Service Network Design for Same-Day Delivery with Hub Capacity Constraints

Haotian Wu, Ian Herszterg, Martin Savelsbergh
H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
haotian.wu@gatech.edu (HW), ihersztreg@gatech.edu (IH), martin.savelsbergh@isye.gatech.edu (MS)

Yixiao Huang
SF Express (Group) Co., Ltd., Shenzhen, China 518048
yixiaohuang@sffmail.sf-express.com (YH)

We study a new service network design problem for an urban same-day delivery system in which the number of vehicles that can simultaneously load or unload at a hub is limited. Due to the presence of both time constraints for the commodities and capacity constraints at the hubs, it is no longer guaranteed that a feasible solution exists. The problem can be modeled on a time-expanded network and formulated as an integer program. To be able to solve real-world instances, we design and implement three heuristics: (1) an integer programming based heuristic, (2) a metaheuristic, and (3) a hybrid matheuristic. An extensive computational study using real-world instances (with different geographies, market sizes, and service offerings) from one of China’s leading comprehensive express logistics service providers demonstrates the efficacy of the three heuristics, but also reveals that offering the desired service in all possible markets (all origin-destination pairs) is not possible. Therefore, a pragmatic variant of the problem is also introduced and investigated.

Key words: service network design; hub capacity constraints; same-day delivery; heuristics

1. Introduction

Package delivery represents a significant part of the transportation industry. Measured by revenue, package delivery has been the fastest growing segment of the freight transport business in the United States in the 21st century, with one representative player (UPS) reporting a revenue increase from $30 billion in 2000 to $72 billion in 2018. A critical aspect of package delivery is the service. Driven by the growth of e-commerce which relies heavily on faster delivery times [Turban et al. 2015], package express carriers have long been aiming to provide same-day delivery service with wide coverage, especially in major metropolitan areas. Being able to do so is expected to increase demand (and profit) significantly given the compelling value proposition of same-day delivery for consumers [Savelsbergh and Van Woensel 2016].

To provide an economically viable (same-day) delivery service, carriers need to carefully allocate and utilize their resources. The challenge is to identify consolidation opportunities (so as to keep
the costs down) while satisfying the service guarantees offered to customers (so as to maintain or increase market share). Service network design problems have long been used to aid in this process, and are usually modeled on a time-expanded network, especially when the time aspect is critical. The goal is to minimize cost through consolidation by choosing the right shipment paths in both space and time. A consolidation transportation system typically employs a complex hub network, in which vehicles transport packages between hubs, and packages are unloaded, sorted, and loaded at hubs. Routing packages through intermediate hubs is key to consolidation, but requires additional time and additional loading and unloading.

The novel feature in the service network design problem investigated in this paper is a restriction on the number of vehicles that can simultaneously be loaded and unloaded at a hub, i.e., the presence of hub capacity constraints. These hub capacity constraints can cause waiting at hubs, which, in turn, can result in certain shipment paths no longer being time feasible, especially in urban same-day delivery environments with aggressive service offerings. In long-haul transportation systems of consolidation carriers such hub capacity constraints are usually not relevant, because the hubs are large (some very large) with many loading and unloading docks. However, in short-haul transportation systems in urban areas such hub capacity constraints are critical, because real-estate is expensive (and will continue to become more expensive) and the hubs are small (some very small) with only a few loading and unloading docks. Although service network design models and methods have been developed for various settings, to the best of our knowledge, none of these have considered accommodating these hub capacity constraints, especially not in a same-day delivery environment.

The restriction on the number of vehicles that can simultaneously be loaded and unloaded at a hub has an important consequence, which is typically not encountered in service network design problems: the existence of a feasible solution is no longer guaranteed. For example, it may not be feasible to send all packages directly from their origin to their destination as this requires a large number of vehicles and the limit on the number of vehicles that can be loaded and unloaded simultaneously at hubs may result in so much waiting that some packages will no longer be able to meet their service guarantee. Therefore, this service network design variant has two objectives: (1) maximize the number of origin-destination markets that can be served, and (2) given the origin-destination markets that are served, minimize the cost of serving these markets.

The variant of the service network design problem investigated is motivated by the settings encountered in China’s major mega-cities at package express carrier SF Express (and the instances used in our computational study are representative of these settings). The same-day service products of interest include the “12-18 service”, the “14-20 service”, and the “12-20 service”. Each service product specifies the latest time customers should present their packages to the carrier and
the latest time that the carrier guarantees delivery takes place. For example, a customer choosing the “12-18 service” product has to present their packages to the carrier before noon, and the carrier guarantees that these packages will be delivered no later than 6 PM (the same day). This does not mean that all packages for service product 12-18 are available at an origin hub at noon, because in the SF Express system couriers collect packages from customers and deliver these packages to an access point (a parcel box), and then riders deliver packages from the access point to the origin hub. Similarly, it does not mean that all packages for service product 12-18 need to be at the destination hub by 6 PM, because riders need to deliver the packages to an access point, where they are picked up by couriers to handle the ultimate delivery. Therefore, a certain amount of time (1.5 to 2 hours) has to be reserved for these “pre-processing” and “post-processing” activities when designing the same-day service network. For example, if 2 hours are reserved for the pre-processing and post-processing activities for service product 12-18, then the carrier needs to transport the package from its origin hub to its destination hub between 2 PM and 4 PM.

The service network design problem with hub capacity constraints that we consider seeks for every market, i.e., for every origin – destination pair of hubs, a path specifying how the packages in this market will be transported from their origin to their destination, and a minimum cost vehicle schedule for transporting the packages that specifies for each vehicle movement, its origin and destination, when loading at the origin starts (and ends), when it departs from the origin (and when it arrives at the destination), and when unloading at the destination starts (and ends), while ensuring that service constraints are satisfied, i.e., loading of packages at their origin starts after they become available and unloading of packages at their destination completes before they are due, and satisfies the operating restrictions at hubs, i.e., the restrictions on the number of vehicles that can simultaneously be loaded and unloaded.

We model this service network design problem using a time-expanded network in which arcs represent the loading, travel, waiting, and unloading of a vehicle and commodities represent packages in a market, and show how this model can be formulated as an integer program (IP). We develop three heuristic algorithms for its solution, because the IP cannot be solved in an acceptable amount of time for real-world instances: (1) an IP-based heuristic, (2) a metaheuristic, and (3) a hybrid matheuristic, which takes advantage of the strengths of the IP-based heuristic and the metaheuristic. Computational experiments show that the hybrid matheuristic produces high-quality solutions in a reasonable amount of time. Because it is not always possible to find feasible solutions, i.e., sometimes not all markets can be served while satisfying service constraints and hub capacity constraints, a pragmatic variant, in which it is allowed to exceed the maximum number of vehicles that can be loaded or unloaded at hub for short periods of time, is also introduced and investigated. The contributions of this research are summarized as follows:
A new service network design variant is introduced for same-day delivery systems in urban areas and a non-trivial integer programming model for its solution is developed;

- An IP-based heuristic, a metaheuristic, and a hybrid matheuristic, which combines the strengths of the IP-based heuristic and the metaheuristic, are designed and implemented; and

- The efficacy of the heuristics is demonstrated on a number of real-world instances.

- A practical variant of the new service network design problem is introduced that incorporates further constraints encountered in real-life and that converts some constraints into soft constraints to ensure an implementable solution is obtained.

The remainder of the paper is organized as follows. In Section 2, we review relevant prior research. In Section 3, we provide a formal description of the same-day delivery system and present a model and formulation using an appropriately constructed time-expanded network. In Section 4, we introduce three heuristics: an IP-based heuristic, a metaheuristic, and a hybrid matheuristic. In Section 5, we present and interpret the results of an extensive computational study, and discuss a variation of the problem able to handle additional real-life complexities. Finally, in Section 6, we finish with conclusions and a discussion of future work.

2. Literature Review

Since we are not aware of any literature addressing the design of a service network for same-day delivery of packages with loading and unloading capacities at hubs, we briefly review literature on same-day delivery, on service network design for package express carriers, on service network design with other types of capacity limitations, and on advances in solving service network design problems.

There is a growing body of literature on same-day delivery, but it is focused almost exclusively on the delivery of packages from a fulfillment center, which gives rise to challenging dynamic routing and scheduling problems (see, e.g., Pillac et al. (2013), Psaraftis, Wen, and Kontovas (2016), Sampaio et al. (2019), and Alnaggar, Gzara, and Bookbinder (2019)), but is quite different from the same-day delivery environment studied in this paper, which is characterized by package flows between hubs. There is literature on courier operations and dial-a-ride services in urban areas, which do involve taking packages or passengers from a pickup location to a drop-off location (see, e.g., Berbeglia, Cordeau, and Laporte (2010), Bouros et al. (2011), and Ho et al. (2018)), but consolidation and transfers are usually not as critical to feasibility and profitability as they are in the same-day delivery environment studied in this paper.

There is a fair amount of literature on optimizing the design and operations of package express carriers service networks, but the focus has been on inter-city package flows rather than same-day delivery of packages within a city. Examples include Kim et al. (1999), Barnhart et al. (2002), Yildiz and Savelsbergh (2019), and Lin, Zhao, and Lin (2020).
Researchers have incorporated capacity limitations into service network design models in a variety of transportation systems. In the context of an air transportation system, Helme (1992) proposes a multi-commodity minimum-cost flow model on a time-expanded network which incorporates the arrival capacity at destination airport of flights. In the context of a maritime transportation system, where the number of available berths in a port strongly impacts the throughput capacity of a port, Yan et al. (2015) formulate the dynamic berth allocation problem as an integer multi-commodity network flow problem, using innovative flexible berth-space utilization scheme based on blocking plans. In rail networks, the capacity of a classification yard limits the formation of blocks, i.e., sets of railcars that travel together on part of their journey from origin to destination. Barnhart, Jin, and Vance (2000) formulate the blocking problem, i.e., the creation of cost effective blocks, as a network design problem, with nodes and arcs representing yards and candidate blocks, respectively. Limited yard capacity is incorporated by imposing maximum in- and outdegrees on the nodes. In the context of service network design for intercity package flows, Lin, Zhao, and Lin (2020) consider the (limited) sorting capacity at terminals.

Recent research advances in exact solution methods for solving service network design problems have come from iterative refinement techniques (Boland et al. 2017, Clautiaux et al. 2017) and branch-and-price-and-cut algorithms (Gendron and Larose 2014, Rothenbächer, Drexl, and Irnich 2016). More effective matheuristics, i.e., approaches integrating metaheuristic concepts and mathematical programming techniques, continue to appear. Gendron, Hanafi, and Todosijević (2018) present a matheuristic that iterates between linear programming and slope scaling for multicommodity capacitated fixed-charge network design. Crainic et al. (2014) combine column generation, metaheuristic, and exact optimization techniques in dealing with service network design with resource constraints. In liner shipping network design, Brouer, Desaulniers, and Pisinger (2014) use an integer program to search for improvements in their matheuristic. For less-than-truckload load plan design, Lindsey, Erera, and Savelsbergh (2016) develop an effective neighborhood search heuristic for solving a natural integer programming model where a modified and restricted version of the integer program is solved to find improving changes during each iteration of the matheuristic.

3. Problem Description

Let $D = (N, A)$ be a flat network with node set $N$ modeling physical locations or hubs and directed arc set $A$ modeling travel between locations. A travel time $\tau_{ij} \in \mathbb{N}_+$ and a travel cost $c_{ij} \in \mathbb{R}_+$ are associated with each arc $a = (i, j) \in A$. Let $K$ denote a set of commodities to be served, each of which has a single source node $o_k \in N$ (later referred to as the commodity’s origin), a single sink node $d_k \in N$ (later referred to as the commodity’s destination), and a quantity $q_k \in \mathbb{R}_+$ that needs to be routed from its origin to its destination along a single geographic path $P_k = (o_k, \ldots, d_k)$.
using a homogeneous fleet of vehicles with capacity \( Q \in \mathbb{N}_+ \). We assume that \( q_k \leq Q \) for all \( k \in K \). Commodity \( k \in K \) becomes available at its origin at time \( e_k \in \mathbb{N}_+ \) and is due at its destination at time \( l_k \in \mathbb{N}_+ \).

We assume that each commodity \( k \in K \) will be loaded every time it departs from a hub \( i \in N \) and will be unloaded every time it arrives at a hub \( i \in N \). This assumption is in line with practices at many package express carriers. It simplifies operations at hubs; selective loading and unloading packages is prone to human error, especially under time pressure. We assume the loading of vehicle takes \( \tau_l \in \mathbb{N}_+ \) and the unloading of a vehicle takes \( \tau_u \in \mathbb{N}_+ \). For ease of presentation, we assume these times are the same for all hubs, but it is easy to accommodate hub-dependent loading and unloading times. Each hub \( i \in N \) has loading capacity \( L_i \in \mathbb{N}_+ \) and an unloading capacity \( U_i \in \mathbb{N}_+ \), which represent the maximum number of vehicles that can be loaded and unloaded at the same time, respectively. We assume that there is enough space at each hub for vehicles to wait if loading or unloading cannot start upon arrival. Finally, we assume, without loss of generality, that a vehicle departs as soon as it is loaded. Thus, a vehicle may only wait before it gets loaded or before it gets unloaded.

The Service Network Design with Hub Capacities (SNDHC) problem seeks to determine a path \( P_k \) for each commodity \( k \in K \) and a vehicle schedule that implies loading and unloading start times at each hub in the path \( P_k \), such that the loading at the origin hub starts at or after \( e_k \) and the unloading at the destination hub starts at or before \( l_k - \tau_u \) and that satisfies the hub capacity constraints, i.e., for every hub \( i \in N \) and at any time \( t \in [\min_{k \in K} e_k, \max_{k \in K} l_k] \), there are no more than \( L_i \) vehicles loading and no more than \( U_i \) vehicles unloading. The restriction on the number of vehicles that can simultaneously be loaded and unloaded at a hub implies that the existence of a feasible solution is no longer guaranteed. Therefore, the objective of SNDHC is to minimize the cost of a vehicle schedule that maximizes the number of commodities for which a feasible path can be found.

We use a time-expanded network to model SNDHC. We derive a time-expanded network \( \mathcal{D} = (\mathcal{N}, \mathcal{A} \cup \mathcal{H}) \) from flat network \( D \) and a set of time points \( T = \bigcup_{i \in N} T_i \) with \( T_i = \{t_1, \ldots, t_{n_i}\} \). The timed node set \( \mathcal{N} \) has a node \((i, t)\) for each node \( i \in N \) and \( t \in T_i \). The holding arc set \( \mathcal{H} \) contains arcs \( ((i, t_g), (i, t_{g+1})) \) for all \( i \in N \) and \( g = 1, \ldots, n_i - 1 \). A holding arc \( ((i, t_g'), (i, t_{g'+1})) \) models the possibility of holding packages at hub \( i \) for a period of time while they wait to be loaded onto a vehicle. The movement arc set \( \mathcal{A} \) contains arcs of the form \( ((i, t), (j, \bar{t})) \), where \((i, j) \in A, t \in T_i, \) and \( \bar{t} \in T_j \). An arc \( ((i, t), (j, \bar{t})) \) models the possibility of sending packages from hub \( i \) to hub \( j \) with the loading of packages starting at \( i \) at time \( t \) and the unloading of packages finishing at \( j \) at time \( \bar{t} \). This implies that the vehicle departs from \( i \) at time \( t + \tau_l \) and waits at \( j \) from \( t + \tau_l + \tau_{ij} \) until \( \bar{t} - \tau_u \). (Thus, we must have that \((\bar{t} - \tau_u) - (t + \tau_l) \geq \tau_{ij}\).)
Observe that arc \((i, t), (j, \bar{t})\) and arc \((i, t), (j, \bar{t}')\) represent a different set of activities even though the vehicle travels at the exact same time (departing at \(i\) at \(t + \bar{\tau}_i\) and arriving at \(j\) at \(t + \bar{\tau}_i + \tau_{ij}\)). This differs from most service network design problems, where it suffices to model vehicle movements, i.e., have arcs of the form \((i, t), (j, t + \tau_{ij})\). However, to be able to accurately model the hub capacity constraints, it is necessary to explicitly embed the loading and unloading of a vehicle into the arc. Different packages traveling from \(i\) to \(j\) on their journey from their origin to their destination can start loading at \(i\) at the same time \(t\) but start unloading at \(j\) at different times \(\bar{t} - \bar{\tau}_u\) at \(\bar{t}' - \bar{\tau}_u\) due to differences in waiting time at \(j\) of the vehicle that transports them.

We use a time-expanded network \(\mathcal{D}^\Delta\) derived from \(\mathcal{D}\) with a regular time discretization controlled by parameter \(\Delta \in \mathbb{N}_+\). Specifically, we let \(T_i = \{E\Delta, (E + 1)\Delta, \ldots, L\Delta\}\) for all \(i \in \mathcal{N}\) where \(E, L \in \mathbb{N}_+\) with \(\min_{k \in \mathcal{K}} e_k / \Delta - 1 < E \leq \min_{k \in \mathcal{K}} e_k / \Delta\) and \(\max_{k \in \mathcal{K}} l_k / \Delta \leq L < \max_{k \in \mathcal{K}} l_k / \Delta + 1\). In the time-expanded network \(\mathcal{D}^\Delta\), for every pair of nodes \((i, t)\) and \((j, \bar{t})\) where \((i, j) \in \mathcal{A}\) and \((\bar{t} - \bar{\tau}_u) - (t + \bar{\tau}_i) \geq \tau_{ij}\), there is a movement arc \(((i, t), (j, \bar{t}))\) in \(\mathcal{A}\), i.e., we consider all possible loading and unloading time options for every possible vehicle travel option. To handle the discretization of time, we adopt a standard mapping that rounds up travel times, rounds up loading and unloading times, rounds up commodity availability times, and rounds down times commodity due times, so that a feasible solution to the service network design model on the time-expanded network \(\mathcal{D}^\Delta\) can always be converted to a feasible schedule in continuous time.

We denote the service network design problem with hub capacities defined on the time-expanded network \(\mathcal{D}^\Delta\) described above by SNDHC(\(\mathcal{D}\)). Let \(y_{ij}^t\) represent the number of times arc \((i, j)\) is used to accommodate dispatches that start loading at hub \(i\) at time \(t\) and finish unloading at hub \(j\) at time \(\bar{t}\). Because multiple vehicles can perform the same movement, the \(y_{ij}^t\) variables have to be integer. Let \(x_{ij}^{kt}\) represent whether commodity \(k \in \mathcal{K}\) travels on movement arc \(((i, t), (j, \bar{t}))\). For convenience, let \(\mathcal{A}^k\) and \(\mathcal{H}^k\) denote the movement and holding arc sets containing movement and holding arcs that can possibly be used by commodity \(k \in \mathcal{K}\), respectively. Let \(\mathcal{N}'\) be a timed node set that contains a node \((i, t)\) for each node \(i \in \mathcal{N}\) and \(t \in T_i'\) where \(T_i' = \{\min_{k \in \mathcal{K}} e_k, \min_{k \in \mathcal{K}} e_k + 1, \ldots, \max_{k \in \mathcal{K}} l_k\}\). Because each commodity must follow a single path from its origin to its destination, the \(x_{ij}^{kt}\) variables have to be binary. Finally, let \(z_k\) be a binary variable indicating whether commodity \(k \in \mathcal{K}\) is served or not. With these variables, SNDHC(\(\mathcal{D}\)) can be formulated as follows:

\[
\begin{align*}
\max \sum_{k \in \mathcal{K}} z_k & \quad \text{(Phase 1)} \\
\min \sum_{((i, t), (j, \bar{t})) \in \mathcal{A}} c_{ij} y_{ij}^t & \quad \text{(Phase 2)} \\
\text{s.t.} & \\
\end{align*}
\]
\[
\sum_{(i,t),(j,\bar{t}) \in H^k \cup A^k} x_{ij}^{kt} - \sum_{(j,\bar{t}),(i,t) \in H^k \cup A^k} x_{ji}^{kt} = \begin{cases} 
+z_k & (i,t) = (o_k, e_k), \\
-z_k & (i,t) = (d_k, l_k), \forall k \in K, (i,t) \in N; \\
0 & \text{otherwise}; 
\end{cases} 
\] (1a)

\[
\sum_{k \in K} q_k x_{ij}^{kt} \leq Q y_{ij}^{t} \quad \forall ((i,t),(j,\bar{t})) \in A; 
\] (1b)

\[
\sum_{((i,s),(j,t)) \in A_{t-\tau } \leq s \leq t} y_{is}^{s} \leq L_i \quad \forall (i,t) \in N'; 
\] (1c)

\[
\sum_{((i,s),(j,t)) \in A_{t} \leq s < t+\tau } y_{is}^{s} \leq U_j \quad \forall (j,\bar{t}) \in N'; 
\] (1d)

\[
x_{ij}^{kt} \in \{0,1\} \quad \forall ((i,t),(j,\bar{t})) \in A^k \cup H^k \quad \forall k \in K; 
\] (1e)

\[
y_{ij}^{t} \in \mathbb{N}_\geq 0 \quad \forall ((i,t),(j,\bar{t})) \in A; 
\] (1f)

That is, SNDHC(D) problem seeks to maximize the total number of commodities to be served on time in Phase 1, and to minimize the total travel cost in Phase 2 while ensuring the maximum number of served commodities. Constraints (1a) ensure that each served commodity departs from its origin after it becomes available and arrives at its destination before it is due. The presence of holding arcs allows a commodity to arrive early at its destination or depart late from its origin. Constraints (1b) ensure that a sufficient number of vehicles is available for the commodities that are sent from hub i to hub j starting loading at time t and completing unloading at time \( \bar{t} \). Constraints (1c) and (1d) ensure that the number of vehicles that are loading or unloading simultaneously does not exceed the specified hub capacity limits across all locations at any time during the planning horizon. Constraints (1e) and (1f) define the variables and their domains.

Since there is a movement arc \( ((i,t),(j,\bar{t})) \in A \) for every pair of nodes \( (i,t) \) and \( (j,\bar{t}) \) with \( (i,j) \in A \) and \( (\bar{t} - w_u)-(t + w_l) \geq \tau_{ij} \), the number of variables \( x_{ij}^{kt} \) and \( y_{ij}^{t} \) is huge; prohibitively large for real-life instances. Furthermore, to ensure the hub capacity restrictions across all locations and at any time of the planning horizon, the number of loading and unloading capacity constraints is huge too; prohibitively large for real-life instances. Therefore, solving SNDHC(D), or even obtaining high-quality solutions, using a standard commercial solver is practically impossible, which motivates the development of heuristics.

4. Methodology

In this section, we start by introducing two heuristic approaches for solving SNDHC(D), one IP-based heuristic and one metaheuristic. Then, we present a hybrid matheuristic that takes advantage of the strengths of both heuristics.
4.1. An IP-based heuristic

Our IP-based heuristic (IP-H) for obtaining a high-quality solution solves multiple small IPs, derived from the original IP, which greatly reduces the computational effort. IP-H solves an IP for every hub that serves as origin or destination for at least one commodity in a pre-specified sequence. Let \( h \in N \) be the hub under consideration. Let \( K_h^1 \subseteq K \) denote the set of commodities for which \( h \) is either the origin or the destination, i.e., \( K_h^1 = \bigcup_{k \in K_0} k \cup \bigcup_{k \in K} d_k = h \). Let \( K_h^2 \subseteq K \setminus K_h^1 \) denote the set of commodities (for which \( h \) is not the origin or the destination) which have been assigned a feasible path when the previous hub in the sequence was considered (if \( h \) is the first hub in the sequence, then \( K_h^2 = \emptyset \)). The set of commodities considered for hub \( h \) is \( K_h = K_h^1 \cup K_h^2 \). We formulate an arc-based hierarchical model (the small IP) that seeks to serve as many commodities \( k \in K_h \) as possible, i.e., seeks a movement path \( P^k = (a_1 = ((i_1, t_1), (j_1, \bar{t}_1)), a_2 = ((i_2, t_2), (j_2, \bar{t}_2)), \ldots, a_g = ((i_g, t_g), (j_g, \bar{t}_g))) \) for each commodity \( k \in K_h \) to be served with timed arc \( a_p \in A \cup H \), \( (j_p, \bar{t}_p) = (i_{p+1}, t_{p+1}) \) for \( p = 1, \ldots, g-1 \), \( (i_1, t_1) = (o_k, e_k) \) and \( (j_g, \bar{t}_g) = (d_k, l_k) \). The associated geographic path is \( P^k = (a_1 = (i_1, j_1), a_2 = (i_2, j_2), \ldots, a_g = (i_g, j_g)) \) with \( a_p \in A \), \( j_p = i_{p+1} \) for \( p = 1, \ldots, g-1 \), \( i_1 = o_k \) and \( j_g = d_k \). For notational convenience, we let \( P^{k'} \) denote the geographic path that is assigned to commodity \( k \in K_h^2 \) when processing the previous hub in the sequence (if hub \( h \) is not the first hub in the sequence).

Commodities in \( K_h^1 \) are allowed to follow any feasible timed path in the network, but commodities in \( K_h^2 \) are forced to follow either the previously assigned path (but possibly a different timed copy) or the direct path from the commodity’s origin to its destination. For each commodity \( k \in K_h \), we denote the set of feasible timed arcs by \( A^{hk} \) and the set of feasible holding arcs by \( H^{hk} \). Let \( A^h = \bigcup_{k \in K_h} A^{hk} \) and \( H^h = \bigcup_{k \in K_h} H^{hk} \). Let \( z_k \) for \( k \in K_h \) be a binary decision variable denoting whether commodity \( k \) is served in the optimization problem for hub \( h \). In Phase 1, we maximize the number of commodities served (from among the commodities in \( K_h \)). In Phase 2, we minimize the total travel distance, while enforcing that the number of served commodities is the number of commodities found in Phase 1. This hierarchical optimization approach captures the objectives of SNDHC(D), but also facilitates serving a large number of commodities as minimizing the total distance in the second phase tends to free up unloading and loading capacity for commodities that have not yet been considered (as they will only be considered when we process hubs that are later in the sequence).

The formulation of the arc-based hierarchical model is as follows:

\[
\begin{align*}
\text{max} & \quad \sum_{k \in K_h} z_k \quad \text{(hierarchical model phase 1)} \\
\text{min} & \quad \sum_{(i, t, (j, \bar{t})) \in A^h} c_{ij} x_{ij}^{k} \quad \text{(hierarchical model phase 2)}
\end{align*}
\]
s.t. \[
\sum_{(i,t),(j,\bar{t})} x^{kti}_{ij} - \sum_{(j,\bar{t}),(i,t)} x^{ktj}_{ji} = \begin{cases} 
+z_k & (i,t) = (o_k,e_k) \\
-z_k & (i,t) = (d_k,l_k) \forall k \in K_h, (i,t) \in N; \\
0 & \text{otherwise}; 
\end{cases} 
\]
\[
\sum_{k \in K_h} q_k x^{kti}_{ij} \leq Q y^{ti}_{ij} \quad \forall ((i,t),(j,\bar{t})) \in A^h; 
\]
\[
\sum_{(i,s),(j,\bar{s})} \gamma^{ss}_{ij} \leq L_i \quad \forall (i,t) \in N'; 
\]
\[
\sum_{(i,s),(j,\bar{s})} \gamma^{ss}_{ij} \leq U_j \quad \forall (j,\bar{t}) \in N'; 
\]
\[
x^{kti}_{ij} \in \{0,1\} \quad \forall ((i,t),(j,\bar{t})) \in H^{hk} \cup A^{hk} \quad \forall k \in K_h; 
\]
\[
y^{ti}_{ij} \in \mathbb{N}_{\geq 0} \quad \forall ((i,t),(j,\bar{t})) \in A^h; 
\]

Constraints (2a) ensure that each commodity departs from its origin after it becomes available and arrives at its destination before it is due. Constraints (2b) ensure that sufficient vehicle capacity is available for planned transportation activities on each arc. Constraints (2c) and (2d) ensure that hub capacities are respected at all locations. Constraints (2e) and (2f) define the variables and their domains.

Observe that at most three different (geographic) paths are assigned to a commodity \( k \in K \) when processing a hub sequence: one when processing the commodity’s origin, one when processing the commodity’s destination, and one when processing the intermediate hub on a non-directed path assigned earlier (recall that only paths with a single intermediate hub are considered).

A simple, but important observation regarding IP-H is captured in the following proposition.

**Proposition 1.** If for some hub sequence, it is not possible to serve all commodities \( k \in K_h \) for the first hub \( h \) in the sequence, then there exists no feasible solution to SNDHC(\( D \)) that serves all commodities \( k \in K \).

**Proof.** Suppose there exists a feasible solution to SNDHC(\( D \)) where all commodities \( k \in K \) are served, such that each commodity has at least one feasible timed path from its origin to destination, and restrictions including vehicle capacities and and hub capacities are all respected. By removing arcs that are only used by commodities \( k \in K \setminus K_h \) from that feasible solution, we still keep the feasible timed paths for commodities \( k \in K_h \), and all restrictions are still satisfied. Since all possible timed paths are considered for \( k \in K_h \) (the network is ‘empty’ when we process the first hub \( h \) in a sequence), the above set of timed paths should have constituted a feasible solution when we process \( h \). Therefore, we have reached a contradiction. \( \square \)

The quality of solution produced by IP-H, of course, depends on the chosen hub sequence. Preliminary computational experiments have shown that processing the hubs in nonincreasing distance from the centroid or geographic center tends to produce high-quality solutions, and this is the sequence used in our computational study.
4.2. Metaheuristic Approach

We next discuss a multi-start iterated local search (MILS) metaheuristic for solving instances of SNDHC(D). Its primary benefit is that it is faster than the IP-based heuristic. The shorter solve time allows MILS to be used for sensitivity analysis, but, maybe more importantly, it can be used to guide construction of the hub sequence used in the IP-based heuristic.

MILS focuses on maximizing the number of commodities served and only uses cost information for breaking ties. MILS has two main loops: an outer loop and an inner loop; see Algorithm 1. In the outer loop, we perform $I_{\text{MS}}^{O}$ iterations where, in each iteration, we start from a greedy initial solution $S$ and perform an iterated neighborhood search to improve it, always keeping track of the best solution found during the execution of the algorithm. After an initial solution has been constructed, an attempt is made to improve that solution by iteratively examining pairs of hubs $i$ and $j$ that either have at least one timed path that connects them, or for which there is a commodity originating at $i$ and destined for $j$ that is not served in $S$. The inner loop terminates after $I_{\text{MS}}^{I}$ consecutive iterations without an improvement.

The loading and unloading operations at a hub can be viewed as a sequence of time periods, each with a given start time. For implementation efficiency, such a sequence of time periods is stored in a binary search tree, as illustrated in Figure 1. The binary search trees allows us to quickly detect conflicts between any loading or unloading operations at a hub, which is critical to ensure that the

Algorithm 1 Multi-Start Iterated Local Search

1: input: The sets of hubs and commodities and the travel time matrix;
2: output: A set of paths $S^*$ serving a large number of commodities;
3: $S^* \leftarrow \emptyset$
4: $I_{\text{MS}} \leftarrow 0$
5: while $I_{\text{MS}} < I_{\text{MS}}^{O}$ do
6:   $S \leftarrow \text{INITIAL SOLUTION}()$; # Outer loop
7:   $\{P, R\} \leftarrow \text{RESET PROBABILITIES}()$;
8:   while termination criteria is not satisfied do
9:     for all pairs of hubs $(i, j)$ in random order do
10:    $S \leftarrow \text{NEIGHBORHOOD SEARCH}(i, j, S, P, R)$;
11:   $S \leftarrow \text{REMOVE INFEASIBLE PATHS}(S)$;
12:   $S \leftarrow \text{CONNECT PATHS}(S)$;
13:   if $c_1(S) > c_1(S^*)$ then
14:     $S^* \leftarrow S$;
15:   end if
16: end for
17: $P \leftarrow \text{UPDATE PROBABILITIES}(P, R)$;
18: end while
19: $I_{\text{MS}} \leftarrow I_{\text{MS}} + 1$;
20: end while
21: return $S^*$;
hub capacity constraints are respected. More specifically, determining whether a given loading or unloading operation (a time period with a given start time) can be added at a hub can be done in $O(\log w)$ time, where $w$ is the number of loading or unloading operations in the search tree. Updating the search tree, i.e., adding or deleting a loading or unloading operation, can also be done in $O(\log w)$ time.

A solution $S$ is represented as a set of timed paths and an assignment of commodities to these paths. Each timed path is composed of a sequence of sets of two actions: (1) a vehicle loading period $v^l_i$ (the time period representing when vehicle $v$ starts and finishes loading at location $i$) and (2) a vehicle unloading period $v^u_j$ (the time period representing when the vehicle $v$ starts and finishes unloading at location $j$. For example, a timed path $(i \rightarrow j \rightarrow k)$ is represented as the sequence $\{v^l_i, v^u_j, v^l_j, v^u_k\}$, where each element is stored in the binary search tree representing the activities at the loading or unloading dock used by the vehicle.

Next, we will describe how we obtain an initial solution in each iteration of the outer loop, the neighborhoods used in the local search procedure, and how the neighborhoods are selected in the local search procedure.

4.2.1. Initial solution construction. We start by listing all possible pairs of hubs $(i, j)$ with $i$ representing the origin of a commodity and $j$ representing its destination, respectively, and partition them into three groups, $G_1$, $G_2$ and $G_3$, based on the travel time between the hubs. The travel time is short for pairs in $G_1$, moderate for pairs in $G_2$, and long for pairs in $G_3$. We start constructing the initial solution $S$ by processing the commodities associated with pairs in $G_1$ in random order, one by one. Given a pair $(i, j) \in G_1$, we try to create feasible timed paths for all commodities originating at $i$ and destined to $j$ (there can be more than one such commodity because of the different service classes) departing as early as possible from $i$, always respecting hub capacities and the available time at the origin and the due time at the destination. If a feasible
timed path is identified, then \((i \rightarrow j)\) is added to the list of timed paths. Before considering the
next pair, we check if the path \((i \rightarrow j)\) enables the creation of additional paths, i.e., if a timed
path \((j \rightarrow k)\) exists and the time path \((i \rightarrow j \rightarrow k)\) is feasible, then \((i \rightarrow j \rightarrow k)\) is created as it can
potentially serve commodities originating at \(i\) and destined to \(k\). Once all pairs in \(G_1\) are processed,
we repeat the same procedure for the pairs in \(G_2\) and \(G_3\).

4.2.2. Local search. MILS employs several neighborhoods to try and improve a given solu-
tion, each involving simple modifications of timed paths or of loading and unloading sequences.
The exploration of the neighborhoods is exhaustive, i.e., the solution modifications or moves are
performed for all hub pairs \((i, j)\), in random order. Whenever a pair of hubs \((i, j)\) is selected in
the neighborhood search, we first check if all commodities originating from \(i\) and destined for \(j\)
are served. In case there is a commodity that is not served, we first try to create a direct timed
path from \(i\) to \(j\) at the earliest possible departure time from \(i\). If no direct timed path is possible,
we try to create an indirect timed path \((i \rightarrow k \rightarrow j)\), where \(k\) is chosen randomly from among the
hubs that are within given distance from either \(i\) or \(j\). Specifically, we check if there exists a timed
path \((k \rightarrow j)\) that arrives at \(j\) before the due time of the commodity and if it is possible to send
a vehicle from \(i\) to \(k\) such that it arrives at \(k\) in time, i.e., we only consider creating a new timed
path from \(i\) to \(k\). In case all commodities are served, or, if we are unable to create paths for all the
commodities that are not yet served, we apply one of the following neighborhoods:

- **MoveLoadingPeriod**(i,j) and **MoveUnloadingPeriod**(i,j): Each neighborhood tries to
delay or advance (chosen with equal probability) the loading or unloading periods of a timed
path \((i \rightarrow j)\) as much as possible. New docks may be assigned at either \(i\) or \(j\), if beneficial.

- **ChangeLoadingDock**(i,j) and **ChangeUnloadingDock**(i,j): Each neighborhood tries to
change the loading or unloading dock of a timed path \((i \rightarrow j)\) in order to create space at the
dock to add new loading or unloading operations (which can then be used for other timed
paths). The start time of the loading or unloading operation is kept the same.

- **SwitchLoadingPeriod**(i,j): The neighborhood tries to switch the loading operation of a
direct timed path \((i \rightarrow j)\) with the loading operation of another, randomly selected, direct
timed path \((i \rightarrow k)\) at the loading dock. New unloading times and docks may be assigned to
the vehicles at both \(j\) and \(k\), if necessary. The switch is only performed if it does not affect
the feasibility of the commodities assigned to the timed paths.

- **CrossoverPaths**(i,j): Let \((i \rightarrow k \rightarrow j)\) and \((l \rightarrow m \rightarrow j)\) be two distinct timed paths in \(S\).
The neighborhood generates new paths by crossing over \((i \rightarrow k \rightarrow j)\) and \((l \rightarrow m \rightarrow j)\) into
two new paths: \((i \rightarrow m \rightarrow j)\) and \((l \rightarrow k \rightarrow j)\). The crossover operation is only applied if the
new paths are feasible and do not violate the loading/unloading capacity constraints at the related hubs.

- **SwitchDirectPath**(i,j): The neighborhood tries to reassign the commodities on the direct timed path \((i \rightarrow j)\) to an indirect timed path \((i \rightarrow k \rightarrow j)\), where \(k\) is randomly selected. If all commodities can (feasibly) be reassigned, then \((i \rightarrow j)\) is removed from \(S\), hence opening space for a new loading operation at hub \(i\) and a new unloading at hub \(j\).

- **Re-ArrangeLoadingPeriods**(i): Consider the hubs \(i, j\) and \(k\) and the movement arcs \(u, v,\) and \(w\) shown in Figure [2a]. Each movement arc reflects the time that the loading starts and the time that the unloading finishes. The accompanying rectangles represent the feasible times at which the loading can start, which are determined by the latest time that one of the commodities transported becomes available at \(i\) and the departure time at \(i\) that ensures an arrival at the time unloading starts. Next, assume that there is a commodity that originates at \(k\) and is destined for \(j\). In this example, it is possible to create an indirect timed path \((k \rightarrow i \rightarrow j)\) by starting the loading of \(v\) earlier and starting the loading of \(w\) later (but the start times remain within their respective windows, as shown in Figure [2b].

This neighborhood attempts to re-organize the loading operations at the loading dock of hub \(i\) by solving a small integer program, which seeks to maximize the number of potential new indirect timed paths for commodities not yet served while ensuring that no existing timed paths become infeasible. For a given movement arc \(w\) originating at hub \(i\), let \(d_w\) denote the destination, let \(e_w\) denote the earliest time that loading of \(w\) can start, let \(l_w\) denote the latest time that loading of \(w\) can start, and let \(s_w\) be a variable that represents the time that the loading of \(w\) starts. Finally, let there be a commodity that originates at \(k\) and is destined for \(d_w\) that is not yet served and a vehicle that originates at \(k\) and finishes unloading at \(i\) at time \(a_{k,d_w}\). We solve the following IP:

\[
\begin{align*}
\max & \sum_{k,d_w} \lambda_{k,d_w} \\
\text{s.t} & \\
& s_w + \bar{\tau} \leq s_y + M \cdot (1 - \delta_{wy}) \quad \forall w, y \in \mathcal{W}, w \neq y \quad (3a) \\
& \delta_{wy} + \delta_{yw} = 1 \quad \forall w, y \in \mathcal{W} \quad (3b) \\
& e_w \leq s_w \leq l_w \quad \forall w \in \mathcal{W} \quad (3c) \\
& s_w \geq a_{k,d_w} \lambda_{k,d_w} \quad \forall w \in \mathcal{W}, \forall (k,d_w) \quad (3d) \\
& \lambda_k \in \{0, 1\} \quad \forall k \quad (3e) \\
& \delta_{wy} \in \{0, 1\} \quad \forall w, y \in \mathcal{W} \quad (3f) \\
& s_w \in \mathbb{R}_+ \quad \forall w \in \mathcal{W} \quad (3g)
\end{align*}
\]
(a) Initial state of the loading operations (large rectangles) \( w,v \) and \( u \) in hubs \( i \) and \( k \), respectively, and the associated time windows \([e,l]\) where \( w \) and \( v \) can start loading.

In this example, there is an unserved timed commodity path from \( k \) to \( j \).

(b) Final state of the loading operations in hub \( i \) after the neighborhood move is applied. It is possible to create the indirect timed path \((k \to i \to j)\) by starting the loading of \( v \) earlier and starting the loading of \( w \) later, while still respecting the associated time windows.

**Figure 2** Initial and final states of the loading operations at a hub after the neighborhood move Re-ArrangeLoadingPeriods is applied.

where \( \mathcal{W} \) is the set of movement arcs originating at \( i \), \( \lambda_{k,d,w} \) is a binary variable indicating whether or not a new indirect timed path can be created to served a commodity originating at \( k \) and destined for \( d_w \), \( \delta_{wy} \) is a binary variable indicating whether or not the start of the loading of \( w \) precedes the start of the loading of \( y \), and \( M \) is a large constant. Constraints (3a) and (3b) ensure that there are no overlaps between loading periods, while constraint (3c) ensure that no existing paths become infeasible, and constraint (3d) links the decision variables \( \lambda_{k,d,w} \) and \( s_w \). Finally, constraints (3e),(3f) and (3g) define the decision variables and their domains.
- **Re-ArrangeUnloadingPeriods(i):** This neighborhood is similar to the previous one, but now focused on re-arranging the unloading operations at an unloading dock of hub $i$.

- **OpenNewLoadingPeriod(i):** The neighborhood tries to re-arrange the loading operations at a loading dock of hub $i$, again using a small integer program, now with the goal of opening space for a new loading operation at the dock. The integer program is similar to the one presented in **Re-ArrangeLoadingPeriods**. It has an additional decision variable representing the start time of the new loading operation and additional constraints that link the variable to the latest possible time of the start of a loading operation for a commodity that is not yet served. If new indirect timed paths are possible for more than one such commodity, a single new direct timed path is chosen randomly with equal probability.

- **OpenNewUnloadingPeriod(i):** This neighborhood is similar to the previous one, but now focused on opening space for a new unloading operation at an unloading dock of hub $i$.

As the neighborhood moves, which are designed to find timed paths for commodities that currently do not yet have a timed path, may also result in destruction of existing timed paths before carrying out a move, we examine the difference between the number of commodities with a timed path before and after the move is performed. If the number of commodities with a timed path remains the same or increases, the move is performed, but if the number of commodities with a timed path decreases, but not by more than a threshold $T_k$, it is carried out with probability $p$. Finally, after performing a move, we first remove the indirect timed paths that have become infeasible, using **RemoveInfeasiblePaths($S$)**, and then try to create new indirect timed paths, using **ConnectPaths($S$)**.

### 4.2.3. Neighborhood selection.

The probability that a neighborhood is selected depends on how many timed paths are added/removed by the neighborhood throughout the execution of the metaheuristic. The more successful a particular neighborhood is in creating new timed paths, the higher the chance this neighborhood will be chosen in subsequent iterations. Let $P_n$ be the current probability of selecting neighborhood $n$ and let $R_n$ be the current reward associated to $n$, i.e., the difference between the total number of timed paths added and removed when using neighborhood $n$ in past iterations. When **UpdateProbabilities($P,R$)** is called, the selection probabilities associated with the neighborhoods are updated, as shown in Algorithm 2, where $N$ is the number of neighborhoods and $\eta$ is a parameter determining how aggressively we increase or decrease the probabilities based on the rewards vector. Before entering the inner loop, **ResetProbabilities()** resets all probabilities and rewards by setting $\{P_n = \frac{1}{N}, R_n = 0\} \forall n \in N$. In order to maintain an effective level of diversification, we reset all probabilities to the discrete uniform distribution $\mathcal{U}(1, N)$ if no improvements were made to the current solution for $\mathcal{U}_R$ consecutive iterations.
Algorithm 2 Update neighborhood selection probabilities

1: **input:** The probability and the rewards vectors $P$ and $R$
2: **output:** The updated probability vector $\tilde{P}$
3: $\tilde{P} \leftarrow P$
4: $W \leftarrow 0$
5: for $n = 1, \cdots, N$ do
6: \hspace{.5cm} $\tilde{P}_n \leftarrow P_n \cdot e^{n \eta R_n}$
7: \hspace{.5cm} $W \leftarrow W + \tilde{P}_n$
8: end for
9: for $n = 1, \cdots, N$ do
10: \hspace{.5cm} $\tilde{P}_n \leftarrow \frac{\tilde{P}_n}{W}$
11: end for
12: return $\tilde{P}$;

4.3. Hybrid Matheuristic

Given that the IP-heuristic presented above processes hubs in a given sequence, it is obvious that this sequence has an impact on the quality of the resulting solution. Therefore, in this section, we propose a hybrid matheuristic (H-MAT), in which solutions produced by the metaheuristic are used to guide the IP-heuristic. More specifically, the solutions produced by the metaheuristic are used to (1) adjust the sequence in which the hubs are processed, and (2) adjust the (geographic) paths options considered for commodities $k \in K^2_h$ when a hub is processed.

To control the solution time, we divide the processing of hubs in stages. There will be $G$ stages and each stage involves the processing of a certain number of hubs. The “staging” can be represented by a vector $(m_1, m_2, \ldots, m_G)$ with $m_g > 0$ and $\sum_{g=1}^{G} m_g = |N|$, and $m_g$ representing the number of hubs processed in Stage $g$. In Stage $g$, we start by running MILS, where we enforce that commodities that are served in the latest solution (obtained when processing the last hub in Stage $g-1$) have to remain served, and recording the commodities that are served in its solution. Then, the hubs are sorted in nondecreasing order of the number of commodities served that originate from or are destined to the hub, and the first $m_g$ as-yet unprocessed hubs are selected to be processed (one by one). When processing hub $h$, each commodity $k \in K^1_h$ is allowed to follow any feasible timed path, and each commodity $k \in K^2_h$ is forced to follow either (1) the direct path from the commodity’s origin $o_k$ to its destination $d_k$, or (2) the previously assigned geographic path $P^{k'}$, or (3) the geographic path in the MILS solution. The arc sets $A^{hk}$ and $H^{hk}$ for $k \in K_h$ and $A^h$ and $H^h$ are defined as before. Algorithm 3 shows the pseudo-code for H-MAT.

The advantage of H-MAT over IP-H is that the sequence in which the hubs are processed is determined dynamically and informed by the solution produced by MILS and that for commodities $k \in K^2_h$ another geographic path is considered (the geographic path it uses in the MILS solution).
Algorithm 3 Hybrid Matheuristic

1: **input:** The sets of hubs, commodities, the travel time/distances between hubs, the number of stages $|G|$ and parameter vector $M$;
2: **output:** The set of timed paths $S^*$;
3: **while** not all hubs $i \in N$ have been processed **do**
4: Run Algorithm 1 while making sure that previously served commodities remain served, and record served commodities and their assigned geographic paths in the updated best MILS result;
5: Sort hubs based on the number of served commodities that origin from or destine to the hub in the updated best MILS result (ascending);
6: Choose the first $m_g \in M$ unprocessed hubs in the sort as the next $m_g \in M$ hubs in the hub sequence $\bar{N}$ to process;
7: **for** hub $h$ in the new chosen $m_g$ hubs **do**
8: identify commodity set $K_h$;
9: identify integrated arc set $A^{hk}$ and holding arc set $H^{hk}$ for every $k \in K_h$;
10: solve the arc-based hierarchical model;
11: end for
12: $g = g + 1$
13: **end while**
14: return $S^*$;

5. **Computational Results**

5.1. **Instances**

The proposed algorithms were used to solve real-world instances of SF Express, each representing same-day delivery service offerings in one of China’s mega-cities. A representative day was used to define the demand to be served by a fleet of homogeneous vehicles, each with a capacity of 400 packages. The demand consists of a set of commodities, each specifying an origin hub, a destination hub, a number of packages, the time the packages are available at the origin hub, and the time the packages are due at the destination hub. Packages that have the same origin and destination hub, but that become available at different times or that have different due times are considered to be different commodities; this happens when more than one service product is offered in a market (e.g., “12-18 service” and “14-20 service”). For each hub a limit on the maximum number of vehicles that can be simultaneously loaded and that can be simultaneously unloaded are given. A commodity will be loaded every time it departs from a hub and unloaded every time it arrives at a hub; each loading or unloading operation takes 10 minutes. Table I summarizes the characteristics of the four instances. For each instance, the demand information includes the number of commodities to be served ($|K|$), the length (hours) of the planning period ($|T|$), i.e., $|\max_{k \in K} l_k - \min_{k \in K} e_k|$, the average and standard deviation of the number of packages per commodity ($\bar{q}_k$, $\tilde{q}_k$), the average and standard deviation of the time that commodities become available ($\bar{e}_k$, $\tilde{e}_k$), the average and standard deviation of the time commodities are due ($\bar{l}_k$, $\tilde{l}_k$), the average and standard deviation of the direct travel time (minutes) for the commodities ($\bar{\tau}_k$, $\tilde{\tau}_k$), and an upper bound on the number
of commodities that can be served (\(|K'|\)), obtained using Proposition 1. For each instance, the network information includes the number of hubs (\(|N|\)), the average and standard deviation of the travel time (minutes) and the distance (kilometers) between hubs (\(\bar{\tau}_{ij}, \tilde{\tau}_{ij}, \bar{c}_{ij}, \tilde{c}_{ij}\)), and the average and standard deviation of the unloading and loading capacity of the hubs (\(\bar{U}_i, \tilde{U}_i, \bar{L}_i, \tilde{L}_i\)). Instance C and D correspond to representative days of different seasons in the same city.

| Instance | \(|K|\) | \(|T|\) | \(q_k\) | \(\tilde{q}_k\) | \(\bar{c}_k\) | \(\tilde{c}_k\) | \(I_k\) | \(\bar{I}_k\) | \(\tilde{I}_k\) | \(|K'|\) | \(|N|\) | \(\bar{\tau}_{ij}\) | \(\tilde{\tau}_{ij}\) | \(\bar{c}_{ij}\) | \(\tilde{c}_{ij}\) | \(U_i\) | \(\tilde{U}_i\) | \(L_i\) | \(\tilde{L}_i\) |
|----------|--------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| A        | 270    | 2.5   | 32     | 28     | 13:09  | 15     | 15:02  | 8      | 43     | 13     | 264    | 17     | 43     | 14     | 25     | 13     | 4      | 0      | 3      | 0      |
| B        | 986    | 1.75  | 34     | 32     | 13:15  | 0      | 15:12  | 15     | 44     | 13     | 980    | 32     | 44     | 13     | 26     | 12     | 5      | 2      | 4      | 3      |
| C        | 1291   | 5.6   | 29     | 20     | 13:38  | 55     | 16:58  | 58     | 60     | 22     | 1286   | 31     | 58     | 23     | 29     | 16     | 3      | 0      | 3      | 0      |
| D        | 1779   | 5.6   | 29     | 19     | 13:38  | 55     | 16:56  | 58     | 56     | 21     | 1774   | 31     | 58     | 23     | 29     | 16     | 3      | 0      | 3      | 0      |

5.2. Analysis

We compare the performance of the proposed algorithms, i.e., the IP-based heuristic (IP-H), the metaheuristic (MILS) and the hybrid matheuristic (H-MAT), on the set of instances described in Section 5.1. IP-H was coded in Python and uses Gurobi 8.1.1 to solve integer programs. MILS was coded in C++ and uses IBM CPLEX Optimizer 12.6 to solve integer programs. All experiments were performed in a single thread of a dedicated Intel Xeon ES-2630 2.3GHz processor with 50GB RAM running Red Hat Enterprise Linux Server 7.4.

Preliminary computational experiments were used to determine values for the parameters of MILS that ensure good quality solutions are found in under three hours for the larger instances. Based on these experiments, we set \(I_{t_{\text{max}}}^O = 10, I_{t_{\text{max}}}^I = 500, T_k = -1, p = 0.05, I_{t_{\text{R}}}^I = 100\) and \(\eta = 0.005\). The maximum time allowed for solving the integer programs was set to five seconds and the incumbent solution is discarded when the time limit is exceeded and when the integrality gap is more than 10%. The maximum distance for choosing an intermediate hub was set to \(\frac{3}{4}\) of the average distance between hubs. The same set of parameter values was used when running MILS within H-MAT.

Geographic paths for commodities are allowed to use at most one intermediate hub. The integer programs solved by IP-H and H-MAT are based on a time-expanded network with a homogeneous time discretization of 2 minutes. To control the number of variables in the integer programs solved by IP-H and H-MAT, the maximum waiting time embedded in a movement arc is 15 minutes for instances A and B and 10 minutes for instances C and D. Moreover, a maximum solution time of 4 hours is imposed for each phase of the hierarchical optimization and in the second phase we seek a solution that is within 5% of optimality. MILS uses a time discretization of 1 minute.
The stages in H-MAT are defined by vector \((4, 4, 3, 3, 3)\) for instance A, \((7, 7, 6, 6, 6)\) for instance B, and \((7, 6, 6, 6, 6)\) for instances C and D. The reason for using five stages is to control the overall solution time (by invoking MILS only five times). We process fewer hubs in the later stages because finding a timed path for a commodity is more difficult towards the end and we benefit from more frequent adjustment of the geographic paths considered for commodities.

In order to assess the performance of the proposed algorithms, we report the following statistics for the solutions produced by each algorithm: the number of commodities served \(|K_S|\), the total travel cost \(C\), the number of commodities served using direct and indirect paths \(|K^d_S|, |K^i_S|\), the average number of segments per path for commodities served \(|G^S|\), the average travel time (minutes) of the indirect paths for commodities served \(\tau^i_S\), the average direct travel time for commodities served \(\tau^d_S\), the average direct travel time for commodities not served \(\tau^d_U\), and the solve time (hours) \(TT\).

| Instance | Algorithm | \(|K^S|\) | \(|K^d_S|\) | \(|K^i_S|\) | \(|G^S|\) | \(\tau^i_S\) | \(\tau^d_S\) | \(\tau^d_U\) | \(TT\) |
|----------|-----------|---------|---------|---------|---------|---------|---------|---------|-------|
| A        | MILS      | 264     | 143     | 121     | 1.46    | 65.21   | 45.03   | 58.33   | 0.21  |
|          | IP-H      | 264     | 149     | 115     | 1.44    | 62.03   | 42.89   | 60.67   | 0.38  |
|          | H-MAT     | 264     | 144     | 120     | 1.45    | 63.58   | 42.96   | 60.17   | 1.56  |
| B        | MILS      | 965     | 395     | 570     | 1.59    | 62.52   | 43.10   | 55.52   | 1.43  |
|          | IP-H      | 971     | 542     | 429     | 1.45    | 63.76   | 41.88   | 65.53   | 15.78 |
|          | H-MAT     | 975     | 498     | 477     | 1.49    | 63.18   | 41.35   | 64.36   | 41.50 |
| C        | MILS      | 1233    | 606     | 627     | 1.51    | 79.80   | 58.15   | 67.98   | 2.64  |
|          | IP-H      | 1275    | 744     | 531     | 1.42    | 78.22   | 58.10   | 75.38   | 38.08 |
|          | H-MAT     | 1282    | 668     | 614     | 1.48    | 79.58   | 56.90   | 79.11   | 71.56 |
| D        | MILS      | 1626    | 890     | 736     | 1.45    | 76.20   | 52.95   | 55.96   | 3.00  |
|          | IP-H      | 1674    | 884     | 790     | 1.47    | 75.14   | 50.90   | 70.54   | 118.93|
|          | H-MAT     | 1689    | 851     | 838     | 1.50    | 74.20   | 51.29   | 69.92   | 174.91|

The results can be found in Table 2. We observe that MILS is significantly faster than IP-H and H-MAT, but the efficiency comes at the price of serving fewer commodities. H-MAT produces the best solutions, but the quality comes at the price of taking a long time (175 hours for Instance D). Interestingly, H-MAT produces solutions in which the largest number of commodities is served, but also with a low cost; the cost is always less than the cost of the solutions produced by IP-H and also less than the cost of the solutions produced by MILS for Instance A and Instance D.

We also see that in the best solutions most commodities that are served are served using a direct path. The tight service constraints, i.e., the difference between due time and available time, implies that relatively few indirect paths, which involve one additional unloading operation and one additional loading operation, and, potentially, additional waiting time at the intermediate hub, are feasible. As expected, the average direct travel time of the commodities that are not served is high compared to the average direct travel time of the commodities that are served and the average
travel time between hubs. Here too, this is due to the tight service constraints, as this implies that (too) few options are available for these commodities.

Next, we examine the differences in the solutions produced by the three algorithms from a capacity utilization perspective. Figure 3 shows the average utilization of loading (or unloading) capacity across the hubs at different times during the planning period, where the capacity utilization at a point in time is computed as the number of docks occupied at that time divided by the number of docks available at that time.

The most striking feature of the graphs shown is the difference between instances A and B and instances C and D. For Instance C and D, with a relatively long planning period, we see two loading and two unloading peaks, clearly reflecting the use of indirect timed paths, i.e., the use of hubs to transfer packages. For Instance A and B, with a relatively short planning period, we see that it is critical to get packages moving as soon as possible and the loading capacity in the early part of the planning period is almost fully utilized, and that we need all the available time as the unloading capacity in the late part of the planning period is almost fully utilized. We also observe that the solution produced by H-MAT utilizes less loading and unloading capacities than the solution produced by IP-H, while serving more commodities, which indicates that hub capacities are used more effectively.

Many interacting factors affect our ability to serve the commodities and understanding which of these factors more strongly impact our ability to serve commodities is informative and valuable. MILS is most suitable for carrying out such an investigation as it is more efficient. More specifically, we explore three scenarios, in which we relax one factor: (i) we reduce the loading and unloading time of a vehicle from 10 to 8 minutes (RedLoadUnloadTime), (ii) we increase the due time of each commodity by 10 minutes (IncrDueTime), and (iii) we modify the loading and unloading capacity at each hub during the first and last ten minutes of the planning horizon by making all docks available for loading in the first ten minutes and making all docks available for unloading in the last ten minutes (AdjLoadUnloadCap).

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Sensitivity Analysis for Operational Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td>Setting</td>
</tr>
<tr>
<td>B</td>
<td>DEFAULT</td>
</tr>
<tr>
<td></td>
<td>RedLoadUnloadTime</td>
</tr>
<tr>
<td></td>
<td>IncrDueTime</td>
</tr>
<tr>
<td></td>
<td>AdjLoadUnloadCap</td>
</tr>
<tr>
<td>C</td>
<td>DEFAULT</td>
</tr>
<tr>
<td></td>
<td>RedLoadUnloadTime</td>
</tr>
<tr>
<td></td>
<td>IncrDueTime</td>
</tr>
<tr>
<td></td>
<td>AdjLoadUnloadCap</td>
</tr>
</tbody>
</table>

The results can be found in Table 3. As expected, reducing the loading and unloading times of vehicles, adjusting the loading and unloading capacities at hubs, and extending the due time of
Figure 3  Average hub capacity utilization.
commodities all enable more commodities to be served. For Instance B, we are not only able to serve more commodities, but we are also able to do it at less cost. For Instance C, we see that minimally adjusting loading and unloading capacities at the hubs (only in the first and last ten minutes) allows the maximum number of commodities to be served, mostly because it allows many more commodities to be served using a direct path. Recall to that MILS always tries to create a direct path between pairs of hubs first, and only if a direct path is not possible due to the hub capacity constraints, it tries to create an indirect path. For this instance, we also see that serving more commodities comes at a significant cost; an increase of less than 5% in the number of commodities served comes at an increase in cost of more than 13%. Overall, these results demonstrate that serving all commodities is very challenging when hub capacity constraints are tight.

5.3. Service network design with hub capacities in practice

In this section, we introduce the variant of SNDHC used at SF Express in their search for a practically viable solution. This variant ensures that a solution that serves all commodities is obtained (a hard business requirement) and incorporates several other relevant practical considerations. First and foremost, because a solution serving all commodities always exists when there are no hub capacity constraints, we switch to minimizing a weighted combination of transportation costs and a penalty for violating hub capacity constraints, i.e., we change the hub capacities into soft constraints with a penalty term in the objective for exceeding the hub capacity. Furthermore, the maximum amount of product transshipped at hub is restricted, as the larger this amount is the larger the chance of misoperations. Also, to avoid, as much as possible, any issues arising from limited parking capacity at a hub, a commodity is not allowed to wait at an intermediate hub. As before, an indirect path contains exactly one intermediate hub.

Let $\bar{e}_k$ be a parameter that indicates the latest possible load start time for commodity $k$ at its origin $o_k$. To generate the feasible direct timed paths, we consider all movement arcs $((o_k,t),(d_k,\bar{t})) \in A$ where $e_k \leq t \leq \bar{e}_k$ and $\bar{t} \leq l_k$. Let $N(i,\kappa)$ and $N^k_{\text{int}}(\kappa)$ denote the set of $\kappa$ nearest hubs to hub $i \in N$ and the set of potential intermediate hubs for commodity $k \in K$, respectively, with $N^k_{\text{int}}(\kappa) = N(o_k,\kappa) \cup N(d_k,\kappa)$. To generate the feasible indirect timed paths, we consider all pairs of movement arcs $((o_k,t),(i,t'))$ and $((i,t'+\bar{\tau}_l+\bar{\tau}_u),(d_k,\bar{t}))$ such that $i \in N^k_{\text{int}}(\kappa)$, $e_k \leq t \leq \bar{e}_k$ and $\bar{t} \leq l_k$.

Let $P^k$ denote the set of timed paths for commodity $k \in K$ with $P^k_{\text{ind}} \subseteq P^k$ the set of indirect timed paths for commodity $k \in K$. Let $A^P^k \subseteq A$ denote the set of movement arcs $((i,t),(j,\bar{t}))$ in timed path $P^k \in P^k$ and let $H^{P^k} \subseteq H$ denote the holding arcs in timed path $P^k \in P^k$. Furthermore, for each movement arc $((i,t),(j,\bar{t})) \in A$, let $K^t_{ij}$ be the set of commodities that have at least one timed path traversing arc $((i,t),(j,\bar{t}))$ and $P^k_{ij} \subseteq P^k$ be the set of timed paths including movement arc $((i,t),(j,\bar{t})) \in A$. 
Let \( v_k^p \) be a binary variable indicating whether or not commodity \( k \in K \) uses timed path \( P^k \in \mathcal{P}^k \), and \( S^t_{i, \text{L}} \) and \( S^t_{i, \text{U}} \) be continuous variables denoting the violation of the loading and unloading capacities at hub \( i \in N \) and time \( t \in T_i \), respectively. Based on the enumerated candidate path sets, we use a path-based formulation to model this variant and solve it via calling a commercial solver. Because loading and unloading capacity at hubs is modeled using soft constraints (i.e., violations are penalized in the objective function), there is no need to incorporate waiting in the movement arcs and they could be represented by \((i,j,t)\). However, we choose to use \((i,j,t,\bar{t})\) for consistency of notation. The formulation is as follows:

\[
\begin{align*}
\min & \quad \sum_{((i,t),(j,\bar{t})) \in A} c_{ij} y^T_{ij} + \sum_{(i,t) \in N'} \mu(S^t_{i, \text{L}} + S^t_{i, \text{U}}) \\
\text{s.t.} & \quad \sum_{p^k \in \mathcal{P}^k} v_k^p = 1 \quad \forall k \in K \quad (4a) \\
& \quad \sum_{k \in K} \sum_{p^k \in \mathcal{P}^k} q_k v_k^p \leq Q y^T_{ij} \quad \forall ((i,t),(j,\bar{t})) \in A \quad (4b) \\
& \quad \sum_{((i,s),(j,\bar{s})) \in A; \bar{t}_i < s \leq t} y^s_{ij} \leq L_i + S^t_{i, \text{L}} \quad \forall (i,t) \in N' \quad (4c) \\
& \quad \sum_{((i,s),(j,\bar{s})) \in A; \bar{t}_j \leq s \leq t + \bar{\tau}_u} y^s_{ij} \leq U_j + S^t_{j, \text{U}} \quad \forall (j,\bar{t}) \in N' \quad (4d) \\
& \quad \sum_{k \in K} \sum_{p^k \in \mathcal{P}^k_{\text{ind}}} q_k v_k^p \leq \gamma \sum_{k \in K} q_k \\
& \quad v_k^p \in \{0,1\} \quad \forall P^k \in \mathcal{P}^k \quad \forall k \in K \quad (4f) \\
& \quad y^T_{ij} \in \mathbb{Z}_{\geq 0} \quad \forall ((i,t),(j,\bar{t})) \in A \quad (4g) \\
& \quad S^t_{i, \text{L}}, S^t_{i, \text{U}} \in \mathbb{N}_+ \quad \forall (i,t) \in N' \quad (4h)
\end{align*}
\]

The objective is to minimize the total travel cost plus the penalty on violations of loading and unloading capacities. Constraints (4a) enforce that every commodity is served. Constraints (4b) ensure that a sufficient number of vehicles are available for the commodities that are sent from node \( i \) (starting loading at time \( t \)) to node \( j \) (finishing unloading at time \( \bar{t} \)). Constraints (4c) and (4d) are the hub capacity constraints. Constraint (4e) restricts the maximum quantity \( (\gamma \sum_{k \in K} q_k, \text{ where } \gamma \text{ is the transshipping ratio s.t. } 0 < \gamma \leq 1) \) of transshipped demand. Finally, constraints (4f), (4g) and (4h) define the decision variables and their domains.

After obtaining a service network plan (i.e., the time-indexed commodity paths), we regard all commodity paths as independent pickup and delivery tasks. For example, a path with a visit sequence of \( A - B - C \) is split into two tasks \( A - B \) and \( B - C \) with the corresponding available time and due time. We then solve a pickup and delivery problem with time windows and docking
capacities via tabu search to further reduce the total number of vehicles. We observe that the decrease in the number of utilized vehicles also reduces the violation of the loading and unloading capacity constraints as two tasks (or timed paths) arriving at the same node within a time period can be potentially assigned to the same vehicle. The solution of this routing problem is the final plan that is handed to the regional planners for the deployment.

A same-day service network produced by the above approach has been deployed in one of the first-tier cities in China for more than one year. By exploiting the fact that many of the commodities are served using relatively short direct paths and that these direct paths can depart later in the day, the service cut off time for many markets (origin-destination pairs) commodities could be extended, sometimes up to 2 hours, which resulted in an increase of same-day package demand of more than 30%.

6. Final remarks
Same-day and instant delivery services are the fastest growing business segments of package express carriers. These services are typically offered in urban areas and require effective consolidation to be profitable. Consolidation can only be achieved by operating hub networks and because real-estate prices in urban areas are high (and are expected to go up even more), it is only possible to operate relatively small hubs and these hubs are likely to have limited loading and unloading capacity (and limited space for vehicles waiting to be loaded or unloaded). We have introduced a variant of the traditional service network design problem that incorporates loading and unloading capacity limits at hubs to study these new delivery environments and have proposed and analyzed different heuristic solution approaches. Much more needs to done to be able to provide truly effective decision support for companies operating in this space. Most importantly, recognizing that many of the model parameters are uncertain, e.g., vehicle loading and unloading time and travel times between hubs, and developing solution approaches that go beyond using “guessing” parameter values or using point estimates as parameter values.

Acknowledgments
This work would not have been possible without the assistance of the City Operations Planning Team and Data Science and Operations Research Center at SF Express. We thank them not only for providing us with real-world instances, but also for the many informative and insightful discussions.

References


