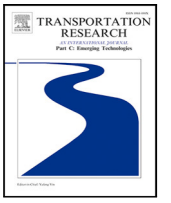
Transportation Research Part C 179 (2025) 105250

Contents lists available at ScienceDirect

Transportation Research Part C

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



Optimizing last-mile delivery through crowdshipping on public transportation networks

Mikele Gajda [a], Olivier Gallay [a], Renata Mansini [b], Filippo Ranza [b,*]

[a] University of Lausanne, Faculty of Business and Economics (HEC Lausanne), Department of Operations, CH-1015, Lausanne, Switzerland
[b] Department of Information Engineering, University of Brescia, via Branze 38, Brescia, 25123, Italy

ARTICLE INFO

Keywords:
Crowdshipping
Last-mile delivery
Public transportation network
Adaptive Large Neighborhood Search
Branch-and-Cut

ABSTRACT

In this paper, we explore an innovative last-mile delivery paradigm that leverages commuters on public transportation (PT) networks as crowdshippers, creating a low-impact delivery model that minimizes environmental footprint while taking advantage of technological advancements, improved infrastructure, and the widespread use of electronic devices. At the beginning of each delivery service period, parcels are routed to selected PT stations by a delivery company, and assigned to a set of crowdshippers (commuters). These crowdshippers collect and deliver the parcels as part of their regular journeys through the PT network, without deviating from their usual routes. The delivery company ensures, through a backup service, the final delivery of parcels that do not reach their final destination. The problem looks for the optimal schedule and route for each parcel while minimizing overall delivery expenses. We call this problem the Public Transportation-based Crowdshipping Problem (PTCP).

We propose a compact Mixed Integer Linear Programming formulation strengthened with valid inequalities and develop an Adaptive Large Neighborhood Search to address large-scale instances. The experimental analysis, conducted on a large set of instances, shows the effectiveness of the proposed heuristic method when compared to the exact model solution. Sensitivity analysis reveals that crowdshipping and backup delivery costs significantly influence the total system cost.

1. Introduction

In the past ten years, e-commerce and on-demand services have experienced significant growth. Industrial reports, like (Chevalier, 2024), foresee these services to grow further over the next few years. This rise has placed an ever-increasing strain on urban centers and has resulted in last-mile logistics becoming a significant contributor to urban issues such as pollution, noise, accidents, and congestion, as discussed by Demir et al. (2015), Savelsbergh and Van Woensel (2016), Manerba et al. (2018). Also, last-mile delivery accounts for more than a quarter of the total delivery cost, making it a significant expense for logistics companies and retailers (Ranieri et al., 2018; Côté et al., 2024). Thus, it is clear that innovative paradigms, both environmentally and economically feasible, that can tackle last-mile delivery, are strongly required (Mansini et al., 2024). In the last decades, several solutions have been proposed to address these issues (Boysen et al., 2021). Among these alternatives, many contributions focus on drones and autonomous robots as new technologies available for widespread delivery (Boysen et al., 2018; Gajda et al., 2024). In this paper, we investigate a crowdshipping model based on public transportation (PT), a new delivery paradigm in which regular

* Corresponding author.
E-mail addresses: mikele.gajda@unil.ch (M. Gajda), olivier.gallay@unil.ch (O. Gallay), renata.mansini@unibs.it (R. Mansini), filippo.ranza@unibs.it (F. Ranza).

https://doi.org/10.1016/j.trc.2025.105250
Received 27 September 2024; Received in revised form 20 June 2025; Accepted 20 June 2025
Available online 12 July 2025
0968-090X/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

commuters (workers commuting to their workplaces, students, and other individuals who regularly use public transportation for their daily activities) are involved in the delivery process. We focus on urban settings equipped with automated parcel lockers (APLs) strategically placed at selected stations of shared transport systems, including subways, buses, trams, urban trains, and ferries (where available). The proposed framework is designed for urban and suburban logistics networks with dense APL deployment and high commuter availability, making same-day delivery feasible. It is particularly suitable for e-commerce platforms, postal services, and logistics providers aiming to consolidate deliveries at APLs instead of distributing parcels to individual households.

PT commuters register on a digital platform, specifying their availability in terms of days, times, and routes. A pairing system then assigns packages to available crowdshippers, aligning with their regular journeys. Each parcel is associated with an origin and destination station. The Logistics Service Provider (LSP), typically responsible for home deliveries, indicates the stations from which each parcel can enter the PT network, while the recipient (the final customer) specifies multiple preferred pick-up stations within a reasonable distance from home. Parcels are transported by crowdshippers along their regular commuting routes, either directly to the recipient's chosen destination station or to an intermediate transfer point, such as a station where commuters switch lines or modes of transport, for further routing. Since crowdshippers cannot deviate from their planned journeys, some parcels may require multiple transfers between different crowdshippers at intermediate stations before reaching their final destination station. PT-based crowdshipping is therefore an innovative paradigm that allows LSPs, typically responsible for last-mile delivery, to bypass urban centers entirely. Instead, packages are delivered to APLs at urban peripheral stations, significantly reducing traffic congestion and the environmental impact in city centers. The crowdshipping system handles small to medium-sized parcels (e.g., e-commerce items under 5 kg), excluding perishable or fragile goods. Parcels assigned for crowdshipping delivery are pre-qualified by the LSP to meet size, volume, and weight requirements, ensuring compatibility with standard slots in designated APLs.

Each operational day, early in the morning and before peak commuter hours, the LSP determines the optimal starting station for each parcel. Parcels are transported by truck and delivered to selected APLs within the PT network, ensuring they are ready for pick up when crowdshippers arrive. A pool of crowdshippers, with known travel routes and journey timings, is available to handle the delivery of each parcel between its source and destination station. Deliveries must be completed within the same operational day, aligning with urban logistics practices, without strict time windows. Once a parcel reaches its destination APL, the last-mile process is considered complete, as the destination APL serves as the customer's designated endpoint. By treating the destination APL as the terminal node in the delivery chain, we align with urban logistics trends where customers opt for self-service pick-up points, thereby reducing failed deliveries and improving flexibility.

In this paper, we study a new problem to optimize parcel delivery through a PT network. We jointly optimize two main segments: from the LSP's central warehouse to the APL at the source station, and from there to the destination APL, intentionally decoupling it from the retrieval phase between the destination APL and the final customer (recipient). The problem looks for:

(i) the selection of a starting station (*source station*) for each parcel. Each source is selected from a subset of stations identified by the LSP and will receive its parcels directly from the central warehouse by the LSP truck.
(ii) the selection of the final station (*destination station*) for each parcel. Each destination is selected from a subset of stations specified by the customer following their preferences.
(iii) parcels assignment to crowdshippers and their routing/scheduling through the network while satisfying APL capacity constraints in each station. Since crowdshippers cannot deviate from their regular commuting routes, the model accounts for the possibility that some parcels may require multiple transfers and temporary storage at intermediate stations, potentially involving coordination between different crowdshippers for the final delivery to their destination stations.

Parcels that cannot be included in daily planning due to capacity constraints or other limitations are delivered via a backup service arranged by the LSP. This ensures that all parcels reach their destination at the end of the day, even if some may require alternative delivery arrangements. The problem aims to minimize the total cost, which includes the remuneration of crowdshippers, the cost of transferring parcels from the warehouse to the source stations, the cost of parcel loading in APL, and the backup cost for undelivered parcels to their destination stations. We call this problem the Public Transportation-based Crowdshipping Problem (PTCP).

From an economic standpoint, the advantages of this delivery paradigm are evident. Crowdshippers receive a fixed travel credit in their customer accounts, which can be redeemed for discounted or subsidized travel with the LSP. On the other hand, through crowdshipping, LSPs can optimize delivery routes and vehicle capacity, thereby reducing operational costs such as fuel and labor. Additionally, this system supports an eco-friendly approach: recipients contribute by opting for crowdshipping rather than requesting door-to-door delivery, which further enhances the sustainability of the overall delivery process.

This paper provides the following contributions:

- Although the issue of crowdshipping in urban PT networks is a known paradigm in the literature (Filippi and Plebani, 2021; Boysen et al., 2021; Wyrowski et al., 2024), We propose a new problem in relation to the state of the art. First, unlike other formulations, PTCP explicitly allows crowdshippers to transfer a parcel to any subsequent station on their journey. This feature is integrated into a multi-step routing system, where a parcel can be delivered by more than one crowdshipper if necessary. Second, the problem evaluates the overall cost-effectiveness of the backup service with respect to the use of crowdshippers. In some cases, relying on the backup service may be preferable if transferring the parcel through multiple crowdshippers results in higher costs. Although none of these features is novel on its own, to the best of our knowledge, this is the first time they have been integrated into a single problem.

- We develop a mixed-integer linear programming (MILP) formulation for the PTCP. Our model minimizes overall costs by deciding the source and destination stations for each parcel, assigning parcels to crowdshippers, and routing them through the PT network while guaranteeing the less expensive option between using crowdshippers or the backup service.
- We strengthen the proposed mathematical formulation by introducing different valid inequalities. We compare their impact when all are included from scratch versus when they are dynamically separated during the solution process.
- We develop an Adaptive Large Neighborhood Search (ALNS) metaheuristic algorithm with two destroy and four repair operators, that can efficiently handle large-scale instances derived from different PT networks and commuter traffic flows.
- We generate a comprehensive set of problem instances reflecting different PT system structures to assess the applicability of our model and the performance of our solution approaches across various scenarios.
- We provide relevant managerial insights for operators in the logistics sector. In particular, our sensitivity analysis results indicate that several factors are crucial for the success of a PT-based crowdshipping system. Specifically, these systems are highly influenced by cost structure, crowdshipper availability, and system scale. Effective implementation requires a careful balance among these factors, maintaining a large crowdshipper network while optimizing the backup delivery system to ensure reliability.
- Although the proposed model is deterministic, we analyze the impact of potential delays and disruptions on travel times, which may affect parcel pick-up schedules and render some delivery plans infeasible. Specifically, we perform a sensitivity analysis to assess how delays along a transit line influence the final solution.

The remainder of the paper is organized as follows. In Section 2, we review the current state of the art in last-mile delivery and the crowdshipping paradigm, both for private and public transportation systems. In Section 3, we introduce the notation and formally state the problem. In Section 4, we describe the mathematical formulation for PTCP and discuss the valid inequalities. In Section 5, we detail the implemented ALNS algorithm. In Section 6, we present the experimental setup and discuss the results for both the MILP model and the ALNS algorithm, along with various sensitivity analyses and managerial insights. Finally, in Section 8, we draw concluding remarks and propose possible future research directions.

2. Literature review

Transportation research has extensively examined the critical role of last-mile delivery in the movement of goods. This final leg of the delivery process has traditionally relied on delivery vans, yet it remains the most inefficient and expensive component, accounting for over 40% of total supply chain costs in the United States (Joerss et al., 2016). The rapid growth of e-commerce, coupled with increasing customer expectations for faster and more flexible deliveries, has intensified pressure on last-mile logistics systems (Boysen et al., 2021). Contemporary challenges include high delivery costs, restricted delivery time windows, and a proliferation of delivery vehicles on urban roads, contributing to congestion, emissions, noise pollution, and increased accident risks (Department of Transportation, NYC, 2019).

The need for innovative last-mile delivery solutions has led researchers to rethink traditional paradigms, exploring approaches like crowdshipping, which leverages commuter mobility for parcel delivery. In parallel, urbanization and sustainability challenges have driven interest in underground freight transportation (UFT) to reduce congestion and improve efficiency. Powell et al. (2024) propose UFT with electric autonomous vehicles in tunnels, modeling it as a fixed-charge multicommodity flow problem. Their study shows that a 45-mile network in Chicago could divert 42% of packages from roads and reduce costs by 40%. Zhang et al. (2022b) analyze 222 studies on underground logistics, focusing on metro-based networks. Their review identifies key research areas like system planning, vehicle design, and operations, while noting a gap between theory and practice, echoing (Powell et al., 2024) on the need for government support.

Integrating freight movement with existing PT assets is another promising strategy. Freight on transit (FoT) systems aim to utilize the surplus capacity on buses, trams, or even metro vehicles to transport cargo. In the context of FoT, Elbert and Rentschler (2022) discuss various operational models and optimization techniques for sharing vehicles and infrastructure between passenger and freight services. This review describes the evolution of FoT models, from small pilot projects (e.g., off-peak metro deliveries) to more comprehensive mathematical frameworks that account for emissions, congestion, and stochastic demand. While mixed-integer programming is able to optimize shared vehicle routing, challenges like passenger resistance and scheduling conflicts require adaptive pricing models that can adjust to changing conditions. Further exploring this integration concept, Derse and Van Woensel (2024) present a conceptual framework for integrated people-and-freight transportation, combining multiple approaches at the strategic, tactical, and operational levels. This study highlights the socio-economic benefits of such a system, including reductions in urban traffic density by 18%–24%, while identifying policy gaps and passenger dissatisfaction as key barriers to adoption. These findings underscore the need for data-driven public–private partnerships to address conflicting interests. (Mo et al., 2023) investigate the Vehicle Routing Problem with Underground Logistics (VRP-UL), which explores transporting freight using a combination of traditional vehicles and subway trains. In this model, freight originates at a depot and can initially be routed via either vehicle or subway. During the delivery process, parcels transferred by subway are offloaded at predetermined stations and then loaded onto trucks for final delivery to the customer. The objective is to minimize the overall costs, including routing, vehicle activation, and the cost associated with transferring freight through the subway system. To solve this problem, the authors propose a MILP model and a specialized ALNS heuristic. Computational experiments using real-world data from an e-commerce company in Beijing demonstrate that VRP-UL can reduce overall costs by up to 18% compared to a standard VRP approach.

Parallel to routing optimization, a critical planning decision in integrated systems using public transit like buses is determining the appropriate number of vehicles to equip for carrying goods. Equipping too few vehicles risks insufficient capacity and poor service quality, while equipping too many leads to excessive costs. To address this capacity planning challenge, Machado et al. (2023) propose a decision-support system. In this context, a MILP model (formulated as a variation of the bin-packing problem) and two Greedy Randomized Adaptive Search Procedures (GRASPs) are introduced to identify the minimum number of buses needed to reliably accommodate parcels across various operational scenarios. Computational experiments demonstrate that the proposed heuristics efficiently produce high-quality results. For readers seeking a comprehensive review of the literature on integrated delivery systems, Cheng et al. (2023) provide an in-depth analysis that highlights the current state of research, outlines future directions, and discusses the practical advantages and challenges of combining passenger and parcel transportation.

Modular vehicle routing studies have explored the consolidation of passenger and freight demand into a single vehicle platoon. In this context, Hatzenbühler et al. (2023) demonstrates how combining passenger and freight demand can result in 48% cost savings through the Modular Multipurpose Pick-up and Delivery Problem framework. A case study in Stockholm confirms a 60% reduction in empty vehicle kilometers, with demand consolidation contributing an additional 9% efficiency gain. Despite challenges like limited depot space and regulations, the study supports the idea that transit and logistics operators should collaborate, especially through adaptive pricing and infrastructure adjustments.

Within this evolving landscape of infrastructural and technological innovations, crowdshipping emerges as a promising paradigm that addresses many last-mile challenges through a fundamentally different approach. Unlike traditional solutions, crowdshipping, also known as crowd logistics, leverages the existing mobility patterns of individual commuters to facilitate parcel delivery. The operational framework of crowdshipping generally manifests mainly through two different models. The first approach involves occasional drivers who integrate parcel deliveries into their personal vehicle commutes, accepting minor detours to transport goods. These drivers utilize spare cargo space in private vehicles, adapting their routine travel patterns to additionally accommodate the delivery they are assigned. The second model, and the primary focus of this analysis, exploits the potential of PT systems. Here, commuters using trains, buses, or metros voluntarily participate as couriers, transporting parcels between stations based on their commute. Crowdshipping represents a shift toward more flexible, adaptive logistics systems that can potentially reduce both costs and environmental impacts by utilizing excess capacity in existing PT networks. This model aligns with the broader principles of the sharing economy while tackling key last-mile delivery challenges. As Carbone et al. (2017) highlight, crowdshipping presents notable advantages over traditional delivery models, particularly in urban density, delivery flexibility, and sustainability performance.

Research on occasional drivers includes several studies. In Archetti et al. (2016), the authors introduce the Vehicle Routing Problem with Occasional Drivers (VRPOD), which involves both regular vehicles and non-professional delivery couriers, and results in cost savings and efficiency improvements. In this work, occasional drivers are required to deviate from their usual routes to deliver packages in exchange for a small financial reward. The authors propose a multi-start matheuristic that integrates both metaheuristic and exact optimization strategies. The approach starts by generating diverse initial solutions using a sequential greedy insertion algorithm, alongside integer programming to assign customers to either regular or occasional drivers based on criteria such as distance, demand, and cost. An internal tabu search is then applied, using moves like 1-move, swap-move, in-move, and out-move, to iteratively improve the solution while avoiding cycling due to temporary tabu restrictions. The study explores various compensation methods for occasional drivers and demonstrates that this crowdsourced delivery system can lead to more cost-efficient solutions. More specifically, significant cost savings can be achieved when a large number of flexible occasional drivers are available for deliveries. Building on this, Arslan et al. (2019-02) extend the analysis by considering a dynamic version of the problem, where an online crowdsourcing platform continuously receives new delivery tasks and driver trip announcements over time. A rolling horizon framework is developed to reoptimize the system whenever new information becomes available. In addition, their heuristic accelerates the routing subproblem by pruning infeasible routes early and storing only a limited number of the most promising candidate routes, particularly under strict route duration constraints for backup vehicles. This approach reduces computational complexity without compromising solution quality. Results indicate that employing occasional drivers can significantly enhance cost-efficiency in last-mile delivery operations, potentially achieving system-wide vehicle-mile savings of up to 37% compared to conventional delivery systems relying solely on dedicated vehicles. (Dayarian and Savelsbergh, 2020) analyze a dynamic setting that accounts for uncertainty in both future demand (online orders) and delivery capacity (owned and crowdsourced). A specific constraint imposes that the delivery of an order must occur within a fixed time window after the order is placed. The authors developed two variants (static and dynamic) of a fast algorithm capable of making real-time decisions based on incoming demand and delivery capacity. In contrast, Di Puglia Pugliese et al. (2023) introduce the VRPOD with Time Windows (VRPODTW), considering uncertain travel times and penalties for missed deliveries. The problem is formulated using a chance-constrained stochastic model to develop a distribution plan that minimizes the total costs, including routing expenses for regular drivers, compensation for occasional drivers, and penalties for violating customer time windows. Additionally, the authors propose a robust model and apply two decomposition techniques: Benders decomposition and column-and-row generation, leveraging the problem's two-stage nature. In Archetti et al. (2021), the online VRPOD is studied, where a company's fleet is supplemented by occasional drivers available within specific time windows. The research focuses on addressing dynamic customer requests, assuming a prior knowledge of crowdshippers' availability. The objective function aims to minimize overall distribution costs and incorporates a penalty function for time window violations. Unlike previous studies, such as (Archetti et al., 2016; Dayarian and Savelsbergh, 2020), the occasional drivers can handle multiple deliveries during their assigned routes. Despite concerns about the negative impacts of delivery vans in urban areas, most research on crowdshipping systems has largely focused on operational challenges and cost efficiency. Although these studies have optimized logistics processes, they have not addressed the impact of additional vehicles and their detours on

urban traffic, congestion, and pollution. Thus, a research gap remains, focusing on strategies that enhance cost-effectiveness while mitigating environmental concerns in crowdshipping systems.

Gatta et al. (2018) highlight the environmental and economic impacts, revealing potential reductions in particulate matter and greenhouse gas emissions through a PT-based crowdshipping system integrated with APLs. Results indicate significant environmental benefits, such as an average annual reduction of 239 kg in particulate matter emissions. Gatta et al. (2019) identify key factors influencing crowdshipping, highlighting that in-station APLs are preferred over external ones, even more than monetary incentives. The authors also emphasize the importance of delivery scheduling flexibility for consumers and the reluctance of crowdshippers to significantly alter their commuting routes. Similarly, Filippi and Plebani (2021) analyze a metro-based crowdshipping service in Brescia, Italy, finding that young and retired metro users are willing to participate without monetary incentives. The authors assess the economic impact of combining metro-based crowdshipping with traditional delivery, using a MILP model. The results show that this integration could reduce delivery costs by up to 66% and vehicle miles traveled by 17% compared to standard van deliveries. Fessler et al. (2022) find that younger individuals, students, and employed people are most willing to carry parcels on PT in Copenhagen. Higher compensation increases participation, while larger parcels decrease willingness.

The concept of PT-based crowdshipping is closely linked to the successful implementation of APLs and their location (Gatta et al., 2019). The adoption of APLs provides a convenient delivery method, with growing popularity worldwide (Zurel et al., 2018). (Seghezzi et al., 2022) analyze the cost benefits of APLs. An analytical model compares the cost of home delivery and delivery through APLs. Results show that APLs can significantly reduce delivery costs by 60% averaging 1.2 €per parcel versus 2.9 €for home delivery. (Vakulenko et al., 2018) highlight that the most important factor for customers is the location of APLs, preferring those close to their destination or on their way to work/home. Balancing this customer-centric approach with environmental concerns, Bonomi et al. (2022) explore the strategic positioning of APLs. This study shows that while the placement of APLs is crucial, the eco-conscious behavior of consumers has the highest impact on reducing emissions. This outcome depends on two key parameters: the maximum distance individuals are willing to travel to reach a locker and that they are willing to cover by foot or bicycle. (Zurel et al., 2018) analyze implications of APLs on logistics and postal systems. The authors unveil APLs' cost efficiency and sustainability benefits by comparing the integration of these systems across different countries and focusing on regulatory considerations accompanying the widespread adoption of APLs.

Relevant research on the operational challenges of PT-based crowdshipping has emerged in recent years, and it is within this specific area of study that our project is situated. Within the context of Singapore's PT system, Zhang et al. (2022a) assess the viability and benefits of the PT-based crowdshipping delivery method. Focusing on the congested urban environment of Singapore, the authors assess the feasibility and impact of crowdshipping on e-commerce logistics. A parcel allocation problem is introduced to match parcels with suitable public bus journeys, but no mathematical formulation is provided. When an insufficient number of crowdshippers are available, traditional delivery vans are used to complete the deliveries. A simple heuristic algorithm is proposed to solve the problem. The findings reveal that crowdshipping can significantly reduce kilometers traveled and air emissions by up to 17%, while the delivery cost per parcel improves by up to 29%, even when limited to off-peak hours. Kızıl and Yıldız (2023) propose a slightly different variant of the problem discussed by Zhang et al. (2022a). The main idea is to place APLs in PT stations and other strategically located spots around the city, known as *satellites*. In this framework, crowdshippers are employed to transfer parcels from the satellites to the PT stations, while the transfers between PT stations are handled by the PT vehicles (i.e., trains) independently of the crowdshippers. If no crowdshipper is available, a backup transfer vehicle is used instead. The authors formulate a two-stage stochastic problem that accounts for uncertainties in demand and crowdshipper capacity. To efficiently solve realistic instances of the problem, they propose a Branch-and-Price algorithm enhanced with a new branching rule. This approach can achieve up to an 84.4% reduction in run time, with an average reduction of 61.5% across all problem instances, compared to the standard Branch-and-Price implementation. Different scenarios involving 10 to 80 crowdshippers and 149 network nodes are considered (39 PT connection locations and 110 satellites outside the PT system). The best algorithm can optimally solve the instance with 80 crowdshippers in 7356 s (approx. 123 min).

A more complete analysis of the PT-based crowdshipping paradigm is provided by Wyrowski et al. (2024). The authors examine a problem where metro users can transfer parcels between stations. They provide a mathematical formulation for the problem and independently test five different objective functions: the maximization of (i) the profit (difference between the income for the delivered parcels and the crowdshipper remuneration cost); (ii) the number of delivered parcels; and (iii) the crowdshipper involvement; the minimization of (iv) the total number of entrainments (joint travel of a crowdshipper and a parcel from one station to another) and (v) the maximum length of entrainment sequences. Unlike the present work, their framework does not consider the first leg of delivery, which involves transshipment of parcels from the warehouse to the starting stations, nor a backup system for parcels that remain unserved by the crowdshippers. We account for both such aspects dealing with the selection of a starting station for each parcel and the assignment of parcels to crowdshippers depending on the initial cost of the logistics service provider to deliver the parcel, as well as the backup cost. Additionally, in the formulation by Wyrowski et al. (2024), a crowdshipper can only transfer a parcel from their current location to the immediate next location on their path. This simplification restricts the set of feasible paths for parcels. In our formulation, we explicitly consider the possibility of a crowdshipper delivering a parcel not only to the next station on their route but to any subsequent station being the final destination or any exchange station. This assumption leads us to introduce a discrete time setting instead of the continuous one used in Wyrowski et al. (2024). Wyrowski et al. (2024) propose a decomposition heuristic to solve the problem, which involves generating numerous parcel paths and selecting the best subset using a MILP model. The authors recognize that their heuristic is generally outperformed by the pure solver approach and is only a viable option when the number of crowdshippers exceeds twice the number of parcels to be delivered. Although their work introduces a first attempt to tackle the complex PT-based crowdshipping problem by focusing on parcel movement through

Table 1
Comparison between the study in Wyrowski et al. (2024) and the present one.

	Wyrowski et al. (2024)	This paper
Backup cost		✓
Loading and delivery costs		✓
Multiple crowdshippers	✓	✓
Valid inequalities	✓	✓
Delivery time windows	✓	
Different objective functions	✓	
Selection of the starting station		✓
Selection of the destination station		✓
Explicit time synchronization		✓
Time management	Continuous	Discrete
Main objective function	Profit maximization	Cost minimization
Parcel transfer	Subsequent station	Any exchange station
Heuristic framework	MILP decomposition	ALNS
Maximal number of parcels	100	250
Maximal number of crowdshippers	300	300
Maximal number of stations	147	44

the PT stations, it does not fully address the integrated parcel delivery system. In Table 1, we summarize the main differences between the study in Wyrowski et al. (2024) and the present work. While recent literature on crowdshipping delivery systems has been expanding and addressing various issues, this work introduces a new MILP formulation for the global PT-based crowdshipping delivery problem. Our approach manages synchronization between crowdshippers coordinated to transport the same parcel, while minimizing the overall system cost.

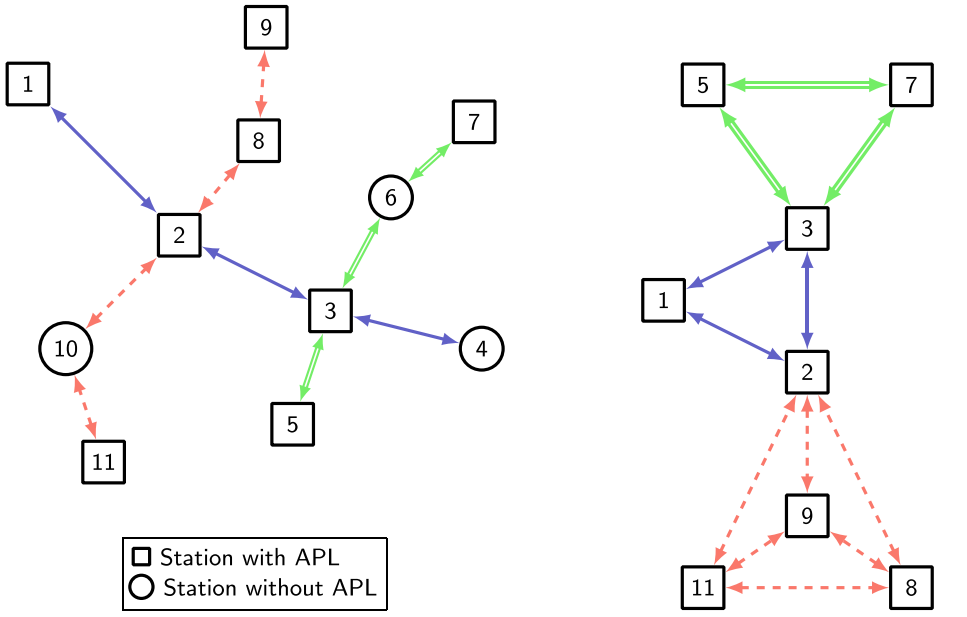
In this work, we develop an efficient ALNS (Ropke and Pisinger, 2006) metaheuristic to deal with real-size instances of the problem. ALNS-based metaheuristics, as highlighted in a recent survey by Windras Mara et al. (2022), have proven to be very effective and adaptable for various combinatorial optimization problems, especially in vehicle routing. The survey highlights the algorithm's ability to handle tightly constrained problems, shown by its successful use in several two-echelon VRPs (Grangier et al., 2016; Hemmelmayr et al., 2012; Jie et al., 2019) and large-scale crowdsourcing routing problems in the food delivery industry with large vehicle networks (Liu et al., 2019), among others.

3. Problem definition and notation

We define the parcel delivery system over a public transport (PT) network consisting of a set $L = \{1, \dots, l_{\max}\}$ of $l_{\max}$ transportation lines and a set $\bar{N}$ of *network stations* equipped with APLs with capacity C_i, representing the number of parcels that can be stored at station $i \in \bar{N}$. Each transportation line $l \in L$ serves a subset of network stations $\bar{N}_l \subseteq \bar{N}$. We denote as $\bar{S} \subset \bar{N}$ the set of *source stations*, indicating the subset of network stations where parcels can enter the network. Which of these stations are actually used as entry points depends on the specific delivery plan and the LSP's scheduled stops. Similarly, the subset $\bar{D} \subset \bar{N}$ defines the set of *destination stations*, where recipients can collect parcels, with the understanding that the final selection of these stations will depend on the choices made by the final recipients. We assume $\bar{S} \cap \bar{D} = \emptyset$. This is not a restrictive assumption: if a station is both a source and a destination, the parcel is directly delivered by the truck during its stop, making crowdshipping unnecessary. Some stations serve as *exchange stations*, enabling passengers and, consequently, parcels to switch between different transportation lines. A station $i \in \bar{N}$ is an exchange station if there exists a set $\mathcal{L}_i \subseteq L$ such that $|\mathcal{L}_i| \geq 2$ and $\cap_{l \in \mathcal{L}_i} \bar{N}_l \neq \emptyset$. Note that the set of network stations $\bar{N} \setminus (\bar{S} \cup \bar{D})$ includes all exchange stations that are not part of $\bar{S}$ or $\bar{D}$, as well as additional network stations (not necessarily exchange stations) that serve as entry or exit points for crowdshippers. Each station $i \in \bar{S}$ has two associated cost coefficients: an activation cost ν_i if the LSP selects the source station as entry point, and a cost σ_i proportional to the number of parcels loaded by the LSP at that station.

A set K of *crowdshippers* operates within the PT network. Each crowdshipper follows the shortest path (in terms of travel time) between their entry and exit stations without deviating from their planned route. The sequence of network stations traversed by a crowdshipper k, where parcels can be picked up or delivered, is denoted as $N_k \subset \bar{N}$ for each $k \in K$. A crowdshipper can transport parcels only if their route meets specific conditions: they must either pick up a parcel at their entry station, which must be a network station, or their route must pass through at least one exchange station. If the crowdshipper picks up the parcel at an exchange station, it must be delivered either to the crowdshipper exit station or another exchange station. We indicate as $K_{ij} \subseteq K$ the subset of crowdshippers who can transfer a parcel from station i to station j, being two stations of their planned route. Each crowdshipper $k \in K$ receives a fixed remuneration $\rho > 0$ for their service. To explicitly prevent professionalization, our system employs non-monetary and distance-independent incentives: crowdshippers earn rewards exclusively as public transit fare credits (e.g., discounted or free tickets), avoiding cash exchanges. This ensures that participation remains casual and aligns with existing mobility patterns of the commuters, rather than encouraging full-time roles.

To formalize the problem, we introduce three key graph representations that progressively refine the structure of the initial PT network:



(a) Real network (b) Transfer graph $\bar{G} = (\bar{N}, \bar{A})$

Fig. 1. Example of a real network and its associated transfer graph.

- **Transfer Graph**: We first construct a *transfer graph* $\bar{G} = (\bar{N}, \bar{A})$, which models the real PT network by considering only the network stations equipped with APLs. We define as $\bar{A}_l = \{(i,j) | i,j \in \bar{N}_l\}$, the set of arcs representing all possible parcel transfers on the line l and $\bar{A} = \bigcup_{l \in L} \bar{A}_l$ all possible transfers on the PT network. Each arc $(i,j) \in \bar{A}$ corresponds to the shortest path between network stations i and j in the real network.
  In Fig. 1(a), we show an example of a small PT network consisting of three transportation lines (blue, red, and green) and 11 stations, where the nodes drawn as squares represent network stations equipped with APLs, whereas nodes represented as circles are stations without lockers. Arcs with the same color and pattern belong to the same PT line. Note that when referring to a transportation line, we consider both directions of travel, which typically operate on two separate tracks. Stations 2 and 3 are the exchange stations where lines blue and red and lines blue and green intersect, respectively. The corresponding transfer graph (represented in Fig. 1(b)) is obtained by excluding the stations without lockers and constructing all possible connections between the network stations belonging to each line.
- **Crowdshipper Delivery Graph**: Given the transfer graph, each crowdshipper $k \in K$ operates on a *delivery graph* $G_k = (N_k, A_k)$, where $N_k \subset \bar{N}$ consists of the network stations that the crowdshipper can interact with, meaning the stations from which parcels can be picked up or delivered without deviating from the planned route. The set $A_k = \{(i,j) \in \bar{A} \mid i,j \in N_k\}$ represents the set of feasible parcel transfers for that crowdshipper.
  In the left part of Fig. 2, we show the individual delivery graphs associated with three crowdshippers traveling on their regular journeys over the transfer graph reported in Fig. 1(b). The routes of the three crowdshippers are: $\{9,2,3,7\}$, $\{1,3,7\}$, and $\{11,2,1\}$, respectively. The arrows in each of Figs. 2(a) to 2(c) show all the possible ways a crowdshipper can transfer parcels. For example, Fig. 2(a) represents the delivery graph of the first crowdshipper. Since this crowdshipper follows the path $\{9,2,3,7\}$, a parcel can be transferred from station 9 to stations 2, 3, or 7; from station 2 to stations 3 or 7; or from station 3 to station 7. Notice that the transfers do not include station 8, even though it is a network station. This is because the station 8 is neither an entry nor an exit station for that commuter, nor does it serve as an exchange station. Similarly, Figs. 2(b) and 2(c) represent the delivery graphs for the second and third crowdshippers, respectively.
- **Global Delivery Graph:** The final directed graph $G = (N, A)$ is the global delivery graph obtained as the union of the delivery graphs of all crowdshippers. The node set $N = \cup_{k \in K} N_k$ includes all stations where at least one crowdshipper can perform a transfer, and the arc set $A = \cup_{k \in K} A_k$ represents all feasible transfers in the system. For each arc $(i,j) \in A$ we indicate as t_{ij} the cumulative travel time across the shortest path between i and j in the transfer graph. Given $G = (N, A)$, we redefine the sets of the source and destination stations, $\bar{S}$ and $\bar{D}$, to include only the stations used by at least one crowdshipper: $S = N \cap \bar{S}$, $D = N \cap \bar{D}$, with $S \cap D = \emptyset$.
  Fig. 2(d) shows the global delivery graph obtained as union of the three crowshippers delivery graphs reported in Figs. 2(a)–2(c).

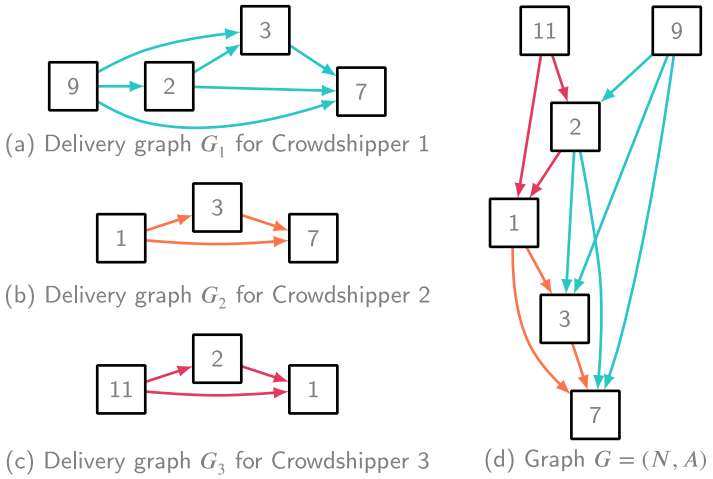


Fig. 2. Union of crowdshipper delivery graphs G_1, G_2, and G_3 into global delivery graph G.

Finally, let P be the set of parcels to be delivered. One or more crowdshippers can transfer parcels. Each parcel $p \in P$ is associated with a set of feasible destination stations $D_p \subseteq D$ selected by the corresponding recipient and a backup cost γ_p, incurred if the parcel is handled by the LSP, which uses a vehicle to retrieve and transport parcels that the crowdshipping system does not deliver. Moreover, once a source station $i \in S$ for a parcel is selected, a fixed cost ν_i is incurred, representing the transportation cost of the parcel from the depot to station i by the LSP vehicle, along with a unit cost σ_i for loading the parcel in the APL. To synchronize the parcel transfer between consecutive crowdshippers while complying with the capacity constraints for the APL at each network station, the delivery day is divided into equally sized time slots. Let T represent the set of available time slots. We assume that each parcel can be picked up from or dropped off at a network station no more than once per time slot. This time discretization allows for accurately calculating the total capacity used at each station during a given time slot and ensures that a parcel is handled by at most one crowdshipper per time slot. To simplify the problem formulation, we assume that the time slot at which each crowdshipper starts from their initial station is known in advance, meaning we are aware of when each crowdshipper begins their journey. We also assume that the average travel time between any station and the next one is known. These assumptions make the problem fully deterministic and allow us to handle time within the model without relying on dedicated time variables. To this end, we denote by K_i^t the set of crowdshippers present at network station $i \in N$ at time slot $t \in T$.

The PTCP aims to find the optimal assignment of parcels to crowdshippers and their optimal route through the network, while determining both the source and destination station for each parcel and complying with capacity constraints of APL at network stations. The problem is framed from the perspective of the LSP, seeking to minimize the total costs. These costs include the remuneration paid to crowdshippers, the backup costs for delivering parcels to their destinations using the LSP vehicles, and the costs for loading parcels at the source and exchange stations. It is worth noting that the backup service is not only employed when no crowdshippers are available to complete a parcel transfer, but also when relying on a large sequence of crowdshippers would result in excessive costs, making vehicle-based transportation a more convenient alternative. As an illustrative example, consider the parcel transfer depicted in Fig. 3, where the parcel starts at source station 5 and is delivered to destination station 9 via a route involving sequentially four distinct crowdshippers. Crowdshipper 1 travels directly from station 5 to 7 without line changes; crowdshipper 2 takes the parcel from station 7 to exchange station 3; crowdshipper 3 travels from station 3 to station 2 leaving the parcel there, whereas crowdshipper 4 enters station 2, retrieves the parcel, and leaves it at destination station 9. Assuming that the remuneration for each crowdshipper is $\rho = 2$ and that the backup cost for the parcel is $\gamma_p = 6$, using the backup service becomes more cost-effective than relying on crowdshippers. Moreover, this simplified comparison considers only the remuneration of crowdshippers, neglecting additional costs such as the potential activation cost of station 5 and the expense of loading the parcel into the APL.

The sets and parameters required for the formal definition of the problem are summarized in Tables 2 and 3, respectively.

4. Mathematical formulation

In this section, we introduce a new compact Mixed-Integer Linear Programming (MILP) formulation for the PTCP. The sets of decision variables required to formulate the model are defined in Table 4.

The MILP for the PTCP is stated as follows:

$$\min \rho \sum_{k \in K} z_k + \sum_{i \in S} \nu_i e_i + \sum_{i \in S} \sigma_i \sum_{p \in P} s_{ip} + \sum_{p \in P} \gamma_p q_p \tag{1}$$

s. to:

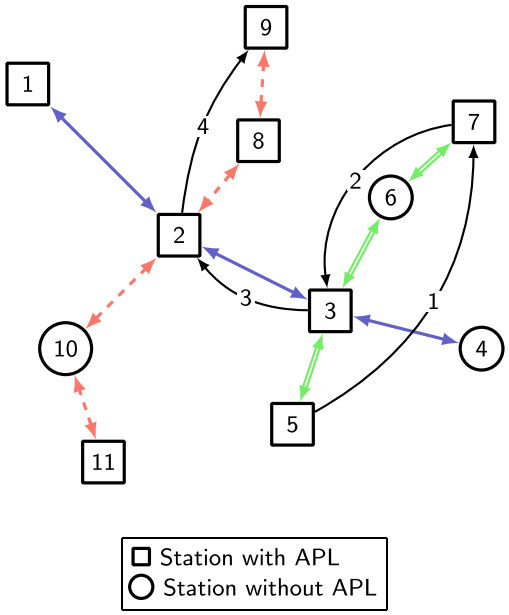


Fig. 3. Example of a parcel transfer with crowdshippers' flow indicated by a black arrow and their id.

Table 2
Sets.

P	Set of parcels to be delivered
K	Set of available crowdshippers
N	Set of network stations visited by at least one crowdshipper
A	Set of arcs connecting nodes in N
S	Set of source stations
D_p	Set of possible destination stations for parcel $p \in P$ provided by the recipient
D	Set of possible destination stations for all parcels, $D = \cup_{p \in P} D_p$
A_k	Set of arcs in A traversed by crowdshipper $k \in K$
T	Set of discrete time slots
K_i^t	Set of crowdshippers visiting stations $i \in N$ at time slot $t \in T$
K_{ij}	Set of crowdshippers moving from station $i \in N$ to station $j \in N$ at any time

Table 3
Parameters.

ρ	Reward granted by the LSP to crowdshippers
v_i	Fixed cost to send a delivery vehicle to source station $i \in S$
σ_i	Unitary cost to load a parcel into source station $i \in S$
γ_p	Backup cost for parcel $p \in P$
C_i	Maximum capacity of the APL located at station $i \in N$

$$\sum_{i \in S} s_{ip} = 1 - q_p \qquad p \in P \tag{2}$$

$$\sum_{i \in D_p} d_{ip} = 1 - q_p \qquad p \in P \tag{3}$$

$$\sum_{p \in P} s_{ip} \leq C_i e_i \qquad i \in S \tag{4}$$

$$\sum_{j \in N} \sum_{k \in K_{ij}} x_{ij}^{kp} - \sum_{j \in N} \sum_{k \in K_{ji}} x_{ji}^{kp} = s_{ip} - d_{ip} \qquad p \in P,\ i \in N \tag{5}$$

$$x_{ij}^{kp} \leq y_{kp} \qquad p \in P,\ k \in K,\ (i,j) \in A_k \tag{6}$$

$$\sum_{p \in P} y_{kp} \leq z_k \qquad k \in K \tag{7}$$

$$l_{ipt} = \sum_{j \in N} \sum_{k \in K_i^t} x_{ij}^{kp} \qquad p \in P, i \in N, t \in T \tag{8}$$

Table 4
Decision variables.

x_{ij}^{kp}	Binary variable taking value 1 if crowdshipper $k \in K$ transfers parcel $p \in P$ from station $i \in N$ to station $j \in N$; 0 otherwise.
z_k	Binary variable taking value 1 if crowdshipper $k \in K$ is selected; 0 otherwise.
y_{kp}	Binary variable taking value 1 if parcel $p \in P$ is assigned to crowdshipper $k \in K$; 0 otherwise.
o_{ipt}	Binary variable taking value 1 if parcel $p \in P$ is at station $i \in N$ at time slot $t \in T$; 0 otherwise.
a_{ipt}	Binary variable taking value 1 if parcel $p \in P$ arrives at station $i \in N$ at time slot $t \in T$; 0 otherwise.
l_{ipt}	Binary variable taking value 1 if parcel $p \in P$ leaves station $i \in N$ at time slot $t \in T$; 0 otherwise.
s_{ip}	Binary variable taking value 1 if parcel $p \in P$ starts its route at source station $i \in S$; 0 otherwise.
d_{ip}	Binary variable taking value 1 if parcel $p \in P$ terminates its route at destination station $i \in D$; 0 otherwise.
e_i	Binary variable taking value 1 if source station $i \in S$ is selected; 0 otherwise.
q_p	Binary variable taking value 1 if parcel $p \in P$ is delivered by the backup service; 0 otherwise.

$$a_{ipt} = \sum_{j \in N} \sum_{k \in K_i^t} x_{ji}^{kp} \qquad p \in P, i \in N, t \in T \tag{9}$$

$$o_{ip(t+1)} \geq a_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T|-1\} \tag{10}$$

$$o_{ip(t+1)} \geq o_{ipt} - l_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T|-1\} \tag{11}$$

$$o_{ip(t+1)} \leq 1 - l_{ipt} \qquad i \in N, p \in P, t \in \{1, \ldots, |T|-1\} \tag{12}$$

$$\sum_{i \in N} o_{ipt} \leq 1 - q_p \qquad p \in P, t \in T \tag{13}$$

$$\sum_{p \in P} o_{ipt} + \sum_{p \in P} a_{ipt} \leq C_i \qquad i \in N, t \in T \tag{14}$$

$$o_{ip1} = s_{ip} \qquad p \in P,\ i \in N \tag{15}$$

$$x_{ij}^{kp} \in \{0,1\} \qquad (i,j) \in A_k,\ k \in K,\ p \in P \tag{16}$$

$$z_k \in \{0,1\} \qquad k \in K \tag{17}$$

$$a_{ipt}, l_{ipt}, o_{ipt} \in \{0,1\} \qquad i \in N, p \in P, t \in T \tag{18}$$

$$s_{ip} \in \{0,1\} \qquad i \in S, p \in P \tag{19}$$

$$d_{ip} \in \{0,1\} \qquad i \in D_p, p \in P \tag{20}$$

$$y_{kp} \in \{0,1\} \qquad k \in K, p \in P \tag{21}$$

$$q_p \in \{0,1\} \qquad p \in P \tag{22}$$

$$e_i \in \{0,1\} \qquad i \in S \tag{23}$$

The objective function (1) minimizes the overall cost that comprises the remuneration for selected crowdshippers, the cost to send delivery vehicles to source stations, the cost for the loading of the parcels, and the backup cost for the parcels undelivered through crwodshipping. Constraints (2) and (3) ensure that, if the LSP does not deliver parcel $p \in P$ through the backup service (i.e., $q_p = 0$), the parcel must have exactly one source and one destination stations. Constraints (4) set variable e_i equal to 1 if station $i \in S$ is used as a source station by at least one parcel. Constraints (5) deal with three distinct cases:

- $s_{ip} - d_{ip} = 1$. In this case, station i is the source station for parcel p ($s_{ip} = 1$ and $d_{ip} = 0$). The constraints impose that parcel p must leave station i without entering it.
- $s_{ip} - d_{ip} = -1$. In this case, station i is the destination station for parcel p ($s_{ip} = 0$ and $d_{ip} = 1$). The constraints impose that parcel p must enter i without leaving it.
- $s_{ip} - d_{ip} = 0$. This configuration holds when $s_{ip} = d_{ip} = 0$, so station i is an exchange station for parcel p. In this case, the constraints impose flow conservation.

According to constraints (6), crowdshipper $k \in K$ can transfer parcel $p \in P$ from station i to j only if parcel p has been assigned to them ($y_{kp} = 1$). Constraints (7) guarantee that each crowdshipper $k \in K$ (if selected) can deliver at most one parcel. Constraints (8) set variable l_{ipt} to 1 if parcel p leaves station i at time t. Since variables l_{ipt} are binary, these constraints prohibit a parcel from leaving

a station more than once. Similarly, constraints (9) set variable a_{ipt} to 1 if parcel p arrives at station i at time t and exclude the parcel from entering station i more than once at time t. Constraints (10) force variable $o_{ip(t+1)}$ to 1 if variable $a_{ipt} = 1$, thus specifying that parcel p is located at station i at time $t+1$, if the parcel arrived at this station at time t. Constraints (11) force the location of parcel p at station i at time $t+1$ if the parcel has not been moved from station i at time t ($l_{ipt} = 0$) and it was already located there at time t ($o_{ipt} = 1$). Constraints (12) force variable $o_{ip(t+1)}$ to 0 if parcel p leaves station i at time t, otherwise the constraint is not binding. Notice that constraints (10) and (12) imply that binary variables a_{ipt} and l_{ipt} are mutually exclusive. If parcel p is managed by crowdshippers (i.e., $q_p = 0$), constraints (13) impose that it can be located in at most one station for each time slot. Otherwise, the LSP will take care of the delivery and the parcel will not travel through the network $\sum_{i \in N} o_{ipt} = 0$. Constraints (14) are capacity constraints, ensuring that the number of parcels located at station i plus the number of parcels arriving at the same station at time t has to be lower than or equal to the station capacity. Constraints (15) set the location of parcel p at time 1 at its source station i when $s_{ip} = 1$. Finally, constraints (16)–(23) are binary and non-negativity conditions on variables.

4.1. Valid inequalities

Our formulation (1)–(23) involves a large set of variables and constraints driven by the number of network stations, parcels, crowdshippers, and the selected time slots discretization. While complete and correct, we introduce the following valid inequality classes to strengthen the relationships among the variables, a common technique to improve the model's structure and optimization performance.

$$\sum_{p \in P} s_{ip} \geq e_i \qquad i \in N \tag{24}$$

$$\sum_{i \in D} d_{ip} \leq \sum_{k \in K} y_{kp} \qquad p \in P \tag{25}$$

$$\sum_{i \in S} s_{ip} \leq \sum_{k \in K} y_{kp} \qquad p \in P \tag{26}$$

$$\sum_{i \in N} \sum_{t \in T} l_{ipt} \leq \sum_{k \in K} y_{kp} \qquad p \in P \tag{27}$$

$$\sum_{i \in N} \sum_{t \in T} a_{ipt} \leq \sum_{k \in K} y_{kp} \qquad p \in P \tag{28}$$

$$y_{kp} \leq 1 - q_p \qquad p \in P, k \in K \tag{29}$$

$$s_{ip} + d_{ip} \leq 1 \qquad i \in N, p \in P \tag{30}$$

$$\sum_{i \in N} o_{ipt} \leq \sum_{i \in S} s_{ip} \qquad p \in P, t \in T \tag{31}$$

$$\sum_{i \in N} o_{ipt} \leq \sum_{i \in D} d_{ip} \qquad p \in P, t \in T. \tag{32}$$

Inequalities (24) impose that if the source station $i \in S$ is selected ($e_i = 1$), at least one parcel must start its path at station i. Inequalities (25) and (26) establish that neither a destination station nor a source station can be selected for parcel $p \in P$ if the parcel has not been assigned to a crowdshipper. Similarly, inequalities (27) and (28) allow a parcel p to leave (resp. arrive) at a station only if a crowdshipper handles this parcel. Each inequality (29) ensures that parcel $p \in P$ cannot be assigned to crowdshipper $k \in K$ if the parcel has already been allocated to the backup service (i.e., $q_p = 1$). Inequalities (30) state that, for a given parcel p, since a station i cannot be jointly its source and destination stations, variables s_{ip} and d_{ip} cannot be both equal to 1. Finally, valid inequalities (31) and (32) ensure that the location variables o_{ipt} for parcel $p \in P$ at time slot $t \in T$ can only be active at a station $i \in N$ if both a starting and a destination stations have been assigned to this parcel.

5. An ALNS algorithm for the PTCP

This section presents our solution method based on the Adaptive Large Neighborhood Search (ALNS) algorithm (Ropke and Pisinger, 2006), adapted to the specific challenges of PTCP. The flexibility of ALNS in handling diverse destroy and repair operators makes it well-suited to our problem, despite differences from (Ropke and Pisinger, 2006) (e.g., APL capacities, time-dependent crowdshipper availability). ALNS effectively explores large solution spaces, balancing intensification and diversification to address the trade-offs in PTCP.

The ALNS framework iteratively applies destroy and repair operators to refine solutions. Destroy operators remove parcel assignments, while repair operators reassign them more effectively. An adaptive scoring mechanism prioritizes effective operators. This process is embedded in an Iterated Simulated Annealing (ISA) framework, which regulates solution acceptance via a temperature-based criterion, aiding in escaping local optima while balancing exploration and exploitation. The stopping condition is the maximum computation time.

In the following, we introduce the concept of *parcel path* to denote the sequence of steps a parcel takes from its source station to its destination, with its length indicating the number of crowdshippers involved. Shorter paths are more cost-effective. Our ALNS prioritizes crowdshipper-based paths over backup services, when they incur higher costs due to additional personnel and vehicles.

This principle guides the development of our destroy and repair operators, which aim to minimize the use of backup services while efficiently using available crowdshippers.

Algorithm 1 outlines our ALNS within an ISA framework, initializing a baseline starting solution and assigning an initial score of 1 to each operator. The outer loop runs until the time limit is reached, initializing operator scores before executing the simulated annealing temperature schedule. In each iteration of the inner loop, a destroy and a repair operator are selected via roulette wheel selection (Line 6). The selected destroy operator removes a predetermined number of parcel assignments (Line 7), reducing node capacities to prevent regenerating the same solution (Line 8). The selected repair operator then reinserts as many parcels as possible while minimizing costs (Line 9). Finally, each node's capacity is reset to its original value. In Lines 11–23, the algorithm decides whether to accept the new solution. Lower-cost solutions are always accepted, while worse solutions are accepted probabilistically based on the current temperature. If the new solution improves the incumbent, the latter is updated and a reheating policy is applied. The new solution is compared to the current solution based on cost difference, and a temperature-dependent criterion is used to determine if the new solution should be accepted. The temperature is systematically reduced by a factor α (Line 24). Finally, in Lines 25–30, operator scores are reset to their initial value after each *Limit* iterations.

Algorithm 1 ALNS within ISA Framework

1: Initialize solution S
2: **while** stopping criteria not met **do**
3: operator scores $\omega_d, \omega_r \leftarrow 1$, temperature $T \leftarrow T_0$
4: iterations $\leftarrow 0$
5: **while** $T < T_{\min}$ **do**
6: Select destroy operator d and repair operator r via roulette wheel selection on ω_d, ω_r
7: $S' \leftarrow$ apply d to remove q parcels from S ▷ Destroy phase
8: Reduce node capacity according to the destroy operator ▷ Avoid to recreate the previous solution
9: $S' \leftarrow$ apply r to reinsert removed parcels into S' ▷ Repair phase
10: Reset node capacity
11: Compute cost difference $\Delta = \text{cost}(S') - \text{cost}(S)$
12: **if** $\Delta < 0$ or $U(0,1) < \exp(-\Delta/T)$ **then** ▷ Acceptance logic, with $U(0,1)$ Uniform random in [0, 1]
13: $S \leftarrow S'$
14: Update scores:
15: **if** S' is a new best solution **then**
16: $\omega_d \leftarrow \omega_d + \eta_{\text{best}}$, $\omega_r \leftarrow \omega_r + \eta_{\text{best}}$ ▷ Reward for new best solution
17: Reheat: $T \leftarrow T + \Delta t$ ▷ $\Delta t = (S' - S)\exp\left(-\frac{t}{T_0}\right)$
18: **else if** S' is better than current solution **then**
19: $\omega_d \leftarrow \omega_d + \eta_{\text{incumbent}}$, $\omega_r \leftarrow \omega_r + \eta_{\text{incumbent}}$ ▷ Reward for new incumbent
20: **else**
21: $\omega_d \leftarrow \omega_d + \eta_{\text{accepted}}$, $\omega_r \leftarrow \omega_r + \eta_{\text{accepted}}$ ▷ Reward for accepted solution
22: **end if**
23: **end if**
24: Cool temperature: $T \leftarrow \alpha T$ $(\alpha \in (0,1))$
25: **if** iterations = Limit **then** ▷ Reset the operator scores after a certain number of iterations
26: operator scores $\omega_d, \omega_r \leftarrow 1$,
27: iterations $\leftarrow 0$
28: **else**
29: iterations $\leftarrow$ iterations + 1
30: **end if**
31: **end while**
32: **end while**
33: **return** best solution found

During iterations, operator scores are updated based on solution quality (Lines 13–20):

- New best solution: Operators are rewarded by η_{best} when finding a new globally improved solution.
- New incumbent solution: Operators are increased by $\eta_{\text{incumbent}}$ when finding a solution better than the current one but not the best.
- Accepted solution: Operators are rewarded by η_{accepted} even if they do not provide any solution improvement.

In the remainder of this section, we describe the destroy and repair operators tailored to PTCP, highlighting their role in addressing the problem's specific characteristics. Then, we present the constructive initial heuristic to generate the starting solution, ensuring a high-quality baseline for ALNS.

5.1. Destroy heuristics

The destroy heuristics remove existing parcel routes and allocate them to a backup set, enabling solution space exploration. Let $H \in \mathbb{N}^{|N|\times|T|}$ be the load matrix where h_{it} denotes parcels at station i at time slot t. Each of the proposed destroy operators influence station prioritization through four metrics:

- Absolute Load: sort stations according to the value $a_i = \max_{t\in T} \ h_{it}$, targeting peak congestion;
- Relative Load: sort stations according to the value $r_i = a_i/C_i$, identifying peak utilization rate;
- Average Load: sort stations according to the value $m_i = \sum_{t\in T} h_{it}/|T|$, identifying sustained usage;
- Random: sort stations according to the value $u_i = U(0,1)$, enabling diversification.

The chosen metrics help identify stations under high stress or congestion. Specifically, the *Absolute Load* metric targets stations with peak demand, the *Relative Load* metric highlights stations operating near capacity, and the *Average Load* metric reflects sustained high usage over time. These metrics directly inform the destroy operators by prioritizing the removal of parcel paths from the most burdened stations. Destroy operators ((D1)–(D2)) apply these metrics to select stations for parcel removal.

(D1) Parcel Paths Elimination. This heuristic targets parcel paths that originate from a selected station, thereby encouraging the formation of completely new paths. The procedure is as follows:

(a) Station Selection: Sort all stations using one of the four metrics and select the top-ranked station.
(b) Path Sorting: Sort all parcel paths originating from the selected station in non-increasing length order.
(c) Determination of Paths to Remove: Let $\bar{q}$ denote the number of parcel paths from the selected station. Define $q^* = \min(\bar{q}, q)$, where q is the pre-specified number of paths to eliminate.
(d) Path Removal and Update: Remove the q^* longest parcel paths and assign them to a backup service. Update the load matrix H accordingly.

(D2) Capacity Reduction. This procedure reduces the effective capacity of a selected subset of stations, forcing the reassignment of parcel paths that no longer comply with the updated capacity constraints:

(a) Station Selection: Sort stations using one of the four metrics and select the top q stations.
(b) Capacity Adjustment: For each selected station i, update its capacity by setting $\tilde{C}_i = \max(a_i - \tilde{q}, 0)$, where a_i is the original capacity of station i and $\tilde{q}$ is the specified reduction amount.
(c) Path Sorting and Removal: For each selected station, sort all incident parcel paths in non-increasing length order. Iteratively remove the longest paths until the load at station i, for every time slot t, satisfies $h_{it} \le \tilde{C}_i$.

5.2. Repair heuristics

The repair heuristics aim to deliver backup set parcels through crowdshipping by constructing feasible delivery paths using a label-setting algorithm. For each undelivered parcel in the backup set, the method aims at (i) assigning a reasonable source station and (ii) computing a feasible path from that source station to the destination, considering the availability of crowdshippers. Two distinct cost functions are used in the labeling process: one that minimizes the number of crowdshippers employed, and another that penalizes overused crowdshippers to promote assignment diversification. Regardless of the specific variant, if a feasible path is found, the parcel is marked as delivered; otherwise, it remains in the backup set.

The proposed labeling algorithm is an extension of Dijkstra's algorithm, and associates with each station j a label given by the tuple (c_j, t_j, k, i), where c_j is the cumulative cost to reach station j; t_j is the arrival time at station j of crowdshipper k who traveled from station i; k identifies the crowdshipper; i is the predecessor station in the path. At the source station, the label is initialized as $(0, 0, \emptyset, \emptyset)$. Each station is assigned at most one label corresponding to the lowest cost (or best) path found so far. When processing a station i with label $(c_i, t_i, \cdot, \cdot)$, the algorithm considers each feasible move to a neighboring station j. For every crowdshipper k not yet employed in the current path and able to depart after time t_i, a new label is computed as $c_j = c_i + f_k(i,j) \quad \text{and} \quad t_j = t_i^k + t_{ij}$, where: t_i^k is the arrival time of crowdshipper k at station i; t_{ij} is the travel time from station i to station j, and $f_k(i,j)$ is a cost function associated with the arc (i,j) for crowdshipper k. A new label replaces the current one at station j if it is the first label or if it has a lower cost than the one previously recorded. The algorithm terminates when a label for a destination station is fixed.

We consider two alternative cost functions:

(i) $f_k(i,j) = 1$ for all $k \in K$ and $(i,j) \in A$, which minimizes the number of crowdshippers used;
(ii) A cost function that counts the number of times crowdshippers were employed in previous solutions. This second function encourages diversification.

In all cases, the cost of the final solution is evaluated according to Eq. (1).

The repair procedure is applied sequentially to all undelivered parcels that are in the backup set. The order in which parcels are processed follows one of the following strategies:

- *As-Is*: Process parcels in their original order.
- *Random*: Process parcels in a randomly shuffled order.
- *Max Backup*: Process parcels in descending order of their backup costs, thus prioritizing those that are most expensive to handle via the backup service.
- *Min Backup*: Process parcels in ascending order of their backup costs, thereby prioritizing parcels that require minimal backup adjustments and can yield early, incremental improvements.

These various processing orders allow the heuristic to target different aspects of solution quality, ranging from aggressively reducing high backup costs to quickly capturing low-hanging improvements, while the baseline (*As-Is*) and randomized orders help ensure diversification. Based on the above, four repair heuristic operators are proposed:

(R1) Parcel Distribution + Shortest Path

(a) Station Assignment: An undelivered parcel in the backup set is assigned to a source station using function $p_i(n)$ that represents the cost of loading $n+1$ parcels at station i. A priority queue is initialized with all stations that are not full, sorted by $p_i(n)$ value. For each parcel, the station with the lowest loading cost is selected; if the station is not yet full after the assignment, it is reinserted into the queue. If no station is left, the process stops.
(b) Path Computation: Using the label-setting algorithm, a path is computed for each parcel from the assigned source station to its destination to minimize the number of crowdshippers employed.

(R2) Delivery Cost Minimization + Shortest Path

(a) Station Assignment: stations are first sorted in ascending order of their base loading cost $p_i(0)$. Undelivered parcels are then assigned to source stations by matching each parcel with the nearest feasible destination from the current station until the station reaches its capacity. This process is repeated with the next station.
(b) Path Computation: The labeling algorithm is used to compute each parcel's shortest path (in terms of the number of crowdshippers).

(R3) Parcel Distribution + Uniform Crowdshipper Load

(a) Station Assignment: As in variant R1.
(b) Path Computation with Load Balancing: Each arc cost in the labeling algorithm is set to the number of times the corresponding crowdshipper has been used in previous solutions. This penalizes overused crowdshippers, thus favoring paths that involve less frequently employed resources.

(R4) Delivery Cost Minimization + Uniform Crowdshipper Load

(a) Station Assignment: As in variant R2.
(b) Path Computation with Load Balancing: As in variant R3.

5.3. Constructive initial heuristic

The Constructive Initial Heuristic (CIH) generates a starting solution used as a baseline in the ALNS. CIH sequentially improves the solution starting from a full backup set containing all the parcels to be delivered. In more detail, CIH:

(1) Uses Repair Heuristic R1 (Parcel Distribution + Shortest Path) for parcel assignment.
(2) Processes parcels in the **As-Is** order.
(3) Employs a uniform arc cost $f_k(i,j) = 1$ to minimize the number of crowdshippers.

All parcels are initially placed in the backup set, and CIH iteratively assigns them to the source stations using the introduced labeling algorithm to compute feasible delivery parcel paths. If no feasible path is found for a parcel, it stays in the backup set. If a feasible path is found, the parcel is removed from the backup set, which ensures that the obtained starting solution is at least as good as the one involving all the parcels in the backup set.

6. Experimental results

In this section, we first introduce the procedure used to generate the synthetic instances, followed by a description of the experimental setup. Next, we discuss the results obtained from the introduced strengthened MILP, as well as the performance of the proposed ALNS algorithm. Finally, we present and analyze the sensitivity of some key problem parameters.

6.1. Instance generation and experimental setup

We generate synthetic instances based on three key components: the transportation network, the crowdshippers, and the parcel demand. We randomly create different PT networks depending on the number of lines and the total number of stations. Each transport line is generated as a random curve on the plane, ensuring that it intersects with at least one other curve and contains no self-intersections. These intersections define exchange stations between lines, while the remaining stations are randomly distributed along the available lines. The travel time between consecutive stations is an integer value in minutes distributed in the range $[5, 15]$. We generated four PT networks with the following configurations in terms of number of lines and stations: (4, 24), (6, 36), (8, 40), (8, 44). The first network contains 3 exchange stations, the second contains 6, and the last two contain 9 each. We created six instance classes, each defined by a given number of crowdshippers and parcels. Each class includes four families, with six instances per family all sharing the same transportation network. In all instances, the number of time slots is set to 85, with each time slot lasting five minutes.

Each station is associated with two probabilities: the likelihood of being selected as a starting point by a crowdshipper and the likelihood of being chosen as a destination station by a customer. These probabilities are higher for traffic-intensive stations, such as terminal or exchange stations. A random variable with a realistic time pattern distribution determines when crowdshippers arrive at their starting stations. A random point on the plane is chosen as the depot for the LSP, and the distance between the depot and each station is used to compute the backup cost, which is regularized to be an integer between 12 and 20. Each instance is solved three times using the CPLEX solver: first on the base model without the valid inequalities, then with the valid inequalities separated dynamically, and finally on the model strengthened with valid inequalities added from scratch. The solver is configured with a 1-hour time limit, with no restrictions on the maximum number of threads or memory available. Each instance is also solved using our ALNS algorithm with different configuration parameters. These parameters cover possible combinations of the cooling rate value, set to $[0.75, 0.8, 0.85, 0.9]$, with the time limit threshold, set to 1, 5, and 10 min, respectively.

The destroy heuristic (D1) is configured with $q = [10, 15, 20]$, whereas for the destroy heuristic (D2) both q and $\tilde{q}$ can take value between 1 and 4. To ensure the robustness of the results, each instance-configuration combination is solved five times. The reported results always present the family-wise average. All experiments were executed on a computer running Debian 12.1, with an AMD Ryzen 5 5600U CPU, and 40 Gb of RAM. The ALNS algorithm is implemented in Rust 1.70, the MILP model is implemented in Java 20, and solved using CPLEX 22.1.

6.2. MILP formulation results: Base CPLEX versus implemented branch-and-cut

This section presents the results of our computational experiments when solving the proposed mathematical formulation. We begin by discussing the outcomes of running the solver without incorporating our valid inequalities (Base CPLEX). Subsequently, we present the results obtained when the solver is enriched with our valid inequalities. Finally, we provide a comparative analysis between the two methods. Henceforth, we will refer to a solution that makes use of the backup service to deliver parcels as a *default solution*. It is important to note that finding a default solution is always straightforward, as it involves simply summing the lowest backup costs for each parcel.

Table 5 presents a summary of the results obtained from running CPLEX without incorporating our valid inequalities. The table is composed of 11 columns representing the following information. The first three columns identify the instance family, specifying the number of stations ($|\boldsymbol{N}|$), crowdshippers ($|\boldsymbol{K}|$), and parcels ($|\boldsymbol{P}|$) in the instance family, respectively. The fourth column (**Gap**) shows the family average MIP gap (between the incumbent integer solution and the best bound) provided by CPLEX. The fifth column (**Opt.**) shows the number of family instances (out of 6) solved to optimality. The sixth column (**Nodes**) shows the average number of nodes explored in the Branch-and-Bound tree. The seventh column (**Obj.**) shows the average objective function value. The column (**Bound**) shows the average bound value. The ninth column (**Time [s]**) indicates the execution time, in seconds, whereas the tenth column (**TTB [s]**) displays the average time to best, in seconds. Finally, the last column (**Def.**) shows the number of instances in the family for which the solver reaches the time limit without improving the default solution.

By examining Table 5, we observe that only 7 out of 144 instances were solved to optimality within the one hour time limit. Even the family with the smallest average solution time, namely $(44, 75, 50)$, still takes more than 37 min on average to find a solution. Looking at the **Nodes** column, we observe that in 17 out of 24 families, the solver is unable to explore the solution tree beyond the root node. Many instance families display very small time-to-best (TTB) values, often below one second. Such findings can be explained by the fact that the solver only finds the default solution. However, in several families, the solver managed to find better solutions than the default one, resulting in more reasonable TTB values. Finally, the MIP gap values are generally high, with an average value of 30.33%. The minimum gap observed is 1.28%, while the maximum reaches 100% (for family $(36, 300, 250)$, the solver is unable to find a feasible solution for any instance). The solver struggles to find improving solutions for numerous large instances, and in several cases, CPLEX could not even find the optimal solution of the linear relaxation. Although the solver performs better with smaller instances, computational times are close to or reach the time limit, even for many small instances. These results suggest that CPLEX alone is not sufficient for handling large-scale instances, therefore heuristic methods should be considered to improve the solution process.

Table 6 summarizes the corresponding results obtained by our implementation of the Branch-and-Cut obtained from running the solver CPLEX with our valid inequalities separated dynamically. The implemented Branch-and-Cut procedure works as follows. At each branch-and-bound node, once the first linear relaxation solution is found, it is inspected for violations of the valid inequalities (24)–(32). Any violated inequalities are added to the model, but only if their inclusion leads to an improvement in the objective

Table 5
Base CPLEX: solution statistics.

$\|N\|$	$\|K\|$	$\|P\|$	Gap	Opt.	Nodes	Obj.	Bound	Time [s]	TTB [s]	Def.
24	75	50	11.01%	1	437.17	613.17	546.28	3029.36	1311.72	0
36	75	50	8.20%	0	1019.67	654.3	600.66	3600.17	1338.78	0
40	75	50	1.28%	3	2455.33	671.67	663.08	2648.68	124.25	0
44	75	50	1.42%	3	2949.83	692.83	683.29	2000.27	181.11	0
24	100	75	14.51%	0	1.17	951.17	814.54	3600.15	2037.67	0
36	100	75	30.53%	0	0.00	1063.33	730.80	3600.19	1138.09	2
40	100	75	7.50%	0	77.67	999.17	924.50	3600.29	1609.20	0
44	100	75	3.93%	0	178.67	1059.33	1018.00	3600.45	1449.46	0
24	150	120	35.11%	0	0.00	1809.83	1173.93	3600.25	46.66	5
36	150	120	36.30%	0	0.00	1862.00	1185.85	3600.38	0.09	6
40	150	120	16.94%	0	0.00	1833.83	1524.86	3600.33	66.89	5
44	150	120	10.99%	0	0.00	1859.67	1653.38	3600.41	532.68	3
24	200	160	38.00%	0	0.00	2501.67	1552.05	3600.45	120.83	5
36	200	160	33.78%	0	0.00	2363.33	1564.88	3600.63	0.13	6
40	200	160	17.72%	0	0.00	2495.67	2052.65	3600.56	164.28	5
44	200	160	16.75%	0	0.00	2487.50	2070.05	3600.60	205.58	5
24	250	225	30.48%	0	0.00	3348.33	2327.01	3600.72	0.13	6
36	250	225	63.00%	0	0.00	3536.50	1293.66	3601.43	0.19	6
40	250	225	16.30%	0	0.00	3437.00	2876.30	3600.83	0.21	6
44	250	225	27.87%	0	0.00	3552.33	2572.97	3601.07	0.21	6
24	300	250	34.61%	0	0.00	3853.00	2518.59	3600.85	0.16	6
36	300	250	100.00%	0	0.00	3740.67	0.00	3602.82	0.24	6
40	300	250	85.53%	0	0.00	3782.33	538.19	3602.15	565.39	5
44	300	250	86.21%	0	0.00	3861.00	530.60	3602.20	0.26	6
Average			30.33%	0.29	340.02	2209.62	1309.06	3482.07	453.82	3.71

Table 6
Implemented Branch-and-Cut: solution statistics.

$\|N\|$	$\|K\|$	$\|P\|$	Gap	Opt.	Nodes	Obj.	Bound	Time [s]	Impr.	TTB [s]	Def.
24	75	50	6.65%	0	877.17	610.83	570.27	3621.28	4.39%	818.85	0
36	75	50	3.86%	0	835.5	649.7	623.92	3626.62	0.70%	311.80	0
40	75	50	0.76%	4	249.00	671.67	666.48	2008.88	0.51%	208.67	0
44	75	50	0.57%	4	242.17	692.83	689.06	1552.07	0.00%	604.80	0
24	100	75	8.09%	0	81.50	936.50	861.71	3619.18	5.79%	1872.32	0
36	100	75	15.58%	0	3.83	944.50	787.18	3619.48	7.71%	2114.86	0
40	100	75	4.53%	0	10.83	1000.17	954.93	3624.80	3.29%	1956.47	0
44	100	75	1.92%	0	31.83	1054.17	1033.95	3624.16	1.57%	1374.48	0
24	150	120	34.00%	0	0.00	1809.83	1194.37	3600.27	1.74%	45.29	5
36	150	120	36.00%	0	0.00	1862.00	1191.43	3600.38	0.47%	0.09	6
40	150	120	16.80%	0	0.00	1833.83	1527.58	3600.36	0.18%	66.44	5
44	150	120	10.53%	0	0.00	1849.83	1652.97	3600.43	−0.02%	1201.96	2
24	200	160	37.85%	0	0.00	2501.67	1555.80	3600.47	0.24%	122.44	5
36	200	160	33.38%	0	0.00	2363.33	1574.24	3600.69	0.60%	0.13	6
40	200	160	17.34%	0	0.00	2495.67	2062.36	3600.63	0.47%	185.06	5
44	200	160	16.49%	0	0.00	2487.50	2076.63	3600.63	0.32%	219.17	5
24	250	225	30.17%	0	0.00	3348.33	2337.39	3600.72	0.45%	0.14	6
36	250	225	63.00%	0	0.00	3536.50	1293.66	3653.59	0.00%	0.20	6
40	250	225	16.29%	0	0.00	3437.00	2876.77	3949.77	0.02%	0.20	6
44	250	225	27.75%	0	0.00	3552.33	2576.98	3823.56	0.16%	0.21	6
24	300	250	34.61%	0	0.00	3853.00	2518.59	3737.91	0.00%	0.17	6
36	300	250	100.00%	0	0.00	3740.67	0.00	3603.02	N.A.	0.26	6
40	300	250	85.53%	0	0.00	3782.33	538.19	3602.21	0.00%	580.21	5
44	300	250	86.21%	0	0.00	3861.00	530.60	3602.22	0.00%	0.26	6
Average			28.66%	0.33	94.81	2203.20	1320.69	3503.52	1.28%	474.90	3.58

value compared to the previous step. The current node is iteratively reoptimized, with the process repeating for each new linear relaxation. If the solution remains unchanged from the prior iteration, no cuts are added. Table 6 consists of 12 columns following the structure of Table 5 with the addition of column "**Impr.**" showing the average percentage improvement in the objective value compared to the initial one found by Base CPLEX. This column provides various insightful observations. Primarily, our cuts are more effective with smaller instances, providing only minimal enhancements for larger instances. Furthermore, as shown in Table 7, no valid inequalities are added in the final three instance families, as the solver is unable to compute the linear relaxation within the time limit. In one instance with 44 nodes, 150 crowdshippers, and 120 parcels, the introduction of valid inequalities resulted in a slight deterioration of the objective bound. This is presumably due to the substantial number of constraints added, resulting in a slower optimization process and hence worsening the results. For class $|N| = 36$, $|K| = 300$, and $|P| = 250$, CPLEX never solved the

Table 7
Statistics on generated number of cuts.

$\|N\|$	$\|K\|$	$\|P\|$	(24)	(25)–(26)	(27)–(28)	(29)	(30)	(31)–(32)
24	75	50	28.00	6171.67	11224.50	10035.33	40.50	95830.83
36	75	50	117.83	3146.50	4345.83	13789.16	96.00	43205.50
40	75	50	7.17	572.50	873.67	2001.17	12.17	15703.50
44	75	50	1.00	1553.50	1930.00	4235.33	15.17	19614.67
24	100	75	0.00	1393.83	2578.50	1747.17	6.50	14601.83
36	100	75	0.50	183.17	321.83	91.50	0.50	1252.67
40	100	75	6.50	226.50	331.83	513.33	2.17	2548.83
44	100	75	6.17	431.33	564.67	815.83	1.33	4024.50
24	150	120	0.00	129.83	145.00	18.83	0.00	252.50
36	150	120	0.00	98.83	134.33	87.00	0.00	192.17
40	150	120	0.00	58.17	68.17	161.17	0.00	153.67
44	150	120	0.17	63.33	74.33	48.33	0.00	140.67
24	200	160	0.00	152.33	184.00	18.17	0.00	152.67
36	200	160	0.00	153.33	192.00	121.83	0.00	293.17
40	200	160	0.00	76.50	95.00	190.33	0.00	303.83
44	200	160	0.00	76.83	109.33	525.50	0.00	727.50
24	250	225	0.00	206.67	225.50	204.50	0.00	338.67
36	250	225	0.00	26.33	41.00	0.83	0.00	21.33
40	250	225	0.00	121.83	135.17	462.50	0.00	458.00
44	250	225	0.00	81.17	91.83	1074.17	0.00	982.67
24	300	250	0.00	195.33	227.00	60.83	0.00	111.00
36	300	250	0.00	0.00	0.00	0.00	0.00	0.00
40	300	250	0.00	0.00	0.00	0.00	0.00	0.00
44	300	250	0.00	0.00	0.00	0.00	0.00	0.00

linear relaxation, hence it is impossible to compute the improvement from the Base CPLEX. For this reason, the corresponding cell contains the placeholder N.A. (No. Available), and the average presented at the bottom of the table is computed without considering this value. The use of valid inequalities allows to solve an additional instance to optimality with respect to Base CPLEX. We observe that, on average, the valid inequalities increase the average TTB by 4.65%. On the other hand, the valid inequalities improve the MIP gap by 1.67%. It suggests that the introduction of valid inequalities forces CPLEX to spend more time on the Branch-and-Cut algorithm, thereby reducing the time available for the internal heuristics, which results in a generally worse TTB. However, the new information given to the Branch-and-Cut allows the solver to find better linear relaxation to the problem, proving the effectiveness of our valid inequalities. With respect to Base CPLEX, we can observe that it has been possible to find solutions that are not the default one for a larger number of instances: 86 against 89. An interesting result to discuss is the average number of nodes explored by the solver for small size classes when compared to Base CPLEX. With a few exceptions, this number is smaller. This can be attributed to the fact that the added cuts enabled the solver to close subproblems without necessitating branching.

In Table 7, we summarize the statistics on the number of generated cuts. The table consists of nine columns, where the first three identify the instance size, the fourth column displays the average number of added cuts for inequalities (24). Similarly, from the fifth to the ninth column, we provide the average numbers of added cuts for valid inequalities (25) and (26) jointly applied for each instance, valid inequalities (27) and (28), (29), (30), and (31) and (32), respectively.

It is interesting to note that the smaller the size of the instance, the larger the number of added cuts. This can be explained by the fact that, given the time limit, in small instances, the solver explored a large number of Branch-and-Cut nodes. On the other hand, no cuts were generated for the last three families where the solver cannot even find the optimal solution of the continuous relaxation within the time limit. By examining the columns of Table 7, we observe that cuts like (31) and (32) were generally applied the most. Cuts (27) and (28) were also applied frequently. This is likely because these cuts operate on larger variable sets, such as the location variables o_{ipt} and the movement variables a_{ipt} and l_{ipt}. The least used cuts are (24) and (30).

Table 8 presents the results obtained when valid inequalities (24)–(32) are incorporated directly into the model, rather than being added dynamically during the solution process. These results are compared with the two previously considered approaches: solving the model using CPLEX without valid inequalities (Base CPLEX) and with a Branch-and-Cut algorithm that dynamically separates these inequalities. Column **Opt.** reports the number of instances solved to optimality within the time limit, totaling 14. In particular, all instances previously solved optimally by both Base CPLEX and Branch-and-Cut algorithm with dynamically separated inequalities were also solved by the current version that incorporates inequalities from scratch. For the configuration with $|N| = 40$, $|K| = 75$ and $|P| = 50$, incorporating valid inequalities directly into the model proves more effective in finding the optimal solution in all instances. On average, the incumbent solution improves by 5.80% (column **Impr.**) compared to Base CPLEX, significantly outperforming the 1.28% improvement achieved with dynamically separated inequalities. The improvement is most significant for medium-sized instances ($|K| = 100$ and 150) with a peak at 23.88%, while for smaller instances ($|K| = 75$), it is negligible or absent. This occurs because the solver finds the same incumbent solution, but struggles to prove optimality without valid inequalities. Regarding MIP gaps (column **Gap.**), two key insights emerge: (1) the average gap is lower than both Base CPLEX and Branch-and-Cut with dynamically added inequalities, and (2) no instances reach the worst-case 100% gap, with even the largest instances ($|K| = 300$) successfully solving their linear relaxations. In terms of computing times, the time to the best solution (column **TTB [s].**) is generally higher than for the other two approaches. However, this must be interpreted in the context of the number of instances solved. Since

Table 8
Valid inequalities incorporated into the model from scratch: solution statistics.

$\|N\|$	$\|K\|$	$\|P\|$	Gap	Opt.	Nodes	Obj.	Bound	Time [s]	Impr.	TTB [s]	Def.
24	75	50	6.51%	2	3348.17	610.67	571.87	3023.85	0.37%	420.53	0
36	75	50	5.58%	0	3363.67	571.33	538.06	3600.22	1.28%	635.36	0
40	75	50	0.00%	6	1118.50	671.67	671.67	731.54	0.00%	98.87	0
44	75	50	0.17%	5	2186.83	692.83	691.64	1000.00	0.00%	144.39	0
24	100	75	8.21%	0	601.33	937.17	860.88	3600.38	1.47%	329.14	0
36	100	75	7.03%	0	251.50	871.50	809.49	3600.61	17.45%	1886.17	0
40	100	75	3.45%	0	1899.83	993.83	959.24	3600.47	0.54%	894.71	0
44	100	75	1.51%	1	1054.17	1054.00	1038.21	3538.96	0.52%	586.34	0
24	150	120	15.44%	0	7.33	1521.50	1274.84	3600.61	16.02%	1278.46	1
36	150	120	8.36%	0	1.17	1418.50	1297.43	3601.13	23.88%	2077.48	0
40	150	120	3.54%	0	243.17	1633.00	1576.18	3601.30	11.03%	2091.54	0
44	150	120	1.83%	0	539.83	1730.67	1697.84	3601.23	6.88%	1084.99	0
24	200	160	21.39%	0	0.00	2154.83	1672.92	3600.51	13.92%	1607.61	2
36	200	160	19.51%	0	0.00	2116.33	1669.03	3600.72	10.49%	1402.88	3
40	200	160	5.20%	0	0.00	2248.83	2131.94	3600.86	9.86%	2415.93	0
44	200	160	4.33%	0	0.00	2241.67	2142.71	3601.31	9.86%	2869.15	0
24	250	225	23.77%	0	0.00	3230.00	2446.32	3601.06	3.58%	514.53	5
36	250	225	88.25%	0	0.00	3536.50	435.27	3601.92	0.00%	0.21	6
40	250	225	26.45%	0	0.00	3334.33	2441.27	3602.15	2.94%	1078.69	4
44	250	225	34.96%	0	0.00	3310.00	2057.00	3601.49	6.88%	1783.58	2
24	300	250	30.90%	0	0.00	3853.00	2661.80	3601.13	0.00%	0.18	6
36	300	250	88.77%	0	0.00	3740.67	423.81	3603.31	0.00%	0.25	6
40	300	250	85.57%	0	0.00	3782.50	570.62	3602.81	0.00%	0.26	6
44	300	250	69.56%	0	0.00	3769.83	1065.55	3602.98	2.37%	599.50	5
Average			23.35%	0.58	606.01	2084.44	1321.12	3340.22	5.80%	990.91	1.92

Table 9
CPLEX and ALNS results for the instances solved to optimality.

$\|N\|$	$\|K\|$	$\|P\|$	ID	CPLEX Obj.	ALNS Obj.	ALNS Gap	CPLEX time [s]	ALNS time [s]
24	75	50	3	638	638	0.00	377	9.0
24	75	50	6	629	632	0.41	3364	269.0
40	75	50	1	750	750	0.00	30	0.0
40	75	50	2	617	618	0.16	93	20.0
40	75	50	3	694	695	0.17	3178	81.0
40	75	50	4	647	647	0.00	94	233.0
40	75	50	5	708	712	0.56	73	1.0
40	75	50	6	614	614	0.00	918	57.0
44	75	50	2	716	716	0.00	48	1.0
44	75	50	3	715	715	0.00	72	1.0
44	75	50	4	646	648	0.31	1248	5.0
44	75	50	5	684	685	0.15	58	0.0
44	75	50	6	718	718	0.00	972	0.0
44	100	75	2	1143	1143	0.00	3231	3.0

fewer instances are solved to their default value, the solver must invest more effort in refining better incumbent solutions, which naturally takes more time. This is confirmed by the overall solution time (column **Time [s].**), which is lower than for both the other two approaches, demonstrating that incorporating valid inequalities directly into the model improves efficiency.

6.3. ALNS results

In this section, we analyze the performance of our ALNS algorithm. Table 9 shows the comparison between the optimal solution values found by CPLEX with the valid inequalities inserted from scratch, and the best value returned by our ALNS algorithm. The table includes only the fourteen instances that were solved to optimality. All these results come from small instances: thirteen from instances with $|K| = 75$ crowdshippers and $|P| = 50$ parcels to deliver, and only one from an instance with $|K| = 100$ crowdshippers and $|P| = 75$ parcels to deliver. The first column indicates the number $|N|$ of stations in the network, the second column represents the number $|K|$ of crowdshippers, the third column the number $|P|$ of parcels, while the fourth column represents the instance id. The fifth and sixth columns show the CPLEX and ALNS objective values, respectively. The last two columns display the execution times (in seconds) for CPLEX and ALNS, respectively. By observing the **ALNS Gap** column, we see that our ALNS algorithm found the optimal value in seven instances. For the instances where our ALNS did not find the optimal value, in six cases, the gap between the optimal value and the heuristic solution is below 1%. From this initial analysis, we can see that our ALNS algorithm can find near-optimal solutions in the vast majority of cases, and in a fraction of the time required by a state-of-the-art solver.

Table 10 summarizes the results obtained from employing the ALNS algorithm with the best configuration for each instance family. Our tests reveal that, for each family, the configuration achieving the best results consistently uses a time limit of 600 s and

Table 10
ALNS results.

$\|N\|$	$\|K\|$	$\|P\|$	TTB [s]	ITB	Obj.	N. Sols	Δ Bound	Δ Obj.
24	75	50	206.35	14948.47	613.63	15.97	7.60%	0.46%
36	75	50	11.89	353.56	684.83	8.33	9.76%	5.41%
40	75	50	65.80	7153.37	672.70	11.00	0.93%	0.15%
44	75	50	19.03	1754.80	693.43	9.83	1.48%	0.86%
24	100	75	236.20	7523.03	940.90	16.77	9.19%	0.47%
36	100	75	55.32	827.07	933.37	13.37	18.57%	−1.18%
40	100	75	150.98	7433.87	996.10	13.13	4.31%	−0.41%
44	100	75	111.30	10530.87	1054.47	12.70	1.98%	0.03%
24	150	120	307.69	4078.70	1450.83	21.17	21.47%	−19.84%
36	150	120	219.16	1159.50	1482.97	18.60	24.47%	−20.36%
40	150	120	242.29	6007.00	1636.03	15.77	7.10%	−10.79%
44	150	120	196.41	9468.60	1725.00	16.47	4.36%	−6.75%
24	200	160	271.20	1826.20	1988.03	19.83	27.78%	−20.53%
36	200	160	287.85	838.40	1871.80	22.03	18.90%	−20.80%
40	200	160	259.53	3335.20	2240.30	17.53	8.63%	−10.23%
44	200	160	262.97	7454.17	2208.77	19.83	6.36%	−11.21%
24	250	225	312.18	1314.33	2789.67	22.00	19.35%	−16.68%
36	250	225	283.21	327.80	2912.67	20.20	125.15%	−17.64%
40	250	225	240.26	1704.13	3112.23	17.87	8.19%	−9.45%
44	250	225	267.37	4537.57	3219.60	22.20	24.94%	−9.37%
24	300	250	328.80	912.77	3130.80	25.07	24.31%	−18.74%
36	300	250	318.09	297.00	2895.57	25.90	N.A.	−22.59%
40	300	250	271.92	1154.87	3417.20	20.00	534.94%	−9.65%
44	300	250	275.87	3686.90	3521.30	19.37	563.64%	−8.80%
Average			216.74	4109.47	1925.01	17.68	64.07%	−9.48%

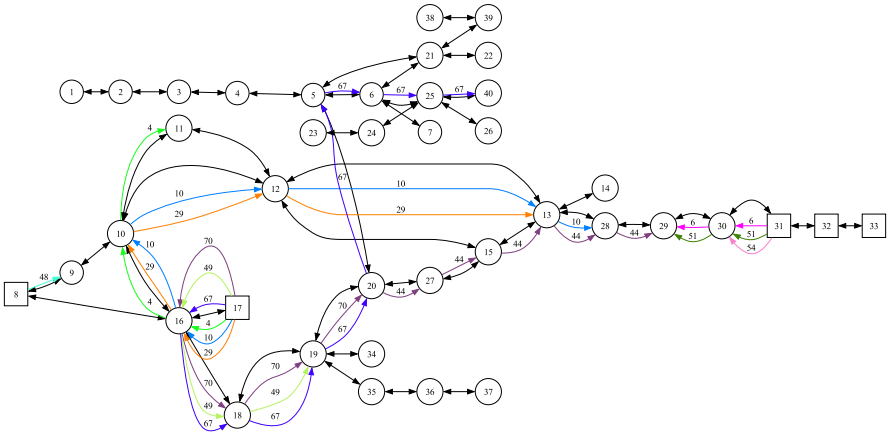
Table 11
Correlation matrix.

	$\|N\|$	$\|K\|$	TTB [s]	ITB	N. Sols
$\|N\|$	–	0.00	−0.35	0.06	−0.34
$\|K\|$	0.00	–	0.79	−0.50	0.84
TTB [s]	−0.35	0.79	–	−0.17	0.93
ITB	0.06	−0.50	−0.17	–	−0.32
N. Sols	−0.34	0.84	0.93	−0.32	–

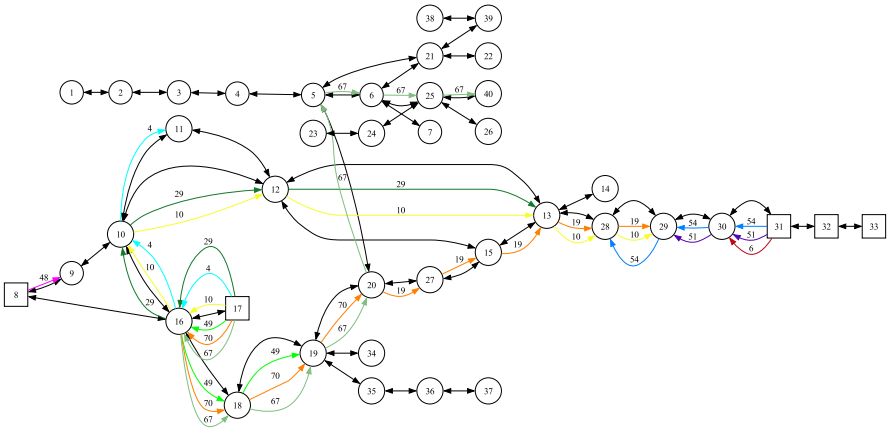
a cooling rate fixed to 0.9. The table consists of nine columns. The first three columns identify the instance family, specifying the number of nodes ($|\boldsymbol{N}|$), crowdshippers ($|\boldsymbol{K}|$) and parcels ($|\boldsymbol{P}|$), respectively. The following two columns (**TTB [s]** and **ITB**) show the average time to the best solution, in seconds, and the average number of ISA iterations to the best solution, respectively. The next column (**Obj.**) shows the average objective function value. The last three columns (**N. Sols**, Δ **Bound**, and Δ **Obj.**) display respectively the average number of improving solutions found during the search, the average difference in percentage between the best solution from ALNS and the best known lower bound from CPLEX, and the average difference in percentage between the ALNS solution value and the CPLEX's one.

An important aspect to highlight concerns the time to the best solution (**TTB [s]**). It is worth noting that the average TTB is consistently below the five-minute mark (300 s), and even below one minute for some instances. Nevertheless, the best configurations are always the ones with a 10 min time limit. Likely, a longer time limit allowed the algorithm to find better solutions in later phases of the optimization without sacrificing lucky runs, in which the incumbent was found in the initial phases. We observe in the Δ **Bound** column that, in general, small gaps from the objective bound exist when the MIP gap itself is small. This indicates that our heuristic approach is able to find solutions that are close to optimal, especially for small instances. For the instance family $|N| = 36$, $|K| = 300$, and $|P| = 250$, the placeholder N.A. (No. Available) is due to the absence of any linear relaxation coming from CPLEX, and the average presented at the bottom of the table is computed without considering this value. Finally, we observe from the Δ **Obj.** column that CPLEX found better average solutions than our ALNS approach in only five instance families. Except for one case, all these gaps are equal to or below 1%. Additionally, there is a noticeable trend of the gap improving as the instance sizes increase.

In order to gain further valuable insights, we examine the correlation coefficients between the different columns presented in Table 10. The most significant insights from this analysis, which are independent of the instances themselves, are summarized in Table 11. We observe a strong positive correlation between the number of crowdshippers ($|\boldsymbol{K}|$) and the number of improved solutions found (**N. Sols**). This correlation suggests that our ALNS algorithm can effectively utilize a larger pool of crowdshippers to its advantage. The observed correlations between the time to the best solution (**TTB [s]**) and the number of stations ($|\boldsymbol{N}|$), as well as the correlation between **TTB [s]** and the number of crowdshippers ($|\boldsymbol{K}|$), are also of interest. Generally, a larger pool of



(a) ALNS Solution



(b) CPLEX Solution

Fig. 4. Visual comparison of the solutions obtained with (a) ALNS and (b) CPLEX for the same instance.

crowdshippers leads to a longer TTB, which is expected due to the increased number of solutions found. On the other hand, a larger set of stations results in shorter TTBs. To explain this further, it is important to consider the existing correlation between the number of stations ($|\boldsymbol{N}|$) and the number of improved solutions found (**N. Sols**). Both of these correlations are negative, indicating that in a larger network, the ALNS algorithm has more possible alternatives for transferring parcels, enabling the heuristic to find solutions more easily. Finally, the nearly negligible correlation between $|\boldsymbol{N}|$ and the average number of ISA iterations to the best solution (**ITB**) suggests that larger networks are not significantly more computationally demanding than smaller ones for our ALNS algorithm.

In Fig. 4, we compare the solution found by our ALNS (Fig. 4(a)) with the one found by CPLEX (Fig. 4(b)) on the same instance, namely the third instance of the family with $|N| = 44$, $|K| = 75$, $|P| = 50$. The solution value remains the same, but the paths followed by the parcels differ slightly. CPLEX consumed the time limit of 1 h with a final MIP gap equal to 0.58%, whereas our ALNS required the assigned 10 min. In each of the two graphs, the nodes represent the PT stations, with square labels indicating source stations. The double-headed arcs between the nodes represent direct connections between the two stations. Each arc represents a step in a parcel's path, with each step associated with the crowdshipper involved. Each parcel is associated with a different color, and the number on each colored arc indicates the crowdshipper responsible for the transfer. For clarity, each parcel's path is represented directly on the original PT network. Both in Fig. 4(a) and Fig. 4(b), we observe that some parcels have been delivered by multiple crowdshippers, even though most parcels are handled by only one. Furthermore, in this specific instance, several parcels in both solutions have completed only one step in the network, while others have followed much longer paths. This behavior arises from the remuneration structure of crowdshippers. Delivering a parcel with a single crowdshipper is always cheaper, and paths with too many crowdshippers can exceed the backup cost.

Table 12
Average objective value obtained from ALNS with various values for the cooling rate and the time limit.

		Cooling rate			
		0.75	0.80	0.85	0.90
TL	1	1929.33	1928.56	1927.73	1926.87
	5	1925.91	1925.40	1924.91	1923.99
	10	1924.81	1924.33	1923.82	1922.91

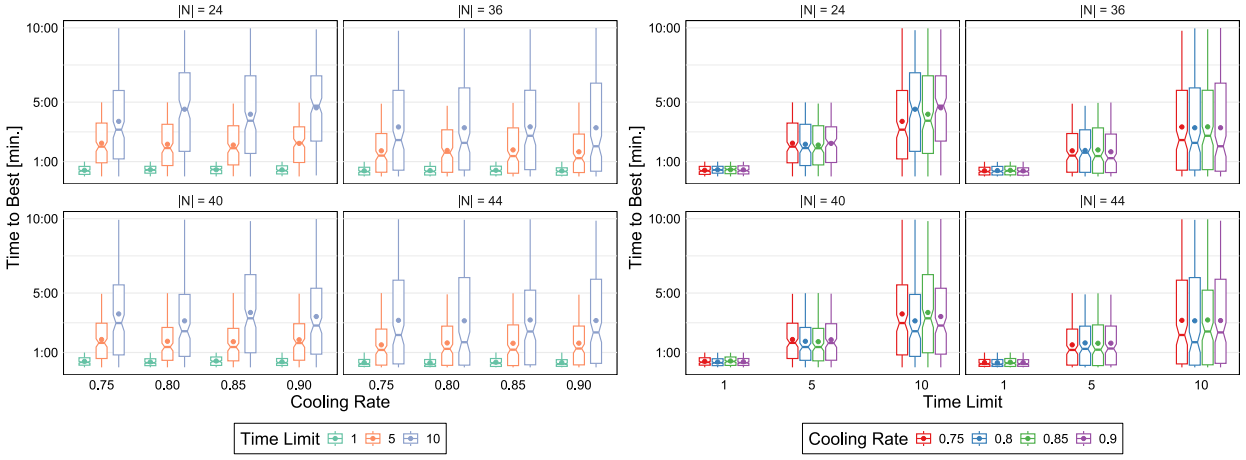
6.4. Tuning of ALNS parameters

This section presents a comprehensive analysis of the main ALNS parameters, namely the cooling rate and the time limit (**TL**), to evaluate the algorithm's robustness under different settings, providing guidance for decision makers. First, we analyze the impact of their values on the solution quality (effectiveness), then on the TTB (efficiency). Finally, we rank the various parameter values to summarize the findings and provide decision makers with a practical rule.

In our computational experiments, we run the ALNS algorithm with all possible combinations of cooling rates in $[0.75, 0.80, 0.85, 0.90]$ and time limits (in minutes) in $[1, 5, 10]$. Table 12 shows the average best objective function value found by our ALNS, computed over all instances without class distinction. In the table, the columns represent the outcomes for the different cooling rates, while the rows show the results for each time limit. As seen in Table 12, the average best solution value returned by ALNS appears to be independent of the values taken by the two parameters. Since the ALNS relies heavily on randomization, it is important to determine whether the differences in outcomes are due to chance alone or if the parameters play a pivotal role in the algorithm's performance. To investigate this behavior in depth, we conduced the Wilcoxon signed-rank paired test on the two parameters (for more information about the Wilcoxon test, see Tiit (2000)). The p-values from the tests comparing the cooling rate outcomes are all below 10^{-10}, indicating that, even if the differences are minimal, the null hypothesis can always be rejected, meaning the differences cannot be attributed to chance alone. Similarly, when comparing the outcomes based on time limits, the p-values are always below 10^{-10}. In this case as well, all tests allow us to reject the null hypothesis, indicating that even a minimal but significant difference exists between the outcomes for different time limit values. Considering the average heuristic results with varying cooling rates and time limits, we observe a limited impact on the solution value. However, Wilcoxon's test results indicate otherwise. Although unexpected, this outcome can be attributed to key aspects of our algorithm's design. First, in our ALNS, the ISA runs repeatedly until the time limit is reached. Unlike standard implementations that use a smooth cooling rate (close to 1) for gradual intensification, we employ multiple restarts within a single run. An aggressive cooling rate (close to 0) shortens each diversification-intensification cycle, increasing the number of runs. Conversely, a smoother cooling rate extends each run, reducing the number of iterations but allowing a deeper search per iteration, often yielding better solutions. Second, the reheating policy prevents premature convergence by increasing the temperature whenever a new best solution is found, effectively restarting diversification. This balances the effects of the cooling rate, ensuring a thorough exploration of the search space and explaining the consistent performance of the algorithm in different settings of parameters.

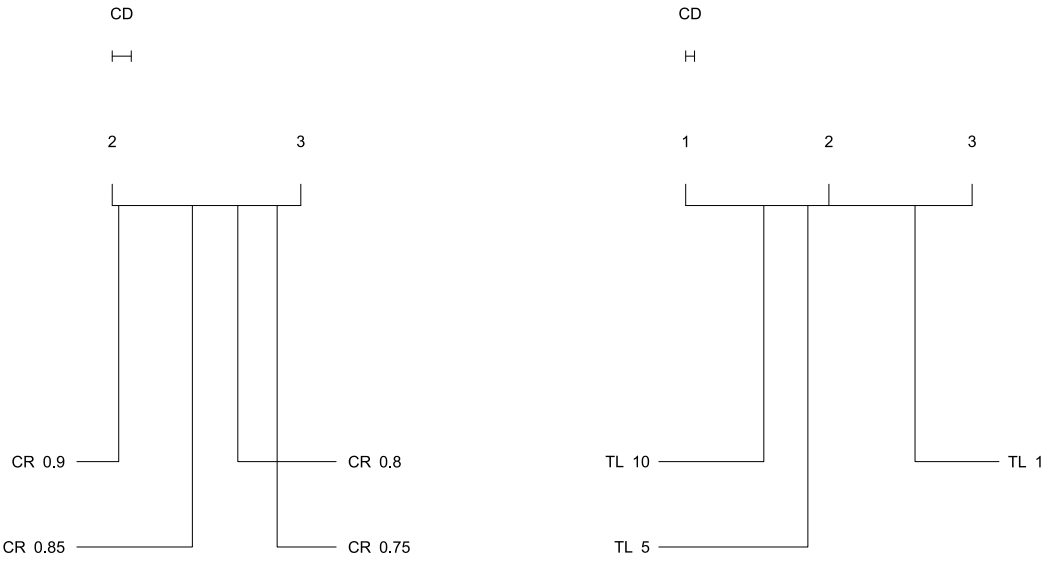
Now, let us focus our analysis on the impact of the cooling rate and time limit on the time to the best solution found (TTB). The goal of this analysis is to assess the impact of the configuration parameters on the efficiency, i.e., the amount of time our ALNS takes to find the best solution provided as output. Fig. 5(a) displays the TTB distribution across the different considered networks, grouped by time limit, under varying cooling rates. In this figure, the boxplots represent the quantiles of the TTB distributions, while the dots indicate the average TTB for each combination of cooling rate, time limit, and network. Unsurprisingly, a lower time limit corresponds to a lower TTB. In general, we observe that the 75^{th} percentile is much lower than the actual time limit. In fact, only in a few extreme cases has the best solution been found at or near the time limit. Fig. 5(b) shows the TTB distribution for each network, grouped by cooling rate, under varying time limits. Naturally, this figure presents the same information as Fig. 5(a) but from a different perspective, serving two purposes. First, it confirms previous observations, such as the fact that longer time limits correspond to larger TTBs. Second, it clearly shows that there is no trend, i.e., no positive or negative correlation between cooling rate and TTB. From the analysis of Fig. 5, it is clear that the time limit significantly impacts the time to the best solution found, demonstrating our algorithm's ability to continuously find new and improved solutions.

To conclude, we present two critical distance plots (see Demšar (2006)) to rank the different parameter values according to their overall performance. In these plots, a lower ranking value indicates better performance of a given configuration. The size of the Critical Distance (CD), computed using the Nemenyi test, determines when the test results can be considered significant (for more information on the Nemenyi test, see Tiit (2000)). Fig. 6(a) shows the CD ranking of the solution values under varying cooling rates. In this plot, the ranking is clear, with the distance between the scores consistently exceeding the critical distance. This image distinctly ranks a cooling rate of 0.9 as the best value among those tested. The second is 0.85, the third is 0.8, and the fourth is 0.75.



(a) Time-to-Best value for different ISA cooling rates (b) Time-to-Best value when changing the ALNS's time limit

Fig. 5. Comparison of Time-to-Best metrics: (a) by cooling rate, (b) by time limit.



(a) Critical Distance (CD) by Cooling Rate (CR) (b) Critical Distance (CD) by Time Limit (TL)

Fig. 6. Comparison of Critical Distance: (a) by Cooling Rate, (b) by Time Limit.

Even if differences are minimal, a smoother cooling rate enhances the overall performance of our ALNS. A similar result is shown in Fig. 6(b), where the CD plot compares the solution quality of the ALNS under varying time limits. In this figure, all the scores exceed the critical distance, indicating that the ranking is significant. The best performing time limit is 10 min, followed by 5 min in second place, and 1 min in third. This indicates that, in general, allowing the heuristic to run longer leads to better solutions. It is noteworthy that there is a significant gap between the one-minute and five-minute scores. However, the gap between the 5 min and 10 min scores is smaller. Increasing the time limit from 1 to 5 min multiplies the run time by five, while increasing it from 5 to 10 min only doubles the available time for the algorithm. Nonetheless, as seen in Table 10, the average TTB is below five minutes for each 10 min ALNS execution. Furthermore, Fig. 5(b) confirms this, showing that 75% of the executions find the incumbent well before the time limit.

Based on the performed CD ranking, a useful recommendation for decision-makers running this algorithm in an application is to configure the ALNS with a smooth cooling rate (≥ 0.85) and allow it to run for a relatively long time, more than 5 min.

Table 13
Sensitivity analysis – percentage of parcels delivered by crowdshippers across different cost configurations.

$(\|P\|,\|K\|)$	$\rho=1$			$\rho=2$			$\rho=3$		
	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$
(10,15)	2.50	26.66	30.00	0.00	15.55	28.33	0.00	10.00	26.67
(10,30)	8.75	56.11	60.43	5.83	42.22	62.77	0.00	20.55	48.23
(10,45)	14.54	70.00	81.81	7.22	53.33	82.77	0.00	41.66	63.89
(20,15)	0.94	14.72	18.00	0.83	14.16	17.77	0.00	7.78	15.83
(20,30)	7.50	32.77	33.33	2.22	25.00	33.88	1.38	20.55	28.88
(20,45)	24.25	50.27	59.44	18.30	43.05	51.38	3.61	32.22	39.79
(30,15)	1.67	11.66	13.51	3.15	10.37	12.40	1.11	9.07	11.85
(30,30)	4.63	21.37	22.74	8.33	16.27	21.76	0.92	16.66	18.82
(30,45)	19.44	40.55	43.51	12.96	35.92	40.74	3.89	32.04	36.85
Average	9.36	36.01	40.31	6.58	28.43	39.09	1.21	21.17	33.14

Table 14
Sensitivity analysis – objective function value across different cost configurations.

$(\|P\|,\|K\|)$	$\rho=1$			$\rho=2$			$\rho=3$		
	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$	$\gamma_p\in[4,8]$	$\gamma_p\in[6,12]$	$\gamma_p\in[8,16]$
(10,15)	57.13	86.88	109.87	61.72	87.27	110.11	59.83	92.00	111.94
(10,30)	57.25	76.94	98.00	59.58	82.11	95.17	58.28	86.88	101.94
(10,45)	57.09	71.00	87.00	58.72	79.11	92.39	57.83	84.38	96.27
(20,15)	111.62	171.27	220.40	120.10	177.88	226.11	120.66	174.44	224.94
(20,30)	111.16	162.72	209.91	116.30	173.22	212.22	118.61	173.05	217.72
(20,45)	107.75	147.72	179.44	117.50	155.55	191.38	121.33	164.88	204.92
(30,15)	170.88	257.66	336.88	178.30	262.61	339.33	176.38	266.22	350.44
(30,30)	174.77	245.94	320.17	178.00	254.35	327.35	180.61	258.55	334.52
(30,45)	162.96	229.05	293.14	178.89	240.44	300.56	181.34	256.15	306.17
Average	112.29	161.02	206.09	118.79	168.06	210.51	119.43	172.95	216.54

6.5. Sensitivity analysis of system efficiency

In this section, we perform a sensitivity analysis to evaluate how different values of the reward ρ granted to the crowdshippers (here referred to as *cs-cost*) and the backup cost γ_p affect the overall cost and efficiency of the delivery system. The cs-cost is the amount paid to the crowdshipper for delivering a parcel, while the backup cost is the expense incurred by the LSP when a parcel is delivered via the backup service. We assume that the cs-cost is always lower than the backup cost, making crowdshipping generally preferable from a cost perspective.

We consider nine sets of instances, each representing a different combination of cs-cost and backup cost ranges. The cs-cost varies between the values [1; 2; 3], while the backup cost ranges are [4, 8], [6,12], and [8, 16]. The backup cost for each parcel depends on its destination and is chosen from the corresponding range. We also vary the number of crowdshippers and parcels to be delivered for each set of instances. The number of crowdshippers takes the values [15; 30; 45], while the number of parcels varies between [10; 20; 30]. These are small instances to ensure that optimality can be reached and the obtained results are not biased. Each set of instances consists of nine subsets, each with a different combination of crowdshippers and parcels.

We can observe how these parameters influence the trade-off between cost and efficiency by varying the cs-cost and backup cost. Lower cs-cost and higher backup cost will encourage crowdshipping, as employing crowdshippers becomes more attractive than opting for the backup delivery alternative. The reverse is true when higher cs-cost and lower backup cost are imposed. We also expect that different public transportation lines will have varying levels of crowdshipping potential, depending on their network structure, frequency, and coverage.

Tables 13 and 14 summarize the results of our sensitivity analysis. The average values are given for each set of instances relative to the percentage of parcels delivered by crowdshippers (in Table 13) and the objective function value (in Table 14). Columns are grouped by cs-cost ($\boldsymbol{\rho} = 1, 2, 3$), with subcolumns representing backup cost ranges ($\gamma_p\in[4,8]$, $\gamma_p\in[6,12]$, $\gamma_p\in[8,16]$), where the backup cost for each parcel is drawn uniformly from the specified range. Rows are labeled with two integers separated by a comma (e.g., **(10,15)**), where the first number indicates the total number of parcels to be delivered, and the second denotes the number of available crowdshippers. For example, the column group $\boldsymbol{\rho = 2}$ with sub-column $\gamma_p \in [8, 16]$ corresponds to a cs-cost of 2 and backup cost ranging between 8 and 16, while the row **(30,45)** represents instances with 30 parcels to be delivered and 45 available crowdshippers.

The results displayed in Tables 13 and 14 confirm our expectations. The percentage of parcels delivered through crowdshipping and the total cost displayed by the objective function value are strongly influenced by the cs-cost and the backup cost. For example, when the cs-cost is set to 1 and the backup cost ranges between [8, 16], we observe in Table 13 that over 80% of parcels are delivered by crowdshippers in the best-case scenario, with an average of 40.30% across the nine instance sets under investigation. On the other hand, when the cs-cost is set to 3 and the backup cost range lies within [4, 8], we observe that crowdshippers do not deliver any parcels, as the cost of crowdshipping becomes comparable to the backup delivery cost, making it much less attractive overall.

Table 15
Sensitivity analysis – objective function value across different time-slot configurations.

$(\|P\|, \|K\|)$	$(\|T\|, \Delta t)$			
	(85,5)	(42,10)	(28,15)	(21,20)
(10,15)	138.58	142.66	141.50	150.33
(10,30)	122.21	127.13	133.33	138.73
(10,45)	109.47	114.60	116.69	135.16
(20,15)	284.91	287.12	293.33	299.21
(20,30)	269.41	277.34	275.87	289.37
(20,45)	235.77	243.18	256.29	279.65
(30,15)	446.00	459.33	451.25	452.91
(30,30)	422.86	413.41	433.47	443.39
(30,45)	389.52	392.05	410.25	425.00
Average	285.02	272.98	296.31	290.42

Table 16
Sensitivity analysis – percentage of parcels delivered by crowdshippers across different time-slot configurations.

$(\|P\|, \|K\|)$	$(\|T\|, \Delta t)$			
	(85,5)	(42,10)	(28,15)	(21,20)
(10,15)	2.76	3.16	2.25	1.16
(10,30)	6.52	5.00	6.20	3.13
(10,45)	8.76	9.08	9.34	5.00
(20,15)	3.08	4.08	3.00	2.08
(20,30)	5.91	5.95	6.12	4.08
(20,45)	11.90	10.80	10.90	6.39
(30,15)	3.54	2.95	3.08	1.34
(30,30)	5.63	7.12	5.78	4.52
(30,45)	11.60	11.80	11.20	7.09
Average	7.12	7.10	7.06	4.20

The average percentage of parcels delivered by crowdshippers in this cost setting is only 1.21%. An additional aspect worth noting from the results of our sensitivity analysis is linked to the ratio of crowdshippers available to the number of parcels to be delivered. We observe that an increasing number of crowdshippers relative to the number of parcels leads to a higher percentage of parcels delivered by crowdshippers. For example, considering the first column where cs-cost = 1 and the backup cost ranges between [8, 16], we observe that the percentage of parcels delivered by crowdshippers increases from 30% to 81.81% as the number of crowdshippers increases from 15 to 45, while delivering the same number of parcels (i.e., 10). This is an important consideration when designing a crowdshipping delivery system, as ensuring a sufficient number of crowdshippers is available to deliver the parcels is crucial. Table 14 shows similar impacts in the trends for the objective function value. We can conclude that changes in the cs-cost and backup cost directly impact the delivery system, influencing both the total cost and the number of crowdshippers that can be employed. Therefore, it is crucial to carefully select the appropriate values for these costs.

To further enhance our sensitivity analysis, we also explore the impact of varying time-slot configurations on our crowdshipping delivery model. Our main testing framework is based on a full delivery day divided into $|T| = 85$ time-slots, each lasting $\Delta t = 5$ minutes. To challenge our approach and broaden our understanding, we conducted additional tests using different $(|T|, \Delta t)$ configurations, while still keeping the cs-cost and backup cost as defined previously (i.e., cs-cost = 2, and backup cost ranging between 12 and 20). Specifically, we examined the following configurations:

- (85,5): The day is divided into 85 time-slots, each lasting 5 min;
- (42,10): The day is divided into 42 time-slots, each lasting 10 min;
- (28,15): The day is divided into 28 time-slots, each lasting 15 min;
- (21,20): The day is divided into 21 time-slots, each lasting 20 min.

Upon analyzing the results shown in Table 15, we observe that the objective function value for the first three configurations (85,5), (42,10), and (28,15) shows only minor variations. More specifically, a modest increase in the objective function value was observed when comparing (85,5) with (42,15) across all instance groups. However, a more pronounced difference emerged when comparing the standard (85,5) and the (21,20) configurations. In the instance set (30,45), the objective function value increased by 9.11% moving from the (85,5) to the (21,20) configuration, highlighting that reducing the number of time-slots significantly impacts overall performance, even for small instances like those under investigation. Additionally, as seen in Table 16, it becomes even more evident that reducing the number of available time-slots significantly affects the percentage of parcels delivered via crowdshipping. This effect is most pronounced in the (21,20) configuration, as shown by the data in the last column.

6.5.1. Managerial insights

Based on the results obtained through our sensitivity analysis, we can draw some key managerial insights that should be taken into consideration while designing a PT-based crowdshipping delivery system:

- Cost structure: The relationship between the cs-cost and the backup delivery cost is crucial. Our results show that when the cs-cost is low (i.e., 1) and the backup cost is high (ranging from 8 to 16), an average of 40.31% of parcels are delivered by crowdshippers. However, this value drops dramatically to just 1.21% when the cs-cost increases to 3 and the backup cost decreases to the [4, 8] range. Therefore, to maximize crowdshipping utilization and minimize total delivery costs, maintaining a favorable cost ratio is essential.
- Crowdshippers availability: Not surprisingly, increasing the number of available crowdshippers relative to the number of parcels to be delivered significantly improves the system efficiency. For example, with 10 parcels to be delivered, increasing the number of crowdshippers from 15 to 45 raises crowdshipping utilization from 30% to 81.81% in the most favorable cost scenario. This underscores the importance of incentivizing and maintaining a large, active crowdshipper network.
- Scalability: As the delivery volume increases, crowdshipping efficiency tends to decrease. When the cs-cost is set to 1 and the backup cost ranges in [8,16], crowdshipping utilization drops from 81.81% for 10 parcels to 43.51% for 30 parcels (with 45 available crowdshippers). This provides useful information on the necessity of defining different distribution strategies based on the volume of parcels to be delivered.
- Pricing: The significant impact of the cs-cost on system performance (e.g., crowdshipping utilization dropping from 40.31% to 1.21% as the cs-cost increases from 1 to 3) suggests that pricing is a decisive factor for the ultimate success of crowdshipping delivery.
- Network effect: Larger systems with more crowdshippers and parcels to be delivered tend to achieve higher crowdshipping percentages. For example, when the cs-cost is set to 1 and the backup cost ranges in [8, 16], the 30–45 (parcels-crowdshippers) configuration achieves 43.51% crowdshipping utilization, compared to 30% for the 10–15 configuration. This implies that there may be a "critical mass" that needs to be achieved to make the system have a positive impact.
- Backup delivery: Even in the most favorable conditions (when the cs-cost is set to 1 and the backup cost ranges in [8,16], 10 parcels to be delivered, 45 available crowdshippers), 18.19% of parcels still rely on backup delivery. This highlights the necessary role of the backup system in ensuring service reliability. When implementing such crowdshipping delivery frameworks, managers should not neglect the optimization of this backup system.
- Route-specific strategies: Although not explicitly quantified in our analysis, different PT lines have varying crowdshipping potentials. This implies that while implementing a PT-based crowdshipping system, one should consider developing and implementing route-specific strategies to maximize efficiency across different parts of the network.

By considering such factors, stakeholders can potentially make more informed decisions. The complex relationship between different factors in the success of a PT-based crowdshipping delivery system suggests the need for a more holistic approach, where changes in one area are carefully evaluated for their impact on the performance of the entire system.

6.6. Sensitivity to travel delays and path length variability

While our problem is formulated within a deterministic environment, assuming PT adheres to predefined timetables, it is essential to consider system performance in real-world scenarios, where delays and disruptions may impact travel times, potentially altering parcel pickup schedules and rendering some delivery plans infeasible. To assess this aspect, we first examine the distribution of parcel path lengths (i.e., the number of crowdshippers required to transfer a parcel) in the solutions obtained by our ALNS. Next, we refine the analysis by focusing on parcel paths involving multiple crowdshippers, specifically analyzing the temporal gap between parcel drop-off and subsequent pickup at exchange stations. These metrics offer valuable insights into the robustness of the obtained solutions when faced with real-world timing variations. Finally, the analysis focuses on the impact of delays introduced along a predefined PT line, simulating disruptions to assess their effect on optimal solutions and the system's resilience.

Table 17 presents the distribution of parcel path lengths across the various network configurations. Specifically, for each network size (denoted by $|\boldsymbol{N}|$ in the first column) and number of available crowdshippers (indicated by $|\boldsymbol{K}|$ in columns 3 to 8), the table shows the percentage of paths requiring from 1 to 4 parcel transfers (**Path Length** column). Table 17 shows a consistent pattern across all network configurations: paths involving a single crowdshipper dominate. Their share is lowest at 70.15% for $|N| = 44$, $|K| = 300$ and highest at 96.30% for $|N| = 40$, $|K| = 100$. These single-crowdshipper paths remain unaffected by typical service disruptions, as they do not require intermediate transfers and synchronization.

The high share of single-crowdshipper deliveries results from the objective function's structure, which inherently minimizes the number of involved crowdshippers to reduce costs. Since using fewer crowdshippers is the most cost-effective option under the reward scheme, our method assigns multiple crowdshippers only when necessary. This is evident in the $|N| = 44$, $|K| = 300$ case, where paths involving three (resp. four) crowdshippers reach just 4.24% (resp. 0.24%). The effectiveness of this mechanism depends on the network topology, as seen in Table 17, where instances with $|N| = 44$ display a higher share of multiple-crowdshipper parcel paths.

Moreover, the cost structure also influences parcel path length, particularly the crowdshipper reward ρ and backup cost γ_p. Ignoring σ_i and ν_i for simplicity, a parcel $p \in P$ cannot be assigned to a path longer than $\left\lfloor \frac{\gamma_p}{\rho} \right\rfloor$, as such paths would exceed the backup delivery cost. This underscores that selecting ρ in relation to γ_p and network topology, while considering business constraints such as minimum crowdshipper involvement and commuter patterns, can also promote shorter paths, reducing exposure to travel disruptions. While single-crowdshipper parcel paths remain unaffected by typical travel disruptions, those requiring synchronization (ranging from 3.70% to 29.85% of cases, depending on the network configuration; Table 17) may be impacted. To evaluate this

Table 17
Percentage of parcel paths for each length, grouped by network size $|N|$ and number of available crowdshippers $|K|$.

$\|N\|$	Path length	$\|K\|$					
		75	**100**	**150**	**200**	**250**	**300**
24	**1**	89.22%	79.93%	81.15%	78.62%	84.40%	78.98%
	2	10.76%	19.84%	18.35%	20.94%	14.99%	20.63%
	3	0.02%	0.23%	0.50%	0.43%	0.60%	0.38%
	4	0.00%	0.00%	0.00%	0.00%	0.01%	0.01%
36	**1**	93.33%	91.59%	94.51%	93.65%	93.96%	93.83%
	2	6.67%	8.41%	5.35%	6.19%	5.87%	5.99%
	3	0.00%	0.00%	0.14%	0.16%	0.17%	0.16%
	4	0.00%	0.00%	0.00%	0.00%	0.00%	0.02%
40	**1**	94.45%	96.30%	93.92%	89.06%	90.38%	85.64%
	2	5.55%	3.66%	5.95%	10.39%	9.21%	13.73%
	3	0.00%	0.04%	0.14%	0.56%	0.42%	0.63%
	4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
44	**1**	84.02%	86.20%	86.92%	81.16%	81.70%	70.15%
	2	14.58%	13.32%	12.95%	16.20%	17.15%	25.37%
	3	1.40%	0.48%	0.13%	2.60%	1.15%	4.24%
	4	0.00%	0.00%	0.00%	0.04%	0.00%	0.24%

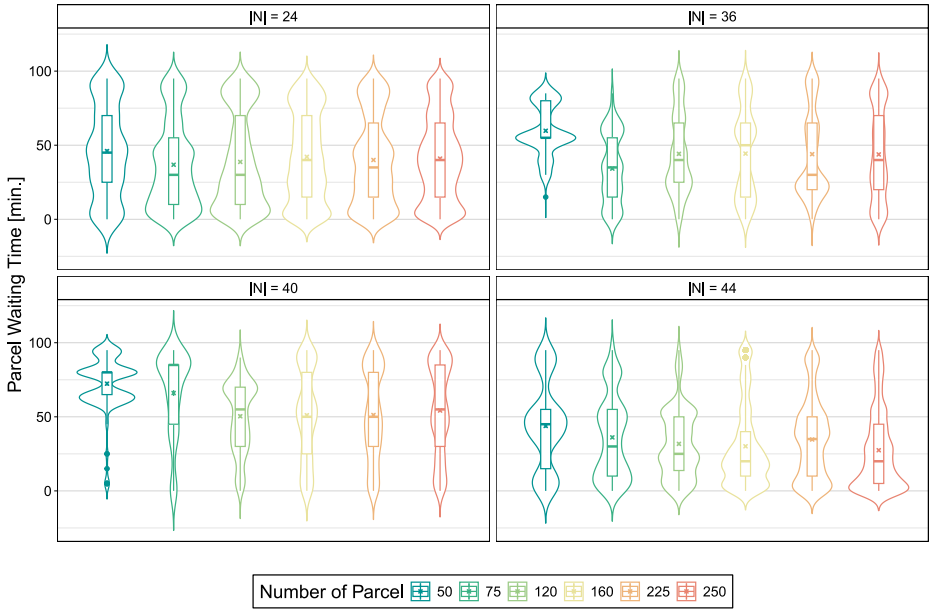


Fig. 7. Distribution of parcel waiting times at exchange stations for paths involving multiple crowdshippers.

risk, we analyze the waiting time between parcel drop-off at exchange stations and pick-up by the next crowdshipper, assessing the likelihood of failed transfers due to upstream delays.

Fig. 7 shows violin plots depicting the distribution of parcel waiting times at exchange stations, with each quadrant representing a different network size and further subdivided by the number of parcels per instance. Inner boxplots highlight the 25th, 50th, and 75th percentiles, while the central cross marks the average waiting time. Across all network sizes and parcel counts, the average waiting time remains around 45 min. The shortest average waiting time, just over 27 min, is observed for $|N| = 44$ and $|P| = 250$. Examining network-wide averages, we find approximately 40 min for $|N| = 24$, 43 min for $|N| = 36$, 53 min for $|N| = 40$, and 31 min for $|N| = 44$.

Overall, the distributions in Fig. 7 indicate that, for the parcel deliveries requiring synchronization between multiple crowdshippers, the waiting time at exchange stations is generally sufficient to prevent missed transfers. Fig. 8 further illustrates this by showing the percentage of parcels involving two or more crowdshippers that fail to transfer at an exchange station (y-axis) as a function of the incurred travel delay before reaching that station (x-axis). The developed crowdshipping framework demonstrates strong resilience to typical travel delays, particularly in reliable transport systems such as trains, metros, and dedicated-lane buses. The results indicate that networks with $|N| = 36$ (resp. $|N| = 40$) are the most robust, with failure rates below 25% for delays under 20 (resp. 30) minutes and below 50% for delays up to 40 (resp. 55) minutes. The failure rate would increase to 50% in the case of travel delays larger than 35 (resp. 25) minutes.

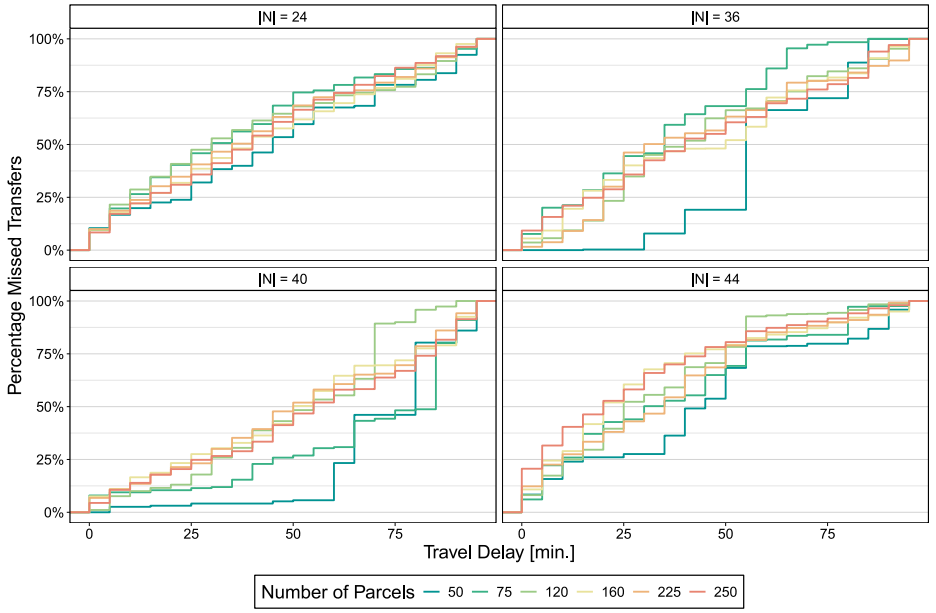


Fig. 8. Percentage of parcels missing planned transfers at exchange stations as a function of the incurred travel delay.

Additional experiments were conducted to evaluate the resilience of the crowdshipping framework, considering an extreme configuration with $|P| = 30$ parcels, $|K| = 30$ crowdshippers, and travel delays ranging from 20% to 50% of the scheduled travel time. The network used for these tests has $|N| = 44$ nodes and $|L| = 7$ PT lines.

Fig. 9 displays the distribution of the number of parcels delivered by crowdshippers under different delay conditions applied to each network line separately. Due to the reduced number of available crowdshippers, only a portion of the parcels can be assigned to crowdshipping in this setup. The cross of each box plot (each line is associated with a different color) represents the average number of parcels delivered in each scenario. The black box plot in the leftmost part of each quadrant corresponds to the base case without any travel delays, serving as a benchmark for comparison. The impact of travel delays depends on which PT line is affected. Interestingly, delays on certain lines can lead to an improvement in the number of parcels ultimately delivered. For example, when line 3 is delayed, the number of delivered parcels is never reduced by more than 4.76%, but it can, in the best case, increase by 28.57%. A similar improvement, though smaller, is also observed for lines 4 and 5. This suggests that delays in certain parts of the network can create new synchronization opportunities, ultimately leading to more crowdshipping possibilities. However, these new opportunities come at a cost, as alternative crowdshippers must be compensated with a reward. In contrast, for line 1, the presence of delays consistently decreases the number of parcels delivered, as they cause missed transfers between crowdshippers. The impact is minimal, with a reduction of 7.14% (resp. 2.38%) in parcels delivered for 30% (resp. 50%) delays. For lines 2 and 7, the number of delivered parcels remains constant despite the incurred delays, indicating that these lines do not serve as primary bottlenecks limiting the system's delivery performance.

These additional experiments confirm that the system's resilience to travel delays is mainly influenced by two factors: the timing of crowdshipper movements and the network topology. The timing factor is especially evident for line 3, where the number of parcels delivered varies with increasing delays, suggesting that certain delivery routes become feasible or infeasible depending on the delay conditions. Regarding the network topology, the length and connectivity (number of intersections) of PT lines impact the effect of travel disruptions, as shown by the variations in Fig. 9 across the different lines.

In summary, the analysis highlights several key points. First, the system is inherently resilient to delays, primarily due to the majority of parcels being delivered by a single crowdshipper, reducing the need for precise synchronization. This is a result of the objective function, which minimizes the delivery costs and, indirectly, the number of crowdshippers involved. Second, for parcel paths requiring multiple crowdshippers, resilience to synchronization issues depends on the network topology and commuter patterns, with most deliveries unaffected by limited delays. Third, when a parcel transfer is missed, the application of dynamic rerouting procedures could ensure delivery by reassigning the parcel to a different crowdshipper. The development of such dynamic algorithms, which would also help handle situations where crowdshipper availability changes on short notice, requires a deeper understanding of the realistic time frames within which commuters can respond to crowdshipping requests. As such, we leave it as a promising avenue for future research. To conclude, travel delays would only significantly impact the system in rare cases involving multiple crowdshippers, missed transfers, and no alternative crowshipping options, with the need for backup services remaining limited.

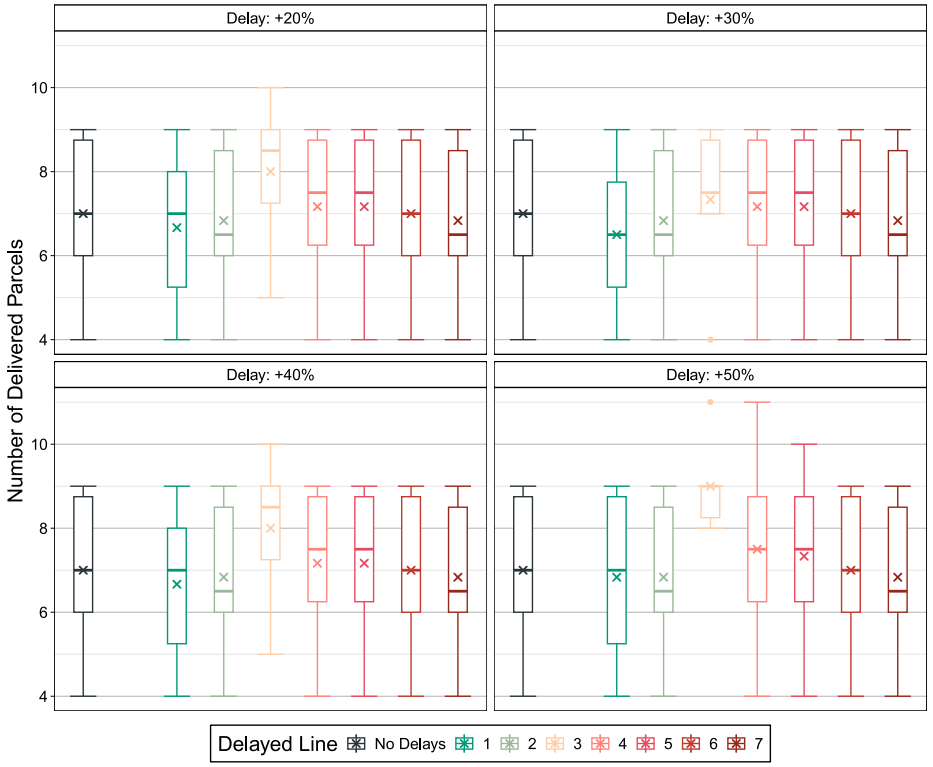


Fig. 9. Number of parcels delivered in presence of travel delays for instances involving $|P| = 30$ parcels, $|K| = 30$ available crowdshippers, $|N| = 44$ stations and $|L| = 7$ public transportation lines.

7. Environmental impact comparison: Crowdshipping versus traditional delivery systems

This section analyzes how a crowdshipping system can reduce environmental impact compared to a scenario in which an LSP vehicle handles all deliveries. Specifically, we compare the total travel effort of the LSP in two scenarios: (i) the one derived from the optimal crowdshipping solution, where the LSP operates in the morning to deliver parcels to source stations and in the evening to perform backup deliveries for parcels not served by the crowdshippers, and (ii) a hypothetical case in which the LSP delivers all parcels directly to the destination stations. For scenario (i), we estimate the LSP's travel effort by solving two Traveling Salesman Problems (TSPs): one in the morning, covering the source stations (identified in our model's optimal solution), and another in the evening visiting the destination stations for parcels not handled by crowdshippers. In contrast, for scenario (ii), a single TSP is solved over the destination stations for all parcels.

We apply this analysis to a representative instance ($|N| = 24$, $|K| = 75$, $|P| = 50$, $Id = 3$), solved to optimality, where each parcel has a fixed, predefined destination station. Starting from this instance, we generate three additional ones by incrementally increasing the number of available crowdshippers, adding 75 randomly generated ones at each step. In this way, we analyze how the environmental footprint of the crowdshipping system evolves as the number of participants increases, thereby expanding the potential volume of parcels delivered via crowdshipping.

Table 18 presents the results and comparative analysis between the crowdshipping system (i) and the traditional delivery scenario (ii). All instances were solved to optimality using our mathematical model: column **Obj.** reports the optimal objective function value (i.e., optimal crowdshipping system cost), while column # indicates the number of parcels delivered by crowdshippers in the optimal solution. As the number of crowdshippers increases, a larger share of parcels is delivered through the public transport network, significantly reducing the dependence on LSP backup intervention and the associated global cost. This trend is captured by **Obj.** value which decreases from 629 to 340 as the number of crowdshippers grows from 75 to 300, and by the # value which shows a corresponding increase in the number of parcels delivered by crowdshippers from 15 to 50 (all parcels delivered by the crowdshippers). These results confirm the ability of our mathematical model to improve cost-efficiency as more crowdshipping options become available.

The remaining columns in Table 18 report the estimated travel effort required by the LSP under both delivery systems, expressed as total travel times (in seconds). Under the **Crowdshipping System**, the morning, the evening and the total travel times are reported for each instance. While morning travel times remain constant across instances, yielding 3354 s and involving the same 3 source stations, as the number of available crowdshippers progressively increases, the travel time required by the backup service (**Evening CS**) decreases accordingly, as fewer parcels remain unassigned to crowdshippers. This leads to a reduced travel effort for

Table 18
Travel time comparison between the Traditional Delivery System and the Crowdshipping System under varying numbers of crowdshippers.

$\|K\|$	Optimal solution		Crowdshipping system			Traditional delivery system	
	Obj.	#	Morning CS	Evening CS	Total CS	TSP	Impr.
75	629.0	15	3354	5439	8793	7043	–24.85%
150	474.0	33	3354	3345	6699	7043	4.88%
225	377.0	48	3354	1674	5028	7043	28.61%
300	340.0	50	3354	0	3354	7043	52.38%

the LSP, from 5439 s when 35 parcels require LSP delivery with 75 crowdshippers, down to 0 s with 300 crowdshippers, when the crowdshipping system can handle all parcel deliveries. In the **Traditional Delivery System**, where all parcels are delivered solely by the LSP, a single TSP is solved on the full set of destination stations, resulting in a constant travel time of 7043 s across all instances. Finally, column **Impr.** shows the percentage improvement of the total LSP travel time in the crowdshipping system over the traditional one, highlighting the reduction in vehicle routing effort due to the involvement of crowdshippers. Interestingly, when the number of available crowdshippers is low, as in the first instance, the traditional delivery system actually outperforms the crowdshipping-based approach, with column **Impr.** showing a negative value of –24.5%. However, as the number of crowdshippers increases, the trend reverses: with 150 crowdshippers, the crowdshipping system already achieves a positive improvement of 4.88%, which further rises to 52.38% when 300 crowdshippers are available.

While the initial comparison is based on travel times, the analysis becomes more impactful when directly considering CO_2 emissions, offering a clearer assessment of the crowdshipping system's effectiveness in reducing the LSP's environmental footprint. Assuming an average driving speed of 30 km/h for the LSP vehicle, travel times can be converted into distances. Considering a conservative fuel consumption of 8.5 liters of diesel per 100 km, typical for a modern urban delivery van, we estimate that traditional delivery systems that require an average daily travel time of 7043 s (approximately 58,69 km) would produce approximately 23 kg of CO_2 per day. In contrast, in the best crowdshipping scenario, with $|K| = 300$, the total travel time (corresponding only to the morning parcel transfer) is equal to 3354 s (about 27,5 km) resulting in about 11 kg of CO_2 emissions. If we assume the service operates 300 days per year and these figures are taken as representative averages, the crowdshipping system would prevent the emission of approximately 3.6 tons of CO_2 annually.

8. Conclusions

We have introduced optimization strategies for addressing the Public Transportation-based Crowdshipping Problem (PTCP), effectively integrating public transportation (PT) networks into the delivery ecosystem. This study directly incorporates PT networks as a relevant part of standard parcel delivery frameworks. We propose a system based on PT stations as pick-up points equipped with automatic parcel lockers and commuters serving as crowdshippers transferring packages between stations. A backup delivery service handles final delivery for packages not reaching their destination through crowdshipping.

We formulate the problem as a MILP capturing the synchronization constraints inherent to such a system. The model aims to minimize the total delivery costs comprising crowdshipping transfers and backup deliveries. To effectively solve problem instances, we have developed two solution approaches: an Adaptive Large Neighborhood Search (ALNS) metaheuristic and a Branch-and-Cut algorithm based on different classes of valid inequalities.

We have generated a wide set of problem instances based on randomized public transit networks to accommodate different scenarios that could appear in real-world situations. Our computational experiments demonstrate the challenges that commercial solvers face in dealing with PTCP in most of the tested instances. On the other hand, our ALNS has proven to be efficient in handling both small and large instances at a faster pace, yielding results of good quality. Specifically, our heuristic consistently achieved an average time to the best solution found of less than five minutes, overcoming CPLEX in most of the cases. In very small instances, CPLEX performs extremely well, outperforming ALNS, which still achieves good results with only marginal gaps (i.e., lower than or equal to 1%). Additionally, we performed an extensive sensitivity analysis around crowdshipping and backup delivery costs. The results highlighted the significant influence of the ratio of such costs on overall system expenses and crowdshipping utilization.

The proposed deterministic framework offers a balance between model tractability and realistic applicability, providing a solid foundation for analyzing cost-optimized crowdshipping. While future research may introduce stochastic elements for a more in-depth validation, several promising developments could enhance its robustness and applicability. These include integrating public transit schedules, developing real-time coordination systems, designing more refined pricing schemes and incentives, and allowing limited diversions from commuters' original paths to increase delivery flexibility. In particular, the adoption of adaptive algorithms and real-time data could improve operational resilience, while behavioral studies could refine incentive mechanisms, for example, by introducing dynamic travel credits linked to demand or time windows. Alternative pricing strategies, such as distance-based rewards instead of fixed per-parcel payments, could better align incentives with crowdshippers' efforts. Finally, an active collaboration between public transit services and logistics providers to co-design dedicated infrastructure and improve coordination among stakeholders would facilitate the transition from theory to practice, strengthening interest in sustainable, public transit-integrated crowdshipping solutions.

CRediT authorship contribution statement

Mikele Gajda: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Conceptualization. **Olivier Gallay:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Conceptualization. **Renata Mansini:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Formal analysis, Conceptualization. **Filippo Ranza:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

References

Archetti, C., Guerriero, F., Macrina, G., 2021. The online vehicle routing problem with occasional drivers. Comput. Oper. Res. 127, 105144.

Archetti, C., Savelsbergh, M., Speranza, M.G., 2016. The vehicle routing problem with occasional drivers. European J. Oper. Res. 254 (2), 472–480.

Arslan, A.M., Agatz, N., Kroon, L., Zuidwijk, R., 2019-02. Crowdsourced delivery—A dynamic pickup and delivery problem with ad hoc drivers. Transp. Sci. (1), 222–235, Publisher: INFORMS.

Bonomi, V., Mansini, R., Zanotti, R., 2022. Last mile delivery with parcel lockers: Evaluating the environmental impact of eco-conscious consumer behavior. IFAC- Pap. 55 (5), 72–77.

Boysen, N., Fedtke, S., Schwerdfeger, S., 2021. Last-mile delivery concepts: a survey from an operational research perspective. OR Spectrum 43 (1), 1–58.

Boysen, N., Schwerdfeger, S., Weidinger, F., 2018. Scheduling last-mile deliveries with truck-based autonomous robots. European J. Oper. Res. 271 (3), 1085–1099.

Carbone, V., Rouquet, A., Roussat, C., 2017. The rise of crowd logistics: a new way to co-create logistics value. J. Bus. Logist. 38 (4), 238–252.

Cheng, R., Jiang, Y., Nielsen, O.A., 2023. Integrated people-and-goods transportation systems: from a literature review to a general framework for future research. Transp. Rev. 43 (5), 997–1020.

Chevalier, S., 2024. Retail E-commerce sales worldwide from 2014 to 2027. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/. (Online; Accessed 6 March 2024).

Côté, J.F., Mansini, R., Raffaele, A., 2024. Multi-period time window assignment for attended home delivery. European J. Oper. Res. 316 (1), 295–309.

Dayarian, I., Savelsbergh, M., 2020. Crowdshipping and same-day delivery: Employing in-store customers to deliver online orders. Prod. Oper. Manage. 29 (9), 2153–2174.

Demir, E., Huang, Y., Scholts, S., Van Woensel, T., 2015. A selected review on the negative externalities of the freight transportation: Modeling and pricing. Transp. Res. Part E: Logist. Transp. Rev. 77.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Department of Transportation, NYC, 2019. Improving the Efficiency of Truck Deliveries in NYC. Nimber.

Derse, O., Van Woensel, T., 2024. Integrated people and freight transportation: A literature review. Futur. Transp. 4 (4), 1142–1160.

Di Puglia Pugliese, L., Ferone, D., Macrina, G., Festa, P., Guerriero, F., 2023. The crowd-shipping with penalty cost function and uncertain travel times. Omega 115, 102776.

Elbert, R., Rentschler, J., 2022. Freight on urban public transportation: A systematic literature review. Res. Transp. Bus. Manag. 45, 100679. http://dx.doi.org/10.1016/j.rtbm.2021.100679, Urban logistics: From research to implementation.

Fessler, A., Cash, P., Thorhauge, M., Haustein, S., 2022. A public transport based crowdshipping concept: Results of a field test in Denmark. Transp. Policy 134, 106–118.

Filippi, C., Plebani, F., 2021. Metro stations as crowd-shipping catalysts: An empirical and computational study. arXiv preprint arXiv:2109.08069.

Gajda, M., Boysen, N., Gallay, O., 2024. Impact of bot return policies in van-and-bot delivery systems. Int. J. Prod. Res. 62 (17), 6360–6379.

Gatta, V., Marcucci, E., Nigro, M., Patella, S.M., Serafini, S., 2018. Public transport-based crowdshipping for sustainable city logistics: Assessing economic and environmental impacts. Sustainability 11 (1), 145.

Gatta, V., Marcucci, E., Nigro, M., Serafini, S., 2019. Sustainable urban freight transport adopting public transport-based crowdshipping for B2C deliveries. Eur. Transp. Res. Rev. 11 (1), 1–14.

Grangier, P., Gendreau, M., Lehuédé, F., Rousseau, L.M., 2016. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. European J. Oper. Res. 254 (1), 80–91. http://dx.doi.org/10.1016/j.ejor.2016.03.040.

Hatzenbühler, J., Jenelius, E., Gidófalvi, G., Cats, O., 2023. Modular vehicle routing for combined passenger and freight transport. Transp. Res. Part A: Policy Pr. 173, 103688.

Hemmelmayr, V.C., Cordeau, J.F., Crainic, T.G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Comput. Oper. Res. 39 (12), 3215–3228. http://dx.doi.org/10.1016/j.cor.2012.04.007.

Jie, W., Yang, J., Zhang, M., Huang, Y., 2019. The two-echelon capacitated electric vehicle routing problem with battery swapping stations: Formulation and efficient methodology. European J. Oper. Res. 272 (3), 879–904. http://dx.doi.org/10.1016/j.ejor.2018.07.002.

Joerss, M., Schröder, J., Neuhaus, F., Klink, C., Mann, F., 2016. Parcel delivery: the future of last mile. Travel. Transp. Logist. 32.

Kızıl, K.U., Yıldız, B., 2023. Public transport-based crowd-shipping with backup transfers. Transp. Sci. 57 (1), 174–196.

Liu, Y., Guo, B., Chen, C., Du, H., Yu, Z., Zhang, D., Ma, H., 2019. FooDNet: Toward an optimized food delivery network based on spatial crowdsourcing. IEEE Trans. Mob. Comput. 18 (6), 1288–1301. http://dx.doi.org/10.1109/TMC.2018.2861864.

Machado, B., Pimentel, C., de Sousa, A., 2023. Integration planning of freight deliveries into passenger bus networks: Exact and heuristic algorithms. Transp. Res. Part A: Policy Pr. 171, 103645.

Manerba, D., Mansini, R., Zanotti, R., 2018. Attended home delivery: Reducing last-mile environmental impact by changing customer habits. IFAC- Pap. 51 (5), 55–60.

Mansini, R., Ranza, F., Zanotti, R., 2024. Towards sustainable last-mile delivery: Introducing an off-peak urban policy to mitigate environmental and social impacts. IFAC- Pap. 58 (2), 186–191, 3rd IFAC Workshop on Integrated Assessment Modeling for Environmental Systems IAMES 2024.

Mo, P., Yao, Y., D'Ariano, A., Liu, Z., 2023. The vehicle routing problem with underground logistics: Formulation and algorithm. Transp. Res. Part E: Logist. Transp. Rev. 179, 103286.

Powell, S., Campbell, A.M., Hosseini, M., 2024. Underground freight transportation for package delivery in urban environments. arXiv preprint arXiv:2405.04618.

Ranieri, L., Digiesi, S., Silvestri, B., Roccotelli, M., 2018. A review of last mile logistics innovations in an externalities cost reduction vision. Sustainability 10 (3), 782.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. 40 (4), 455–472.

Savelsbergh, M., Van Woensel, T., 2016. 50Th anniversary invited article—City logistics: Challenges and opportunities. Transp. Sci. 50.

Seghezzi, A., Siragusa, C., Mangiaracina, R., 2022. Parcel lockers vs. home delivery: a model to compare last-mile delivery cost in urban and rural areas. Int. J. Phys. Distrib. Logist. Manage. 52 (3), 213–237, Publisher: Emerald Publishing Limited.

Tiit, E.M., 2000. Nonparametric Statistical Methods. Myles Hollander and Douglas A. Wolfe, Wiley, Chichester, 1999. Wiley Online Library, No. of pages: xiii+ 779. Price:£ 39.95. ISBN 0-471-19045-4.

Vakulenko, Y., Hellström, D., Hjort, K., 2018. What's in the parcel locker? Exploring customer value in e-commerce last mile delivery. J. Bus. Res. 88, 421–427.
Windras Mara, S.T., Norcahyo, R., Jodiawan, P., Lusiantoro, L., Rifai, A.P., 2022. A survey of adaptive large neighborhood search algorithms and applications. Comput. Oper. Res. 146, 105903. http://dx.doi.org/10.1016/j.cor.2022.105903.
Wyrowski, A., Boysen, N., Briskorn, D., Schwerdfeger, S., 2024. Public transport crowdshipping: moving shipments among parcel lockers located at public transport stations. OR Spectrum.
Zhang, M., Cheah, L., Courcoubetis, C., 2022a. Exploring the potential impact of crowdshipping using public transport in Singapore. Transp. Res. Rec.: J. Transp. Res. Board 036119812211232.
Zhang, H., Lv, Y., Guo, J., 2022b. New development direction of underground logistics from the perspective of public transport: A systematic review based on scientometrics. Sustainability 14 (6).
Zurel, Ö., Van Hoyweghen, L., Braes, S., Seghers, A., 2018. Parcel lockers, an answer to the pressure on the last mile delivery? In: New Business and Regulatory Strategies in the Postal Sector. In: Topics in Regulatory Economics and Policy, Springer International Publishing.