Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

A SIMULATION DRIVEN OPTIMIZATION ALGORITHM FOR SCHEDULING SORTING CENTER OPERATIONS

Supratim Ghosh
Aritra Pal
Prashant Kumar
Ankush Ojha
Aditya A. Paranjape
Souvik Barat
Harshad Khadilkar

Tata Consultancy Services Research
Galaxy Business Park, Sector - 62
Noida - 201309, UP, India

ABSTRACT

Parcel sorting operations in logistics enterprises aim to achieve a high throughput of parcels through sorting centers. These sorting centers are composed of large circular conveyor belts on which incoming parcels are placed, with multiple arms known as chutes for sorting the parcels by destination, followed by packing into roller cages and loading onto outbound trucks. Modern sorting systems need to complement their hardware innovations with sophisticated algorithms and software to map destinations and workforce to specific chutes. While state of the art systems operate with fixed mappings, we propose an optimization approach that runs before every shift, and uses real-time forecast of destination demand and labor availability in order to maximize throughput. We use simulation to improve the performance and robustness of the optimization solution to stochasticity in the environment, through closed-loop tuning of the optimization parameters.

1 INTRODUCTION

A sorting center is a sophisticated cyber-physical system comprising of infeeds, conveyor belts, different kinds of chutes, workforce, and outfeeds. For optimal operation, packages should move through the conveyor belt and the chutes in as little time as possible, with the additional goal of avoiding rejected parcels (ones that the automated system is unable to route, thus requiring human intervention).

1.1 Motivation for this work

The parcel delivery industry was worth 500 billion USD in 2019 (Research and Markets 2020), with further growth due to the pandemic in 2020. The large growth in an already high-demand industry has led to stress on existing infrastructure. Sorting centers are critical components of the parcel delivery logistics industry (Boysen, Briskorn, Fedtke, and Schmickerath 2019), since they are the points in the network where incoming parcels are aggregated and then segregated into onward destinations (see Section 2 for details). The current practice in sorting terminal operations is largely manual and experience-driven. In order to maximize the throughput of these centers while retaining the existing infrastructure, one can attempt to optimize the sorting logic and workforce allocation to ensure a smooth flow of parcels. Such an algorithm is described in Section 3. However, the stochasticity inherent in real-world operations cannot be completely captured in a formal optimization approach. Therefore, we use a high-fidelity simulation of sorting center

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

operations (see Section 3) to tune the constraints and parameters of the formulation, in order to ensure robustness (in the control theoretic sense, (Sastry and Bodson 2011)) while maximising throughput.

1.2 Literature Review

Optimization of a sorting center can be considered as a highly coupled, multi-level optimization problem. At the highest level, one can optimize the layout of the sorting center to maximize the efficiency (Werners and Wülfing 2010). At the next level (with a given sorting center layout), the optimization problems of interest typically involve (i) mapping chutes to destinations, (ii) mapping labor to chutes, and (iii) mapping individual parcels to chutes. This *sort plan* is currently prepared in state-of-the-art applications once every few weeks based on expected demand (Jarrah, Qi, and Bard 2016; Novoa, Jarrah, and Morton 2018). However, to handle higher demand variability and with the benefit of earlier insight into demand through advanced sensing and forecasting, one can produce better solutions with smaller planning horizons.

At a fundamental level, the sorting problem bears resemblance to well-known queueing problems (Coffman Jr., Gilbert, Greenberg, Leighton, Robert, and Stolyar 1995). To the best of our knowledge, theoretical tools used to study such ring networks have found little application in sorting, and this is very likely because they do not prescribe any particular optimal algorithm. Instead, most of the literature on sorting has focused on linear programming and its variants. For instance, a robust planning approach to two-stage sorting operations is described in (Khir, Erera, and Toriello 2021). Their solution is robust to demand stochasticity from known uncertainty sets, but excludes labor assignment and assumes a fixed upper bound for the total number of parcels assigned to a chute. However, the real problem is dynamic because parcels are often loaded onto roller cages and cleared from the chute, thus creating space for additional parcels. It is worth noting that learning-based solutions have yet to be investigated in the literature for the sorting problem. However, reinforcement learning has been used for related problems such as inbound and outbound truck loading (Shahmardan and Sajadieh 2020). A recent study (Kim, Choi, Hwang, Kim, Hong, and Han 2020) uses reinforcement learning for the sortation problem, though this is defined as the task of routing parcels from 'emitters' to 'removers' through a grid. The definition is thus closer to a routing problem than the one considered in this study.

In this paper, we consider the combined problem of destination, parcel, and workforce mapping to individual chutes with the objective of maximising throughput with minimal disruptions (parcel rejections and chute blockages). Workforce mapping is an aspect that is often overlooked in literature. The problem involves assignment of ℓ near-identical laborers to k chutes for a fixed time interval. Such problems can be solved in a static (as against sequential) setting using well-established techniques such as auctions (Vickrey 1961; Ausubel 2000) or linear programming. However, the combination with destination and parcel mapping increases the complexity substantially. Given the high complexity and degree of coupling, we develop a simulation scheme for driving the optimization formulation towards realistic, implementable solutions. Among prior simulation studies, there are a few that use a software called FlexSim to simulation sorting center operations (Li, Hong, Zheng, and Chen 2009; Zhang and Tian 2017). However, the flexibility of this software is not sufficient for the present work. In particular, we need to establish an iterative optimization-simulator loop for fine-tuning the solutions, as described in Section 3.

1.3 Contributions

In the present work, we focus on a single-stage automated sorting center similar to the description in (Fedtke and Boysen 2017), which discusses a number of design alternatives. A schematic layout is shown in Figure 1, and a detailed problem description is provided in the next section. We solve the three individual sub-problems listed above (destination-chute mapping, labor allocation and parcel-chute mapping) separately, allowing us to handle multiple time scales inherent in the problem. Our contributions are:

1. Optimization algorithms for an offline destination→chute plan and online parcel→chute mapping;
2. Integrating labor force allocation into the sorting center optimization problem;

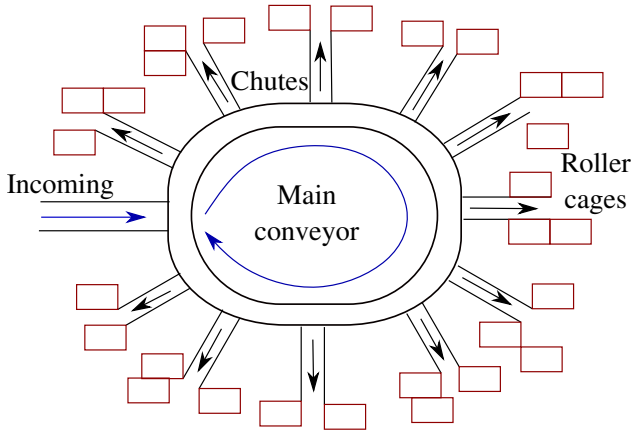

Figure 1: Schematic plan of a sorting center, showing the main oval shaped conveyor belt and outgoing chutes. Each chute is associated with a fixed number of roller cages, and each roller cage may be assigned to at most one destination, though multiple roller cages can serve the same destination.

3. Using a simulation environment (digital twin) to (i) validate the optimization algorithms, and (ii) to iteratively improve the solution performance and robustness to stochasticity.

2 PROBLEM DESCRIPTION

In this section, we describe the specific sorting center layout assumed for the present work, and also describe the sorting problem in words. A mathematical treatment of the same is covered in the next section.

2.1 System Layout

A schematic of the layout is shown in Figure 1, adapted from (Fedtke and Boysen 2017). Parcels arrive at the sorting terminal via inbound trucks, and are processed in *waves*. Each wave consists of a batch of incoming parcels for which a sort plan is computed. A complete shift (8 hours) typically consists of multiple waves (often around 10). The objective of the system is to sort the incoming parcels by destination, and place them in roller-cages which are then loaded onto trucks for outbound transport.

The parcels to be processed in a wave are first processed by an OCR reader (not shown in the figure), which captures the intended destinations and the dimensions of the parcels. After this, the parcels enter the main conveyor. The system consists of multiple chutes, each connecting the conveyor (upper level) to a loading area with roller cages (lower level). Each parcel is diverted from the conveyor into the chute after allocation by an automated system for processing (placing into a roller-cage for onward transportation to its destination). We consider two types of chutes in this paper:

- *Spiral Chutes:* These chutes consist of long metal tubes connecting the two levels. On the lower level, typically a human sorter or a robot picks up the parcel when it arrives and places it in the relevant roller cage according to destination. The number of destinations that this type of chute can handle is limited only by the number of roller cages it can hold.
- *Direct chutes:* These chutes cater to exactly one location and are *unsupervised*. In these chutes, the parcels are dropped from the upper level directly into roller-cages which are placed below (one cage per chute). Once, the cage is filled, it is immediately replaced by a new one. Direct chutes are usually reserved for destinations which have very high forecasted loads.

Each parcel on the main conveyor is either allocated a chute or is sent directly to the rejection chute (if no feasible chute is found). In addition, if a parcel is not able to enter a chute at the first attempt (due to chute

blockage or dimension incompatibility), it can make a fixed number of re-attempts before being diverted to the rejection chute. The total number of parcels processed (packed into roller-cages and sent off for transportation) represents the *throughput* of the sorting system and is measured in *pph* (parcels per hour).

2.2 The Sorting Problem

The objective of our current work is to maximize the throughput of the sorting system subject to physical constraints on chute allocation, chute capacity, human resource allocation and capacity, and avoiding chute blockages. We solve the optimal sorting problem in two stages, *planning* and *execution*:

- *Planning:* This stage takes the details of the projected load profile as input (total demand for each destination) and produces a matching between destinations and chutes *for the entire shift* in such a way that the total number of parcels processed is maximized. Note that the matching is many-to-many: one chute can handle multiple destinations (equal to number of roller cages at that chute), and one destination can be mapped to multiple chutes. Special cases with constraints include (i) direct chutes, and (ii) layout constraints. Both are described in detail in the next section. Clearly, the forecast is noisy and other unexpected events may occur after planning. Thus a second online execution phase is necessary.
- *Execution:* Once the planning is done and the parcels start arriving, the execution stage starts. This stage comprises of three parts: (i) scanning a parcel to get its dimensions and intended destination, (ii) allocating the parcel to a specific chute, and (iii) final processing after parcel comes out of the chute using roller-cages and trucks. The optimization algorithm performs the second part; i.e., it allocates specific chutes to each parcel in the upcoming wave (typically composed of thousands of parcels). This is in contrast to the planning stage, which only mapped destinations to a subset of chutes. The input to the algorithm is the destination for each parcel and the designated chutes for the destination as defined by the planning stage. Other constraints such as maximum parcel holding capacity of each chute are also considered, and so is workforce allocation.

The importance of simulation in this process is emphasised by the multi-step nature of the planning and execution phases. Since the planning phase is restricted to partial demand knowledge and labor availability (forecasts), an open loop system could potentially result in sub-optimal operations. Furthermore, the optimization constraints (defined later) are designed to handle worst-case scenarios in terms of chute blockage and parcel rejection, and may result in overly conservative plans. In order to avoid both these problems, we use the simulator in the planning phase to roll out multiple demand scenarios and to adjust the constraint thresholds for the optimization problem. The next section covers this aspect in detail.

3 METHODOLOGY

We rely on optimization and its improvements via feedback from a simulation environment (called the digital twin) to solve the optimal sorting problem as shown in Figure 2. We first present and implement the optimization formulation OPTSORT (a mixed-integer linear program) for the two stages described in Section 2. Following this we validate our results using digital twin (described later in this section). We then investigate how some of the constraints in this formulation can be gradually relaxed with the goal of improving the throughput without creating any chute blockages and violation of other physical constraints.

3.1 Optimization Algorithm OPTSORT

3.1.1 Planning

The objective of this stage is to produce a destination-chute matching based on the projected load so that a maximum of such parcels are processed. Let the set $\mathscr{C} \triangleq \{C_1, \ldots, C_k\}$ denote the k chutes available in the sorting center and $\mathscr{D} \triangleq \{D_1, \ldots, D_n\}$ denote the n destinations to which the parcels in the projected load

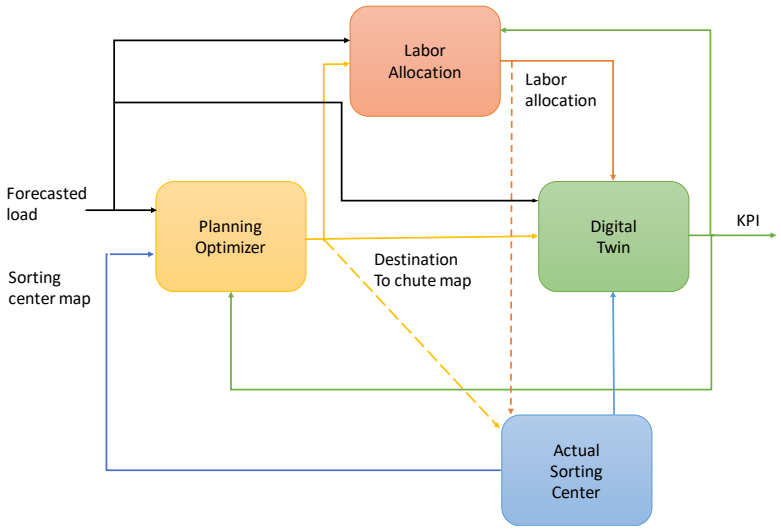

Figure 2: An overview of our methodology. Dashed lines indicate information flow.

will be sent. Correspondingly, let B_i denote the number of parcels in the projected load that will be sent to destination D_i. We will denote the total shift length by T and the time taken to process one parcel at a chute C_j by t_j. The quantity t_j depends upon a variety of factors such as type of chute (Direct/Spiral); number of human resources manning the chute; and efficiency of those resources. We will present a separate algorithm of human resource allocation later in the paper. We need to define the following variables:

$$X_{ij} \in \{0,1\} = \begin{cases} 1 \text{ if destination } D_i \text{ is allocated to chute } C_j, \\ 0, \text{ otherwise,} \end{cases}$$

$$Y_{ij} \in \mathbb{Z}_{\geq 0} = \text{number of parcels of destination } D_i \text{ allocated to chute } C_j,$$

Without specific limits, a many-to-many matching between destinations and chutes will be created. To avoid the situation where a destination is distributed to *too* many chutes and vice versa, we impose certain limits: M_i (respectively, N_j) will denote the maximum number of chutes (resp., destinations) the destination D_i (resp., chute C_j) can be matched to, for $i \in \{1,\ldots,n\}$ (resp., for $j \in \{1,\ldots,k\}$); Note that M_i and N_j can be adjusted based on designer preferences. The following MILP produces the destination-chute matching with the goal of maximizing the number of processed parcels from projected load:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{k} X_{ij}$$

$$\text{subject to } 1 \leq \sum_{j=1}^{k} X_{ij} \leq M_i \text{ for all } i \tag{1}$$

$$\sum_{i=1}^{n} X_{ij} \leq N_j \quad \text{for all } j \tag{2}$$

$$\sum_{j=1}^{k} Y_{ij} \leq B_j \quad \text{for all } i \tag{3}$$

$$\sum_{i=1}^{n} Y_{ij} \leq \left\lfloor \frac{T}{t_j} \right\rfloor \quad \text{for all } j \tag{4}$$

$$X_{ij} \leq Y_{ij} \leq \mathscr{M} X_{ij} \text{ for all } i, j, \tag{5}$$

where $i \in \{1,\ldots,n\}$, $j \in \{1,\ldots,k\}$, and $\mathscr{M}$ is a sufficiently large positive integer. Constraint (1) ensures that every destination is matched to at least one chute; and that destination D_i is not matched to more than M_i chutes (for every $i \in \{1,\ldots,n\}$). A similar bound with respect to destinations per chute is imposed in (2). Constraint (3) ensures that no more than B_i parcels can be processed for destination D_i while (4) guarantees that the number of parcels allocated to chute C_j do not exceed the maximum number that can be processed at C_j. The final constraint (5) ensures that if destination D_i is matched to chute C_j, then at least one parcel intended for destination D_i is processed at C_j. This MILP formulation is suitable for situations where we have only Spiral chutes with no layout restrictions. Furthermore, we have two special cases:

- *Direct chutes:* We follow a simple rule for Direct chutes: allocate the destinations with highest projected loads to Direct chutes. The remaining chutes and destinations can be matched according to the above formulation.
- *Restricted Layouts:* These situations involve additional restrictions of destination-chute matching wherein a destination can only be matched to a (strict) subset of $\mathscr{C}$. Such a context arises for e.g.; when the departure bays for the trucks departing to destinations are categorized by directions or zones and we are trying to minimize the distance travelled by the roller-cages from chutes to departing trucks. To impose this restriction, we introduce a new parameter $A_{ij} \in \{0,1\}$ such that $A_{ij} = 1$ if and only if, destination D_i *can be* allocated to chute C_j and add an additional constraint $X_{ij} \leq A_{ij}$ for all $i \in \{1,\ldots,n\}$ and $j \in \{1,\ldots,k\}$ in the formulation.

3.1.2 Execution

After the shift planning is completed with the creation of a destination-chute matching; real-time processing of parcels start. The parcels are processed in waves (mini-batches) which typically have to be processed in about an hour. The inputs to OPTSORT in this phase are: (i) parcel-destination data for the wave (i.e., which parcel goes to which destination); (ii) list of time-stamps at which the parcels enter the main conveyor; and (iii) destination-chute matching from the planning phase. We now propose the MILP for creating the parcel-to-chute allocation based on these inputs: Let the set $\mathscr{P} = \{P_1,\ldots,P_N\}$ denote the N parcels arriving as input to the current wave. Let the intended destination for parcel P_m be D_{i_m} with $m \in \{1,\ldots,N\}$ and $i_m \in \{1,\ldots,n\}$. Using the destination-chute matching from the planning phase, we then create parameters $Q_{mj} \in \{0,1\}$ with $m \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,k\}$ where $Q_{mj} = 1$ if and only if $X_{i_m j} = 1$; i.e.; the intended destination for P_m is matched to chute C_j and hence, parcel P_m *can be* allocated to chute C_j (the exact chute allocation is provided by OPTSORT).

Assumption 1 We make the following assumptions about the operation of the system: (i) the conveyor moves at constant speed; i.e., the time taken by any parcel to reach the opening of a particular chute from the OCR reader is fixed and known; (ii) the parcels in a wave arrive sequentially in increasing order of their ids; and (iii) for each destination matched to a chute; at least one roller-cage is open all the time which is immediately replaced after it is filled.

Let τ_m for $m \in \{1,\ldots,N\}$ be the time-stamp at which parcel P_m enters the sorting system; i.e., it crosses the OCR reader. Also, let $\bar{\tau}_j$ with $j \in \{1,\ldots,k\}$ denotes the time taken for a (any) parcel to reach the opening of chute C_j from the OCR reader. Correspondingly, we can define $\tau_{mj} = \tau_m + \bar{\tau}_j$ as the time-stamp at which parcel P_m is present at the opening of chute C_j. Assumption 1 guarantees that $\tau_{m_1} \leq \tau_{m_2}$ and consequently, $\tau_{m_1 j} \leq \tau_{m_2 j}$ whenever $m_1 < m_2$ for all $m_1, m_2 \in \{1,\ldots,N\}$ and for all $j \in \{1,\ldots,k\}$. Let T_w denote the total time within which a wave needs to be processed. For a chute C_j, we define t_j as the time taken to process one parcel, L_j as the bound on maximum number of parcels that can be processed within one wave (if such a bound exists based on labor laws etc.), and $\mathscr{C}_j$ as the carrying capacity of chute C_j. Essentially, $\mathscr{C}_j$ denotes the total number of parcels that can stay inside chute C_j without blocking it.

Note that the quantity t_j depends upon the number of human resources allocated to a chute and their efficiency. We present an algorithm for human resource allocation later in this section. Armed with these constants, we now define the binary variable $\bar{P}_{mj}$ such that $\bar{P}_{mj} = 1$ if and only if parcel P_m is allocated

to chute C_j for $m \in \{1,\ldots,N\}$ and $j \in \{1,\ldots,k\}$. A parcel is considered *processed* if it is allocated to a chute. The objective of OPTSORT is to maximize the total number of parcels processed; i.e., to

$$\text{maximize} \sum_{m=1}^{N} \sum_{j=1}^{k} \bar{P}_{mj}. \tag{6}$$

The following constraint ensures that parcel P_m *is* allocated to chute C_j only if it *can be* allocated to C_j:

$$\bar{P}_{mj} \leq Q_{mj} \quad \forall m \in \{1,\ldots,N\} \text{ and } \forall j \in \{1,\ldots,k\}. \tag{7}$$

Next we have the bound on total number of parcels that can be allocated to chute C_j within a single wave:

$$\sum_{m=1}^{N} \bar{P}_{mj} \leq \min\left\{ \left\lfloor \frac{T_w}{t_j} \right\rfloor, L_j \right\} \quad \forall j \in \{1,\ldots,k\} \tag{8}$$

Following this, we have the constraint which ensures that one parcel is allocated to at most one chute:

$$\sum_{j=1}^{k} \bar{P}_{mj} \leq 1 \quad \forall k \in \{1,\ldots,N\} \tag{9}$$

Finally, we formulate a constraint to ensure that the chutes are never blocked. This is done by guaranteeing that a situation where a spate of parcels enter a chute before the earlier ones are processed *never* happens:

$$\sum_{m:\ \tau_{mj} \in [r, r+\mathscr{C}_j t_j]} \bar{P}_{mj} \leq \bar{\mathscr{C}}_j \quad \forall r \in \{1,\ldots,T-\mathscr{C}_j t_j\};\ \forall j \in \{1,\ldots,k\}. \tag{10}$$

The initial value of $\bar{\mathscr{C}}_j$ is set equal to $\mathscr{C}_j$ initially, which is refined based on feedback from the digital twin. This optimization formulation allocates parcels to specific chutes. In situations where we may have extra human resources available or we want to allocate more (or less) than one person per chute depending upon actual load; a resource allocation algorithm needs to be implemented which is described next.

3.2 Allocation of Human Employees to Chutes

The problem of assigning humans to chutes is equivalent to mapping n almost identical objects (e.g., resources with figures of merit drawn from a normal distribution with a small covariance) to p identical bins (e.g., tasks). We note one particular challenge in the context of sorting centers, namely that some chutes (e.g., those that receive heavy or bulky packages) may require at least two humans for operation.

Let $z_{ij} \geq 0$ denote the penalty associated with C_i when it is assigned $j-1$ workers which can be interpreted as a function of the parcels that will be *unprocessed* if $j-1$ workers are assigned to C_i.

Remark 1 It is possible to use more complex functions of the number of unprocessed parcels; for instance, the square of the unprocessed fraction (taken with respect to the capacity) can be used to relax the penalty if the number of unprocessed parcels is relatively small. We assume the following for all i, j:

$$\begin{aligned} \text{Non-increasing penalty:} \quad & z_{ij} \geq z_{i,j+1} \\ \text{Non-increasing marginal penalty:} \quad & z_{ij} - z_{i,j+1} \geq z_{i,j+1} - z_{i,j+2} \text{ except if } z_{ij} = z_{i,j+1} > 0 \end{aligned}$$

Remark 2 The second condition helps satisfy a necessary condition for local optimality, namely that neighboring allocations yield a higher penalty than the one obtained from the algorithm. We have that $z_{ij} = z_{ij+1} > 0$ when the addition of an exactly one extra individual to the i^{th} chute does not help reduce the penalty. This can happen for chutes wherein each package needs at least two handlers. Finally, we define the matrix of penalties and and its columns as

$$Z = [\mathbf{z}_1, \ldots, \mathbf{z}_{n+5}]^\top = \{z_{ij}\}, \quad Z \in \mathscr{R}^{p \times n+5} \tag{11}$$

The reason for adding four extra columns is purely to support the notation employed in our algorithm (see Algorithm 1). As such, all entries of those columns can be trivially set to 0. The assignment problem can be cast formally as follows.

Problem 1 Suppose we are given a matrix $Z \in \mathbb{R}^{p \times n+1}$, where $z_{ij} \geq z_{i,j+1} \geq 0$ for all i, j. We wish to determine an assignment $(i, \sigma(i))$ to solve the following problem:

$$\min_{\sigma} \sum_{i=1}^{p} z_{i\sigma(i)} \quad \text{s.t.} \quad \sum_{i=1}^{n} \sigma(i) = p \tag{12}$$

Algorithm 1 Assignment of workers to chutes

Initialize: $\sigma(i) = 0 \; \forall i$; $k_{\text{last}} = \emptyset$ (previous assignment)
Initialize: $C_0 = \mathbf{z}_1$ (current assignment); $C_1 = \mathbf{z}_2$ (assign one individual to a chute); $C_2 = \mathbf{z}_3$
Initialize $a = \sum_i \sigma(i)$
Define $P_2 = \{k \in [1, p] \mid C_0(k) = C_1(k) > 0\}$; !chutes with packages needing two handlers
while $a < p$ AND $\max_k C_0(k) > 0$ **do**
  $k = \arg\max_k (C_0(k) - C_1(k))$
  $\delta_1 = C_0(k) - C_1(k)$!most profitable reduction of penalty
  ! Check if two people should be assigned to a single chute in P_2
  if $a \in (1, n-2]$ AND $P_2 \neq \emptyset$ **then**
    $\delta_0 = C_0(k_{\text{last}}) - C_1(k_{\text{last}})$! last profitable allocation
    $\delta_c = \max_{j \in P_2} (C_0(j) - C_2(j))$! reduction in penalty from allocating to a P_2 chute
    if $\delta_c > \delta_0 + \delta_1$ **then**
      $k = \text{argmax}(\delta_c)$ $\sigma(k) \leftarrow \sigma(k) + 2$, $a \leftarrow a + 2$
      $C_0(k) = C_2(k)$; $C_1(k) = \mathbf{z}_{\sigma(k)+3}(k)$; $C_2(k) = \mathbf{z}_{\sigma(k)+4}(k)$
    else
      $\sigma(k) \leftarrow \sigma(k) + 1$, $a \leftarrow a + 1$
      $C_1(k) = C_2(k)$; $C_2(k) = C_3(k)$; $C_3(k) = \mathbf{z}_{\sigma(k)+3}(k)$
    end if
  end if
  $k_{\text{last}} = k$
end while
Output: $\sigma(1 : p)$

The assignment algorithm is listed in Algorithm 1. It assigns individuals sequentially, aiming to achieve the largest possible reduction in penalties at each step. We now describe the simulation environment, an *enterprise digital twin* of a sorting center terminal which will act as both a medium of validation and provide feedback to OPTSORT for improvement.

3.3 Digital Twin

We adopt a pragmatic modelling & simulation methodology for evaluating optimization outcomes using a close-to-real environment. Principally, we use a purposive hi-fidelity digital twin (DT) of a sorting terminal that captures all relevant entities, behavioural aspects, inherent uncertainties and pragmatic considerations. A schematic representation of a typical sorting terminal digital twin is shown using a class diagram in Figure 3. It has three parts: a cyber-physical system, a *planner* (in this case it is the optimizer) and a set of *human resources*. From a structural perspective, the conveyor belt has multiple *slots* (to hold and control

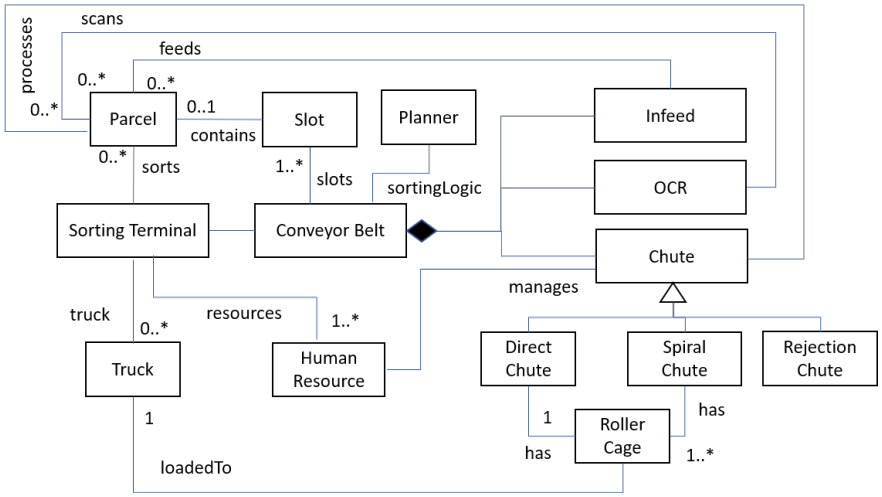

Figure 3: A schematic representation of sorting digital twin.

parcel movements) and a set of sub-systems: *infeeds*, *OCRs*, and *chutes* (Direct/Spiral/Rejection). Once the parcels are processed at Direct or Spiral chutes; they are put into roller cages which are finally loaded onto different *trucks* for final dispatch. A three-step process to construct a faithful DT is described below:

Construction & contextualization: We constructed a configurable sorting terminal digital twin by adopting a bottom-up actor-based (Agha, Mason, Smith, and Talcott 1997) modelling paradigm and an actor-based specification language - ESL (Clark, Kulkarni, Barat, and Barn 2017). Behaviours and interaction patterns of the entities shown in Figure 3 are captured as *actors*. For example, all parcels, conveyor belts, slots, chutes, and resources are realised as actors. Entity behaviours, such as movement of conveyor-belts and slots, tilting of slots at specific chute, and resource emptying parcels from Spiral chute to roller cages, are realized as actor behaviours. All practical considerations and inherent uncertainties of the entities, such as varying efficiency of resources and arrival time of parcels, are encoded as probabilistic actor behaviours.

Appropriate configuration parameters are introduced to represent a specific sorting terminal of a logistic company that includes specific number of conveyor belts, slots for each conveyor belt, number and position of in-feed, OCRs, and chutes, chute sizes, and resource efficiencies.

Validation: We use operational validity technique as suggested by (Sargent 2010) to establish the faithfulness of a contextualized digital twin. Essentially, parcels' arrival details and specific sorting logic of a set of historical days are fed into contextualized digital twin and their simulation results are compared with real outcome to establish faithfulness.

What-if simulation: A validated digital twin is used as an environment to evaluate the efficacy of optimization under different practical considerations. In this step, a sequence of parcels and an optimized sorting logic are fed into the digital twin and specific KPIs (i.e., key performance indicators), such as throughput, chute blockage, number of rejections, etc. are observed through simulation as shown in Figure 3. Simulation of the digital twin involves executing the probabilistic micro-behaviour of all entities and observing the emergent macro-behaviour to understand the efficacy of a sorting logic under different parcel loads. A sorting logic can be evaluated for different parcel loads by feeding different streams of parcels to the digital twin. Similarly, different sorting laws for the same parcel load and a specific sorting logic with different parcel loads under various practical considerations (e.g., limited personnel/hour, workers with less productivity, mechanical failures) can be tested using our method shown in Figure 2.

4 RESULTS AND DISCUSSION

In this section, we present the results of our experiments. The key ideas of our experiments are:

- Implement and validate the results produced by OPTSORT;
- If the chute blockage constraint (10) turns out to be conservative, can we gradually relax it to improve throughput and KPIs (defined later) without adding chute blockages?; and
- Study the robustness of OPTSORT to variations in human efficiency.

We classify our experiments into three groups: i) no restrictions on destination-chute matching (*unrestricted*); ii) restrictions on which destination can be mapped to which chutes (restricted layouts); and iii) Direct chutes combined with restricted layout situations. We assume the following parameters for experiments: $n = 300$, $k = 30$ with $M_i = 5$, and $N_j = 15$ for every $i \in \{1,\ldots,n\}$ and $j \in \{1,\ldots,k\}$. Every wave is to be processed in 3000 seconds. For the planning stage, we assume a load that is distributed randomly for the 300 destinations to be processed within $30,000$ seconds (10 waves). Moreover, for the constrained layout scenario we assume the following matching restrictions: $D_1 - D_{60} \rightarrow C_1 - C_6$; $D_{61} - D_{120} \rightarrow C_7 - C_{12}$; $D_{121} - D_{180} \rightarrow C_{13} - C_{18}$; $D_{181} - D_{240} \rightarrow C_{19} - C_{24}$; and $D_{241} - D_{300} \rightarrow C_{25} - C_{30}$. Furthermore, we assume that there is one person manning one chute who is able to process 120 parcels/hour; i.e., time taken to process a parcel at a chute is equal to $t_j = 30$ seconds for every $j \in \{1,\ldots,k\}$. Thus, no more than $1,000$ parcels can be processed at a chute in $30,000$ seconds, and hence, the maximum possible load for the operation for $k = 30$ chutes turns out to be $30,000$. We assume a projected load of $29,335$ parcels (close to peak capacity). The planning algorithm is able to process all parcels in the case of unrestricted layouts; and $29,148$ parcels in the case of restricted layouts.

To compare the performance of our algorithm, we use a greedy heuristic (GREEDY) as a baseline. This heuristic pushes a parcel P_m into the first free chute C_j such that $Q_{mj} = 1$; i.e., a parcel is pushed into the first free chute if it can be allocated to that chute. We compare the performance of the two algorithms based on the following key performance indicators: (i) *Number of re-circulations Rc* (Number of parcels circulated more than once before entering a chute/rejection chute), (ii) *Number of rejections Rj* (total number of parcels entering the rejection chute in a wave), and (iii) *Average Sorting time* S_t (time elapsed between the point at which a parcel crosses the OCR reader (enters the system) and the point when it enters a chute/rejection chute, averaged over all parcels in a wave).

We ran several experiments with varying load capacities and profiles for waves e.g., loads with *benign* capacity ($\approx 50\%$ capacity); waves with load-profiles (i.e., parcel-destination matching) similar to that of the forecast, and random load-profiles. In the case of waves with benign load capacity, both GREEDY and OPTSORT performed equally well with zero rejections and re-circulations (with slight variations in average sorting times). On the other hand, once can observe a marked difference in KPIs once the load capacity increases. To demonstrate this, we present the results of one set of such experiments here. Two waves were sent in comprising of $N = 2,523$ parcels ($\approx 85\%$ of max. capacity) each, with random τ_m for $m \in \{1,\ldots,2523\}$. Figure 4 shows the KPIs for GREEDY with no layout restrictions. Other figures are not presented due to space constraints.

The KPIs for GREEDY with no layout restrictions are $Rc = Rj = 162$ and $S_t = 0.6$ min; while the same for OPTSORT with $\bar{\mathscr{C}}_j = 50$, they are $Rc = 0$, $Rj = 190$, and $S_t = 0.499$ min. Thus, OPTSORT results in no re-circulation with a much lower average sorting time. However, the number of rejections for OPTSORT is higher than GREEDY. This is due to the fact that $\bar{\mathscr{C}}_j = 50$ turns out to be too conservative. Increasing $\bar{\mathscr{C}}_j$ to 55, removes the rejections while maintaining zero re-circulations and smaller average sorting time. The KPIs for unrestricted and restricted layout scenarios are captured in Table 1. The results for the Direct-chute+restricted layout scenario are almost similar and are omitted due to space constraints.

Experiments with varying human efficiency/productivity (randomly between 0.8 and 1.2) were also simulated using the digital twin. Variations in human efficiency lead to variations in processing time per parcel and consequently, the overall throughput of the system may change. Our objective was to study the robustness of OPTSORT against such variations and its iterative improvement via feedback from digital twin. For instance, for the unrestricted layout scenario, with $\bar{\mathscr{C}}_j = 60$, no parcels were rejected or re-circulated with varying human efficiency (as opposed to $\bar{\mathscr{C}}_j = 55$ with nominal human efficiencies). However, there is one caveat to this study. OPTSORT assumes that the efficiency of person manning a chute is known *a*

KPI's	Values
Total number of parcel arrived from Infeed onto Belt	5046
Total number of parcel sorted	5046
Number of parcel with circulation (> 1)	162
Total number of circulation	374.5057471264359
Average number of re-circulation	2.311763871150839
Number of parcel left in the terminal	0
Average sorting time (in minutes)	0.605630118617529
Minimum sorting time (in minutes)	0.0823754789272031
Maximum sorting time (in minutes)	6.473180076628353

Figure 4: KPIs for GREEDY in unrestricted layout, captured in simulator interface.

Table 1: Key Performance Indicators for various test situations.

Algorithm+Situation	Rc	Rj	$S_t(min)$
Unrestricted GREEDY	162	162	0.606
Unrestricted OPTSORT ($\bar{\mathscr{C}}_j = \mathscr{C}_j = 50$)	**0**	190	0.499
Unrestricted OPTSORT ($\bar{\mathscr{C}}_j = 55$)	**0**	**0**	**0.495**
Restricted GREEDY	55	55	0.557
Restricted OPTSORT ($\bar{\mathscr{C}}_j = \mathscr{C}_j = 50$)	**0**	341	0.507
Restricted OPTSORT ($\bar{\mathscr{C}}_j = 65$)	**0**	**0**	**0.491**

priori; i.e., it can be different from nominal (100%); but it needs to be known. Such an assumption may not be feasible in a real-world scenario; and work is in progress to reconcile this difference.

5 CONCLUSIONS

We studied how a combination of optimization and high-fidelity simulatable digital twin can be used to improve the operations of a sorting center terminal. We developed a two-stage optimization procedure OPTSORT intended to maximize throughput of a sorting terminal, validated and refined using feedback from a simulatable digital twin of the sorting terminal. We showed that the digital twin is a critical component, helping to balance performance with robustness. Future work in this regard is threefold: first, to model more real-world constraints and scenarios such as uncertain human efficiency and time delays in processing; second, use efficient heuristics instead of MILP for faster processing; and finally, integrate various logistics operations such as vehicle routing with sorting logic to solve large-scale enterprise problems.

REFERENCES

Agha, G., I. Mason, S. Smith, and C. Talcott. 1997. "A Foundation for Actor Computation". *Journal of Functional Programming* 7(1):1–72.

Ausubel, L. 2000. "A Generalized Vickrey Auction". In *Econometric Society World Congress 2000 Contributed Papers*. Econometric Society. paper number 1257.

Boysen, N., D. Briskorn, S. Fedtke, and M. Schmickerath. 2019. "Automated Sortation Conveyors: A Survey from an Operational Research Perspective". *European Journal of Operational Research* 276(3):796–815.

Clark, T., V. Kulkarni, S. Barat, and B. Barn. 2017. "ESL: An Actor-based Platform for Developing Emergent Behaviour Organisation Simulations". In *Practical Applications of Agents and Multi-Agent Systems*, 311–315. Springer.

Coffman Jr., E. G., E. N. Gilbert, A. G. Greenberg, F. T. Leighton, P. Robert, and A. L. Stolyar. 1995. "Queues Served by a Rotating Ring". *Communications in Statistics. Stochastic Models* 11(3):371–394.

Fedtke, S., and N. Boysen. 2017. "Layout Planning of Sortation Conveyors in Parcel Distribution Centers". *Transportation Science* 51(1):3–18.

Jarrah, A. I., X. Qi, and J. F. Bard. 2016. "The Destination-loader-door Assignment Problem for Automated Package Sorting Centers". *Transportation Science* 50(4):1314–1336.

Khir, R., A. Erera, and A. Toriello. 2021. "Two Stage Sort Planning for Express Parcel Delivery". *Insitute of Industrial and Systems Engineering Transactions*.

Kim, J.-B., H.-B. Choi, G.-Y. Hwang, K. Kim, Y.-G. Hong, and Y.-H. Han. 2020. "Sortation Control Using Multi-Agent Deep Reinforcement Learning in N-Grid Sortation System". *Sensors* 20(12):3401.

Li, X., Y.-t. Hong, H. Zheng, and X.-j. Chen. 2009. "Flexsim System Simulation Software in the Distribution Center Sorting System Design [J]". *Logistics engineering and management* 1:37–39.

Novoa, L. J., A. I. Jarrah, and D. P. Morton. 2018. "Flow Balancing with Uncertain Demand for Automated Package Sorting Centers". *Transportation Science* 52(1):210–227.

Research and Markets 2020, December. "Global Parcel Delivery Market Insight Report 2020". Technical report.

Sargent, R. G. 2010. "Verification and validation of simulation models". In *Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, 166–183. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sastry, S., and M. Bodson. 2011. *Adaptive control: stability, convergence and robustness*. Courier Corporation.

Shahmardan, A., and M. S. Sajadieh. 2020. "Truck Scheduling in a Multi-door Cross-docking Center with Partial Unloading – Reinforcement Learning-based Simulated Annealing approaches". *Computers & Industrial Engineering* 139:106–134.

Vickrey, W. 1961. "Counterspeculations, Auctions and Competitive Sealed Tenders". *The Journal of Finance* 16(1):8–37.

Werners, B., and T. Wülfing. 2010. "Robust Optimization of Internal Transports at a Parcel Sorting Center operated by Deutsche Post World Net". *European Journal of Operational Research* 201(2):419–426.

Zhang, F., and C. Tian. 2017. "Study on Modeling and Simulation of Logistics Sorting System Based on Flexsim". In *2017 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, 504–506. Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

SUPRATIM GHOSH is a scientist with TCS Research. He earned his Ph.D and M. S. in Electrical Engineering, and M. A. in Mathematics from the Pennsylvania State University. His research interests include applications of control and optimization techniques to supply chain and robotics. His email address is supratim.ghosh2@tcs.com.

ARITRA PAL is a researcher with TCS Research. He earned his M. Tech in Mathematics and Computing from Indian Institute of Technology, Patna. His research interests include probability, statistics and optimization. His email address is p.aritra@tcs.com.

PRASHANT KUMAR is a Researcher at TCS Research. He holds a B.Tech in Information Technology from the BIT Sindri. His research interests are in the area of actor based simulations. His email address is kumar.prashant10@tcs.com.

ANKUSH OJHA is a researcher with TCS Research. He earned his M. Tech degree in Industrial and Management Engineering from Indian Institute of Technology, Kanpur. His research interests include application of reinforcement learning and optimization to problems in supply chain and transportation. His email address is ojha.ankush@tcs.com.

ADITYA A. PARANJAPE is currently a Lecturer in the Department of Aeronautics at Imperial College London, and on sabbatical from TCS Research where he holds the position of Scientist in Software Systems and Services. He earned his Ph.D. in Aerospace Engineering from the University of Illinois at Urbana-Champaign in 2011. His research interests are centered around control theory, multi-agent systems, and reinforcement learning. His email address is aditya.paranjape@tcs.com.

SOUVIK BARAT is a Principal Scientist at Tata Consultancy Services Research. He earned his Ph.D. in Computer Science from Middlesex University. His research interests include simulation, agent and actor technology, enterprise modeling and digital twin. His email address is souvik.barat@tcs.com..

Harshad Khadilkar is a senior scientist with TCS Research. He holds a PhD in Aeronautics and Astronautics from the Massachusetts Institute of Technology. His research interests are in the area of optimal decision making for industrial operations. His email ID is harshad.khadilkar@tcs.com.