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
August 21, 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 primary objective is to minimize operational cost subject to machine capacity and parcel deadline constraints. Since the full optimization problem can be difficult to solve, we propose an alternative problem formulation 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 piles. 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 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 (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 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 approach while ensuring 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 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 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

The volume of parcels arriving to a sortation hub varies considerably by time-of-day depending on parcel’s service class and final destination. For that, sort hub operations are generally organized following a shift structure that takes into account such variations for better planning of resource requirements. Let \( \mathcal{T} \) be a set of equally spaced discrete time points that constitutes a given 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. 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 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 $S = \{1, 2\}$ be the set of sort modes. Furthermore, let $C$ be a set of time deadlines to be associated with piles, where $C \subset T$. A pile with deadline $c \in C$ requires completing the sort requirement of all its assigned parcels’ no later than its time deadline $c$.

The secondary sort area consists of multiple stations that are manually operated. A typical secondary sort station is a cabinet with $n$ sort positions (cubicles), and each commodity in pile with sort mode $s = 2$ must be assigned to exactly one of these positions, with a different position for each commodity. 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. Additionally, suppose that the primary sorter processes a maximum of $\mu_p$ parcels per unit time, and each secondary sort station can process up to $\mu_s$ parcels per unit time (measured in the discretization of $T$).

Turning now to the demand side on the sortation system, let $K$ be the set of parcel commodities arriving to the hub during the sort shift. Each commodity $k \in K$ is a unique pair of final destination terminal and sort deadline time $d^k \in T$. 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 commodity $k$ an arrival profile given by a set of arrival quantities $q^k_t$ for each time $t \in T$. The total arriving quantity of commodity $k$ during the shift is $q^k = \sum_{t \in T} q^k_t$, in parcels. A commodity with deadline $d^k$ can only be assigned to piles with deadline $c \leq d^k$. Additional commodity-to-pile compatibility rules can also be considered and implemented in preprocessing steps. For instance, a commodity can be assigned to a pile if its last arrival occurs at least e.g. 20 minutes before the pile deadline $c$.

Our primary focus is to generate plans that determine the assignment of commodities to primary sort piles and their corresponding feasible dispatch schedule to secondary sorting. For that, we assume that the unloading process to primary sort area has already been planned and therefore, we do not model the sequencing of unloading and queuing operations prior to and during primary sort operations; we assume instead that $\mu_p(t_2 - t_1) \geq \sum_{k \in K} \sum_{t_1 < \tau \leq t_2} q^k_\tau$ for all $t_1 < t_2$ in $T$. For simplicity, then, we assume $q^k_t$ is the arrival quantity that has completed primary sort by time $t$.

Finally, we use the following key assumptions and modeling choices when building our models.

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

2. *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.

3. *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.

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. We focus primarily on constraints that define the space of feasible solutions, given that there are multiple objective functions of importance that are explained in more details in later Sections.

### 2.2 The Sort Plan Design (SPD) problem formulation

In this formulation, 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 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.

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 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 $a = ((p, t), (p, t + 1)) \in A^{pile}$ represent the staging of parcels in their pile location from time $t$ to time $t + 1$. Similarly, arcs $a = (((p, ss), t), ((p, fs), t + 1)) \in A^{ss}$ represent the secondary sort of pile $p$ parcels starting at time $t$ and finishing at time $t + 1$ while arcs $a = (((p, fs), t), (load, t)) \in A^{fs}$ represent the completion of parcels’ sort into containers and their dispatch for loading. Moreover, we define $\delta^+(i)$ and $\delta^-(i)$ to be the set 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 $f_{a,c}^c \in \mathbb{Z}^+$ denote the flow on arc $a$ with deadline $c$, in parcels, and $w_{a,c}^c \in \{1, 0\}$ denote whether or not there is a positive flow in arc $a \in A^{pile}$ with deadline $c$.

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

Minimize $C(x, y, f, w)$

subject to

$$\sum_{s \in S} \sum_{p \in P} \sum_{c \in C} x_{k,p,c,s} = 1 \quad \forall k \in K,$$  \hspace{1cm} (1b)

$$\sum_{p \in P} \sum_{c \in C} \sum_{s \in S} y_{p,c,s} \leq |P| \quad \forall p \in P, c \in C, s = 2,$$  \hspace{1cm} (1c)

$$\sum_{k \in K} x_{k,p,c,s} \leq n y_{p,c,s} \quad \forall p \in P, c \in C, s = 2.$$  \hspace{1cm} (1d)

$$\sum_{k \in K} x_{k,p,c,s} \leq y_{p,c,s} \quad \forall p \in P, c \in C, s = 1.$$  \hspace{1cm} (1e)
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. Constraints (1d) and (1e) 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 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 \( \mu_s \) is the processing rate of the secondary sort stations measured in parcels per time period. Constraints (1l) ensure that every parcel completes its sort requirement by its deadline. Constraints (1m) 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 optional and applicable only when there is no binding dispatch capacity. Finally, constraints (1n) are coupling constraints between the flow and flow indicator variables.
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. Recall that in the \( SPD \) problem, pile assignment decisions are critical decisions as they have direct impact on costs associated with sorting activities. For example, piles assigned single commodities are more cost effective as they do not require additional manual dispatch and secondary sorting. Pile assignment decisions are also critical to the feasibility of flow decisions. Specifically, piles with multiple commodities must be grouped to ensure all assigned parcels are sorted while respecting secondary sort capacities and commodity deadlines.

In this section, we propose an alternative formulation that focuses on generating pile assignments that are guaranteed to induce feasible flows given secondary sort capacities and parcel deadlines. We discuss how the generated solutions compare to solutions generated using the \( SPD \) problem described earlier from both feasibility and optimality perspectives. Since the output of the proposed model is guaranteed to induce feasible flows, we also propose algorithms to aid in generating dispatch schedules from primary to secondary sort areas for piles that requires secondary sorting.

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.

Minimize \( D(x, y) \)

subject to

\[
\begin{align*}
\sum_{s \in S} \sum_{p \in P} \sum_{c \in C} x_{p,c,s}^k &= 1 & \forall k \in K, & \tag{2a} \\
\sum_{k \in K} \sum_{t+1 \leq \tau \leq c} q_{p,c}^k x_{p,c,s}^k &\leq \mu_s(c - t) & \forall t \in T, p \in P, c \in C, s = 2, & \tag{2b} \\
\sum_{p \in P} \sum_{c \in C} \sum_{s \in S} y_{p,c,s} &\leq |P|, & \forall k \in K, & \tag{2c} \\
\sum_{k \in K} x_{p,c,s}^k &\leq n y_{p,c,s} & \forall k \in K, p \in P, c \in C, s = 2, & \tag{2d} \\
\sum_{k \in K} x_{p,c,s}^k &\leq y_{p,c,s} & \forall k \in K, p \in P, c \in C, s = 1, & \tag{2e} \\
x_{p,c,s}^k &\leq y_{p,c,s} & \forall k \in K, p \in P, c \in C, s = 2, & \tag{2f} \\
y_{p,c,s} &\geq y_{p+1,c,s} & \forall p \in P, c \in C, s \in S & \tag{2g}
\end{align*}
\]
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) - (1g) 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

$$
\sum_{k \in K} \sum_{1 \leq \tau \leq t} q_{p, c, s}^k x_{p, c, s}^k \geq \mu_s t + b_{p, c, s} \quad \forall t, c, t \in T, p \in P, c \in C, s = 2,
$$

where $b_{p, c, s} = \sum_{k \in K} \sum_{1 \leq \tau \leq c} q_{p, c, s}^k - \mu_s c$ 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 $\mu_s$ 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 (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.

The presented form of constraints (2c) assumes that there is no required travel time between the primary sort and secondary sort locations. This assumption, however, can be easily modified when travel times are not negligible. For instance, if it takes $X$ time periods (measured in the discretization $T$) 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 losses due to travel times.
3.2 Relationship to the SPD problem formulation

Next, we discuss the relationship between the original SPD problem and the alternative PA formulation. We show that the PA formulation 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 SPD problem and the PA problem.

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, then the time feasibility 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 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

\[
\sum_{a \in A^{ss} : t \leq \tau < c} f_a^c \geq \sum_{k \in K} \sum_{t+1 \leq \tau \leq c} q_k^{p,s,t} x_{p,s,t}^c \quad \forall t \in T, p \in P, c \in C, s = 2
\]  

(4)

where \(a = ((ss, p), \tau), (fs, p), \tau + 1\)) 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),

\[
\sum_{a \in A^{ss} : t \leq \tau < c} f_a^c \leq \mu_s(c - t) \quad \forall t \in T, p \in P, c \in C, s = 2.
\]  

(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. \(\square\)

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.

Since we are mainly concerned about pile assignment decisions, a primary objective of interest is to maximize 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. It also depends on assignment-related decisions only, making the alternative PA formulation suitable for replacing the intractable SPD formulation discussed earlier. Since commodity arrival patterns may vary from nominal projections, other secondary objectives can also be considered including balancing the workload assigned to secondary sort stations and maximizing available schedule slack time to improve the robustness of the plan; we discuss these 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 feasible operational flows, 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. 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, 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.

With this objective in mind, we propose backward recursion algorithms for solving the Secondary Sort Dispatch (SSD) problem, as detailed in what follows. Note that in these algorithms we seek to minimize the dispatch costs into secondary sort, and that dispatch decisions are represented by variables \( w_a \) 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 \( \mu_s \) 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. 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 on. 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: \( A_{p,c}^0(t) = \sum_{k \in K} \sum_{t \in T} q_{k,t}^p \cdot x_{k,p,c}^t \);
    initialize: \( i \leftarrow 1 \);
    initialize: \( \bar{t} \leftarrow c \);
    initialize: \( t \leftarrow \text{argmin}_{t \in T} \{A_{p,c}^0(t)\} \);
    while \( A_{p,c}^i(t) > 0 \) do
      \( S_{p,c}^i(t) = \mu_s(\bar{t} - t) \);
      \( A_{p,c}^{i+1}(t) = A_{p,c}^i(t) - S_{p,c}^i(t) \);
      \( \tau = \text{argmin}_{t \in T} \{A_{p,c}^{i+1}(t)\} \);
      \( i \leftarrow i + 1 \);
      \( \bar{t} \leftarrow t \);
      \( t \leftarrow \tau \);
      dispatch \( S_{p,c}^i(t) \) 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.

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 alternative formulation 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.

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

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

4.2 Performance analysis

The main goal of the following experiment is to illustrate the practical usefulness of the proposed alternative formulation. We mainly focus on evaluating how the PA formulation performs when compared to solving the problem directly using the SPD formulation, and when using other heuristic approaches. As a proxy for minimizing costs associated with sort activities, the primary objective we consider when solving the PA and SPD formulations is maximizing the number of parcels that are directly sorted using primary sort equipment as all parcels have to go through at least one sort stage for consolidation. More formally, the objective function we consider is:

\[
\text{Maximize } x, y \sum_{p \in P} \sum_{c \in C} \sum_{s : s = 1} \sum_{k \in K} q^k_{p,c,s}
\]

We focus our analysis on the performance of the PA problem formulation when compared to the following approaches:

- **SPD**: This approach involves solving the SPD formulation, which models both assignment and dispatch decisions using time-space 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 as described earlier.

- **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.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Approach</th>
<th>Daysort Instances (its size, in parcels)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S2_1</td>
<td>S2_2</td>
</tr>
<tr>
<td></td>
<td>(3034)</td>
<td>(14635)</td>
</tr>
<tr>
<td>Run time (in seconds)</td>
<td>FF BP</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>FF BP Variant</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>SPD</td>
<td>1437</td>
</tr>
<tr>
<td></td>
<td>PA</td>
<td>1250</td>
</tr>
<tr>
<td># of parcels requiring one-pass sort</td>
<td>FF BP</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>FF BP Variant</td>
<td>1106</td>
</tr>
<tr>
<td></td>
<td>SPD</td>
<td>1106</td>
</tr>
<tr>
<td></td>
<td>PA</td>
<td>1106</td>
</tr>
<tr>
<td># of required manual secondary sort stations</td>
<td>FF BP</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>FF BP Variant</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>SPD</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>PA</td>
<td>11</td>
</tr>
<tr>
<td>Feasibility level</td>
<td>FF BP</td>
<td>100.0%</td>
</tr>
<tr>
<td></td>
<td>FF BP Variant</td>
<td>100.0%</td>
</tr>
<tr>
<td></td>
<td>SPD</td>
<td>100.0%</td>
</tr>
<tr>
<td></td>
<td>PA</td>
<td>100.0%</td>
</tr>
</tbody>
</table>

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 that are optimal in the number of parcels assigned to one-pass sort. However, its performance with respect to time 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 time 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.

### 4.3 Operational analysis

In this subsection, we take a closer look at the characteristics of sort plans generated using our PA formulation and SDD 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

---

1. measured as the percentage of parcels that can finish their sort requirement by the associated deadlines under the generated pile assignment.
• **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 standpoint, and thus, generates more robust assignments.

The detailed solution procedure is described in the Appendix. The idea here 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. This generated assignments are then used to derive minimum-cost dispatch schedule for every pile that requires secondary sort using Algorithm 1 described in Section 3. We discuss the results of our experiments in what follows.

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 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.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Assigned piles</th>
<th>Assigned parcels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>% total</td>
</tr>
<tr>
<td>Twilight Shift</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3_1</td>
<td>30</td>
<td>79%</td>
</tr>
<tr>
<td>S3_2</td>
<td>31</td>
<td>82%</td>
</tr>
<tr>
<td>S3_3</td>
<td>30</td>
<td>79%</td>
</tr>
<tr>
<td>S3_4</td>
<td>29</td>
<td>76%</td>
</tr>
<tr>
<td>S3_5</td>
<td>29</td>
<td>76%</td>
</tr>
<tr>
<td>S3_6</td>
<td>29</td>
<td>76%</td>
</tr>
<tr>
<td>S3_7</td>
<td>29</td>
<td>76%</td>
</tr>
<tr>
<td>Midnight Shift</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4_1</td>
<td>27</td>
<td>71%</td>
</tr>
<tr>
<td>S4_2</td>
<td>28</td>
<td>74%</td>
</tr>
<tr>
<td>S4_3</td>
<td>28</td>
<td>74%</td>
</tr>
<tr>
<td>S4_4</td>
<td>28</td>
<td>74%</td>
</tr>
<tr>
<td>S4_5</td>
<td>25</td>
<td>66%</td>
</tr>
<tr>
<td>S4_6</td>
<td>28</td>
<td>74%</td>
</tr>
<tr>
<td>S4_7</td>
<td>27</td>
<td>71%</td>
</tr>
<tr>
<td>Average</td>
<td>28</td>
<td>75%</td>
</tr>
</tbody>
</table>

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 that are of interest as summarized in Table 4. Dispatch-related results are generated using Algorithm 1, where we assume dispatching is uncapacitated.

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

From a resource requirement perspective, an average of ten manual stations are required to operate 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.

![Cumulative arrivals (in parcels)](time)

<table>
<thead>
<tr>
<th>Time</th>
<th>Cumulative arrivals (in parcels)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Expected arrival profile (nominal case)</td>
</tr>
<tr>
<td></td>
<td>Scaled arrival profile (robust case)</td>
</tr>
</tbody>
</table>

**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.

![Resource and workload allocation](figure)

**Figure 7:** Direct 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. 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.
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 approach that solves it, generating cost-effective sort plans that are time feasible with respect to parcel service delivery guarantees. We showed that our alternative formulation that separates first-stage sort decisions from second-stage sort decisions performs well when compared to solving the full problem directly using a detailed time-space model that is difficult to solve, and when using other heuristic approaches. In particular, we proved that our alternative formulation generates solutions that are guaranteed to be optimal when the objectives of interest are only dependent on assignment-related decisions. The proposed approach 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. It would be interesting to investigate both the tradeoffs between the level of conservatism and its impact on the operational costs, along with other similar objectives, as well as the uncertainty present in arrival times. 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


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 $p \in P$ do
  for $c \in C$ do
    set: $A_{p,c}^0(t) = \sum_{k \in K} \sum_{t \in T} q_k x_{p,c,k}^t$;
    initialize: $i \leftarrow 1$;
    initialize: $i \leftarrow c$;
    initialize: $t \leftarrow \text{argmin}_{t \in T} \{ A_{p,c}^0(t) \}$;
    while $A_{p,c}^i(t) > 0$ do
      $S_{p,c}^i(t) = \min \{ \mu_{ss}(i-t), \text{CAP} \}$;
      $A_{p,c}^{i+1}(t) = A_{p,c}^i(t) - S_{p,c}^i(t)$;
      $\tau = \text{argmin}_{t \in T} \{ A_{p,c}^{i+1}(t) \}$;
      $i \leftarrow i + 1$;
      $i \leftarrow t$;
      $t \leftarrow \tau$;
    end
    dispatch $S_{p,c}^i(t)$ 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.

$$v_1^* = \max_{x,y} \sum_{p \in P} \sum_{c \in C} \sum_{s \in S} \sum_{k \in K} q_k x_{p,c,s}^k$$

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

\[
\sum_{p \in \mathcal{P}} \sum_{c \in \mathcal{C}} \sum_{s=1}^{s} \sum_{k \in \mathcal{K}} q^k x^k_{p,c,s} \geq \Delta_1 v_1^i
\]

where \( \Delta_1 \in [0, 1] \) is a relative tolerance that can be used to allow degradation from the optimal objective value \( v_1^i \) 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.

\[
v_2^i = \text{Minimize} \quad z = \max_{p,c,s=2} \sum_{k \in \mathcal{K}} q^k x^k_{p,c,s}
\]

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 which we use for reoptimization.

\[
v_2^i = \text{Minimize} \quad z
\]

subject to:

\[
\sum_{k \in \mathcal{K}} q^k x^k_{p,c,s} \leq z \quad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2,
\]

constraints (2b) - (2h), (6).

**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:

\[
z \leq \Delta_2 v_2^i
\]

where \( \Delta_2 \in [0, 1] \) is a relative tolerance that can be used to allow degradation from the optimal objective value \( v_2^i \) 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 (9) provides its mathematical equivalence.

\[
v_3^i = \text{Maximize} \quad u = \min_{p,c,s=2} cy_{p,c,s} - \frac{q^k x^k_{p,c,s}}{\mu_s s}
\]

Having a large slack time reduces the assignment’s sensitivity to changes in arrival profiles from a feasibility standpoint, 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 reoptimized to generate the final primary sort pile assignment plan.

\[
v_3^c = \max_{x,y,z,u} \quad u
\]

subject to:  \( cy_{p,c,s} - \frac{q^k x_{p,c,s}}{\mu_s} \geq u \quad \forall p \in \mathcal{P}, c \in \mathcal{C}, s = 2, \)

constraints \( [2h] - [2h], [6], [6]. \)

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.