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

May 15, 2020

Recent years have brought significant changes in the operations of parcel transportation services, most notably due to the growing demand for e-commerce worldwide. Parcel sortation systems are used within sorting facilities in these transportation networks to enable the execution of effective consolidation plans with low per-unit handling and shipping costs. Designing and implementing effective parcel sort systems has grown in complexity 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 introduce 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 sort stations for final sort. We define a sort planning optimization problem in this setting using mixed-integer programming, where the objective is to minimize operational cost and workload imbalance subject to machine capacity and parcel deadline constraints. Since the full optimization problem can be difficult, we propose a decomposition approach for the problem that we show is guaranteed to generate optimal solutions for many practical objective functions. We illustrate the effectiveness of the proposed approach using real-world instances obtained from a large express delivery service provider.

1 Introduction

Parcel delivery services (PDS) 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 U.S. dollars in revenue from express parcel delivery in 2019, which corresponds to a 38% increase in its revenue from 2016 [8]. Leading players in the PDS industry are investing in capacity expansion projects to serve this growth. SF-Express, for example, expanded their ground network to 531 transit hubs in China alone [15]. To enable a fast and effective consolidation network design, 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.

Service providers operate different types of parcel sortation systems that vary mainly in terms of layout design, 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. We assume that decisions have been made at the tactical level to ensure that appropriate equipment with appropriate capacities have been installed such that feasible sorting operations exist at the hub for typical operating days.

Existing research in this area focuses mainly on design and operational planning for fully automated, single-stage sortation systems. Important decisions 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 the 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 then directed to a discharge point located on either side of the primary sort belt where it gets pushed via chute and grouped into pre-sort piles. Pre-sort piles that contain parcels with different geographic destinations and/or dispatch deadlines are then moved to secondary sort stations for detailed 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 destination and deadline. At this point, 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. Fully packed containers (often parcel bags) are finally placed on a conveyor system that transfers them to their corresponding shipping docks for loading.

For such a system, we focus on developing 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 pre-sort piles and then how parcels in pre-sort piles are moved over time to secondary sort stations for fine sorting and packing.

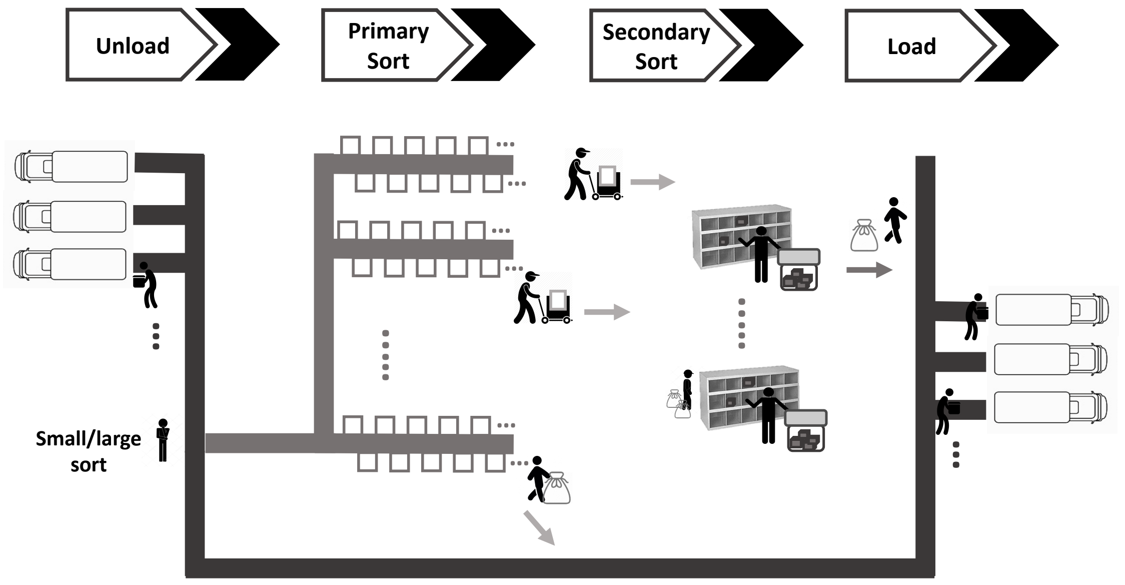

Figure 1: *An illustration of a two-stage sort process*

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 (see [1, 5, 7]) or tactical service network design problems (see [6]) 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, [2] present an extensive review of these problems with a focus on fully-automated hubs. Similarly, [10] 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 [3, 12, 13]), while only few studies address the planning of the sort process itself.

A few papers address planning of manual and semi-automated sorting processes. [11] developed 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 effectively and efficiently. Another research effort, described in [16], 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, [4] used dynamic programming to schedule the loading process for parcels onto sortation conveyors to minimize makespan. The proposed optimization procedure was found to be particularly useful when the decision space was confined to finding the assignment of shipments to loading stations, and the loading sequence was considered as fixed and given. 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. [9] 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, [14] proposed a stochastic programming approach for assigning loading destinations to secondary sort workcenters in an automated package sort facility. The objective was to generate a balanced workload assignment to workcenters while accounting for uncertainty in arrivals during every sort shift. The problem considered in this research focused 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, they proposed and tested 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 sortation 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 sort plans for two-stage sort systems by assigning parcels to sort equipment and 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 service guarantees for on-time delivery that are commonly offered in practice.
- For tractability, we propose a decomposition approach 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, commodity time feasibility) for the second-stage. The decomposition allows a detailed time-space model to be replaced by a much simpler approach to ensure time feasibility.
- 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 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 explore different formulations and present our solution approach. In Section 4, we present the results of our computational study using real-life instances obtained from an international PDS 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* problem that aims to find a cost-effective parcel assignment 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

Let $\mathcal{T}$ be a set of equally spaced discrete time points that constitutes a sort shift, and suppose that a sort plan is to be designed for 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. The primary sorting system has a fixed number of discharge points, and a *pile* of parcels accumulates at each discharge point; suppose that the primary sorter processes a maximum of μ_p parcels per unit time (measured in the discretization of $\mathcal{T}$). Let $\mathcal{P}$ be the set of primary sort piles. Furthermore, let $\mathcal{C}$ be a set of time deadlines to be associated with piles, where $\mathcal{C} \subset \mathcal{T}$.

In a two-stage sorting system, a primary sort pile may require a secondary sort before its constituent commodities are considered sorted. Let the sort mode s of a pile define its follow-on sorting requirements, where $s = 1$ indicates that a pile requires only a first-phase sort, and $s = 2$ that a pile also requires a secondary sort; let $\mathcal{S} = \{1, 2\}$ be the set of sort modes. Suppose that the hub has enough secondary sort stations to devote one to each of the primary sort piles, but that no pile can be assigned to multiple secondary stations. Each secondary sort station has n sort positions, and each commodity in the pile must be assigned to exactly one of these positions, with a different position for each commodity. A typical secondary sort station is a cabinet with n cubicles. Suppose that each secondary sort station can process μ_s parcels per unit time.

Turning now to the demands on the sortation system, let $\mathcal{K}$ be the set of parcel *commodities* arriving to the hub during the sort shift. Each commodity $k \in \mathcal{K}$ is a unique pair of final destination terminal and sort deadline time $d^k \in \mathcal{T}$. To meet on-time service expectations, all arriving parcels for commodity k must complete the sort process no later than time d^k. The parcels that comprise commodity k arrive to the sortation hub over time, so we define for each k an *arrival profile* given by a set of arrival quantities q^k_t for each time $t \in \mathcal{T}$. The total arriving quantity of commodity k during the shift is $q^k = \sum_{t \in T} q^k_t$, in parcels. Finally, in this work we assume that no queuing occurs at the primary sort: formally, $\mu_p(t_2 - t_1) \geq \sum_{k \in \mathcal{K}} \sum_{t_1 < \tau \leq t_2} q^k_\tau$ for all $t_1 < t_2$ in $\mathcal{T}$. For simplicity, then, we assume q^k_t is the arrival quantity that has completed primary sort by time t.

Given these inputs, the *sort plan design* problem determines the assignment of parcels to primary sort piles and a schedule of dispatches from primary sort to secondary sort stations such that each parcel completes the sort by its commodity deadline. The primary objective of the problem is to minimize the costs associated with sorting all parcels. Since commodity arrival patterns may vary from nominal projections, secondary objectives include balancing the workload assigned to secondary sort stations and maximizing available schedule slack time to improve the robustness of the plan.

We formulate the sort plan design problem as a mixed integer program while using the following assumptions and modeling choices.

1. *A time-space network model is used to represent parcel assignments and dispatch decisions.* Figure 2 illustrates an example of a time-space network for this problem, where each node represents a pair of sort location and a time point. Each arc represents either a dispatch arc that models the movement of parcels from one location to another, or a holding arc that models the staging of parcels at a particular location over time. For notational simplicity, we assume transfer activities take zero time, while sorting and staging activities take unit time; a slight modification to the network and model accommodate general activity times.

2. *Time is discretized.* Using a fine enough discretization is important, since the time span of sort operations is short and parcel availability in the hub is constrained by tight deadlines. However, as with any model that employs such a discretization, we must trade off accuracy against computational difficulty. The choice of time unit may be also determined by how granular the arrival profile is, and as such is

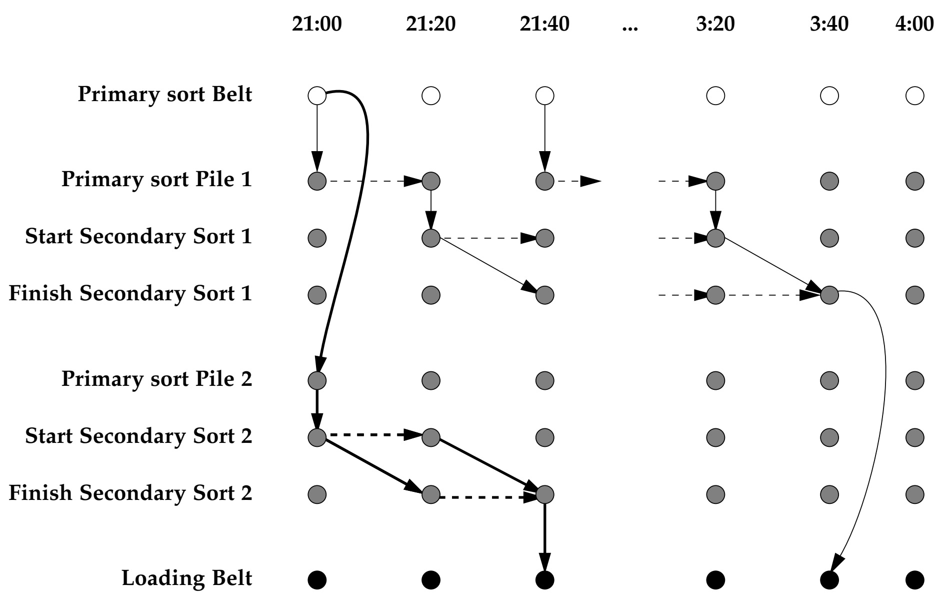

Figure 2: *An illustration of a time-space network modeling a two-stage sort operation, with solid arcs representing dispatch arcs, and dashed arcs representing holding arcs.*

also partly just an input. In our experiments, we use a 10-minute time discretization to accurately model sort operations with tight service guarantees.

3. *Commodity splitting into multiple primary sort piles is not allowed.* Each commodity must be assigned to one pile. While it may be possible to better balance workload among secondary sorting stations by splitting some commodities across piles, it is simpler to consider an operation where the primary sort piles are fixed during the entire sort and where each commodity is dedicated to a single pile.

4. *Secondary sort can only be skipped when a primary sort pile contains one commodity.* Each commodity is packed for loading in containers (bags), and only primary sort piles with one commodity can be packed without secondary sorting. Dedicating a primary sort position to a single commodity reduces the total sorting cost by removing those parcels from the secondary sort process.

2.2 Problem formulation

First, we introduce some notation for a mixed-integer programming model of this problem. We focus primarily on constraints that define the space of feasible solutions, given that there are multiple objective functions of importance. Let the parcel sort network $G = (\mathcal{N}, \mathcal{A})$ model a two-stage sort process 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 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 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 parcels' sort into containers and their dispatch for loading.

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:

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

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

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

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

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

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

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

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

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

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

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

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

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

$$f_a^c \leq \sum_{k \in K} q^k w_a^c \qquad \forall a \in \mathcal{A}^{pile} \tag{1n}$$

Constraints (1b) ensure that each commodity is assigned to a single primary sort pile, i.e. 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. Constraints (1d) and (1e) place an upper bound on the number of commodities that can be assigned to a pile. Piles that require one-pass sorting are allowed to be assigned at most one commodity, while piles that require two-pass sorting are allowed to be assigned up to n commodities, where n is the number of cubicles in the secondary sort put wall. Constraints (1f) are coupling constraints to ensure that a commodity is assigned to a pile only if the pile is used. Constraints (1g) are added to break symmetry between primary sort piles for computational efficiency. Constraints (1h) - (1j) are flow conservation constraints. Constraints (1k) 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. Constraints (1l) ensure that every parcel completes its sort requirement by its deadline. Constraints (1m) are logical constraints to ensure that a pile is either fully dispatched to its assigned secondary sort station or fully staged at its location. Finally, constraints (1n) are coupling constraints between the flow and flow indicator variables. 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.

3 Solution Approach

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. To make the problem more tractable, we propose a decomposition into two sub-problems, the *primary sort assignment* problem to determine the commodity-to-pile assignment and the *secondary sort dispatch* problem to determine the schedule of transferring parcels from the primary sort area to the secondary sort area. In this section, we formulate both problems and discuss the impact of the decomposition approach on the quality of generated solutions.

3.1 Primary sort assignment sub-problem

In this problem, we seek to determine the assignment of commodities to primary sort piles using a mixed-integer program. The *primary sort assignment* (*PA*) feasibility problem is formulated as follows. We discuss multiple objectives of interest in Section 3.4.

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

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

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

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

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

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

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

Constraints (2a) and (2c) - (2g) are identical to constraints (1b) - (1g) described earlier in Section 2.2.

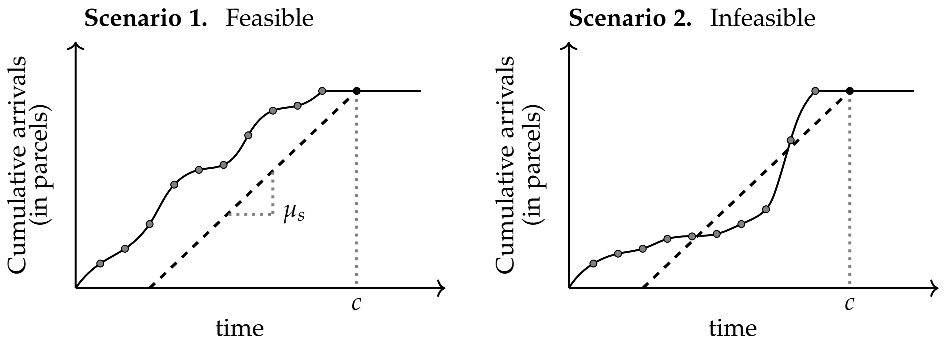

Figure 3: *An illustration of how constraints (2b) work to ensure feasible secondary sort solutions. Scenario 1 is feasible because every point on the arrival curve (grey curve) for a pile lies above its secondary-sort deadline line (black dashed), while scenario 2 is infeasible.*

Constraints (2b) ensure that the plan meets sort deadlines for commodities that are dispatched to the secondary sort. Recall that we assume no travel time is required between the primary sort and secondary sort locations. To understand these constraints, consider the set of conditions

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

where $b_{p,c,s} = \sum_{k \in \mathcal{K}} \sum_{1 \leq \tau \leq c} q_\tau^k x_{p,c,s}^k - \mu_s c$ 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 deadline line has a slope of the secondary sort rate μ_s and an intercept (not greater than zero) equal to the difference in the sort capacity $\mu_s c$ and the total number of parcels that will accumulate in the pile before the deadline c. Simplifying these inequalities yields constraint set (2b). 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 server closer to the deadline.

3.2 Secondary sort dispatch sub-problem

Given a feasible primary sort assignment from the *PA* model, the *secondary sort dispatch (SSD)* problem generates a cost-effective dispatch schedule for primary sort piles assigned more than one commodity to the secondary sort area. We assume that the dispatch process is operated manually by workers with push carts. 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 dispatch process involves workers, a reasonable proxy for minimizing operational 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. With this objective in mind, we propose a backward recursion procedure for solving the *SSD* problem, as detailed in what follows.

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 5, and described formally in Algorithm 1.

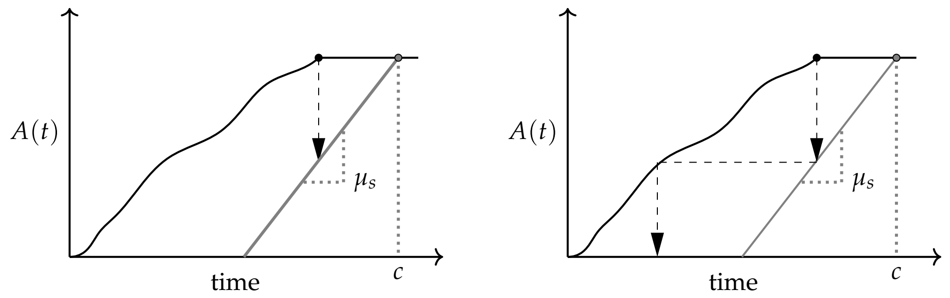

Figure 4: *An illustration of how the backward recursion algorithm works*

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. 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 1. *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. $\square$

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.

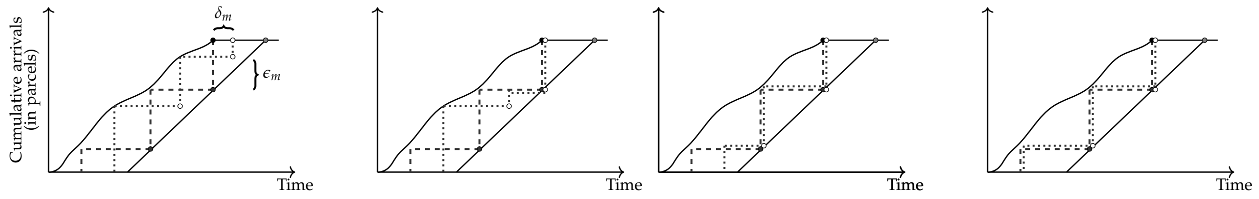

Figure 5: *An illustration of how to modify the optimal solution to one that is equivelent to the solution generated by the backward recursion algorithm while maintaining feasibility and optimality*

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 2. *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.*

3.3 The impact of decomposition

Next, we discuss the relationship between the original SPD problem and its decomposition into the PA and SSD sub-problems. Recall that the purpose of decomposition is to circumvent the computational complexity associated with solving practical-sized instances directly using SPD models. In what follows, we show that the proposed decomposition preserves the feasibility of the original SPD problem, and in fact also preserves the optimality in many cases. The problem's inputs, assumptions and modeling choices, such as how time is discretized, are identical for both the original problem and its sub-problems.

First, we establish the relationship between the PA and SPD formulations. Notice that in our SPD formulation, we use flow-based variables to ensure that the generated sort plans are service feasible. We now show that the assignment-based service feasibility constraints used in PA are valid for SPD.

Proposition 3. *Every inequality* (2b) *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, then constraints (1l) along with all the flow conservation constraints (1h)- (1j) 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 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

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

must hold and therefore, are valid for SPD. In addition, constraints (1k) 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 (1k), we establish an aggregate form of constraints (1k),

$$\sum_{a=((ss,p),\tau),(fs,p),\tau+1))\in\mathcal{A}^{ss}:t\leq\tau<c} f_a^c \leq \mu_s(c-t) \qquad \forall t\in\mathcal{T}, p\in\mathcal{P}, c\in\mathcal{C}, s=2. \tag{5}$$

Substituting by the right-hand side of (4) in (5), we obtain (2b) and, thus, the inequality is valid for SPD. □

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

Theorem 4. *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 (2b) 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 proposed in Section 3.2, 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 problem. When the objective we use in SPD depends only on the (x,y) variables, it follows that PA preserves optimality as well.

Corollary 5. *Given an objective function* (1a) *that only depends on the (x,y) variables, an optimal solution for PA can be extended into an optimal solution for SPD and vice versa.*

Note first that the *SSD* problem seeks 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* problem considers a case where the *SPD* model does not depend only on the x and y variables. It also follows that if the objective function includes the flow variables f and/or w, the solution generated by our sequential decomposition approach 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.

3.4 Using the decomposition approach with multiple objectives

To address the multi-objective nature of the *SPD* problem, we propose solving the *PA* sub-problem using a hierarchical optimization framework. In this section, we describe such an approach with three objectives. The optimized output, which includes primary sort pile assignments' to commodities, is then used to solve the *SSD* problem using one of the backward recursion algorithms proposed in subsection 3.2. We describe the procedure in more detail in what follows.

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 integer program is defined as follows.

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

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

$$\sum_{p\in\mathcal{P}}\sum_{c\in\mathcal{C}}\sum_{s\in S:s=1}\sum_{k\in K} q^k x_{p,c,s}^k \geq \delta_1 v_1^* \tag{6}$$

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. 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 solved under the objective of minimizing the maximum secondary sort pile size to balance workload across manual stations modeled as follows.

$$v_2^* = \underset{x,y}{\text{Minimize}} \quad z = \max_{p,c,s=2}\sum_{k\in K} q^k x_{p,c,s}^k \tag{7}$$

Notice that equation (7) is non-linear, and therefore, we introduce $|\mathcal{P}| \times |\mathcal{C}|$ additional constraints to transform the problem into the following linear counterpart.

$$\begin{aligned} v_2^* = \underset{x,y,z}{\text{Minimize}} \quad & z \\ \text{subject to:} \quad & \sum_{k \in K} q^k x_{p,c,s}^k \leq z \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \\ & \text{constraints}(2a) - (2g), (6). \end{aligned}$$

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:

$$z \leq \delta_2 v_2^* \tag{8}$$

Subject to this additional constraint, a more restricted version of *PA* is solved 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 (9) provides its mathematical equivalence.

$$v_3^* = \underset{x,y,z,u}{\text{Maximize}} \quad u = \min_{p,c,s=2} cy_{p,c,s} - \frac{q^k x_{p,c,s}^k}{\mu_s s} \tag{9}$$

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 (9) is also non-linear. The corresponding restricted *PA* after linearization is modeled in what follows and then solved to generate the final primary sort pile assignment plan.

$$\begin{aligned} v_3^* = \underset{x,y,z,u}{\text{Maximize}} \quad & u \\ \text{subject to:} \quad & cy_{p,c,s} - \frac{q^k x_{p,c,s}^k}{\mu_s s} \geq u \qquad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \\ & \text{constraints}(2a) - (2g), (6), (8). \end{aligned}$$

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. The resulting plan determines the time of each pile dispatch, as well as an estimate for each dispatch size.

4 Computational Study

In this section, we present the results of our computational experiments. The objective is to analyze the performance of our decomposition solution approach 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 service class.

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. We focus on the daysort, twilight

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, 15:45, 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

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.

4.2 Performance analysis

The main objective of the following experiment is to illustrate the practical usefulness of the proposed solution approach. We mainly focus on evaluating how the decomposition approach performs when compared to solving the problem directly without decomposition, i.e., when using the SPD formulation, and when using other heuristic approaches. Since the objective function of the original SPD problem involves assignment-related decisions only, we focus our analysis on the performance of the PA sub-problem part of our approach when compared to the following approaches:

- *SPD*: This approach involves solving the SPD formulation, which models both assignment and dispatch decisions using time-space network representation. The objective is to generate a plan that maximizes the number of parcels that require one-pass sorting, i.e., skipping the secondary sort.
- *First Fit Bin Packing*: This approach involves running a First Fit Bin Packing *(FF BP)* algorithm, which assigns commodities to pre-sort piles in order of priority (from earliest to latest deadline) while respecting the pre-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 pre-sort pile assignment that minimizes the number of pre-sort 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 pre-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 pre-sort piles n_{i+1}^*. If $n_{i+1}^* < n_i^*$, we repeat the process until all pre-sort piles are assigned to at least one commodity.

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

As can be seen in Table 2, both the *FF BP* and its variant were solved very efficiently, in seconds. *FF BP variant*, in particular, generated solutions thar are optimal in the number of parcels assigned to one-pass sort. However, its performance with respect to service feasibility deteriorated considerably as the instance size grew, i.e., the number of arrivals increased. The *PA* approach, on the other hand, guaranteed both optimality and service feasibility, just like SPD. It also ran 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.

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

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

Metrics	Approach	Daysort Instances (its 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 requiring one-pass sort	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%

4.3 Operational analysis

In this subsection, we take a closer look at the characteristics of the sort plans generated using our decomposition approach. We focus our analysis on the Twilight and Midnight shifts, since the majority of arrivals occurs in their operational time windows.

In Table 3, 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 pre-sort equipment. This allocation corresponds to an average of 75% of pre-sort 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 pre-sort piles that require secondary sort. In our analysis, we consider the objective functions we optimized for as well as other performance metrics that are of interest as summarized in Table 4. Dispatch-related results are generated using Algorithm 1, where we assume dispatching is uncapacitated.

From a *resource requirement* perspective, an average of 10 manual stations are required to operate

Table 3: *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%

Table 4: *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

for secondary sort during night shifts. Assuming that each manual station requires a sorter and a transporter/packer, the solutions suggest having an average of 20 workers on duty during night shifts to complete secondary sort activities.

From a *workload allocation* perspective, each sorter is responsible for sorting an average of 1438 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.

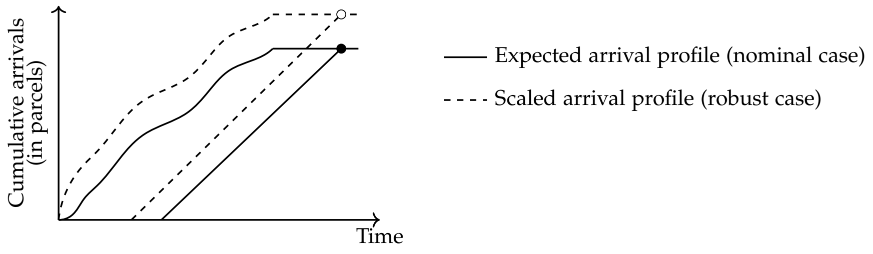

Figure 6: *An example of scaled arrival profile used as input for generating robust sort plans*

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.

As can be seen in Figures 7 and 8, introducing robustness comes at the expense of utilizing fewer pre-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. 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

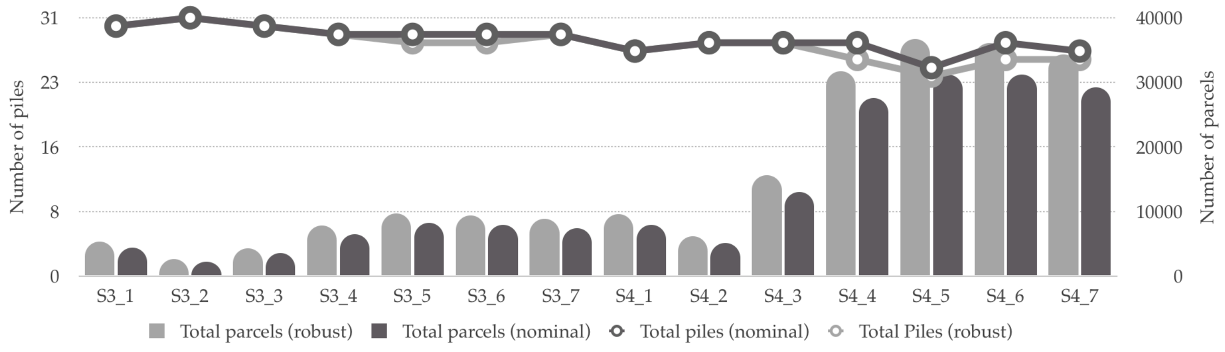

Figure 7: *Direct sort assignment*

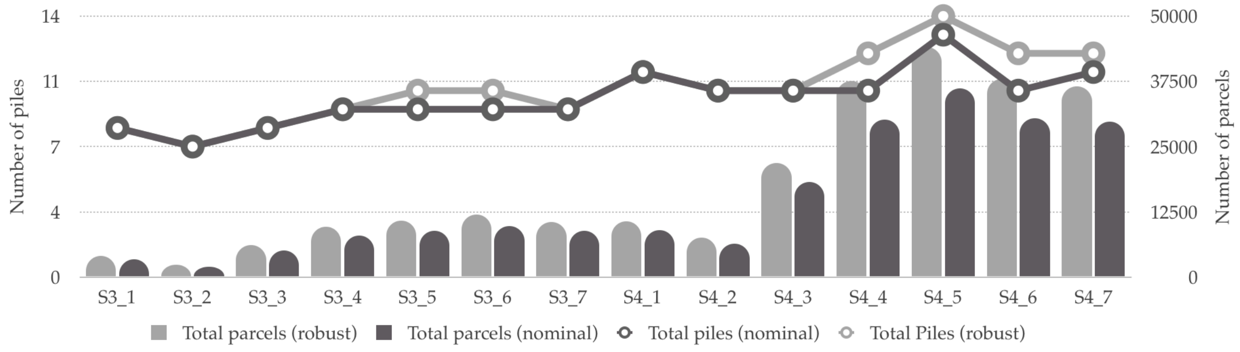

Figure 8: *Two-stage sort assignment*

counterparts.

In some cases, operating robust plans also corresponds to adding up to two more fine 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.

5 Conclusion

In this paper, we have introduced the sort plan design problem in the context of semi-automated sortation systems operating as part of express parcel delivery networks. We proposed an optimization-based decomposition approach that solves it, generating cost-effective sort plans that are time feasible with respect to parcel service delivery guarantees. We showed that our approach performs well when compared to solving the problem directly without decomposition and when using other heuristic approaches. In particular, we proved that our decomposition approach generates solutions that are guaranteed to be optimal when the objectives of interest are only dependent on assignment-related decisions. The proposed approach

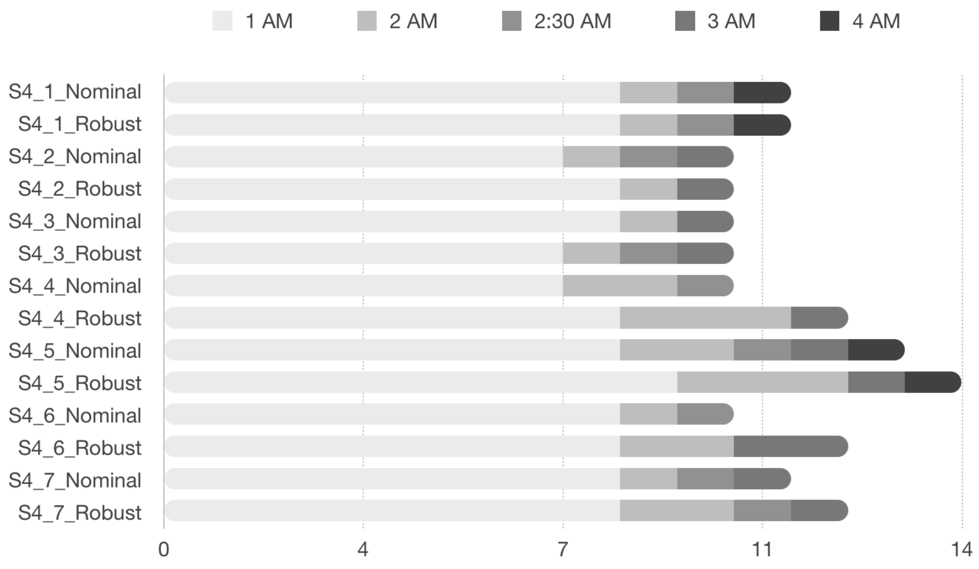

Figure 9: *two-sort pile assignment by deadline*

is not only practical for optimizing day-to-day operations, it can also be used for gaining insights into the feasibility of tactical decisions related to equipment sizing and resource requirement.

Further work includes considering a more dynamic and less conservative robust setting. We showed that our approach can be easily adopted to generate robust plans that remain feasible when arrivals' quantities increase. This approach is, however, very conservative as it plans for all arrivals taking their worst value. Therefore, it would be interesting to investigate the tradeoffs between the level of conservatism and its impact on the operational costs, along with other similar objectives. Additionally, another potential avenue to investigate is the value of moving to a dynamic setting, allowing operational decisions to be updated as more parcel arrival data is revealed.

Other interesting extensions of our problem include: (i) integrating sort decisions with other downstream decisions related to dispatch windows, which in our case is tantamount to optimizing for the last-mile and/or line-haul truck scheduling decisions, and (ii) introducing wave-based release policies to examine their impact on balancing resource requirements, especially in peak hours.

References

[1] 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.

[2] 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).

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

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

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

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

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

A Appendix

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
```