.8%	1244.85	1.90	1.6%	1224.90	11.86	0.0%	**1224.83**	0.47	0.0%
ER	150	6 669	0.60	42	**2715.21**	115.15	0.0%	2746.24	2.79	1.1%	2717.62	15.88	0.1%	2733.42	0.46	0.7%
ER	200	376	0.02	56	22.42	0.50	7.5%	24.28	3.56	16.4%	23.21	2.02	11.3%	**20.85**	0.03	0.0%
ER	200	1 963	0.10	56	279.39	8.05	1.9%	288.69	4.10	5.3%	**274.14**	6.44	0.0%	277.23	0.38	1.1%
ER	200	4 051	0.20	56	823.80	32.54	0.5%	851.39	4.22	3.8%	**820.00**	12.14	0.0%	826.58	0.57	0.8%
ER	200	5 918	0.30	56	1590.36	68.56	1.7%	1624.93	4.61	3.9%	**1564.30**	16.75	0.0%	1581.68	0.70	1.1%
ER	200	7 923	0.40	56	2807.91	122.71	1.2%	2844.38	7.18	2.5%	**2775.27**	21.21	0.0%	2796.87	0.73	0.8%
ER	200	11 860	0.60	56	7033.98	281.62	0.7%	7086.98	9.84	1.4%	**6986.91**	32.36	0.0%	7087.15	1.19	1.4%

Table 17
Results of the large-scale instances with different graph types.

Graph type	No. of nodes	No. of edges	Density	Capacity	Jabeja			KCSA			ALNS			MITS			Feasibility
					Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap	
BA	1000	8 919	0.02	184	1 627.18	105.11	5.3%	1 628.56	552.49	5.4%	**1 544.61**	185.26	0.0%	1 667.93	3.00	8.0%	Yes
BA	1000	46 599	0.09	184	34 339.65	2 262.94	0.3%	34 744.71	349.27	1.5%	**34 228.14**	1143.73	0.0%	36 363.35	10.12	6.2%	Yes
BA	1000	89 199	0.18	184	248 030.24	7 652.41	0.1%	248 741.07	487.96	0.4%	**247 753.43**	1643.22	0.0%	262 521.08	15.31	6.0%	Yes
BA	1000	126 799	0.25	184	1 059 152.99	14 854.45	0.1%	1 061 046.49	255.08	0.3%	**1 057 643.59**	2283.94	0.0%	1 121 891.29	20.52	6.1%	Yes
BA	1000	159 399	0.32	184	3 356 773.63	22 479.82	0.1%	3 362 487.18	149.47	0.2%	**3 354 345.92**	2551.89	0.0%	3 607 995.88	23.18	7.6%	Yes
BA	1000	209 599	0.42	184	19 068 818.45	38 539.65	0.0%	19 077 787.52	505.39	0.1%	**19 059 541.93**	3209.56	0.0%	20 861 391.39	26.50	9.5%	Yes
WS	1000	9 000	0.02	184	427.46	34.09	335.9%	**98.05**	88.30	0.0%	116.59	125.05	18.9%	179.76	0.44	83.3%	Yes
WS	1000	49 000	0.10	184	7 884.92	871.26	10.1%	7 190.19	59.14	0.4%	**7 162.33**	695.84	0.0%	7 186.34	5.70	0.3%	Yes
WS	1000	99 000	0.20	184	95 749.93	5 388.24	0.1%	**95 637.82**	24.36	0.0%	98 228.08	1066.13	2.7%	96 062.66	16.01	0.4%	Yes
WS	1000	149 000	0.30	184	695 812.12	15 230.51	1.4%	**686 005.13**	64.00	0.0%	694 558.22	1925.25	1.2%	752 707.14	27.74	9.7%	Yes
WS	1000	199 000	0.40	184	**3 804 194.15**	29 378.80	0.0%	3 807 917.87	40.98	0.1%	3 807 328.28	2612.98	0.1%	4 097 096.69	34.98	7.7%	Yes
WS	1000	299 000	0.60	184	77 168 888.89	68 762.90	0.4%	76 977 947.17	99.39	0.1%	**76 876 277.11**	4368.22	0.0%	80 001 309.30	6.70	4.1%	No
GNM	1000	9 990	0.02	184	1 574.01	90.79	2.6%	1 692.07	443.77	10.3%	**1 533.50**	176.51	0.0%	1 559.62	5.81	1.7%	Yes
GNM	1000	49 950	0.10	184	25 127.89	2 054.99	0.7%	26 268.11	497.41	5.3%	**24 956.89**	816.65	0.0%	25 301.57	23.20	1.4%	Yes
GNM	1000	99 900	0.20	184	180 505.93	8 017.59	0.3%	184 840.67	622.93	2.7%	**179 931.96**	1719.17	0.0%	182 386.61	40.06	1.4%	Yes
GNM	1000	149 850	0.30	184	953 330.38	18 300.72	0.6%	969 402.00	514.44	2.3%	**947 457.10**	2388.33	0.0%	965 907.36	41.38	1.9%	Yes
GNM	1000	199 800	0.40	184	4 428 596.13	31 862.66	0.4%	4 492 442.32	390.67	1.8%	**4 411 420.59**	2919.00	0.0%	4 466 437.19	78.63	1.2%	Yes
GNM	1000	299 700	0.60	184	79 778 448.35	74 501.40	0.1%	80 568 112.24	450.31	1.0%	**79 735 585.73**	4309.48	0.0%	81 767 514.27	6.83	2.5%	No
ER	1000	9 925	0.02	184	1 552.81	88.00	1.7%	1 667.12	587.58	9.2%	**1526.16**	169.60	0.0%	1 546.70	5.46	1.3%	Yes
ER	1000	49 929	0.10	184	25 067.09	2 023.81	0.3%	26 114.32	622.17	4.5%	**24 986.66**	877.35	0.0%	25 292.07	23.98	1.2%	Yes
ER	1000	99 851	0.20	184	180 330.80	8 115.77	0.6%	185 148.54	431.31	3.3%	**179 241.61**	1600.59	0.0%	181 452.60	45.10	1.2%	Yes
ER	1000	149 559	0.30	184	945 288.25	18 047.01	0.7%	959 508.82	612.45	2.3%	**938 338.77**	2335.90	0.0%	953 495.28	40.29	1.6%	Yes
ER	1000	199 436	0.40	184	4 374 301.58	32 464.61	0.4%	4 431 994.43	412.53	1.8%	**4 354 843.64**	3030.08	0.0%	4 418 473.65	68.96	1.5%	Yes
ER	1000	299 089	0.60	184	78 411 882.80	71 632.47	0.2%	79 201 128.49	240.62	1.2%	**78 240 718.72**	4230.27	0.0%	80 428 972.96	6.69	2.8%	No

Table 18
Results of the instances with different number of partitions and capacities across graph types.

Graph type	No. of nodes	No. of edges	Density	No. of partitions	Capacity	Jabeja			KCSA			ALNS			MITS			Feasibility
						Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap	Objective	Time/s	Gap	
BA	200	6 279	0.32	2	105	1258.51	68.00	3.1%	1285.80	1.52	5.3%	**1221.24**	16.42	0.0%	1256.32	0.26	2.9%	Yes
BA	200	6 279	0.32	2	120	1093.27	59.94	2.4%	1114.51	1.70	4.4%	**1067.56**	15.90	0.0%	1256.32	0.29	17.7%	Yes
BA	200	6 279	0.32	2	140	847.63	46.58	2.5%	867.39	1.86	4.9%	**826.90**	14.89	0.0%	1256.47	0.40	51.9%	Yes
BA	200	6 279	0.32	4	53	2025.32	86.06	0.6%	2026.98	4.40	0.7%	**2012.90**	20.60	0.0%	2072.31	0.50	3.0%	Yes
BA	200	6 279	0.32	4	60	1910.83	83.61	1.4%	1918.20	7.89	1.8%	**1883.69**	20.05	0.0%	2059.44	0.58	9.3%	Yes
BA	200	6 279	0.32	4	70	1761.55	82.17	1.8%	1764.51	6.74	2.0%	**1729.78**	22.77	0.0%	2062.43	0.53	19.2%	Yes
BA	200	6 279	0.32	8	27	**2459.05**	85.33	0.0%	2478.00	17.53	0.8%	2471.35	25.35	0.5%	2555.27	0.33	3.9%	Yes
BA	200	6 279	0.32	8	30	2401.69	85.98	0.1%	2419.61	13.61	0.9%	**2398.30**	29.98	0.0%	2529.07	0.56	5.5%	Yes
BA	200	6 279	0.32	8	35	**2330.40**	86.85	0.0%	2333.81	7.30	0.1%	2344.07	31.86	0.6%	2540.56	0.39	9.0%	Yes
BA	200	8 319	0.42	2	105	2354.69	125.23	3.4%	2374.71	4.85	4.3%	**2277.31**	21.59	0.0%	2338.07	0.51	2.7%	Yes
BA	200	8 319	0.42	2	120	2046.36	112.03	2.5%	2057.80	5.81	3.1%	**1996.41**	21.29	0.0%	2338.31	0.46	17.1%	Yes
BA	200	8 319	0.42	2	140	1595.57	87.54	3.4%	1590.23	4.71	3.1%	**1542.57**	20.61	0.0%	2338.65	0.50	51.6%	Yes
BA	200	8 319	0.42	4	53	3672.37	151.15	0.6%	3704.84	8.13	1.5%	**3648.67**	25.76	0.0%	3743.21	0.72	2.6%	Yes
BA	200	8 319	0.42	4	60	3487.66	149.85	1.2%	3532.67	8.08	2.5%	**3446.03**	26.43	0.0%	3772.15	0.56	9.5%	Yes
BA	200	8 319	0.42	4	70	3252.80	146.32	2.4%	3225.77	6.13	1.5%	**3176.97**	30.33	0.0%	3762.88	0.92	18.4%	Yes
BA	200	8 319	0.42	8	27	**4437.66**	146.28	0.0%	4442.20	11.62	0.1%	4461.79	34.28	0.5%	4542.46	0.64	2.4%	Yes
BA	200	8 319	0.42	8	30	**4353.94**	147.11	0.0%	4356.78	8.64	0.1%	4362.50	38.25	0.2%	4527.90	0.25	4.0%	No
BA	200	8 319	0.42	8	35	**4190.97**	148.28	0.0%	4237.11	9.09	1.1%	4217.99	38.89	0.6%	4535.99	0.30	8.2%	No
WS	200	7 800	0.39	2	105	1480.65	81.05	1.2%	**1463.50**	3.26	0.0%	1468.92	18.41	0.4%	1473.01	0.25	0.7%	Yes
WS	200	7 800	0.39	2	120	1443.02	80.42	2.4%	1443.74	1.43	2.5%	**1408.83**	16.98	0.0%	1458.98	0.33	3.6%	Yes
WS	200	7 800	0.39	2	140	1296.38	72.71	1.8%	1279.23	1.60	0.5%	**1273.24**	17.57	0.0%	1456.49	0.27	14.4%	Yes
WS	200	7 800	0.39	4	53	2384.43	105.89	1.2%	2368.32	8.65	0.5%	2375.70	22.17	0.8%	**2357.12**	0.64	0.0%	Yes
WS	200	7 800	0.39	4	60	2278.95	100.50	0.4%	2272.60	1.55	0.1%	**2270.90**	22.22	0.0%	2358.81	0.48	3.9%	Yes
WS	200	7 800	0.39	4	70	2024.81	98.65	1.0%	2035.35	1.63	1.5%	**2004.37**	25.45	0.0%	2357.74	0.71	17.6%	Yes
WS	200	7 800	0.39	8	27	**3045.75**	109.65	0.0%	3049.84	16.08	0.1%	3057.21	29.81	0.4%	3163.22	0.85	3.9%	Yes
WS	200	7 800	0.39	8	30	**2952.30**	110.30	0.0%	2997.50	3.16	1.5%	2973.90	32.58	0.7%	3147.28	0.46	6.6%	Yes
WS	200	7 800	0.39	8	35	**2823.19**	108.09	0.0%	2861.16	4.63	1.3%	2857.47	31.80	1.2%	3171.14	0.81	12.3%	Yes
WS	200	11 800	0.59	2	105	4644.60	236.69	2.7%	4600.20	3.31	1.7%	**4524.63**	26.06	0.0%	4542.84	0.49	0.4%	Yes
WS	200	11 800	0.59	2	120	4389.42	228.20	0.5%	4495.43	3.41	3.0%	**4365.42**	25.66	0.0%	4547.45	0.60	4.2%	Yes
WS	200	11 800	0.59	2	140	3869.55	201.62	2.8%	3970.02	2.04	5.4%	**3765.44**	27.05	0.0%	4543.69	0.73	20.7%	Yes
WS	200	11 800	0.59	4	53	7042.56	267.70	1.4%	7092.35	5.38	2.1%	**6944.29**	33.82	0.0%	6971.43	1.16	0.4%	Yes
WS	200	11 800	0.59	4	60	6710.40	261.96	0.7%	6843.14	4.57	2.7%	**6662.04**	33.44	0.0%	6947.89	1.93	4.3%	Yes
WS	200	11 800	0.59	4	70	6191.24	265.43	0.7%	6250.02	2.70	1.7%	**6145.24**	34.95	0.0%	6951.40	1.14	13.1%	Yes
WS	200	11 800	0.59	8	27	**8245.11**	253.57	0.0%	8283.60	12.70	0.5%	8250.62	46.25	0.1%	8282.44	1.62	0.5%	Yes
WS	200	11 800	0.59	8	30	8136.92	261.39	0.8%	8156.05	9.54	1.1%	**8070.11**	44.63	0.0%	8281.98	1.21	2.6%	Yes
WS	200	11 800	0.59	8	35	7888.15	258.72	1.1%	7935.54	4.17	1.7%	**7805.94**	50.48	0.0%	8311.69	1.57	6.5%	Yes
GNM	200	7 960	0.40	2	105	1904.87	103.15	4.4%	1889.99	2.50	3.6%	**1824.54**	18.71	0.0%	1833.49	0.46	0.5%	Yes
GNM	200	7 960	0.40	2	120	1813.23	103.42	1.9%	1829.60	4.39	2.8%	**1779.49**	17.77	0.0%	1853.71	0.37	4.2%	Yes
GNM	200	7 960	0.40	2	140	1549.74	85.06	3.3%	1607.89	2.45	7.2%	**1499.75**	17.67	0.0%	1846.08	0.33	23.1%	Yes
GNM	200	7 960	0.40	4	53	2886.72	121.05	1.9%	2886.05	10.06	1.9%	2842.06	22.23	0.3%	**2833.23**	0.94	0.0%	Yes
GNM	200	7 960	0.40	4	60	2742.93	117.19	1.3%	2827.09	2.86	4.5%	**2706.58**	22.58	0.0%	2831.87	0.62	4.6%	Yes
GNM	200	7 960	0.40	4	70	2547.94	117.41	2.6%	2560.48	2.84	3.1%	**2484.22**	23.45	0.0%	2842.45	1.16	14.4%	Yes
GNM	200	7 960	0.40	8	27	3391.60	116.66	0.6%	3408.90	11.83	1.1%	**3371.10**	30.15	0.0%	3373.97	1.40	0.1%	Yes
GNM	200	7 960	0.40	8	30	3316.49	117.93	0.6%	3344.86	14.44	1.4%	**3298.22**	32.99	0.0%	3400.77	0.72	3.1%	Yes
GNM	200	7 960	0.40	8	35	3222.63	117.39	0.5%	3236.17	10.60	0.9%	**3207.35**	32.47	0.0%	3403.94	1.47	6.1%	Yes
GNM	200	11 940	0.60	2	105	4793.05	236.75	1.9%	4771.80	3.66	1.5%	**4702.93**	26.56	0.0%	4721.22	0.50	0.4%	Yes
GNM	200	11 940	0.60	2	120	4574.45	232.28	1.9%	4626.45	2.82	3.1%	**4488.26**	27.15	0.0%	4731.69	0.65	5.4%	Yes
GNM	200	11 940	0.60	2	140	3923.95	199.20	1.9%	4058.62	4.69	5.4%	**3851.00**	26.26	0.0%	4698.08	0.71	22.0%	Yes
GNM	200	11 940	0.60	4	53	7265.69	271.35	1.1%	7342.82	7.20	2.1%	**7189.93**	32.70	0.0%	7219.29	1.00	0.4%	Yes
GNM	200	11 940	0.60	4	60	6951.11	271.92	0.8%	7095.75	3.50	2.9%	**6894.04**	33.02	0.0%	7282.25	1.02	5.6%	Yes
GNM	200	11 940	0.60	4	70	6386.74	266.41	0.4%	6497.48	3.37	2.1%	**6363.87**	35.89	0.0%	7189.04	1.98	13.0%	Yes
GNM	200	11 940	0.60	8	27	8519.15	259.98	0.0%	8532.17	20.41	0.2%	**8517.12**	44.32	0.0%	8570.85	3.32	0.6%	Yes
GNM	200	11 940	0.60	8	30	8382.52	261.96	0.6%	8406.87	18.03	0.9%	**8330.06**	50.23	0.0%	8583.02	1.56	3.0%	Yes
GNM	200	11 940	0.60	8	35	8118.54	261.53	0.6%	8157.33	10.99	1.1%	**8072.51**	50.40	0.0%	8559.66	1.20	6.0%	Yes
ER	200	7 923	0.40	2	105	1894.21	102.70	3.9%	1859.41	2.15	2.0%	**1822.65**	18.82	0.0%	1833.25	0.34	0.6%	Yes
ER	200	7 923	0.40	2	120	1797.88	101.11	2.3%	1812.61	2.39	3.1%	**1757.91**	18.32	0.0%	1834.11	0.41	4.3%	Yes
ER	200	7 923	0.40	2	140	1541.44	86.01	3.0%	1584.73	3.04	5.9%	**1495.91**	18.95	0.0%	1818.59	0.58	21.6%	Yes
ER	200	7 923	0.40	4	53	2850.16	118.75	1.5%	2882.83	4.03	2.7%	2820.02	23.01	0.4%	**2807.86**	0.71	0.0%	Yes
ER	200	7 923	0.40	4	60	2720.36	119.64	1.4%	2792.61	5.45	4.0%	**2684.08**	23.10	0.0%	2803.66	1.02	4.5%	Yes
ER	200	7 923	0.40	4	70	2502.75	113.96	1.6%	2540.06	2.81	3.1%	**2463.28**	26.48	0.0%	2817.87	0.73	14.4%	Yes
ER	200	7 923	0.40	8	27	**3344.59**	118.58	0.0%	3370.65	15.77	0.8%	3344.60	33.93	0.0%	3346.30	1.16	0.1%	Yes
ER	200	7 923	0.40	8	30	3296.51	117.23	0.6%	3322.31	16.67	1.4%	**3275.44**	34.10	0.0%	3355.49	0.90	2.4%	Yes
ER	200	7 923	0.40	8	35	3193.02	119.66	0.4%	3213.79	9.75	1.1%	**3179.84**	34.46	0.0%	3363.79	0.65	5.8%	Yes
ER	200	11 860	0.60	2	105	4705.13	233.53	2.0%	4726.17	2.72	2.4%	**4613.71**	27.21	0.0%	4638.29	0.53	0.5%	Yes
ER	200	11 860	0.60	2	120	4531.69	231.47	2.0%	4579.86	2.41	3.1%	**4443.97**	28.53	0.0%	4639.33	0.89	4.4%	Yes
ER	200	11 860	0.60	2	140	3881.87	206.39	1.8%	4024.78	2.34	5.6%	**3812.30**	27.81	0.0%	4651.96	0.50	22.0%	Yes
ER	200	11 860	0.60	4	53	7141.25	271.21	0.9%	7226.35	6.22	2.1%	**7076.02**	34.57	0.0%	7076.55	1.15	0.0%	Yes
ER	200	11 860	0.60	4	60	6853.08	267.91	0.8%	6939.78	4.16	2.0%	**6801.91**	34.49	0.0%	7093.66	0.75	4.3%	Yes
ER	200	11 860	0.60	4	70	6327.30	259.96	1.4%	6351.87	3.17	1.7%	**6242.65**	38.01	0.0%	7098.34	0.94	13.7%	Yes
ER	200	11 860	0.60	8	27	8405.24	257.72	0.3%	8424.32	11.30	0.5%	**8378.37**	46.66	0.0%	8402.56	1.26	0.3%	Yes
ER	200	11 860	0.60	8	30	8242.45	259.49	0.5%	8274.59	20.32	0.9%	**8201.16**	48.59	0.0%	8403.39	1.60	2.5%	Yes
ER	200	11 860	0.60	8	35	7998.90	262.23	0.6%	8055.90	7.90	1.3%	**7952.15**	50.59	0.0%	8430.53	2.09	6.0%	Yes

Table 19
Effects of KCSA deployment in JD.com using monthly data.

DCs	Before optimization		After optimization	
	Split count	Split rate	Split count	Split rate
Beijing	803,375	8.94%	789,091	7.00%
Shanghai	518,003	6.75%	515,607	6.26%
Guangzhou	665,594	10.69%	649,067	8.13%
Nanjing	198,317	8.32%	195,315	6.68%
Hangzhou	215,240	5.47%	214,476	5.10%

6. Conclusions and future research directions

This study addresses the order splitting problem in collaboration with JD.com, China's largest online retailer by revenue. Existing algorithms prove inadequate for JD.com's real-world scale, prompting our transformation of the problem into a capacitated graph partition formulation—an approximation method aligned with Catalán and Fisher (2012) and Zhu et al. (2021). This approach yields negligible objective deviation for orders containing fewer than 5 SKUs, which represents the majority of real-world orders. To solve the graph partition problem efficiently, we develop a community detection-based algorithm (KCSA) that exploits SKU category connectivity patterns: starting from an initial solution, we iteratively optimize category assignments between warehouse pairs. Each iteration applies community detection to identify densely connected category clusters, then assigns communities to warehouses when it reduces order splits without violating capacity constraints. Improved assignments are retained after each pairwise optimization. Evaluated on JD.com's real order datasets, KCSA achieves near-optimal solutions significantly faster than Gurobi in small-scale tests. For large-scale instances, it outperforms both KLCA and BESTSELLER algorithms in solution quality and computational efficiency. Validation under category-count capacity constraints (consistent with literature baselines) further confirms robustness. Implemented for JD.com's Fresh and Frozen Foods division (requiring cold storage), KCSA reduced order splits by 440,000 annually across five distribution centers, yielding material cost savings of 2.6 million RMB. The

annual carbon dioxide equivalent emissions saved for the refrigerant are approximately 132 tons.

We suggest several potential directions for future research, outlined as follows:

- A limitation of the current model lies in its use of static capacity constraints based on historical average inventory levels for SKU allocation. In practice, the inventory levels can experience significant fluctuations, particularly during major promotions or seasonal peaks. Although warehouse operations often have some elastic capacity to temporarily absorb the inventory surges, the static allocation may face robustness challenges under extreme volatility. This limitation can be mitigated by deploying the proposed algorithm on a monthly or weekly basis to dynamically adjust the allocation using the latest inventory data. Moreover, it would be promising to extend the model by introducing specific constraints, such as promotional peak capacity limits or seasonal inventory fluctuation constraints. That enables the model to proactively address inventory volatility, ensuring a strictly feasible and robust allocation even under extreme conditions;
- While transforming the hypergraph partitioning problem into a graph partitioning problem is computationally efficient, it inevitably introduces an approximation bias into the objective function. Although the bias is negligible for small orders in this study, it is compelling to design new algorithms within the native hypergraph partitioning framework to avoid this approximation bias. This direction may achieve better performance in situations with massive orders; and
- The analysis of the 3-warehouse/66-category instance reveals that when warehouse capacity constraints necessitate splitting densely connected clusters, KCSA may yield suboptimal solutions with relatively high gaps. To address this weakness of KCSA in this scenario, a dedicated refinement phase could be introduced. After the initial community detection stage, the communities can be refined back to the original node level. Then, node swap strategies across warehouses are applied to escape local optima. Additionally, penalizing communities which are large relative to warehouse capacities may help to reduce the oversized communities, thus, alleviating this phenomenon.

CRediT authorship contribution statement

Hao Hu: Writing – original draft, Visualization, Validation, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization. **Yongzhi Qi:** Writing – review & editing, Validation, Supervision, Methodology, Formal analysis, Data curation. **Ningxuan Kang:** Writing – review & editing, Project administration, Methodology, Formal analysis, Conceptualization. **Zhe Chen:** Writing – review & editing, Writing – original draft, Methodology, Formal analysis, Conceptualization. **Ling Wang:** Writing – review & editing, Supervision, Resources, Project administration. **Jian Chen:** Writing – review & editing, Supervision, Resources. **Zuo-Jun Max Shen:** Supervision, Resources, Project administration, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Zhe Chen reports financial support was provided by JD.com Inc. Hao Hu, Yongzhi Qi, Ningxuan Kang, Zuo-Jun Max Shen reports financial support was provided by JD.com Inc. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is partially supported by the Hong Kong RGC General Research Funding (17504324).

Appendix. Pseudocode for KCSA

Algorithm 3 K-Community Swap Algorithm

Require: Graph $G = (V, E)$, Number of warehouses K, Warehouse capacity limit $C_{\max}$, Max iterations N_{iter}.
Ensure: Best feasible order splitting partition P_{best}
  $P_0 \leftarrow$ Initialize partitions greedily by sorting nodes by weight and filling K warehouses subject to $C_{\max}$.
  $P_{\text{current}} \leftarrow P_0$, $P_{\text{best}} \leftarrow P_0$
  Operators $D \leftarrow \{\text{Louvain}, \text{Surprise}\}$
  for $iter = 1$ **to** N_{iter} **do**
    Randomly select an operator $op \in D$.
    Let $P_{\text{new}} \leftarrow P_{\text{current}}$.
    Sort all warehouse pairs (i, j) in P_{new} descendingly by their inter-warehouse cut weight.
    $improved \leftarrow$ False
    for each pair (i, j) in sorted order **do**
      Extract subgraph G_{sub} consisting of all nodes in warehouse i and j.
      Total weight $W_{\text{total}} \leftarrow W_i + W_j$.
      Run 2-CSA algorithm in warehouses (i, j) and obtain new assignment $(Assign_1, Assign_2)$.
      Evaluate new cut penalty for $(Assign_1, Assign_2)$.
      if new cut penalty < current cut penalty **then**
        Update P_{new} by replacing old partitions i and j with $Assign_1$ and $Assign_2$.
        $improved \leftarrow$ True
        break
      end if
    end for
    if $improved$ **and** Objective(P_{new}) < Objective(P_{best}) **then**
      $P_{\text{current}} \leftarrow P_{\text{new}}$
      $P_{\text{best}} \leftarrow P_{\text{new}}$
    end if
  end for
  return P_{best}

References

Acimovic, J., & Graves, S. C. (2015). Making better fulfillment decisions on the fly in an online retail environment. *Manufacturing & Service Operations Management*, *17*(1), 34–51.

Airoldi, E. M., Blei, D., Fienberg, S., & Xing, E. (2008). Mixed membership stochastic blockmodels. *Advances in Neural Information Processing Systems*, *21*.

Albán, H. M. G., Cornelissens, T., & Sörensen, K. (2024). A new policy for scattered storage assignment to minimize picking travel distances. *European Journal of Operational Research*, *315*(3), 1006–1020.

Aldecoa, R., & Marín, I. (2013). Surprise maximization reveals the community structure of complex networks. *Scientific Reports*, *3*(1), 1060.

Ales, Z., & Knippel, A. (2020). The K-partitioning problem: Formulations and branch-and-cut. *Networks*, *76*(3), 323–349.

Ardjmand, E., Bajgiran, O. S., Rahman, S., Weckman, G. R., & Young, W. A., II (2018). A multi-objective model for order cartonization and fulfillment center assignment in the e-tail/retail industry. *Transportation Research Part E: Logistics and Transportation Review*, *115*, 16–34.

Benlic, U., & Hao, J.-K. (2011). An effective multilevel tabu search approach for balanced graph partitioning. *Computers & Operations Research*, *38*(7), 1066–1075.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*(10), Article P10008.

Boysen, N., & De Koster, R. (2025). 50 years of warehousing research—An operations research perspective. *European Journal of Operational Research*, *320*(3), 449–464.

Boysen, N., De Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, *277*(2), 396–411.

Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., & Schulz, C. (2016). Recent advances in graph partitioning. *Algorithm Engineering: Selected Results and Surveys*, 117–158.

Catalán, A., & Fisher, M. (2012). Assortment allocation to distribution centers to minimize split customer orders. Available At SSRN 2166687.

Çatalyürek, ü., Devine, K., Faraj, M., Gottesbüren, L., Heuer, T., Meyerhenke, H., Sanders, P., Schlag, S., Schulz, C., Seemaier, D., et al. (2023). More recent advances in (hyper) graph partitioning. *ACM Computing Surveys*, *55*(12), 1–38.

Çetinkaya, S., & Bookbinder, J. H. (2003). Stochastic models for the dispatch of consolidated shipments. *Transportation Research Part B: Methodological*, *37*(8), 747–768.

Chevalier, C., & Pellegrini, F. (2008). PT-Scotch: A tool for efficient parallel graph ordering. *Parallel Computing*, *34*(6–8), 318–331.

Chockalingam, A., Atan, Z., & Marandi, A. (2019). *Optimizing multi-warehouse assortment allocation in an online retail environment* Master's Thesis, Eindhoven University of Technology.

Clauset, A., Newman, M. E., & Moore, C. (2004). Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, *70*(6), Article 066111.

Deng, W., Li, W., Cai, X., & Wang, Q. A. (2011). The exponential degree distribution in complex networks: Non-equilibrium network theory, numerical simulation and empirical data. *Physica A. Statistical Mechanics and its Applications*, *390*(8), 1481–1485.

Drobyshevskiy, M., & Turdakov, D. (2019). Random graph modeling: A survey of the concepts. *ACM Computing Surveys*, *52*(6), 1–36.

Feng, X., Jiang, H., & Jiang, L.-p. (2020). Study on community detection of shipping network based on modularity. In *Green, smart and connected transportation systems: Proceedings of the 9th international conference on green intelligent transportation systems and safety* (pp. 365–374). Springer.

Fiduccia, C. M., & Mattheyses, R. M. (1988). A linear-time heuristic for improving network partitions. In *Papers on twenty-five years of electronic design automation* (pp. 241–247).

Fiedler, M. (1975). A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, *25*(4), 619–633.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3–5), 75–174.

Fortunato, S., & Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, *104*(1), 36–41.

Girvan, M., & Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *99*(12), 7821–7826.

Gleiser, P. M., & Danon, L. (2003). Community structure in jazz. *Advances in Complex Systems*, *6*(04), 565–573.

Glover, F. (1989). Tabu search—part I. *ORSA Journal on Computing*, *1*(3), 190–206.

Gong, Q., Wang, K., Fan, X., Fu, X., & Xiao, Y.-b. (2018). International trade drivers and freight network analysis-The case of the Chinese air cargo sector. *Journal of Transport Geography*, *71*, 253–262.

Gottesbüren, L., Heuer, T., Sanders, P., Schulz, C., & Seemaier, D. (2021). Deep multilevel graph partitioning. arXiv preprint arXiv:2105.02022.

Guimera, R., Mossa, S., Turtschi, A., & Amaral, L. N. (2005). The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, *102*(22), 7794–7799.

Hall, R. W. (1987). Consolidation strategy: Inventory, vehicles and terminals. *Journal of Business Logistics*, *8*(2), 57.

Hanbazazah, A. S., Abril, L., Erkoc, M., & Shaikh, N. (2019). Freight consolidation with divisible shipments, delivery time windows, and piecewise transportation costs. *European Journal of Operational Research*, *276*(1), 187–201.

Hendrickson, B., & Leland, R. (1993). *The chaco users guide. version 1.0*: *Tech. rep.*, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

Holland, P. W., Laskey, K. B., & Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social Networks*, *5*(2), 109–137.

Huang, M., Zhao, Y., Wang, Y., & Guo, Q. (2024). Application of a green logistics strategy to fulfil online retailing: Split-order consolidated delivery in large-scale online supermarkets. *International Journal of Logistics Research and Applications*, *27*(5), 746–759.

Ishfaq, R., & Raja, U. (2018). Evaluation of order fulfillment options in retail supply chains. *Decision Sciences*, *49*(3), 487–521.

Jasin, S., & Sinha, A. (2015). An LP-based correlated rounding scheme for multi-item ecommerce order fulfillment. *Operations Research*, *63*(6), 1336–1351.

Jia, Y., Zhang, Q., Zhang, W., & Wang, X. (2019). Communitygan: Community detection with generative adversarial nets. In *The world wide web conference* (pp. 784–794).

Jin, D., Yu, Z., Jiao, P., Pan, S., He, D., Wu, J., Yu, P. S., & Zhang, W. (2021). A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*(2), 1149–1170.

Karrer, B., & Newman, M. E. (2011). Stochastic blockmodels and community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, *83*(1), Article 016107.

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, *20*(1), 359–392.

Karypis, G., & Kumar, V. (1999). Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th annual ACM/IEEE design automation conference* (pp. 343–348).

Kernighan, B. W., & Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, *49*(2), 291–307.

Kipf, T. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Lei, Y., Jasin, S., & Sinha, A. (2018). Joint dynamic pricing and order fulfillment for e-commerce retailers. *Manufacturing & Service Operations Management*, *20*(2), 269–284.

Lin, H., Li, X., & Liu, F. (2025). Multi-warehouse assortment selection: Minimizing order splitting in e-commerce logistics. *Production and Operations Management*, Article 10591478251365581.

Mahar, S., & Wright, P. D. (2009). The value of postponing online fulfillment decisions in multi-channel retail/e-tail organizations. *Computers & Operations Research*, *36*(11), 3061–3072.

Mara, S. T. W., Norcahyo, R., Jodiawan, P., Lusiantoro, L., & Rifai, A. P. (2022). A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research*, *146*, Article 105903.

Meyerhenke, H., Sanders, P., & Schulz, C. (2014). Partitioning complex networks via size-constrained clustering. In *International symposium on experimental algorithms* (pp. 351–363). Springer.

Meyerhenke, H., Sanders, P., & Schulz, C. (2017). Parallel graph partitioning for complex networks. *IEEE Transactions on Parallel and Distributed Systems*, *28*(9), 2625–2638.

Newman, M. E., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*(2), Article 026113.

Osipov, V., & Sanders, P. (2010). N-Level graph partitioning. In *European symposium on algorithms* (pp. 278–289). Springer.

Palla, G., Derényi, I., Farkas, I., & Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814–818.

Pujol, J. M., Flache, A., Delgado, J., Sangüesa, R., & Sanguessa, R. (2005). How can social networks ever become complex? Modelling the emergence of complex networks from local social exchanges. *Jasss-the Journal of Artificial Societies and Social Simulation*, *8*(4).

Rahimian, F., Payberah, A. H., Girdzijauskas, S., Jelasity, M., & Haridi, S. (2013). Ja-be-ja: A distributed algorithm for balanced graph partitioning. In *2013 IEEE 7th international conference on self-adaptive and self-organizing systems* (pp. 51–60). IEEE.

Ramakrishna, K. S., Sharafali, M., & Lim, Y. F. (2015). A two-item two-warehouse periodic review inventory model with transshipment. *Annals of Operations Research*, *233*(1), 365–381.

Rosvall, M., & Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, *105*(4), 1118–1123.

Sanders, P., & Schulz, C. (2011). Engineering multilevel graph partitioning algorithms. In *European symposium on algorithms* (pp. 469–480). Springer.

Schlag, S., Heuer, T., Gottesbüren, L., Akhremtsev, Y., Schulz, C., & Sanders, P. (2023). High-quality hypergraph partitioning. *ACM Journal of Experimental Algorithmics*, *27*, 1–39.

Stenius, O., Karaarslan, A. G., Marklund, J., & De Kok, A. (2016). Exact analysis of divergent inventory systems with time-based shipment consolidation and compound Poisson demand. *Operations Research*, *64*(4), 906–921.

Torabi, S., Hassini, E., & Jeihoonian, M. (2015). Fulfillment source allocation, inventory transshipment, and customer order transfer in e-tailing. *Transportation Research Part E: Logistics and Transportation Review*, *79*, 128–144.

Traag, V. A., Aldecoa, R., & Delvenne, J.-C. (2015). Detecting communities using asymptotical surprise. *Physical Review E*, *92*(2), Article 022816.

Trapp, M., Lütjen, M., Arango Castellanos, J. D., Jelsch, O., & Freitag, M. (2017). Life cycle assessment for frozen food distribution schemes. *vol. 24*, In *Digitalization in maritime and sustainable logistics: City logistics, port logistics and sustainable supply chain management in the digital age. Proceedings of the Hamburg international conference of logistics (HICL)* (pp. 267–284). Berlin: Epubli GmbH.

Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications* (pp. 7–15). Springer.

Wagner, S., & Mönch, L. (2023). A variable neighborhood search approach to solve the order batching problem with heterogeneous pick devices. *European Journal of Operational Research*, *304*(2), 461–475.

Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1225–1234).

Wang, F., Li, T., Wang, X., Zhu, S., & Ding, C. (2011). Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, *22*(3), 493–521.

Wiedmer, R., & Griffis, S. E. (2021). Structural characteristics of complex supply chain networks. *Journal of Business Logistics*, *42*(2), 264–290.

Xu, P. J., Allgor, R., & Graves, S. C. (2009). Benefits of reevaluating real-time order fulfillment decisions. *Manufacturing & Service Operations Management*, *11*(2), 340–355.

Yang, Y., Jia, B., Yan, X.-Y., Zhi, D., Song, D., Chen, Y., de Bok, M., Tavasszy, L. A., & Gao, Z. (2023). Uncovering and modeling the hierarchical organization of urban heavy truck flows. *Transportation Research Part E: Logistics and Transportation Review*, *179*, Article 103318.

Yang, D., Tang, M., & Ni, Y. (2025). Robustness of automotive supply chain networks based on complex network analysis. *Electronic Commerce Research*, *25*(5), 3609–3636.

Zhang, Y., Hu, X., & Tian, Q. (2023). Order consolidation for the last-mile split delivery problem with green labels. *Annals of Operations Research*, 1–25.

Zhang, Y., Huang, M., Hu, X., & Sun, L. (2018). Package consolidation approach to the split-order fulfillment problem of online supermarkets. *Journal of the Operational Research Society*, *69*(1), 127–141.

Zhang, Y., Lin, W.-H., Huang, M., & Hu, X. (2021). Multi-warehouse package consolidation for split orders in online retailing. *European Journal of Operational Research*, *289*(3), 1040–1055.

Zhang, Y., Sun, L., Hu, X., & Zhao, C. (2019). Order consolidation for the last-mile split delivery in online retailing. *Transportation Research Part E: Logistics and Transportation Review*, *122*, 309–327.

Zhu, Y., Bao, Y., Qin, L., Sun, Q., Shia, B.-C., & Chen, M.-C. (2025). Resilience analysis based on multi-layer network community detection of supply chain network. *Annals of Operations Research*, 1–25.

Zhu, S., Hu, X., Huang, K., & Yuan, Y. (2021). Optimization of product category allocation in multiple warehouses to minimize splitting of online supermarket customer orders. *European Journal of Operational Research*, *290*(2), 556–571.

Zhuang, Y., Zhou, Y., Hassini, E., Yuan, Y., & Hu, X. (2024). Improving order picking efficiency through storage assignment optimization in robotic mobile fulfillment systems. *European Journal of Operational Research*, *316*(2), 718–732.