Two-Stage Sort Planning for Express Parcel Delivery

Reem Khir, Alan Erera, Alejandro Toriello

H. Milton Stewart School of Industrial and Systems Engineering

Georgia Institute of Technology

Atlanta, GA, 30332, USA

Parcel transportation services have grown in importance in modern logistics and have experienced significant recent changes, most notably a shift to operations with more time-guaranteed volume moving faster between origins and destinations, driven by growing demand for e-commerce. Piece-level sortation systems are used within sorting facilities in parcel transportation networks to enable the execution of effective consolidation plans with low per-unit handling and shipping costs. The design and control of effective sortation systems have become more complex as both the volume of parcels and also the number of time-definite service options offered by parcel carriers has grown. In this paper, we describe approaches for planning two-stage parcel sort operations that explicitly consider time deadlines and sorting capacities. In two-stage sorting, parcels are sorted into groups by a primary sorter and then parcels from these groups are dispatched to secondary stations for final sort. We define a sort planning optimization problem in this setting using mixed-integer programming, where the primary objective is to minimize operational cost subject to machine capacity and parcel deadline constraints. Since a detailed optimization problem for sort planning based on flows in a time-space network is difficult to solve for realistically-sized instances, we develop an alternative formulation that is easier to solve and shares the same feasible region of first-stage sorting decisions with the detailed model; for many practical objective functions, this simpler model can be used to find cost-optimal solutions to the detailed model. We illustrate the proposed modeling approach and its effectiveness using real-world instances obtained from a large parcel express service provider.

1 Introduction

Parcel delivery services are experiencing significant volume growth driven by e-commerce, and much of the growth focuses on express parcels that need to be moved quickly to final destinations. FedEx, for example, recorded more than $29 billion in revenue from express parcel delivery in 2019, a 38% increase from 2016 [11]. Leading players in the industry are investing in capacity expansion projects to serve this growth. SF Express, for example, has expanded its ground network to 531 transit hubs in China alone [23]. To enable a fast and effective consolidation network, more than 60% of these hubs are equipped with parcel sortation systems that allow for rapid unloading, sorting, and reloading of trucks and trailers; 14% of these sortation systems are fully automated. Once installed, parcel sortation systems need to be operated with carefully

designed sort plans that ensure that the capacity of these systems is well utilized while meeting objectives for on-time parcel delivery.

Parcel sortation systems vary in layout, equipment capacity, and automation level. In a typical sorting hub, parcels arrive throughout the day via inbound trucks, and they are required to be sorted to outbound trucks and dispatched by specific deadlines that allow the carrier to meet the time-definite service guarantees provided to customers. In this paper, we consider the problem of determining a cost-effective and feasible operational plan for sortation systems installed within a single sorting hub. Such a plan is developed in advance of operations and is typically modified every few weeks using updated demand forecasts and new consolidation (parcel flow) plans. In this work, we assume that decisions have been made at the tactical level to ensure that sorting equipment with appropriate capacities has been installed such that feasible sorting operations exist at the hub for typical operating days.

Existing research on parcel sorting focuses mainly on design and operational planning for single-stage sortation systems. Important decisions in these problems include the layout design, the assignment of truck unloading/loading docks to destinations, and the operational sequencing of unloading/loading operations. In contrast, we focus on detailed operational planning for two-stage systems, including those with a manual secondary sort. While increasing volumes and complexity often point toward automated solutions, these systems are often very costly and inflexible once installed. In the following subsection, we describe a two-stage semi-automated sortation system that motivates our work.

1.1 Parcel sortation system description

We consider a sortation hub that operates a two-stage sortation system for flats and small parcels, as illustrated in Figure 1. The operations start at the receiving docks where parcels arrive to the hub in vans or trucks and are unloaded onto a conveyor. The conveyor system is used to transfer small parcels to a cross-belt sorter, referred to hereafter as a primary sort belt. Each parcel is scanned (e.g. with a bar code or RFID system), and based on its label is directed to one of the discharge points located on either side of the primary sort belt, where it is pushed via chute and grouped into piles. An urban logistics hub receives a large mix of parcels with different destinations and service requirements, often exceeding the number of pile positions available at the primary sort area, thereby prompting the need for secondary sorters to complete the sort process. Piles that are assigned a single geographic destination and dispatch deadline complete their sort process at the primary sort area, while piles that contain parcels with different geographic destinations and/or dispatch deadlines are moved to secondary sort stations for detailed and final sorting. Each secondary sort station is operated manually with the aid of a put wall of cubicles open on one side for detailed sorting and on the other side for packing. Each cubicle is assigned a container direction, defined in our case as a pair of final terminal destination and a deadline for arrival. A sorter,

guided by a put-to-light system, scans each parcel and places it in its corresponding cubicle and these cubicles are subsequently unloaded into containers. Packed containers (often parcel bags) are finally placed on a conveyor system for transfer to their corresponding shipping docks for loading onto outbound trucks.

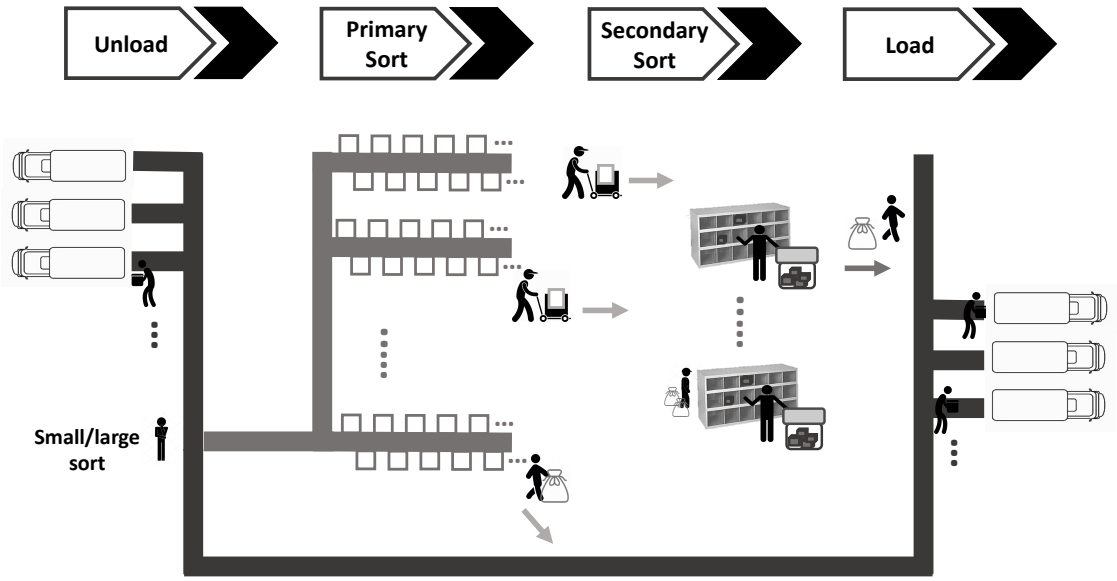

Figure 1: *An illustration of a two-stage sort process*

For such a system, we consider the problem of specifying an operational sort plan for the two-stage sorting process, focusing on sorting inbound parcels by dispatch deadlines. For this two-stage system, a sort plan determines how parcels are grouped by the primary sort into piles and then how parcels in piles are moved over time to secondary sort stations for final sorting and packing.

1.2 Related literature

The existing literature on sortation hubs for consolidation transportation systems addresses a broad array of decision problems. Network design problems, such as the strategic hub location problem [2, 7, 10] or tactical service network design problems [9] consider how to design and operate systems of transfer hubs. Our work focuses on operations within a parcel sortation hub once network design decisions have been fixed. Research has focused on tactical and operational problems, ranging from layout design to resource planning and scheduling. In recent work, [3] present an extensive review of these problems with a focus on fully automated hubs. Similarly, [14] provide a survey on the planning and control of automated material handling systems operated by parcel and postal services. As both of these review papers make clear, most of the work in this area focuses on the inbound and outbound operations that interface with sorting systems

(see for example [4, 17, 18]), while few studies address the planning of the sort process itself.

A few papers address planning of manual and semi-automated sorting processes. [16] develop a mixed integer program to aid in planning the sort operations of a large package express delivery service. The decision problem is to generate a sort plan that determines the assignment of inbound loads to bins and outbound loads to racks, as well as the number of forklifts required to perform the sort. The objective is to minimize the daily costs associated with employing and operating forklifts. To solve this computationally challenging model, they proposed a decomposition procedure that could be used to solve small and moderately sized instances efficiently. Another research effort [24] focuses on the optimization of transportation within a sortation hub and proposes a mixed integer program to minimize the manual transportation required to move parcels from sort areas to loading gates. A three-dimensional assignment model is developed which is then solved using a hierarchical decomposition approach.

Turning to research on fully automated sortation systems, several studies investigate different item release policies to improve the throughput of automated sorters and service performance in warehouses and order fulfillment centers (see [5, 8, 13]). [6] proposes a dynamic programming model to schedule the loading process for parcels onto sortation conveyors to minimize makespan. The proposed optimization procedure was found to be most useful when the decisions considered were limited to determining which inbound shipments to assign to which loading stations, with the sequence of shipment loading treated as fixed and given; we make a similar assumption in our work. Computational experiments demonstrated that using simple rules of thumb, such as an earliest delivery rule, could result in considerably sub-optimal solutions and that the solutions found by optimization provided significant benefit. [12] evaluated different design alternatives for closed-loop tilt tray sortation conveyors and compared throughput performance. They proposed an optimization model for assigning inbound and outbound destinations to unloading and loading docks, proved its NP-hardness, and developed heuristic solution procedures using greedy randomized adaptive search and simulated annealing.

More closely related to our research, [20] proposed a stochastic programming approach for assigning loading destinations to secondary sort work centers in an automated package sort facility. The objective was to generate a balanced workload assignment to work centers, while accounting for uncertainty in arrivals during every sort shift. This research focuses on fully automated sort systems where primary and secondary sorts are operated continuously over time, a key difference from the research problem we address, where decisions about when to dispatch parcels to secondary sort stations are critical. Considering a planning horizon of a day, the authors propose and test different mixed-integer nonlinear optimization formulations that account for operational constraints such as door loading capacities and workstation-to-destination compatibility for material handling. Constraints related to service or time requirements were not explicitly considered, another key difference between this work and our own.

1.3 Contribution

Our contributions are in planning two-stage sorts for parcel delivery hubs. Unlike previous work in this area, we are the first to address the planning of systems with multiple stages that serve modern high-velocity parcel operations with many sort deadlines during the operating day. Specifically:

- We develop optimization methods to generate time-feasible sort plans for two-stage sort systems by assigning parcels to first-stage sort positions (piles) and determining a dispatch schedule from the first to the second sort stage. The primary optimization objective is to minimize the costs of sorting, but we also consider additional objectives, including balancing workload and maximizing schedule slack time to create plans that are robust to fluctuations in sort demand.
- We explicitly model time deadlines for sorting that enable the parcel carrier to meet on-time delivery service guarantees that are commonly offered in practice.
- For tractability, we propose an integer programming formulation for solving sort plan optimization problems that separates first-stage sort decisions from second-stage decisions but, importantly, ensures that our first-stage decision model preserves feasibility (namely, parcel time feasibility) for the second-stage. This formulation allows a detailed time-space model to be replaced by a much simpler model that can be readily solved exactly for large-scale instances found in practice.
- We demonstrate that we can solve instances of realistic scale and composition in moderate CPU time using input data from a parcel carrier partner, and demonstrate that the solutions found outperform those identified by other optimal and heuristics solution approaches.

The remainder of the paper is organized as follows. In Section 2, we introduce the sort plan design problem, describe our assumptions and modeling choices, and present a mixed integer programming model. In Section 3 we propose an alternative model formulation and algorithms and discuss their relation to the sort plan design problem formulation introduced in Section 2. In Section 4, we present the results of our computational study using real-life instances obtained from an international parcel delivery service provider. In Section 5, we summarize our conclusions and discuss future work.

2 Problem Statement and Model Formulation

In this section, we introduce a *sort plan design* optimization problem to find a cost-effective assignment of parcels to primary sort equipment and a corresponding dispatch schedule for secondary sort operations. We first describe the problem parameters, modeling choices and assumptions. We then give a problem formulation using mixed-integer programming.

2.1 Problem description

The parcel volume arriving to a sortation hub varies considerably by time of day. Arriving parcels have different final destinations and service classes (which have associated deadlines for departing the hub), and the arriving mix also varies over time. Sort hub operations are often organized into shifts taking into account such variations for better planning and utilization of resources. Suppose that a sort plan is to be designed for a shift, and furthermore suppose that an appropriate time discretization has been chosen for the shift. As with any discretized planning problem, choosing a finer discretization may improve model accuracy but increases the computational burden. Let $\mathcal{T}$ be the set of equally-spaced discrete time points that constitutes the shift. For simplicity, time points in this set are simply the integers $1, 2, \ldots, |\mathcal{T}|$.

The sortation hub includes a primary sorting system and secondary sorting stations. Arriving parcels are loaded onto the primary sorter which is operated at a constant maximum parcel processing rate of μ_p. The primary sorting system has a fixed number of discharge points, and a *pile* of parcels accumulates at each discharge point. Let $\mathcal{P}$ be the set of primary sort piles corresponding to these discharge points. In a two-stage sorting system, a primary sort pile may or may not require a secondary sort before its constituent parcels are considered sorted. Let the sort mode s of a pile define whether or not secondary sorting is required, where $s = 1$ indicates that no secondary sorting is required and $s = 2$ indicates that secondary sorting is required; let $\mathcal{S} = \{1, 2\}$ be the set of sort modes. Furthermore, let $\mathcal{C}$ be a set of time deadlines to be associated with piles, where $\mathcal{C} \subset \mathcal{T}$. A pile with deadline $c \in \mathcal{C}$ is one where all of its constituent parcels must complete sorting (primary and secondary, if necessary) no later than time c.

The secondary sort system consists of multiple sorting stations. A typical secondary sort station is a manually operated cabinet with n sort positions (cubicles), and each parcel in a pile with sort mode $s = 2$ must be assigned to exactly one of these positions. Suppose that the hub has enough secondary sort stations to devote one to each of the primary sort piles, but that no pile is assigned to multiple secondary stations. Additionally, suppose that each secondary sort station can process up to μ_s parcels per unit time (measured in the discretization of $\mathcal{T}$). Parcels that have completed sorting are loaded into intermediate containers (usually bags) and then transferred to loading docks and loaded onto outbound trucks.

Turning now to the demand side, suppose that each parcel arriving to the hub during the shift is categorized as one of the parcel *commodities*; let $\mathcal{K}$ be the set of commodities. Each commodity $k \in \mathcal{K}$ can be referred to by a unique pair of final destination terminal and sort deadline time $d^k \in \mathcal{T}$ that is common across all commodity parcels. To meet on-time service expectations, all arriving parcels for commodity k must complete primary and secondary sorting no later than time d^k. The parcels that comprise commodity k arrive to the sortation hub over time via inbound trucks, mixed with parcels from many other commodities. Parcels from arriving trucks are loaded onto the primary sorting system, often in first-in and first-out (FIFO) order but sometimes staged. In this problem, we assume that the inbound truck arrival times and

primary sort loading decisions are fixed and that the primary sorting system is continuously operated with maximum rate μ_p. Thus, it is possible to determine for each commodity k an *arrival profile* representing the times when each parcel of this commodity has completed primary sorting. Let this profile be given thus by a set of arrival quantities q_t^k, measured in parcels, for each time $t \in \mathcal{T}$. The total arriving quantity of commodity k during the shift is $q^k = \sum_{t \in T} q_t^k$.

Given these inputs, the *sort plan design* problem is to determine an assignment of parcels to primary sort piles and a schedule of dispatches from primary sort to secondary sort stations. The primary consideration in this problem will be the time-feasibility of the sort plan design, where a plan is time-feasible if each parcel for each commodity completes the sort by its service deadline. The objective of the problem is to minimize the total cost of sorting parcels. Costs accrue in the sorting system when resources are used to sort parcels or to transfer them from one zone of the hub to another. A key to reducing costs is to have as many parcels as possible skip the secondary sorting process, thus avoiding an extra transfer and sorting step within the facility. Although we also consider other objectives in subsequent modeling sections, the primary objective is minimizing the number of parcels sent to secondary sorting. We assume that secondary sorting for a primary sort pile is not necessary only if it contains a *single* commodity. We also assume that feasible sort plans require *all-or-nothing* assignments, where all parcels for a commodity must be assigned to a single primary sort pile. All-or-nothing assignments do not allow commodity splitting across piles (and subsequent secondary sorting stations) and are simpler to implement in practice.

2.2 The Sort Plan Design (SPD) problem formulation

In this section, we develop an optimization formulation for the sort plan design problem. Here we model parcel assignments and dispatch decisions using a time-space network, as shown in Figure 2. Each node represents a pair of sort location and a time point. Each arc represents either a sorting activity, a dispatch that models the transfer of parcels from one zone of the hub to another, or a holding activity that models the staging of parcels at a particular location over time. For notational simplicity in this paper, we assume transfer activities take zero time, while sorting and staging activities take unit time; it is not difficult to modify the model to accommodate different activity times.

We now introduce some notation for a mixed-integer programming model of this problem. Let the parcel sort network $G = (\mathcal{N}, \mathcal{A})$ model two-stage sorting using a time-space representation. Each node in the timed-node set $\mathcal{N}$ represents a parcel position at a particular point of time, and each arc in the timed-arc set $\mathcal{A}$ represents parcel sorting, movement, or staging over time. The node set $\mathcal{N}$ is partitioned into subsets of nodes that represent parcel position, $\mathcal{N} = \{\mathcal{N}^{PS-belt} \cup \mathcal{N}^{pile} \cup \mathcal{N}^{ss} \cup \mathcal{N}^{fs} \cup \mathcal{N}^{load-belt}\}$. Nodes $\mathcal{N}^{PS-belt}$ represent times of entry into the primary sorting process, $\mathcal{N}^{pile}$ represent times when primary sorting is finished and parcels enter piles, $\mathcal{N}^{ss}$ and $\mathcal{N}^{fs}$ represent times when secondary sort

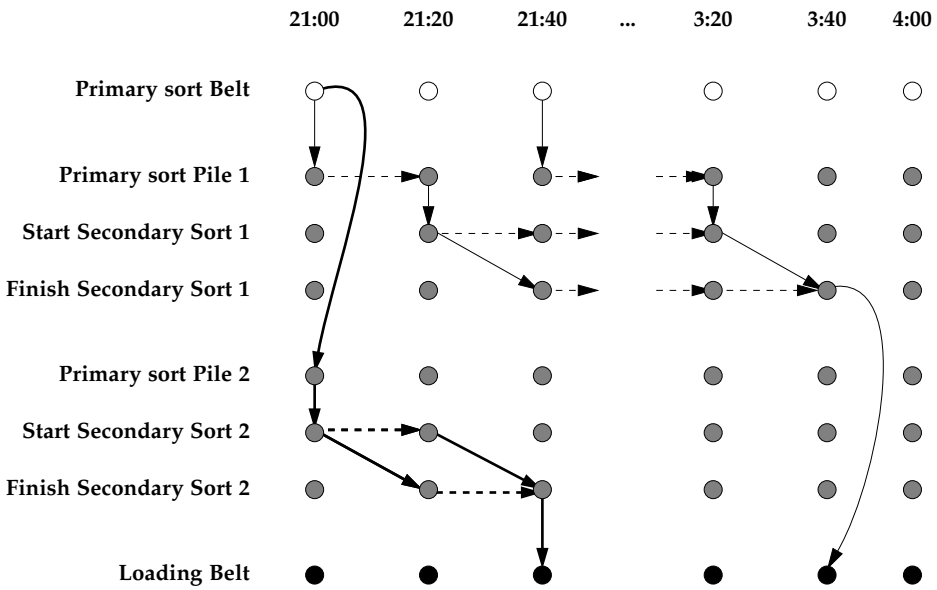

Figure 2: *An illustration of a time-space network that models a two-stage sort system with two pile positions. The depicted arcs show an example sort plan solution with solid arcs representing parcels being sorted/dispatched and dashed arcs representing parcels being held at the same location over time.*

start and complete (respectively), and $\mathcal{N}^{load-belt}$ represent times when the entire sort process has been completed and parcels are packed and ready for loading. The arc set $\mathcal{A}$ is partitioned similarly, where the categorization is defined by the tail node. Arcs $a = ((PS-belt, t), (p, t)) \in \mathcal{A}^{PS-belt}$ represent the unloading of parcels from primary sort belt into pile $p \in \mathcal{P}$ at time t. Arcs $a = ((p, t), ((p, ss), t) \in \mathcal{A}^{pile}$ represent the dispatch of pile p parcels to their pre-assigned manual station for secondary sort at time t while arcs $a = ((p, t), (p, t+1) \in \mathcal{A}^{pile}$ represent the staging of parcels in their pile location from time t to time $t+1$. Similarly, arcs $a = (((p, ss), t), ((p, fs), t+1) \in \mathcal{A}^{ss}$ represent the secondary sort of pile p parcels starting at time t and finishing at time $t+1$ while arcs $a = (((p, fs), t), (load, t) \in \mathcal{A}^{fs}$ represent the completion of parcel sorting into outbound containers and their dispatch to loading docks. Let $\delta^+(i)$ and $\delta^-(i)$ represent the sets of arcs leaving and entering node $i \in \mathcal{N}$, respectively.

We use decision variables $y_{p,c,s} \in \{1, 0\}$ to indicate whether or not pile p with deadline c and sort mode s is used and $x^k_{p,c,s} \in \{1, 0\}$ to indicate whether or not commodity k is assigned to pile p with deadline c and sort mode s. Moreover, variables $f^c_a \in \mathbb{Z}^+$ denote the flow on arc a with deadline c, in parcels, and $w^c_a \in \{1, 0\}$ denote whether or not there is a positive flow in arc $a \in \mathcal{A}^{pile}$ with deadline c.

The sort plan design integer programming formulation (SPD) is then given by:

$$\underset{x,y,f,w}{\text{Min}} \quad C(x,\ y,\ f,\ w) \tag{1a}$$

$$\text{s. t.} \quad \sum_{s \in S} \sum_{p \in \mathcal{P}} \sum_{c \in \mathcal{C}} x^k_{p,c,s} = 1 \qquad \forall k \in K, \tag{1b}$$

$$\sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S} y_{p,c,s} \leq |\mathcal{P}| \tag{1c}$$

$$\sum_{s\in S}\sum_{c\in\mathcal{C}} y_{p,c,s} \leq 1 \qquad \forall p \in \mathcal{P}, \tag{1d}$$

$$\sum_{k\in K} x^k_{p,c,s} \leq n y_{p,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \tag{1e}$$

$$\sum_{k\in K} x^k_{p,c,s} \leq y_{p,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 1, \tag{1f}$$

$$x^k_{p,c,s} \leq y_{p,c,s} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \tag{1g}$$

$$y_{p,c,s} \geq y_{p+1,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s \in S, \tag{1h}$$

$$f^c_a = \sum_{k\in K} q^k_t x^k_{p,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, a = ((PS-belt,t),(p,t)) \in \mathcal{A}^{PS-belt}, \tag{1i}$$

$$\sum_{a\in\delta^+((i,t))} f^c_a = \sum_{a\in\delta^-((i,t))} f^c_a \qquad \forall (i,t) \in \mathcal{N} \setminus \{\mathcal{N}^{PS-belt} \cup \mathcal{N}^{load-belt}\}, c \in \mathcal{C}, t \leq c, \tag{1j}$$

$$f^c_a = \sum_{k\in K} q^k x^k_{p,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, a = (((p,fs),c),(load,c)) \in \mathcal{A}^{fs}, \tag{1k}$$

$$f^c_a \leq \mu_s \qquad \forall a \in \mathcal{A}^{ss} \tag{1l}$$

$$\sum_{a\in\delta^-((i,t))} f^c_a = 0 \qquad \forall (i,t) \in \mathcal{N}^{load-belt}, t > c \tag{1m}$$

$$\sum_{a\in\delta^+((p,c),t)} w^c_a \leq 1 \qquad \forall (p,t) \in \mathcal{N}^{pile} \tag{1n}$$

$$f^c_a \leq \sum_{k\in K} q^k w^c_a \qquad \forall a \in \mathcal{A}^{pile} \tag{1o}$$

$$y_{p,c,s} \in \{0,1\} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s \in S \tag{1p}$$

$$x^k_{p,c,s} \in \{0,1\} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s \in S \tag{1q}$$

$$f^c_a \in \mathbb{Z}^+ \qquad \forall c \in \mathcal{C}, a \in \mathcal{A} \tag{1r}$$

$$w^c_a \in \{1,0\} \qquad \forall c \in \mathcal{C}, a \in \mathcal{A}^{pile} \tag{1s}$$

The objective function (1a) here is written in a generic form $C(x,y,f,w)$, and is assumed to be some linear function of these decision variables; later sections describe specific possible objective functions.

Constraints (1b) ensure that each commodity is assigned to a single primary sort pile, *i.e.,* commodity splitting is not allowed. Constraints (1c) ensure that the number of primary sort pile positions that can be used is no more than the number of primary sort pile positions available in the primary sort area and (1d) ensure that a pile position is assigned at most one sort mode and deadline. Constraints (1e) and (1f) place an upper bound on the number of commodities that can be assigned to a pile. Piles with sort mode 1 do not require secondary sorting and are allowed to be assigned at most one commodity, while piles with sort mode 2 require secondary sorting and are allowed to be assigned up to n commodities, where n is

the number of cubicles in the put wall of each secondary sorting station. Constraints (1g) are coupling constraints to ensure that a commodity is assigned to a pile only if the pile is used. Constraints (1h) are added to break symmetry between primary sort piles for computational efficiency. Constraints (1i) - (1k) are flow conservation constraints. Constraints (1l) place an upper bound on the number of parcels that can be sorted by a secondary sort station during a time period, where μ_s is the processing rate of the secondary sort stations measured in parcels per time period. Constraints (1m) ensure that every parcel completes its sort requirement by its deadline. Constraints (1n) are added to ensure that if a pile is dispatched from primary to secondary sort area, all parcels accumulated should be dispatched. Otherwise, they should all be staged to the next period. This set of constraints is relevant and applicable when there is no capacity on how many parcels can be dispatched in one transfer. Finally, constraints (1o) are coupling constraints between the flow and flow indicator variables for dispatch arcs.

3 Alternative Model Formulation

Finding optimal solutions to *SPD* can be computationally challenging for large-sized instances commonly found in practice. Parcel sort facilities often must handle a large number of commodities (especially during overnight peak operating hours), and must plan operations over a finely discretized time horizon, all of which make the *SPD* problem difficult to solve. To build a model that is more computationally useful, in this section, we propose an alternative formulation that focuses only on the commodity-to-pile assignment decisions and seeks those that are guaranteed to induce time-feasible flows given secondary sort capacities and parcel deadlines. Note that in practice, pile assignment decisions are critical and have perhaps the greatest impact on sorting costs; for example, building piles that contain only a single commodity can allow those parcels to skip secondary sorting entirely and the associated costs. As we will see, it is possible to construct an alternative model for pile assignment decisions, denoted the primary-sort assignment (*PA*) model below, that is guaranteed to yield time feasible flows for the *SPD* model and, for some objective functions, cost-optimal *SPD* flows. Since another important cost component is the transfer of parcels from primary sort to the secondary sorting stations given pile assignment decisions, we also propose models for minimizing these transfer costs and algorithms for finding cost-optimal solutions.

3.1 Primary-sort assignment (PA) problem formulation

In this problem, we seek to determine the assignment of commodities to primary sort piles using an integer program. The *primary sort assignment* (*PA*) problem is formulated as follows.

$$\underset{x,y}{\text{Minimize}} \quad D(x,\, y) \tag{2a}$$

$$\text{subject to} \quad \sum_{s\in S}\sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}} x^k_{p,c,s} = 1 \qquad \forall k \in K, \tag{2b}$$

$$\sum_{k\in K}\sum_{t+1\le\tau\le c} q^k_\tau x^k_{p,c,s} \le \mu_s(c-t) \qquad \forall t \in \mathcal{T}, p \in \mathcal{P}, c \in \mathcal{C}, s = 2 \tag{2c}$$

$$\sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S} y_{p,c,s} \le |\mathcal{P}|, \tag{2d}$$

$$\sum_{k\in K} x^k_{p,c,s} \le n y_{p,c,s} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s = 2 \tag{2e}$$

$$\sum_{k\in K} x^k_{p,c,s} \le y_{p,c,s} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s = 1 \tag{2f}$$

$$x^k_{p,c,s} \le y_{p,c,s} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s = 2 \tag{2g}$$

$$y_{p,c,s} \ge y_{p+1,c,s} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s \in S \tag{2h}$$

$$y_{p,c,s} \in \{0,1\} \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s \in S \tag{2i}$$

$$x^k_{p,c,s} \in \{0,1\} \qquad \forall k \in K, p \in \mathcal{P}, c \in \mathcal{C}, s \in S \tag{2j}$$

The objective function (2a) here is again written in a generic form $D(x, y)$, and is assumed to be some linear function of these decision variables; later sections describe specific possible objective functions. Constraints (2b) and (2d) - (2h) are identical to constraints (1b) - (1h) described earlier in Section 2.2.

Constraints (2c) ensure that the plan meets sort deadlines for commodities that are dispatched to the secondary sort. To understand these constraints, consider the set of conditions

$$\sum_{k\in\mathcal{K}}\sum_{1\le\tau\le t} q^k_\tau x^k_{p,c,s} \ge \mu_s t + b_{p,c,s} \qquad \forall t \le c, t \in \mathcal{T}, p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \tag{3}$$

where $b_{p,c,s} = \sum_{k\in\mathcal{K}}\sum_{1\le\tau\le c} q^k_\tau x^k_{p,c,s} - \mu_s c$ for every pile p with deadline c requiring two-stage sort. These conditions, as depicted in Figure 3, guarantee that every point on the cumulative arrival curve for every pile lies above its secondary sort deadline line. This line, the right-hand side of (3), has as slope of the secondary sort rate μ_s and an intercept $b_{p,c,s}$ that equals the difference in the available sort capacity ($\mu_s c$) and the total number of parcels that will accumulate in the pile before its deadline c. Simplifying these inequalities yields constraint set (2c). Scenario 2 in the figure shows a case where the total capacity of the sort would be sufficient (since the intercept of the deadline line is negative), but many parcels arrive too late and overwhelm the secondary sorter closer to the deadline.

For simplicity, constraints (2c) are presented for the special case where the travel time between the primary sort and secondary sort locations is assumed to be zero time units. The model, however, can be easily modified when these travel times are positive. For instance, if it takes x time periods (measured in the discretization $\mathcal{T}$) to transfer a pile from the primary to secondary sort area, the right-hand side of constraints (2c) can be changed to $\mu_s(c - t - x)$ to account for capacity loss due to travel time. This

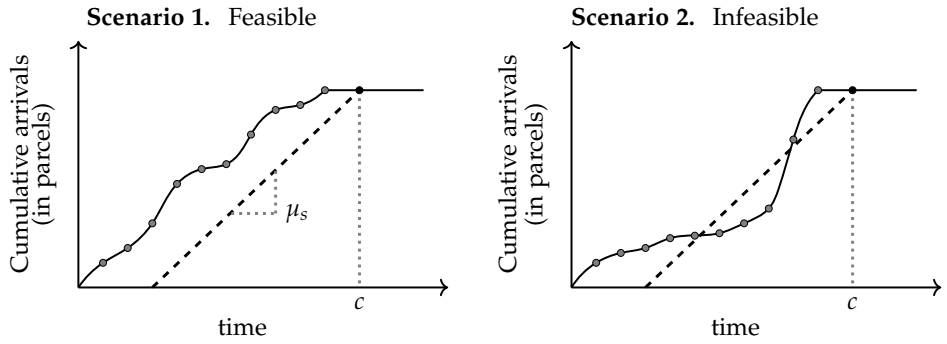

Figure 3: *An illustration of how constraints (2c) ensure feasible secondary sort solutions. Scenario 1 is feasible because every point on the arrival curve (grey curve) for a pile lies above its secondary-sort deadline line (black dashed), while scenario 2 is infeasible.*

corresponds to shifting the secondary-sort deadline line in Figure 3 by x time units to the left, reducing the remaining available secondary sort time and capacity.

3.2 Relationship between SPD and PA models for sort plan design

Next, we discuss the relationship between the original *SPD* problem and the alternative *PA* formulation. We show that each feasible solution to the *PA* formulation corresponds to a feasible solution to the *SPD* and vice versa; thus, these two models share the same feasible region of pile assignment decisions. Given this result, we therefore also discuss for which objective functions the *PA* formulation can be used to find cost-optimal solutions to the *SPD* model. Of course, in this section we assume that the problem instance's inputs, assumptions and modeling choices, such as how time is discretized, are identical for both the original *SPD* model and the *PA* model.

Recall that in the *SPD* formulation, we use flow-based variables to ensure that the generated sort plans are time feasible. We now show that the assignment-based time feasibility constraints used in *PA* are valid for *SPD*.

Proposition 1. *Every inequality* (2c) *in PA is valid for SPD.*

Proof. Suppose (x, y, f) is a feasible solution for *SPD*, where x, y are assignment-related decision vectors and f is flow-related decision vector. Since f represents feasible flows, the time feasibility constraints (1m) along with all the flow conservation constraints (1i)-(1k) hold, and therefore, the flow of parcels assigned to each primary sort pile is guaranteed to be completed no later than the pile deadline c. This implies that for every pile there must exist a total flow in the secondary sort arcs A^{ss} that includes all parcels arriving between any time t and c. More formally, constraints

$$\sum_{a \in \mathcal{A}^{ss}: t \le \tau < c} f_a^c \ge \sum_{k \in K} \sum_{t+1 \le \tau \le c} q_\tau^k x_{p,c,s}^k \qquad \forall t \in \mathcal{T}, p \in \mathcal{P}, c \in \mathcal{C}, s = 2 \tag{4}$$

where $a = ((ss, p), \tau), (fs, p), \tau + 1))$ must hold and therefore, are valid for SPD. In addition, constraints (1l) must also hold by the flow f feasibility, to ensure that there is sufficient capacity at the secondary sort stations to process its assigned parcels. By summing over all arcs from time t up to the pile deadline c in constraints (1l), we establish an aggregate form of constraints (1l),

$$\sum_{a \in \mathcal{A}^{ss}: t \leq \tau < c} f_a^c \leq \mu_s(c - t) \qquad \forall t \in \mathcal{T}, p \in \mathcal{P}, c \in \mathcal{C}, s = 2. \tag{5}$$

where $a = ((ss, p), \tau), (fs, p), \tau + 1))$. Substituting by the right-hand side of (4) in (5), we obtain (2c) and, thus, the inequality is valid for SPD. □

We next show that the PA model is in fact the projection of SPD model, meaning it exactly captures the feasibility of the x and y variables.

Theorem 2. *The PA formulation is the projection of the SPD formulation onto the space of (x, y)-variables. In other words, if (x, y) is feasible for PA, then there must exist a flow f such that (x, y, f) is feasible for SPD. Similarly, if (x, y, f) is feasible for SPD, then (x, y) is also a feasible solution of PA.*

Proof. Suppose (x, y, f) is a feasible solution for the SPD. For convenience, define Q to be the projection of the SPD constraints to the (x, y) space, and P to be the set of constraints given by the PA model. It follows from Proposition 3 that constraints (2c) are valid for SPD, which implies that $P \supseteq Q$. Therefore, any feasible solution for SPD is also feasible for PA.

To establish the converse, let $(x, y) \in P$, *i.e.* , (x, y) is a feasible solution for PA. We want to show now that we can construct a feasible flow f such that $(x, y, f) \in Q$. Notice that such a flow is guaranteed to exist. One way of constructing it is to use the backward recursion algorithm we propose later in Section 3.3, which generates a dispatch schedule for any feasible (x, y) such that the total number of required dispatches for each pile is minimized. Another way of constructing a feasible flow starting from a feasible (x, y) is to move parcels from one node to another at every time period. Therefore, there must exist at least one feasible dispatch schedule that corresponds to a feasible flow f in SPD such that $(x, y, f) \in Q$. □

Theorem 4 shows that the PA formulation preserves the feasibility of the original SPD problem. When the objective we use in SPD depends only on the (x, y) variables, it follows that PA preserves optimality as well.

Corollary 3. *Given an objective function* (1a) *that only depends on the (x, y) variables, i.e. $C(x, y, f, w) = D(x, y)$, an optimal solution for PA can be extended into an optimal solution for SPD and vice versa.*

One important objective of interest is to minimize the total cost of primary and secondary sorting, focusing on the resources required to conduct sort operations feasibly. A reasonable proxy for minimizing

total sorting costs is minimizing the number of parcels that require secondary sorting (or, equivalently, maximizing the number that are directly sorted using primary sort equipment). Other cost-related objectives that can be considered include minimizing the number of required secondary sort stations (piles) or minimizing the number of operating hours required for secondary sort stations. Note that such objectives depend only on the commodity-to-pile assignment decisions, and thus the alternative *PA* formulation can be used to replace the computationally challenging *SPD* formulation without loss of optimality. Other objective functions may also be important, and we will consider some of them as secondary objectives in a hierarchical optimization framework. For example, it may also be useful to balance the workload assigned to secondary sort stations or to maximize the available slack time in the sorting schedule to buffer against uncertainty in the number or timing of arriving parcels and improve the robustness of the plan; we discuss these secondary objectives in more detail in Section 4.

3.3 Secondary sort dispatch algorithms

Now that we have an approach for generating pile assignments that are cost-effective and are guaranteed to induce time-feasible sorting plans, we present complementary algorithms that use a solution of the *PA* model to generate dispatch schedules from primary to secondary sort areas for piles that require secondary sorting, *i.e.*, piles with sort mode 2. Recall that each pile has its own dedicated resources, including a secondary sort station, a sorter and a transporter. We consider a dispatch process that is operated manually by a worker with the aid of a push cart. We consider two cases:

- *Case 1: dispatch capacity is not binding.* There is no constraint on how many parcels can be moved in one dispatch from the primary sorting area to the secondary sorting area. Therefore, a dispatch that includes 1 or 500 parcels costs the same. This assumption is reasonable since we deal with small parcels like flat envelopes, with sizes that are small compared to cart sizes.
- *Case 2: dispatch capacity is binding.* There is a maximum number of parcels that can be moved in one dispatch. Therefore, the corresponding dispatch policies are further constrained by a capacity threshold.

Since the transfer process involves workers, we are interested in generating schedules that are also cost-effective given the assignment generated by the *PA* model. A reasonable proxy for minimizing dispatch-related cost is to minimize the number of required dispatches from primary to secondary sort areas. The challenge is then to generate a dispatch schedule that minimizes the number of dispatches while maintaining the time feasibility of sort plans, *i.e.*, sorting all parcels by the pile deadline, and respecting the threshold capacity constraint, if applicable.

We define a Secondary Sort Dispatch (SSD) problem that determines a partitioning of parcels into batches for transfer from primary to secondary sort areas, taking into account parcel deadlines and the

secondary sorter's capacity. The goal is to determine a dispatch schedule for each secondary sort pile while minimizing the number of required dispatches. The *SSD* problem can be viewed as a variation of the *Batch Delivery Scheduling* problem that appears in the production and scheduling literature. Example areas where batching decisions arise include production and manufacturing operations to reduce setup costs, multi-stage production systems to improve machine utilization, and other applications where multiple customer orders can be grouped together to reduce shipping costs. Previous research in this area addresses batch delivery decisions as well as scheduling, production, or distribution decisions simultaneously to improve coordination between production and transportation activities. Different variations of the problem appear in the literature, with various heuristic and metaheuristic solution methods [1, 15, 19, 22, 25]. See [21] for a comprehensive review of scheduling and batching problems, their complexity and related solution algorithms. The *SSD* problem focuses on partitioning parcels into batches for a given sequence of parcel arrivals that is determined after solving the *PA* model. Unlike much of the past work in this space, *SSD* focuses on minimizing dispatching costs that are independent of the number of parcels in the batch and does not involve any holding cost component. It also treats sort deadlines as hard constraints, i.e. tardiness is not allowed. Deadlines in our setting are also not batch-dependent; a deadline is common for all parcels (and thus all batches) in the same pile.

In what follows, we describe the proposed backward recursion algorithms for solving the (SSD) problem. Note that in these algorithms we seek to minimize the dispatch costs into secondary sort, and that dispatch decisions are represented by variables w_a^c in the *SPD* model; thus, the *SSD* algorithms consider a case where the *SPD* objective does not depend only on the x and y variables. It follows that if the objective function includes the flow variables f and/or w in the *SPD* problem, the solution generated by our alternative formulation is a heuristic solution for the original *SPD* problem, *i.e.*, it is a lower bound for a maximization problem and an upper bound for a minimization problem.

The backward recursion algorithm without dispatch capacity threshold

Define (p, c) as a primary sort pile p with deadline c. Each pile (p, c) has a feasible commodity assignment generated by the *PA* model. Let $A_{p,c}(t)$ denote the total number of parcels accumulating in pile (p, c) from the shift start time up to time t, and μ_s denote the secondary sort capacity measured in parcels per unit time. For the algorithms to follow, suppose that a pile arrives at secondary sort instantaneously after dispatch from primary sort. If a travel time is modeled and a *PA* solution has been found that is feasible given the travel time as described above, then the same algorithm below can be used by simply reducing c by the travel time to secondary sort.

The *SSD* problem can be solved using a backward recursion algorithm, which first determines the last dispatch time and size, and then moves backward in time to determine earlier dispatches until all parcels

are included in a dispatch. The idea of the algorithm is illustrated in Figure 4, and described formally in Algorithm 1. Recall that this algorithm is only required for piles with secondary sort requirements, *i.e.*, with sort mode 2.

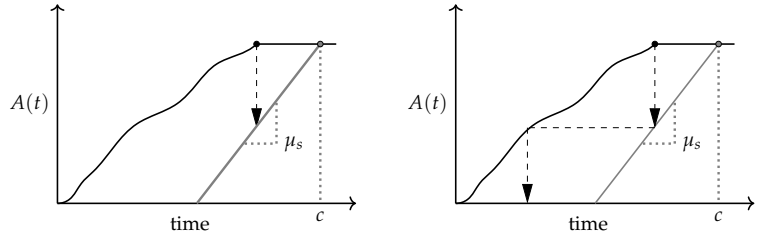

Figure 4: *An illustration of how the backward recursion algorithm works for a given pile assignment. It first determines the last dispatch time and size, and then moves backward in time to determine the second to last dispatch size and time and so in. In this example, two dispatches are required while maintaining time feasibility.*

Algorithm 1: Backward recursion algorithm without dispatch capacity threshold

Result: primary sort pile dispatch schedule to secondary sort that minimizes number of dispatches

for $p \in \mathcal{P}$ **do**

  for $c \in \mathcal{C}$ **do**

    set: $A^0_{p,c}(t) = \sum_{k\in\mathcal{K}} \sum_{t\in\mathcal{T}} q^k_t x^k_{p,c}$;

    initialize: $i \leftarrow 1$;

    initialize: $\bar{t} \leftarrow c$;

    initialize: $t \leftarrow \text{argmin}_{t\in\mathcal{T}}\{A^0_{p,c}(t)\}$;

    while $A^i_{p,c}(t) > 0$ **do**

      $S^i_{p,c}(t) = \mu_s(\bar{t} - t)$;

      $A^{i+1}_{p,c}(t) = A^i_{p,c}(t) - S^i_{p,c}(t)$;

      $\tau = \text{argmin}_{t\in\mathcal{T}}\{A^{i+1}_{p,c}(t)\}$;

      $i \leftarrow i + 1$;

      $\bar{t} \leftarrow t$;

      $t \leftarrow \tau$;

    end

    dispatch $S^i_{p,c}(t)$ parcels at time t;

  end

end

The proposed algorithm determines both the time and size of each pile dispatch throughout the sort shift duration for piles with sort mode $s = 2$. Note that our proposed algorithm generates a dispatch plan that has the following two properties:

Property 1. The last dispatch time of a primary sort pile to the secondary sort area coincides with the time of its last arrival, which is the earliest feasible last dispatch time.

Property 2. A pile dispatch at time t includes all accumulated parcels up to time t that are still at the primary sort area and were not moved in a previous dispatch, which represents the maximum feasible dispatch size.

Proposition 4. *For a given primary sort assignment, the backward recursion algorithm without capacity threshold determines a dispatch schedule that minimizes the number of pile dispatches required to complete the secondary sort requirements by the pile's deadline.*

Proof. Suppose we are given an optimal solution with a minimum number of pile dispatches, say m. Consider also the feasible solution generated by the backward recursion algorithm. Let t_m^* and s_m^* be the time and size of the last dispatch in the optimal solution, and t' and s' be the time and size of the last dispatch in the solution generated by the backward recursion algorithm. Notice that $t' \leq t_m^*$ and $s' \geq s_m^*$, which follows from the two properties that characterize the backward recursion algorithm.

Consider modifying the given optimal solution by shifting t_m^* backward in time by an amount equal to $\delta_m = t_m^* - t'$, while keeping everything else unchanged, and denote this time by $\hat{t}_m^*$. Notice that the resulting new solution has the same number of pile dispatches m as in the original optimal solution. Now, consider modifying further the new solution by shifting an amount equal to $\epsilon_m = s' - s_m^*$ parcels from the previous dispatch at time t_{m-1}^* to the dispatch at time $\hat{t}_m^*$ while keeping everything else unchanged. The resulting new solution maintains the same number of dispatches m; and thus, is still optimal. Similarly, repeat the procedure for dispatches $m-1$ down to the first pile dispatch where all arrivals are covered. As a result of this procedure, as illustrated in Figure 5, we have converted the optimal solution to one that is identical to the solution generated by the backward recursion algorithm while maintaining the same number of pile dispatches m. Therefore, the solution generated by the backward recursion algorithm is optimal, and gives the minimum possible number of pile dispatches. □

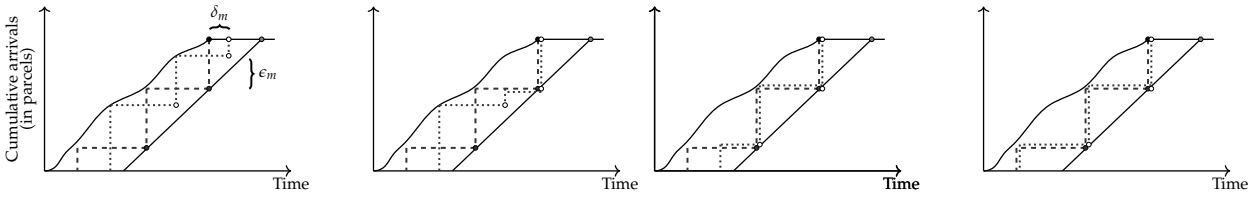

Figure 5: *An illustration of how to modify an optimal solution to one that is equivelent to the solution generated by the backward recursion algorithm while maintaining feasibility and optimality*

The backward recursion algorithm with dispatch capacity

We consider a minor variation in the backward recursion algorithm described in Algorithm 1 to account for limited dispatch capacity. Unlike case 1, where the dispatch size is only dependent on secondary sort capacity, dispatch sizes in case 2 are dependent on both secondary sort capacity and a dispatch capacity. Therefore, the only required change in Algorithm 1 is setting the dispatch size equal to the minimum between the secondary sort capacity and the available dispatch capacity. The modified algorithm is described in more detail in Algorithm 2 in the Appendix. It is not difficult to see that the solution of Algorithm 2 is optimal, and is lower-bounded by the solution generated by Algorithm 1.

Proposition 5. *For a given primary sort assignment, the backward recursion algorithm with capacity threshold determines a pile dispatch schedule that minimizes the number of pile dispatches required to complete the secondary sort requirements by the pile deadline. Furthermore, the solution's number of dispatches is lower-bounded by the solution generated using the backward recursion algorithm without capacity threshold.*

4 Computational Study

In this section, we present the results of our computational experiments. The objective is to analyze the performance of our sort plan design formulations and examine the characteristics of the generated sort plans. All algorithms and models were coded in Python 2.7.11, and solved using Gurobi Optimizer 7.5.1. A 3.1 GHz Dual-Core Intel Core i7 processor with 16 GB of RAM was used to carry out the computations.

4.1 Description of test instances

Our test instances are based on real data obtained from an international express delivery company. The company operates a number of semi-automated sortation hubs with different setups and capacities. In our experiments, we consider a hub with the following configuration. The primary sort area consists of four cross-sorter belts with a combined capacity of 60,000 parcels per hour and 38 pile positions. These belts are dedicated to processing parcels that require shipping using the carrier's linehaul network, and therefore, referred to henceforth as intercity parcels. The secondary sort area can have up to 38 stations, each uniquely assigned to a primary sort pile for processing, and is operated by a sorter working at a rate of 800 parcels per hour. Stations include 37 compartments that are used to segregate parcels based on their final destination and deadline.

The hub receives more than 90,000 intercity parcels on an average weekday. These parcels are conveyable and require sortation and consolidation into transport containers before being dispatched to the carrier's linehaul network for shipping. The volume of these arrivals varies greatly from one shift to another, with

peak arrivals taking place during night shifts. Table 1 provides a summary of shift structure as well as arrival volumes experienced by the company in one of its major sortation hubs.

Table 1: *Characteristics of test instances grouped by operational shift*

	Shift	Start time	End time	Average arrivals (in parcels)	Maximum arrivals (in parcels)	Parcel deadlines
S1	Sunrise	4:10	12:00	1093	1875	12:00
S2	Daysort	12:10	16:00	12338	19669	14:00, 15:30, 15:40, 16:00
S3	Twilight	16:10	21:00	14931	20937	19:10, 19:40, 21:00
S4	Midnight	21:10	4:00	45433	70179	1:00, 1:30, 2:00, 2:30, 3:00, 4:00

We focus on the daysort, twilight and midnight shifts, as they encompass the majority of parcels. We use 21 instances, each representing one of these shifts in a day covering one week from Monday to Sunday. To accurately model sort operations with tight service guarantees, we use a 10-minute time discretization for each shift.

4.2 Computational performance analysis

The main goal of the following experiments is to illustrate the practical usefulness of the proposed alternative modeling approach with respect to run times and quality of generated solutions. We focus our analysis on the performance of the $PA + SSD$ approach when compared to the SPD formulation, as well as other simple heuristics that are typically implemented in practice, described below.

- *SPD*: This approach involves solving the SPD formulation, which models both assignment and dispatch decisions using time-space representation.
- *First Fit Bin Packing*: This approach involves running a First Fit Bin Packing *(FF BP)* algorithm, which assigns commodities to primary sort piles in order of priority (from earliest to latest deadline) while respecting the primary sort pile capacity, which, in our case, is measured in the number of distinct commodities that can be mixed together. The resulting solution determines a primary sort pile assignment that minimizes the number of piles used, i.e. the number of manual secondary sort stations required.
- *First Fit Bin Packing Variant*: This approach involves running a *FF BP* algorithm as described above. The resulting solution determines an assignment with a number of required primary sort piles n_i^*. We then modify the commodity list by excluding the $|P| - n_i^*$ largest commodities and assign them to one-pass sort piles. After that, we re-run the algorithm on the modified commodity list. Again, the solution generates a plan with a number of required primary sort piles n_{i+1}^*. If $n_{i+1}^* < n_i^*$, we repeat the process until all primary sort piles are assigned to at least one commodity.

In the first set of experiments, we consider an objective that depends only on assignment-related

decisions when solving the *SPD* and *PA* problems. Specifically, we consider the objective of maximizing the number of parcels that are directly sorted using primary sort equipment, as modeled in (6). This objective can be viewed a proxy for minimizing sortation costs as all parcels have to go through at least one sort stage for consolidation:

$$\underset{x,y}{\text{Maximize}} \quad \sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S:s=1}\sum_{k\in K} q^k x^k_{p,c,s}. \tag{6}$$

We test instances using the Daysort shift (S2) data, and cross compare their performance using the four different approaches. Table 2 shows the results of these experiments.

Table 2: *Performance comparison between heuristic and exact solution approaches*

Metrics	Approach	Daysort Instances (size, in parcels)						
		S2_1 (3034)	S2_2 (14635)	S2_3 (19256)	S2_4 (19669)	S2_5 (19470)	S2_6 (29504)	S2_7 (38940)
Run time (in seconds)	FF BP	8	8	8	7	8	6	7
	FF BP Variant	13	19	19	23	15	17	15
	SPD	1437	588	627	605	2757	36000	36000
	PA	1250	950	731	305	849	610	471
# of parcels skipping secondary sorting	FF BP	0	0	0	0	0	0	0
	FF BP Variant	1106	3815	5602	5748	5408	8622	10816
	SPD	1106	3815	5602	5748	5408	-	10816
	PA	1106	3815	5602	5748	5408	8622	10816
# of required manual secondary sort stations	FF BP	12	16	16	16	16	16	16
	FF BP Variant	11	15	15	15	16	16	16
	SPD	11	15	15	15	16	-	16
	PA	11	15	15	15	16	15	16
Feasibility level[1]	FF BP	100.0%	92.0%	75.3%	75.3%	78.2%	61.3%	54.1%
	FF BP Variant	100.0%	100.0%	97.4%	92.4%	93.9%	86.2%	75.3%
	SPD	100.0%	100.0%	100.0%	100.0%	100.0%	-	100.0%
	PA	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

As can be seen in Table 2, both the *FF BP* and its variant were can be very efficiently, in seconds. *FF BP variant*, in particular, generates solutions that are optimal in the number of parcels assigned to one-pass sort. However, its performance with respect to time feasibility deteriorates considerably as the instance size grows, i.e., as the number of arrivals increases. The *PA* approach, on the other hand, guarantees both optimality and time feasibility, just like *SPD*. It also runs efficiently, taking at most 20 minutes to find optimal solutions; in contrast, for instance S2_6, *SPD* reached our time limit of 10 hours without even finding a solution.

In the second set of experiments, we consider an objective that involves both first- and second-stage decisions when solving *SPD*. In particular, we consider the objective of maximizing the number of parcels

[1]Feasibility is measured as the percentage of parcels that can finish their sort requirement by the associated deadlines under the generated pile assignment.

that skip secondary sorting while minimizing the number of required pile dispatches from primary to secondary sort areas, formulated as

$$\underset{x,y}{\text{Maximize}} \quad \sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S:s=1}\sum_{k\in K} q^k x^k_{p,c,s} - \alpha \sum_{a\in\mathcal{A}^{Pile}}\sum_{c\in\epsilon\mathcal{C}} w^c_a. \tag{7}$$

Here, α is a user-defined weight that reflects the importance given to minimizing the number of dispatches relative to maximizing the number of parcels that skip secondary sort. In our experiments, we set $\alpha = 0.005$ to give a very high priority to having as many parcels as possible skip secondary sorting. We use the Daysort instances considered earlier to test the performance of the *SPD* formulation using objective (7) and the proposed $PA + SSD$ approach; we use objective (6) when solving the *PA* problem and Algorithm 1 when solving the *SSD* problem. Table 3 summarizes our results. Note that the run times for the *PA* formulation are as reported earlier in Table 3 and that the *SSD* algorithm runs very efficiently (in a few seconds).

Table 3: *Performance comparison between SPD and PA + SSD*

	Approach	Daysort instances (size, in parcels)						
		S2_1 (3034)	S2_2 (14635)	S2_3 (19256)	S2_4 (19669)	S2_5 (19470)	S2_6 (29504)	S2_7 (38940)
% Optimality Gap[2]	PA + SSD	0.859	1.175	1.175	0.718	2.615	0.718	2.613
	SPD (0.5 hr)	0.859	4.090	1.174	0.716	-	-	-
	SPD (3 hrs)	0.859	1.169	1.174	0.716	2.611	-	5.760
# of parcels skipping secondary sort	PA + SSD	1106	3815	5602	5748	5408	8622	10816
	SPD (0.5 hr)	1106	3708	5602	5748	-	-	-
	SPD (3 hrs)	1106	3815	5602	5748	5408	-	10494
Avg. # of dispatches per secondary sort pile	PA + SSD	1	4.06	2.53	2.68	3.75	4	3.74
	SPD (0.5 hr)	1	1.67	1.64	1.71	-	-	-
	SPD (3 hrs)	1	1	1.33	1.43	1.60	-	2.94

We ran the *SPD* formulation for 30 minutes to compare its solutions with solutions generated using the $PA + SSD$ approach, which took at most 20 minutes to solve. We also ran it for three hours as an additional benchmark. As can be seen in Table 3, we were unable to obtain feasible solutions for all of the tested instances when using the *SPD* formulation. Our results show that the *SPD* formulation provides good solutions for small- to medium-sized instances and can therefore be used for planning sort operations in hubs where dispatch activities are costly, if computationally possible. The $PA + SSD$ approach consistently has good computational performance, and is able to generate high quality solutions within reasonable computing time for all instances of realistic scale and composition (including the large-sized instances that we were unable to solve using the *SPD* formulation). Since these models are solved every few weeks in advance of operations, the $PA + SSD$ approach is shown to be an efficient and effective alternative for

[2]We use the SPD 3-hour solve as a benchmark and therefore, the reported optimality gaps for all solutions are computed using the best bounds found when solving the *SPD* model for three hours.

generating sort plans.

4.3 Operational analysis

In this subsection, we take a closer look at the characteristics of sort plans generated using our *PA* formulation and *SSD* algorithms. We focus our analysis on the Twilight and Midnight shifts, since the majority of arrivals occurs in their operational time windows. When carrying out these experiments, we consider optimizing for multiple objectives. We use a hierarchical optimization framework that addresses three objectives of interest in order of priority as described in what follows:

- *Objective 1:* maximize the number of parcels that are directly sorted using primary sort equipment without the need for secondary sort; as formally described earlier in Section 4.2
- *Objective 2:* minimize the maximum secondary sort pile size to balance workload across manual secondary sort stations.
- *Objective 3:* maximize the minimum slack time for primary sort piles that require secondary sort. The slack time can be defined as the amount of time that a pile's secondary sort start time can be delayed without delaying the pile's sort completion time beyond its deadline. Having a large slack time reduces the assignment's sensitivity to changes in arrival profiles from a feasibility stand point, and thus, generates more robust assignments.

The detailed solution procedure is described in the Appendix. The idea is to generate a primary sort pile assignment that maximizes the number of parcels that can be sorted in one pass, while also balancing the workload allocated to secondary sort stations and reducing the piles' sensitivity to changes in arrival profiles. These generated assignments are then used to derive minimum-cost dispatch schedules for every pile that requires secondary sort using Algorithm 1.

In Table 4, we summarize the number of piles that can skip the secondary sort, as well as the corresponding number of parcels that can be handled in one-pass sort without the need for manual secondary sort. On average, 47% of arrivals during evening shifts can be directly sorted using primary sort equipment. This allocation corresponds to an average of 75% of piles that do not require either dispatch plans nor manual processing for secondary sort, thereby reducing the operational costs associated with dispatch and sortation.

Taking a closer look at the generated solutions, commodities that are picked for direct sort are commodities with the highest number of arrivals in a shift. Our approach determines how many of them can be accommodated directly using primary sort resources, while ensuring that the assignment of the remaining commodities is still time feasible.

We now provide further details about the structure of the assignment of piles that require secondary sort. In our analysis, we consider the objective functions we optimized for as well as other performance metrics

Table 4: *Summary of direct sort assignment during night shift*

Instances		Assigned piles		Assigned parcels	
		Count	% total	Count	% total
Twilight Shift	S3_1	30	79%	4417	56%
	S3_2	31	82%	2212	53%
	S3_3	30	79%	3561	41%
	S3_4	29	76%	6504	45%
	S3_5	29	76%	8225	48%
	S3_6	29	76%	7934	45%
	S3_7	29	76%	7409	46%
Midnight Shift	S4_1	27	71%	7956	47%
	S4_2	28	74%	5110	44%
	S4_3	28	74%	13000	42%
	S4_4	28	74%	27603	48%
	S4_5	25	66%	31245	46%
	S4_6	28	74%	31192	51%
	S4_7	27	71%	29230	49%
Average		28	75%	13257	47%

that are of interest as summarized in Table 5. Dispatch-related results are generated using Algorithm 1, where we assume that the dispatching process is uncapacitated.

Table 5: *Summary of two-sort assignment during night shift*

Instances		Number of required piles	Avg. pile size (in parcels)	Max. pile size (in parcels)	Avg. pile slack time (in hrs)	Min. pile slack time (in hrs)	Max. number of required dispatches per pile	Avg. dispatch size per pile (in parcels)
Twilight Shift	S3_1	8	430	697	3.61	3.02	1	430
	S3_2	7	282	387	3.93	2.99	1	282
	S3_3	8	643	720	3.28	2.46	2	501
	S3_4	9	890	1295	2.92	2.53	2	557
	S3_5	9	987	1517	2.8	2.56	3	566
	S3_6	9	1091	1483	2.67	2.29	4	552
	S3_7	9	979	1347	2.81	2.41	3	479
Midnight Shift	S4_1	11	813	863	3.62	2.93	2	461
	S4_2	10	640	748	3.8	3.33	2	448
	S4_3	10	1826	1945	2.17	1.87	4	755
	S4_4	10	3012	3567	0.68	0.36	7	635
	S4_5	13	2780	3175	1.23	0.57	6	652
	S4_6	10	3048	3878	0.54	0.3	8	564
	S4_7	11	2712	3856	1.11	0.54	6	629
Average		10	1438	1820	2.51	2.01	4	537

From a *resource requirement* perspective, a weekly average of ten manual stations is required for secondary sorting during night shifts. Assuming that each manual station requires a sorter and a transporter/packer, the solutions suggest that a hub require having 20 workers on duty on average per week during night shifts

to complete secondary sort activities.

From a *workload allocation* perspective, each sorter is responsible for sorting an average of 1,438 parcels during a night shift. This allocation is clearly higher for midnight shift sorters, where each sorter is responsible for sorting up to 554 parcels per hour compared to around 300 parcels per hour allocated to twilight shift sorters. Similarly, each midnight shift transporter is responsible for up to 8 dispatches, each having an average of 592 parcels, while each twilight shift transporter is responsible for up to 4 dispatches, each having an average of 481 parcels. Such results indicate that operating a shared-resource setting, instead of the dedicated-resource setting under study, may be more cost effective since utilization can be improved for both sorters and transporters, especially during twilight shifts.

From a *plan robustness* perspective, the availability of slack times implies that manual stations are able to accommodate, to some extent, increases in their assigned workload without compromising the time feasibility of their operations. The generated solutions suggest that each manual station operating during a twilight shift has a slack time of at least 138 minutes while each manual station operating during a midnight shift has a slack time of at least 20 minutes. Assuming that expected arrival times do not change, this corresponds to the ability of each station to accommodate an additional 2100 and 430 parcels during twilight and midnight shifts, respectively, on top of their expected workload without delaying their sort completion time beyond the piles' deadlines.

A more practical way of addressing robustness is by explicitly accounting for uncertainty in arrival quantities when generating sort plans. For that, we test robust variations from the nominal cases we considered thus far. We use worst-case scaled deviations from expected arrival profiles as inputs to generate what we refer to henceforth as *robust sort plans*. A robust plan is guaranteed to remain feasible under different possible increases in expected arrival volumes using a user-defined deviation. Figure 6 shows an example of a commodity's nominal and robust arrival profiles.

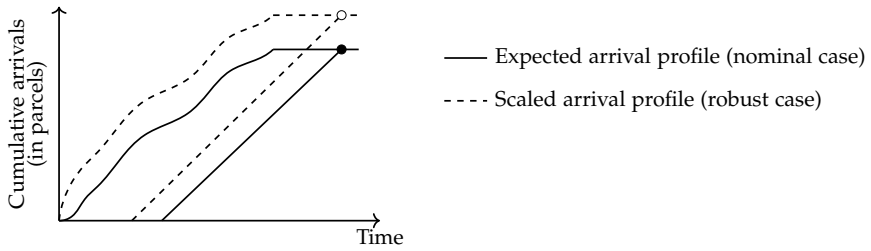

Figure 6: *An example of scaled arrival profile used as input for generating robust sort plans*

Recall that the solutions we presented so far correspond to nominal cases that rely on using expected arrival profiles, which we compare now with robust cases that rely on increasing all arrival quantities by 20% from their expected values, while keeping arrival times unchanged. Figures 7 and 8 illustrate the resource and workload allocation under both the nominal and robust cases.

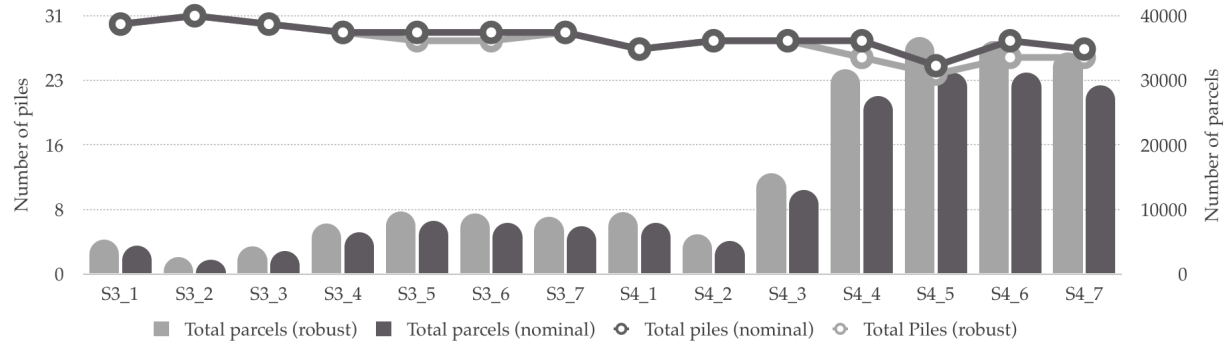

Figure 7: *Direct sort assignment*

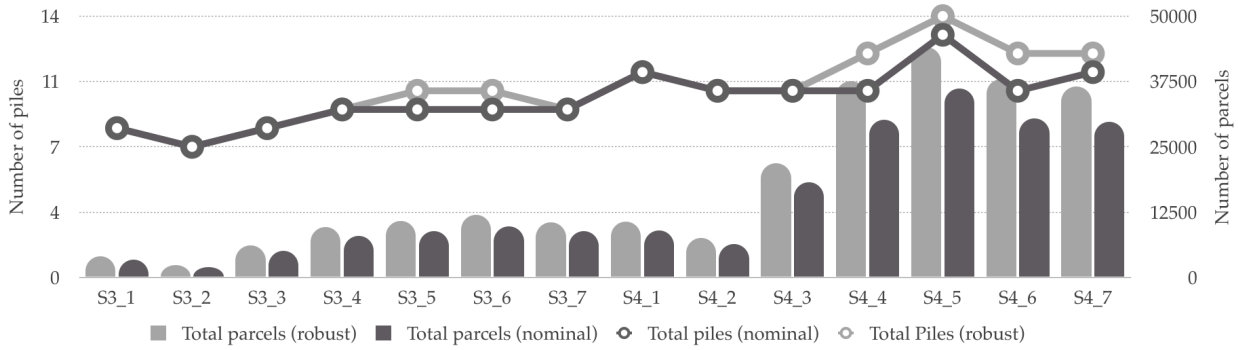

Figure 8: *Two-stage sort assignment*

As can be seen in Figures 7 and 8, introducing robustness comes at the expense of utilizing fewer primary sort pile positions for direct sort, and more manual stations for secondary sort. In some cases, it also comes at the expense of increased urgency in operations. Note that even though the arrival quantities are much smaller during the twilight shift (S3 instances) when compared to arrivals during the midnight shift (S4 instances), we still require operating a relatively similar number of two-stage piles during both shifts. This is expected as the number of commodities arriving during each of the shifts is quite similar.

Figure 9 shows the distribution of two-stage piles by deadline, where more urgent piles are created under the robust plans when compared to their nominal counterparts. In some cases, operating robust plans also corresponds to adding up to two more secondary sort stations to protect against uncertainty in arrival quantities, thereby resulting in higher operational costs. It is worth noting that the robustness approach presented here is very conservative, as it protects against all commodities taking their worst value, whereas in practice we would expect only some to do so.

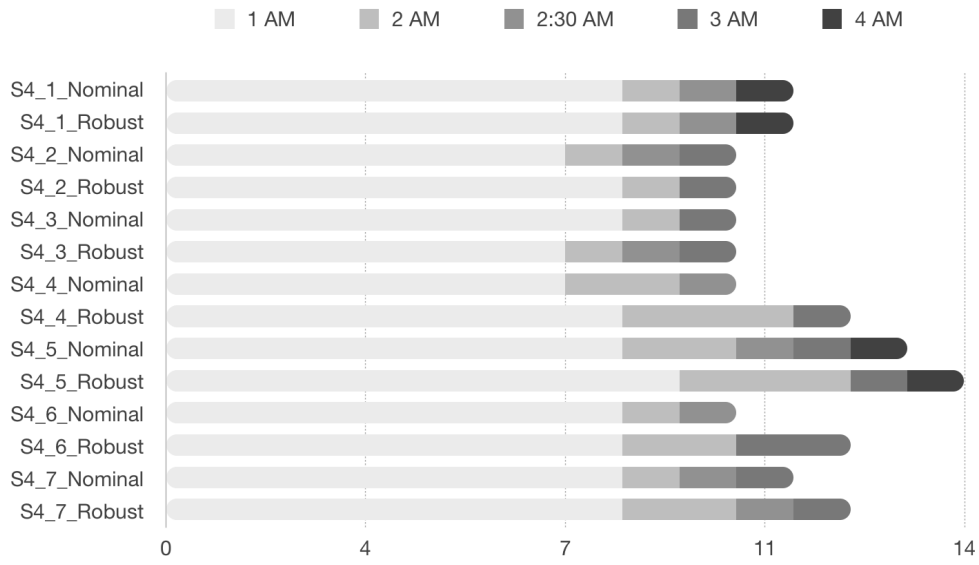

Figure 9: *two-sort pile assignment by deadline*

5 Conclusion

We have introduced a sort plan design problem in the context of two-stage sort systems operating within express parcel delivery networks. We first develop a detailed operational model based on flows in a time-space network capturing both first-stage and second-stage sort decisions. Since this model is difficult to solve for realistic-sized instances typically encountered in practice, we propose an alternative modeling approach that separates first-stage from second-stage sort decisions while importantly preserving the feasibility of secondary sort operations. We show that this alternative formulation is efficient to solve and is guaranteed to generate optimal sort plans for many practical objective functions. The proposed approach can be used by operations managers to design cost-optimal plans for primary sort systems while meeting objectives for on-time delivery. It can also be used for gaining insights into the feasibility of tactical decisions related to equipment sizing and resource requirements at the hub level.

Potential extensions of our problem could address demand uncertainty to generate plans that are cost-effective and time-feasible under different possible demand realizations. This could be done using stochastic programming or robust optimization approaches similar to the ones proposed in [20]. We showed that our approach can be easily adopted to generate robust plans that remain feasible when arrival quantities increase within pre-specified deviations from nominal values. While such robust solutions are safer to operate from a plan's time-feasibility point of view, they can be overly conservative and expensive to operate as they plan for all arrivals taking their worst value simultaneously, which is not likely. We are currently working on extensions related to the primary-sort assignment problem to generate plans that are protected

against various sources of uncertainty while allowing the user to control the desired level of conservatism using uncertainty budgets. In particular, we consider different ways of modeling demand uncertainty that take into account potential changes in arrival quantities and/or arrival times. The main goal is to generate robust models that allow hub managers to investigate trade-offs between plan robustness and the associated operational costs, to ultimately operate cost-effective sort plans that are protected against common sources of uncertainty for improved system and service performance.

Other possible extensions of our problem include: (a) investigating the value of moving from a static to a dynamic planning setting by allowing a primary sort plan to be modified once information about the actual demand is revealed, (b) integrating sort decisions with other downstream decisions related to dispatch windows, which in our case is tantamount to simultaneously optimizing for the last-mile and/or line-haul truck scheduling decisions, and (c) analyzing the impact of using wave-based release policies on balancing resource requirements in express parcel sort hubs, especially during peak operating hours.

References

[1] Alessandro Agnetis, Mohamed Ali Aloulou, and Mikhail Y Kovalyov, *Integrated production scheduling and batch delivery with fixed departure times and inventory holding costs*, International Journal of Production Research **55** (2017), no. 20, 6193–6206.

[2] Sibel Alumur and Bahar Y Kara, *Network hub location problems: The state of the art*, European journal of operational research **190** (2008), no. 1, 1–21.

[3] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Marcel Schmickerath, *Automated sortation conveyors: A survey from an operational research perspective*, European Journal of Operational Research (2018).

[4] Nils Boysen, Stefan Fedtke, and Felix Weidinger, *Truck scheduling in the postal service industry*, Transportation Science **51** (2017), no. 2, 723–736.

[5] Nils Boysen, Stefan Fedtke, and Felix Weidinger, *Optimizing automated sorting in warehouses: The minimum order spread sequencing problem*, European Journal of Operational Research **270** (2018), no. 1, 386–400.

[6] Dirk Briskorn, Simon Emde, and Nils Boysen, *Scheduling shipments in closed-loop sortation conveyors*, Journal of Scheduling **20** (2017), no. 1, 25–42.

[7] James F Campbell and Morton E O'Kelly, *Twenty-five years of hub location research*, Transportation Science **46** (2012), no. 2, 153–169.

[8] Erdem Ceven and Kevin R Gue, *Wave release policies for order fulfillment systems with multiple customer classes*, Submitted for publication (2014).

[9] Teodor Gabriel Crainic, *Service network design in freight transportation*, European Journal of Operational Research **122** (2000), no. 2, 272–288.

[10] Reza Zanjirani Farahani, Masoud Hekmatfar, Alireza Boloori Arabani, and Ehsan Nikbakhsh, *Hub location problems: A review of models, classification, solution techniques, and applications*, Computers & Industrial Engineering **64** (2013), no. 4, 1096–1109.

[11] FedEx, *Fedex express' package revenue from fy 2013 to fy 2019 (in million u.s. dollars)*, 2019.

[12] Stefan Fedtke and Nils Boysen, *Layout planning of sortation conveyors in parcel distribution centers*, Transportation Science **51** (2014), no. 1, 3–18.

[13] Jérémie Gallien and Théophane Weber, *To wave or not to wave? order release policies for warehouses with an automated sorter*, Manufacturing & Service Operations Management **12** (2010), no. 4, 642–662.

[14] SWA Haneyah, Johannes MJ Schutten, PC Schuur, and Willem HM Zijm, *Generic planning and control of automated material handling systems: Practical requirements versus existing theory*, Computers in industry **64** (2013), no. 3, 177–190.

[15] Min Ji, Yong He, and TC Edwin Cheng, *Batch delivery scheduling with batch delivery cost on a single machine*, European journal of operational research **176** (2007), no. 2, 745–755.

[16] Paul McAree, Lawrence Bodin, and Michael Ball, *Models for the design and analysis of a large package sort facility*, Networks: An International Journal **39** (2002), no. 2, 107–120.

[17] Douglas L McWilliams, *Genetic-based scheduling to solve the parcel hub scheduling problem*, Computers & Industrial Engineering **56** (2009), no. 4, 1607–1616.

[18] Douglas L McWilliams, Paul M Stanfield, and Christopher D Geiger, *The parcel hub scheduling problem: A simulation-based solution approach*, Computers & Industrial Engineering **49** (2005), no. 3, 393–412.

[19] A Noroozi, M Mazdeh, M Heydari, and M Rasti-Barzoki, *Coordinating order acceptance and integrated lot streaming-batch delivery scheduling considering third party logistics*, Uncertain Supply Chain Management **7** (2019), no. 1, 73–96.

[20] Luis J Novoa, Ahmad I Jarrah, and David P Morton, *Flow balancing with uncertain demand for automated package sorting centers*, Transportation Science **52** (2016), no. 1, 210–227.

[21] Chris N Potts and Mikhail Y Kovalyov, *Scheduling with batching: A review*, European journal of operational research **120** (2000), no. 2, 228–249.

[22] Mohammad Rostami, Omid Kheirandish, and Nima Ansari, *Minimizing maximum tardiness and delivery costs with batch delivery and job release times*, Applied Mathematical Modelling **39** (2015), no. 16, 4909–4927.

[23] Ltd. S.F. Holding Co., *2018 s.f. holding co., ltd. annual report*, 2019.

[24] Brigitte Werners and Thomas Wülfing, *Robust optimization of internal transports at a parcel sorting center operated by Deutsche Post World Net*, European Journal of Operational Research **201** (2010), no. 2, 419–426.

[25] Yunqiang Yin, TCE Cheng, Chou-Jung Hsu, and Chin-Chia Wu, *Single-machine batch delivery scheduling with an assignable common due window*, Omega **41** (2013), no. 2, 216–225.

6 Appendix

6.1 Description of Algorithm 2

We formally describe Algorithm 2 that solves the SSD problem by generating dispatch schedule for every pile the requires secondary sorting under both secondary sort and dispatch capacities.

Algorithm 2: Backward recursion algorithm with dispatch capacity threshold

Result: cost-effective pre-sort pile dispatch schedule to secondary sort area

for $p \in \mathcal{P}$ **do**

  for $c \in \mathcal{C}$ **do**

    set: $A^0_{p,c}(t) = \sum_{k\in\mathcal{K}} \sum_{t\in\mathcal{T}} q^k_t x^k_{p,c}$;

    initialize: $i \leftarrow 1$;

    initialize: $\bar{t} \leftarrow c$;

    initialize: $t \leftarrow \text{argmin}_{t\in\mathcal{T}}\{A^0_{p,c}(t)\}$;

    while $A^i_{p,c}(t) > 0$ **do**

      $S^i_{p,c}(t) = \min\{\mu_{ss}(\bar{t} - t), \text{CAP}\}$;

      $A^{i+1}_{p,c}(t) = A^i_{p,c}(t) - S^i_{p,c}(t)$;

      $\tau = \text{argmin}_{t\in\mathcal{T}}\{A^{i+1}_{p,c}(t)\}$;

      $i \leftarrow i + 1$;

      $\bar{t} \leftarrow t$;

      $t \leftarrow \tau$;

    end

    dispatch $S^i_{p,c}(t)$ parcels at time t;

  end

end

6.2 Description of the hierarchical objectives procedure

We describe in more detail the multi-objective hierarchical optimization procedure we followed when carrying out the computational experiments in Section 4.3.

Step 1. Solve the PA problem under the objective of maximizing the number of parcels that are directly sorted using primary sort equipment without the need for secondary sort. This objective can be viewed as a proxy for minimizing operational costs associated with sortation activities since all parcels have to go through at

least one sort stage for consolidation. The corresponding *PA* integer program is defined as follows.

$$\begin{aligned} v_1^* = \quad & \underset{x,y}{\text{Maximize}} \quad \sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S:s=1}\sum_{k\in K} q^k x^k_{p,c,s} \\ \text{subject to:} \quad & \text{constraints}(2b)-(2h). \end{aligned}$$

Step 2. Update the *PA* problem by appending the following constraint to its original constraints set (2b) - (2h):

$$\sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S:s=1}\sum_{k\in K} q^k x^k_{p,c,s} \geq \Delta_1 v_1^* \tag{8}$$

where $\Delta_1 \in [0,1]$ is a relative tolerance that can be used to allow degradation from the optimal objective value v_1^* as desired. In our experiments, we used $\Delta_1 = 1$. Subject to this additional constraint that places a lower bound on the amount of workload that can be sorted in one pass, a restricted version of *PA* is reoptimized under the new objective of minimizing the maximum secondary sort pile size to balance workload across manual stations modeled as follows.

$$v_2^* = \underset{x,y}{\text{Minimize}} \quad z = \max_{p,c,s=2}\sum_{k\in K} q^k x^k_{p,c,s} \tag{9}$$

Notice that equation (9) is non-linear, and therefore, we introduce $|\mathcal{P}| \times |\mathcal{C}|$ additional constraints to transform the problem into the following linear counterpart which we use for reoptimization.

$$\begin{aligned} v_2^* = \underset{x,y,z}{\text{Minimize}} \quad & z \\ \text{subject to:} \quad & \sum_{k\in K} q^k x^k_{p,c,s} \leq z \qquad \forall p\in\mathcal{P}, c\in\mathcal{C}, s=2, \\ & \text{constraints}(2b)-(2h), (8). \end{aligned}$$

Step 3. Update the *PA* problem once more by adding the following bounding constraint on the the size of each primary sort pile that requires secondary sort:

$$z \leq \Delta_2 v_2^* \tag{10}$$

where $\Delta_2 \in [0,1]$ is a relative tolerance that can be used to allow degradation from the optimal objective value v_2^* as desired. In our experiments, we used $\Delta_2 = 1$. Subject to this additional constraint, a more restricted version of *PA* is reoptimized under the objective of maximizing the minimum slack time for primary sort piles that require secondary sort. The slack time can be defined as the amount of time that a pile's secondary sort start time can be delayed without delaying the pile's sort completion beyond its

deadline. Constraint (11) provides its mathematical equivalence.

$$v_3^* = \underset{x,y,z,u}{\text{Maximize}} \quad u = \min_{p,c,s=2} cy_{p,c,s} - \frac{q^k x^k_{p,c,s}}{\mu_s s} \tag{11}$$

Having a large slack time reduces the assignment's sensitivity to changes in arrival profiles from a feasibility stand point, and thus, generates more robust assignments. Notice that constraint (11) is also non-linear. The corresponding restricted *PA* after linearization is modeled in what follows and then reoptimized to generate the final primary sort pile assignment plan.

$$\begin{aligned} v_3^* = \underset{x,y,z,u}{\text{Maximize}} \quad & u \\ \text{subject to:} \quad & cy_{p,c,s} - \frac{q^k x^k_{p,c,s}}{\mu_s s} \geq u \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \\ & \text{constraints}(2b) - (2h), (8), (10). \end{aligned}$$

The resulting solution determines a primary sort pile assignment that maximizes the number of parcels that can be sorted in one pass while balancing the workload allocated to secondary sort stations and reducing the piles' sensitivity to changes in arrival profiles.

Step 4. Given the primary sort assignment generated in step 3, run Algorithm 1 or 2 to generate a minimum-cost dispatch schedule for every pile separately. In our experiments, we ran Algorithm 1. The resulting plan determines the time of each pile dispatch, as well as an estimate for each dispatch size.