Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

A SIMULATION-AIDED DEEP REINFORCEMENT LEARNING APPROACH FOR OPTIMIZATION OF AUTOMATED SORTING CENTER PROCESSES

Deepak Mohapatra
Aritra Pal
Ankush Ojha
Supratim Ghosh
Marichi Agarwal
Chayan Sarkar

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

ABSTRACT

Operations in a parcel sorting center (SC) are multi-fold which lead to multiple NP-hard optimization problems, namely, parcel-chute assignment, online bin-packing, scheduling, and routing. The advent of multi-agent robotics has accelerated the process of automation in sorting centers which has led to the requirement of sophisticated algorithms to optimize these operations within an SC. To this end, we propose $RL-SORT$: a simulation-aided deep reinforcement learning based algorithm which *jointly* optimizes the parcel-chute assignment and online roller-cage (RC) packing problems. Through experimentation on our simulation framework, we show that $RL-SORT$ not only outperforms baselines, but also has a low computational burden. Further, it is able to significantly reduce the number of RCs used, thereby, reducing the transportation costs.

1 INTRODUCTION

The advent of the pandemic has significantly accelerated trends in the parcel delivery industry and it is expected to grow at a rate of at least 10% year-on-year for the next few years (reaching a volume of \$675bn by 2027 from its current levels of \$376bn) (Motor Intelligence 2022). Sorting of parcels is a key cog in the supply chain that the parcels have to go through to reach end-users. Such a tremendous growth and increased adoption of automation have started posing significant challenges to the existing hardware and algorithms used for sorting operations (Harrison 2019; Benzi et al. 2019). The primary operations in a sorting center (SC) are: scheduling of inbound parcels, segregation of parcels to destinations by means of chutes, packing of parcels in roller-cages (RCs) in an online fashion, loading of RCs onto trucks, and outbound routing/scheduling of trucks for transportation. Each of these processes yields an NP-hard/NP-complete optimization problem (e.g., scheduling, assignment, knapsack, routing), finding optimal solutions to which are generally very difficult.

Optimization of SC processes have attracted significant attention of the research community in recent years. However, most of the existing research either focuses on layout optimization (Werners and Wülfing 2010) or isolated solutions to these problems (McWilliams et al. 2005; Jarrah et al. 2016; Khir et al. 2021). Existing techniques include either mixed integer programs (Khir et al. 2021; Ghosh et al. 2021) or meta-heuristics (Balcou and Yadavalli 2017). Independent optimization of these processes often leads to sub-optimal solutions overall. Sophisticated algorithms, under the umbrella of *Intralogistics* are the need of the hour which provide an *end-to-end and joint* optimization of these processes (Harrison 2019). However, exact optimization techniques leave a lot to be desired (computationally slow) when they are applied to joint optimization of large-scale logistics problems.

On the other hand, deep reinforcement learning (RL) based methods have made significant progress in producing nearly optimal results for several combinatorial optimization problems (such as vehicle routing (Gupta et al. 2022), online bin packing (Zhao et al. 2021), and production scheduling (Hubbs et al. 2020)) with significantly reduced computational costs. Deep RL based approaches have also been used in sortation control by routing parcels from 'emitters' to 'receivers' in an SC (Kim et al. 2020). Thus, one naturally wonders if deep RL based algorithms can provide scalable and nearly optimal solutions to joint optimization problems in sorting center logistics. We provide an answer to this question in the affirmative, by formulating the first (to the best of our knowledge) simulation-aided deep RL based algorithm $RL-SORT$ for joint optimization two critical processes in an automated sorting center.

Our algorithm $RL-SORT$ uses a combination of deep reinforcement learning and heuristics to *jointly* optimize the parcel-chute assignment and online 3-D RC packing problem in an automated sorting center. These two processes individually represent two of the most famous combinatorial optimization problems: the assignment/allocation problem and the knapsack problem, both of which are NP-hard. Thus, existing solution techniques only provide approximate solutions to large-scale versions of these problems. However, as one would observe in Section 4, optimizing these processes in silos results in highly sub-optimal results thereby, increasing transportation and other costs. $RL-SORT$ not only resolves this by jointly optimizing the two problems, but it does so at an impressive computational speed thereby, making itself amenable for deployment in a real-world automated SC using multiple robots. Further, since $RL-SORT$ cannot be trained in a real-world SC, we also develop a queue and grid based simulation framework to train the algorithm. To the best of our knowledge, our formulation and approach are among the very first ones to provide AI based solutions for intralogistics problems in an SC. Specifically, the paper's contributions are:

1. Formulation of a *joint* optimization problem related to processes in an SC;
2. Development of deep-RL based $RL-SORT$ and other baselines for solution to the joint problem;
3. Experiments and simulations to showcase efficacy of $RL-SORT$ compared to other baselines.

2 PROBLEM SETUP

The focus of this paper is to describe an algorithm that *jointly* optimizes the processes in a sorting center in a scalable fashion. Figure 1 shows the basic setup of a sorting center under consideration (Ghosh et al. 2021; Fedtke and Boysen 2017). The following stages describe the initial processes of an SC:

- *Entry of parcels:* The parcels received via inbound trucks are first processed through an OCR reader before being placed onto the main oval shaped conveyor. The OCR reader captures the parcel's destination, dimensions, and weight; and redirects this information to the sorting logic algorithm.
- *Assignment:* Once the information about the parcel reaches the sorting logic, the parcel is sent to a *spiral chute* for further processing. Each *chute* is mapped to multiple destinations and contains at least one open roller-cage for each destination.
- *Processing and Packing:* Once a parcel enters a chute, it is processed in a first-in-first-out fashion by either a human being or a robotic arm; i.e., parcels which enter the chute first are processed earlier. This stage involves picking up the parcel and putting it into an RC earmarked for its destination.

Further downstream processes include moving out RCs to trucks as soon as they are filled, outbound scheduling, and routing of trucks which are currently beyond the scope of this paper. The *processing and packing* stage can involve either a human being or a robotic arm for packing of parcels. A human being takes a larger time (≈ 30 sec/parcel) to process each parcel, but is able to achieve high efficiency in packing (due to reshuffling capability of humans).

On the other hand, in an *automated* SC, a robotic arm carries out the processing and packing stage with a much smaller time/parcel (≈ 10 sec/parcel) while maintaining a reasonable packing efficiency. This

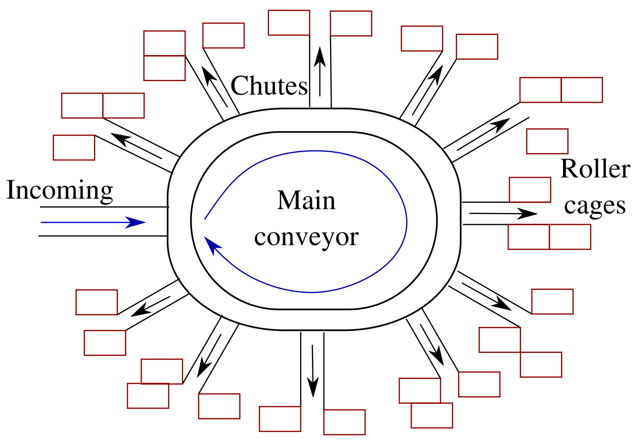

Figure 1: Schematic of a sorting center, showing the main conveyor belt, chutes, and associated (RCs). The overall optimization comprises of two interlinked problems: parcel-chute allocation and online bin-packing.

results in a larger number of parcels processed in a day thereby, increasing the overall productivity of the SC. However, in order to minimize the overall costs (which include transportation), one needs to design scalable algorithms that *jointly* optimize the two processes: *parcel-chute assignment* and *online 3-D bin (RC) packing*. As is well known, both of these problems are variants of well-known NP-hard combinatorial optimization problems: the generalized assignment and knapsack, respectively; thereby, ruling out use of exact optimization approaches for large-scale instances. In this paper, we present a simulation-aided deep reinforcement learning approach to carry out these operations in an approximately optimal fashion.

2.1 Costs and Efficiencies

To make the overall joint optimization problem tractable; we make the following assumptions:

1. The parcels are cuboidal and non-deformable;
2. Parcels arrive in *waves* (with a max processing time $T_{\max}$) and each shift processes multiple waves.
3. Each chute has exactly one RC open per destination, and
4. If no feasible packing location is found for an incoming parcel, one of the open RCs is closed and immediately replaced with an empty one.

Optimization of the sorting process involves acting on the following two levers:

- *Sorting efficiency:* This measures the number of parcels processed and packed in a sorting center:

$$\eta_s = \frac{\text{\# parcels assigned to chutes for a wave}}{\text{\# total number of parcels arriving in a wave}} \times 100(\%).$$

  We assume that if a parcel is not assigned to a chute, it is either re-circulated a fixed number of times or rejected, both of which bring down η_s.
- *Bin Packing efficiency:* This measures the volume utilization of the RCs closed during the process.

$$\eta_b = \text{average}\left(\frac{\text{total volume of boxes in a closed bin}}{\text{Volume of the closed bin}}\right) \times 100(\%).$$

  In other words, η_b measures the average of fill-rates for all the bins that are *closed and replaced* during a wave. We ignore the fill-rates of open RCs since they will be utilized in subsequent waves. Improvement in η_b reduces the number of RCs in use, thereby, reducing the number of trucks, which leads to decrease in transportation costs.

2.2 Problem Statement

The overall objective in this paper is to simultaneously maximize the overall throughput/efficiency of the sorting center and minimize the overall transportation costs. Mathematically, this leads to maximizing $w_1\eta_s + w_2\eta_b$ respecting the constraints of bin packing (Ojha et al. 2021) and sorting of parcels (Ghosh et al. 2021). The weights w_1 and w_2 can be tuned to designer's preferences.

3 METHODOLOGY

In this section we describe the simulation and deep reinforcement learning-cum-heuristic frameworks for joint optimization of processes in an automated sorting center.

3.1 Simulation Framework

We develop a queue and grid based simulation environment in Python to model the processes in a sorting center which is used to train, test, and model what-if scenarios for our deep RL-cum-heuristic based algorithms. The overall simulation environment can be broken down into the following stages:

Simulating Chutes: First-in-first-out (FIFO) queues are used to mimic the behavior of parcels entering and leaving chutes. In other words, parcels entering chutes first are processed earlier. It is assumed that each chute (queue) has a fixed length (L), and can hold parcels $P_1,\ldots,P_k$ as long as $\sum_{i=1}^{k}\ell_i \leq L$ where ℓ_i denotes the length of parcel P_i entering the chute. Further, each parcel is associated with a processing time, t_s (30 sec/parcel and 10 sec/parcel for human and robotic packing, respectively) after they exit a chute. To simplify the problem setup, we assume that the time-taken by a parcel to move inside a chute is negligible. If a parcel P_i enters a non-empty chute C_j, it must wait for the parcels ahead in the queue to be processed, before its processing can begin.

Simulating bin-packing: A 2-dimensional cellular array is used to simulate the interior of a roller-cage (RC) or bin. The size of the array is $L_B \times W_B$ where L_B and W_B denote the length and width of the bin under consideration (in cms), respectively. Each cell (i,j) with $i \in \{1,\ldots,L_B\}$ and $j \in \{1,\ldots,W_B\}$ in the array is comprised of a non-negative integer h_{ij} which describes the height of the object occupying the cell This height is updated whenever an object is placed on top of the object already placed in the cell. Also, $h_{ij} \leq H_B$, for every $i \in \{1,\ldots,L_B\}$ and $j \in \{1,\ldots,W_B\}$ where H_B denotes the height of the bin. One can also see that the resolution of box placement in the bin is 1cm. In other words, whenever a new parcel P with dimensions (ℓ,b,h) is supposed to be placed at a location starting at $(\bar{x},\bar{y})$ on the $x-y$ plane, we change the values h_{ij} to $h_{ij}+h$ for $i \in \{\bar{x},\ldots,x+\ell\}$ and for $j \in \{\bar{y},\ldots,\bar{y}+b\}$ subject to constraints that $\bar{x}+\ell \leq L_B$, $\bar{y}+b \leq W_B$, and $h_{ij}+h \leq H_B$ to ensure that the box fits within the bin's dimensions.

Data-generation: We use a cutting plane technique to generate the set of boxes for experiments. In other words, boxes of random integral dimensions are carved out from the interior of a bin. This ensures that the dimensions of all the boxes match up to the dimensions of the bin. Each data-set contains a pre-specified number of boxes (from $1,000$ to $15,000$) depending on use-case.

The environment records the time-stamp of entry τ_i for parcel P_i, its intended destination D_k, and its dimensions $(\ell_i \times b_i \times h_i)$ at the outset. The time to entry each allowable chute C_j (chutes C_j where P_i *can* enter; i.e., the chutes to which D_K is mapped) for parcel P_i denoted by τ_{ij} is computed according to $\tau_{ij} = \tau_i + \bar{\tau}_j$; where $\bar{\tau}_j$ is fixed time to reach C_j's entrance from the OCR reader. The parcel is then assigned a chute using the sorting and bin packing logic, and are sent to the queue of the destined. Once a parcel is processed at the chute, it is placed into the bin intended for their destination by changing the values of the cells of the corresponding bin array.

3.2 *RL-SORT*: Deep RL based technique for Sorting

We begin this section with a brief description of the optimization problems under consideration. Further details can be found in (Ghosh et al. 2021; Ojha et al. 2021).

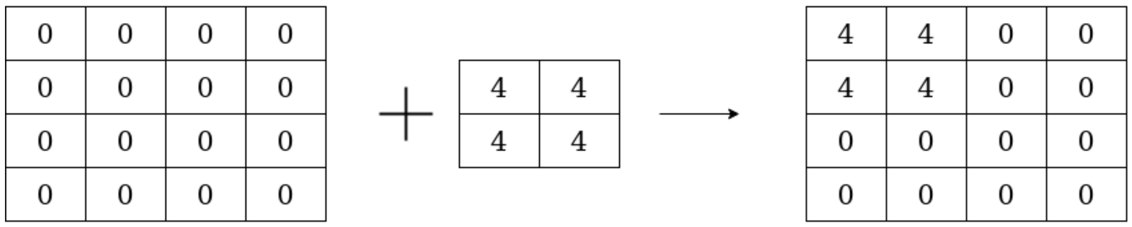

Figure 2: Representation of bin-packing using simulation.

3.2.1 The Sorting Problem:

This problem primarily deals with optimal allocation of parcels to chutes so that the throughput of the system (alternatively, the sorting efficiency η_s) is maximized. This problem comprises of two (stages):

- *Planning*: This stage is carried out to create a sort-plan which consists of a many-to-many mapping between destinations and chutes. This is carried out offline and is based on projected load with the objective to maximize the number of processed parcels in the projected load.
- *Execution*: Once the sort plan is ready and the parcels start arriving, this stage involves assigning chutes to parcels in real-time with the intent to maximize the number of parcels processed.

The approach titled OPTSORT in (Ghosh et al. 2021) used Mixed Integer Linear Programs (MILP) improved iteratively via a digital twin based simulation to solve the sorting problem. However, while the planning stage (carried out offline) can be solved using MILPs, real-time execution of large-scale waves using an exact approach like MILP is infeasible due to large computational times. This is primarily due to the fact that OPTSORT solved the execution problem for the *entire* wave in a single shot thereby maximizing efficiency at the cost of the computational speed. Algorithms to be used for real-time deployment for large-scale sorting need to have a *dynamic* point of view to solve the chute assignment problem for each parcel individually, thereby achieving a trade-off between optimality and computational speed.

3.2.2 The Bin-Packing Problem

This problem deals with placement of 3-dimensional boxes arriving in an online fashion in a bin using a robotic arm so that the bin-fill rate (η_b) is maximized. One may impose constraints on allowable rotations for the robotic arm (depends of degrees of freedom of the arm) and computational time/box (since if the computation time exceeds the processing time/box, the chute will start getting congested thereby blocking the entry of subsequent parcels). Heuristic algorithms such as first-fit, best-fit as well as MILP and RL-based algorithms perform with a reasonable degree of optimality (Ojha et al. 2021; Zhao et al. 2021).

3.2.3 *RL-SORT*: Description

In this section, we will describe $RL-SORT$, a deep reinforcement learning based technique for the *joint* optimization of the sorting assignment and online 3-dimensional bin packing problem in an automated sorting center. The overall problem comprises of three individual stages: (a) Planning; (b) Execution; and (c) 3-D Bin packing. $RL-SORT$ attempts to design a dynamic *execution* protocol that takes a feedback signal from the downstream 3-D bin packing algorithm with the intent to optimize the overall efficiencies. The planning phase which involves creation of sort-plan uses an MILP based technique and is exactly similar to the sort plan creation stage in OPTSORT (Ghosh et al. 2021). On the other hand, we use an online version of the extreme-point based best-fit heuristic (BF) for 3-D bin packing (Crainic et al. 2008; Ojha et al. 2021). This choice is not mandatory, and one can replace BF with any other bin packing heuristic such as first fit (FF), next fit (NF), Jampack (JP), or MPackLite (MPL) (Ojha et al. 2021). Online heuristics have the following aspects: (a) information about the boxes in future is unavailable; except for a few upcoming boxes; and (b) computation time/box should be low enough to avoid chute congestion.

In the case of an automated sorting center, the robotic arm has to pack the box nearest to it; i.e., it works with a look-ahead of one (1). Further, whenever the upcoming box cannot fit into any of the open bins (one bin per destination per chute), BF heuristic mandates that the bin with the highest fill-ratio is closed and is replaced with an empty bin.

The overall dynamic *execution* problem; i.e., allocation of chute to a parcel can be viewed as a Markov Decision Process $(\mathscr{S},\mathscr{A},\mathscr{R},\gamma)$, where S represents the three observed states (described later), $\mathscr{A}$ denotes the set of actions, $\mathscr{R}$ denotes reward and γ denotes the discounting factor.

States: The RL agents gets three observations from the simulation environment described in Section 3.1. Normalized versions of these observations act as input states for the RL agent. For a parcel P_i intended for destination D_k, let the set $\mathscr{C}_i$ contains the chutes to which D_k is mapped in the planning phase. The vector $S(i,j)$ contains the three states for each pair (P_i,C_j) with $C_j \in \mathscr{C}_i$. The first state represents how quickly can a parcel P_i reach a chute $C_j \in \mathscr{C}_i$. Let τ_{ij} denote the time-stamp by which parcel P_i can reach the entrance of chute C_j for a chute $C_j \in \mathscr{C}_i$, and let T denote the entire wave duration. The first state is represented as $T_{ij} = \frac{\tau_{ij}}{T}$.

The second state $CO(j,i)$ represents whether or not the chute C_j in consideration will be able to accommodate parcel P_i. We use parcel and chute lengths to determine this. To this end, we assume that parcels never enter a chute simultaneously, i.e., at no moment, are two parcels placed *side-by-side* inside a chute, and that every chute is wide and high enough for any parcel to enter. In other words, if (ℓ_i,b_i,h_i) and (L_j,B_j,H_j) denote the dimensions of the P_i and C_j, respectively; then $\ell_i \leq L_i$, $b_i \leq B_j$, and $h_i \leq H_j$ for any parcel P_i and chute C_j. In SCs, typically all chutes have equal dimensions and hence the notation j is dropped. The second state $CA(i,j)$ is described by:

$$CA(i,j) = \frac{CO(j,\tau_i)+CON(j,\tau_i)+\ell_i - CL(j,\tau_{ij})}{L},$$

where $CO(j,\tau_i)$ denotes the total length of parcels already inside C_j at time τ_i, $CON(j,\tau_i)$ denotes the total length of parcels that are on the conveyor ahead of P_i and are already assigned to C_j (i.e., they will enter C_j before τ_{ij} or by the time P_i reaches C_j), and $CL(j,\tau_{ij})$ denotes the length of the parcels inside C_j at τ_i that will be processed (and hence leave C_j) by τ_{ij}. The third state acts as a feedback signal from the 3-D bin packing algorithm, (in this case BF). It contains a priority score for the chute $C_j \in \mathscr{C}_i$ which depends on the state of bin (RC) open for the destination D_k. The chute corresponding to the bin with the highest fill-rate is given the highest rank and so on (with the caveat that the bin should be able to accommodate P_i).

Actions: The action space of the RL agent comprises of the following: assign P_i to a chute from $\mathscr{C}_i$. If no such chute is found, then send the parcel to the rejection chute. A parcel is assigned to a chute only if all feasibility conditions (described later) are satisfied and it has the highest Q-value among $\mathscr{C}_i$.

Rewards: The objective of $RL-SORT$ is to maximize both η_s and η_b. Thus, there are no obvious step rewards for $RL-SORT$. Hence, we provide a discounted version of the episodic terminal reward as a proxy for step reward for each parcel. The total terminal reward R_{ter} for the episode is represented as:

$$R_{ter} = w_1 \cdot \eta_s + w_2 \cdot \eta_b, \tag{1}$$

where η_s and η_b denote the sorting and bin-packing efficiency of the episode under consideration. To give equal preference to both η_s and η_b, we set $w_1 = 0.7$ and $w_2 = 1$ in (1). These values are chosen since average values of η_s and η_b are around 90 and 63 respectively. If P_i denotes the ith arriving parcel (i.e., $P_1,\ldots,P_{i-1}$ have arrived before it) that is assigned a chute $C_j \in \mathscr{C}_i$ and N denotes the total number of parcels for the episode, then $R(i,j) = \gamma^{N-i} * R_{ter}$ with the discounting factor $\gamma = 0.99$.

Neural Network Architecture: A neural network with experience replay is used to train the RL agent and approximate the values $Q(i,j)$ corresponding to each allowable chute-parcel pair (P_i,C_j) with $C_j \in \mathscr{C}_i$. The neural network has a fully connected architecture with 1 input layer: 3 nodes, 1 output layer with one node, and 1 hidden layer with 4 nodes. The input and hidden layers have ReLu activation functions whereas the output layer has a linear activation function. The network is trained using *Tensorflow 2.5.0* in *Python 3.8.3* with SGD optimizer with a learning rate of 0.01, a momentum of 0.9, a batch size of 64 samples and mean squared error (MSE) loss. The replay memory buffer is the collection of 50000 latest samples from which 64 samples are collected randomly for training.

For a parcel P_i intended for destination D_k with allowable chute set $\mathscr{C}_i$, we first evaluate whether it is possible to place P_i in any of the open bins earmarked for D_k at the chutes in $\mathscr{C}_i$. If none of the open bins can accommodate P_i, then the bin with the highest fill-rate is closed and replaced with an empty one. On the other hand, suppose there is at least one bin that into which P_i can be placed. In such a situation, we deliberately mask those bins (and their corresponding chutes) which will not be able to accommodate P_i to increase η_b. Let the set of such chutes by denoted as $\hat{\mathscr{C}_i}$. We then compute the the values $Q(i,j)$ for every $C_j \in \mathscr{C}_i \setminus \hat{\mathscr{C}_i}$. The chute $C_o = \text{argmax}_{C_j \in \mathscr{C}_i \setminus \hat{\mathscr{C}_i}} Q(i,j)$ is chosen greedily (using ε-greedy) for allocation to P_i. At this point, we verify whether the chosen C_j is *feasible* for P_i; i.e., whether C_j can accommodate P_i length-wise and process it before the end of the wave. If any of the conditions are violated, P_i is sent to the rejection chute. The RL training procedure is described formally in Algorithm 1.

Algorithm 1: RL Training

```
1  Initialize the neural network with weights $\phi$, batch size $\beta = 64$, and replay buffer $B$
2  for episode = 1 to TotalNumEpisodes do
3      Randomly choose data instance from training set. Reset environment and get initial states.
4      Initialise parcel index, i = 0 and temporary replay buffer $B_t$.
5      while $i <= NumParcels$ do
6          $i = i + 1$
7          Compute $\mathscr{C}_i$ for parcel $P_i$ intended for $D_k$ with dimensions $(\ell_i, b_i, h_i)$
8          if $P_i$ cannot be packed into any open bins for $D_k$ at $\mathscr{C}_i$ then
9              Close the bin with maximum fill-rate and replace it with an empty bin
10         else
11             Mask chutes $C_m \in \mathscr{C}_i$ whose bins cannot accommodate $P_i$. Let $\hat{\mathscr{C}_i}$ be the set of such $C_m$.
12         Calculate $Q_i(j) = \phi(S(i,j)) \; \forall \; C_j \in \mathscr{C}_i \setminus \hat{\mathscr{C}_i}$.
13         Choose $C_o \in \mathscr{C}_i \setminus \hat{\mathscr{C}_i}$ pair using $\varepsilon$-greedy assignment.
14         if $P_i$ can be accommodated and processed at $C_o$ then
15             Allocate $C_o$ to $P_i$.
16         else
17             $P_i$ is rejected.
18     Calculate $R_{ter}$. Compute $R(i,o)$ for each $(P_i, C_o)$ assignment in the episode.
19     Add $[S(i,o); R(i,o); Q(i,o)]$ for each $(P_i, C_o)$ assignment to $B_t$.
20     Add $B_t$ to $B$. Delete oldest entries in $B$ if size exceeds buffer capacity.
21     Draw $\beta = 64$ samples from $B$ and update $\phi$ by min. MSE loss between $Q(i,o)$ & $R(i,o)$.
```

We also created a variant of $RL-SORT$ known as $RL-ISO$ which solves the sorting assignment problem in silo without feedback from the bin-packing algorithm. The main differences between $RL-SORT$ and $RL-ISO$ are: the state vector $S(i,j)$ contains only two states omitting the bin packing priority score; dimension of the neural network $(2,4,1)$, actions regarding bin packing (lines $11-15$ in Algorithm 1) are not carried out, and R_{ter} and consequently, $R(\cdot,\cdot)$ contains only η_s terms.

4 EXPERIMENTS AND RESULTS

4.1 Training Data and Performance

We train $RL-SORT$ using instances drawn randomly from a sample of 50 data-sets. The bin sizes are fixed at $(80cm \times 45cm \times 45cm)$. Each of these data-sets has 1004 parcels, directed to 10 destinations and to be assigned to 5 chutes. These training data-sets have varying destination densities; i.e., parcels/destinations. Some of them have nearly identical densities for destinations and others are skewed in favor of one or multiple destinations. To balance exploration and exploitation and explore the whole action space, we implement an ε-greedy policy for $RL-SORT$ with ε exponentially decaying from 1 to 0.001 over 2000 episodes. The value of ε dictates the probability of choosing a random action over C_o. Here each episode implies one complete sorting and bin packing operation; i.e., either all the parcels are sorted and packed or $T_{\max}$ units of time have elapsed. Figure 3 captures the training performance $RL-SORT$ and shows the evolution of episodic rewards (MER, η_s, and η_b) and the MSE loss. Here MER is the mean episodic reward which is calculated as the average of total rewards of every processed parcel in an episode. In the first, second, and fourth subfigures of figure 3, the ensemble of green curves represent the reward components ($MER/\eta_s/\eta_b$) for eleven different runs with random initialization seeds (i.e., the environment instances for each run are chosen randomly from the pool). On the other hand ,the blue and red curves represent the mean of all the different seeds and a moving average of the mean with a window of 100 episodes, respectively. One can observe η_s and η_b stabilize around 90% and 63%, respectively. For subsequent testing, the same trained model parameters are used without refinement.

4.2 Test Data

To test the performance of $RL-SORT$ and $RL-ISO$ over baselines, we generate various data-sets with varying load profiles, destination densities and scale:

- *Varying Load Profiles*: The data-sets contain parcels that are either uniformly (relatively identically) or non-uniformly (skewed in favor of a few) distributed among destinations.
- *Destination densities*: The data-sets classified into three types with respect to average number of parcels per destination: (a) 100; (b) 125; and (c) 150.
- *Scale*: Data-sets comprise of various scales starting from around small-scale 1000 parcels/wave to large-scale $15,000$ parcels/wave scenarios for waves of $T_{\max} = 2000$ seconds.

The number of chutes and destinations is chosen so that most of the parcels can be packed. Small scale data-sets may have 10 destinations/5 chutes; large-scale ones can have up to 150 destinations/75 chutes.

4.3 Baselines

To test the performance of *joint optimization* carried out by $RL-SORT$, we develop the following baselines:

- *HEURSORT*: This algorithm takes into account two priority scores: namely sorting priority S_{C_j}, and bin-packing priority B_{C_j} for a chute C_j. The score S_{C_j} is generated taking into account the time-stamp by which P_i at C_j will be processed if P_i was assigned to C_j. Lower time-stamps yield higher S_{C_j}. The score B_{C_j} is generated (only for feasible chutes) using fill-rates of the bin (higher fill-rates will have higher B_{C_j}), only if the bin is able to accommodate P_i. If the bin is not able to do so, an empty bin will be used to generate B_{C_j}. The chute $C_j \in \mathscr{C}_i$ with the highest combined priority $S_{C_j} + B_{C_j}$ is selected for P_i's assignment.
- *JOINTSORT*: This is an MILP-cum-heuristic based algorithm (variation of MPackLite (Ojha et al. 2021)) and works in conjunction with priority score S_{C_j} for integrated sorting and bin-packing. The objective function of MPackLite is changed to reflect the priority score S_{C_j} (rather than first-fit) with the added constraint that bins corresponding to infeasible chutes are not selected.

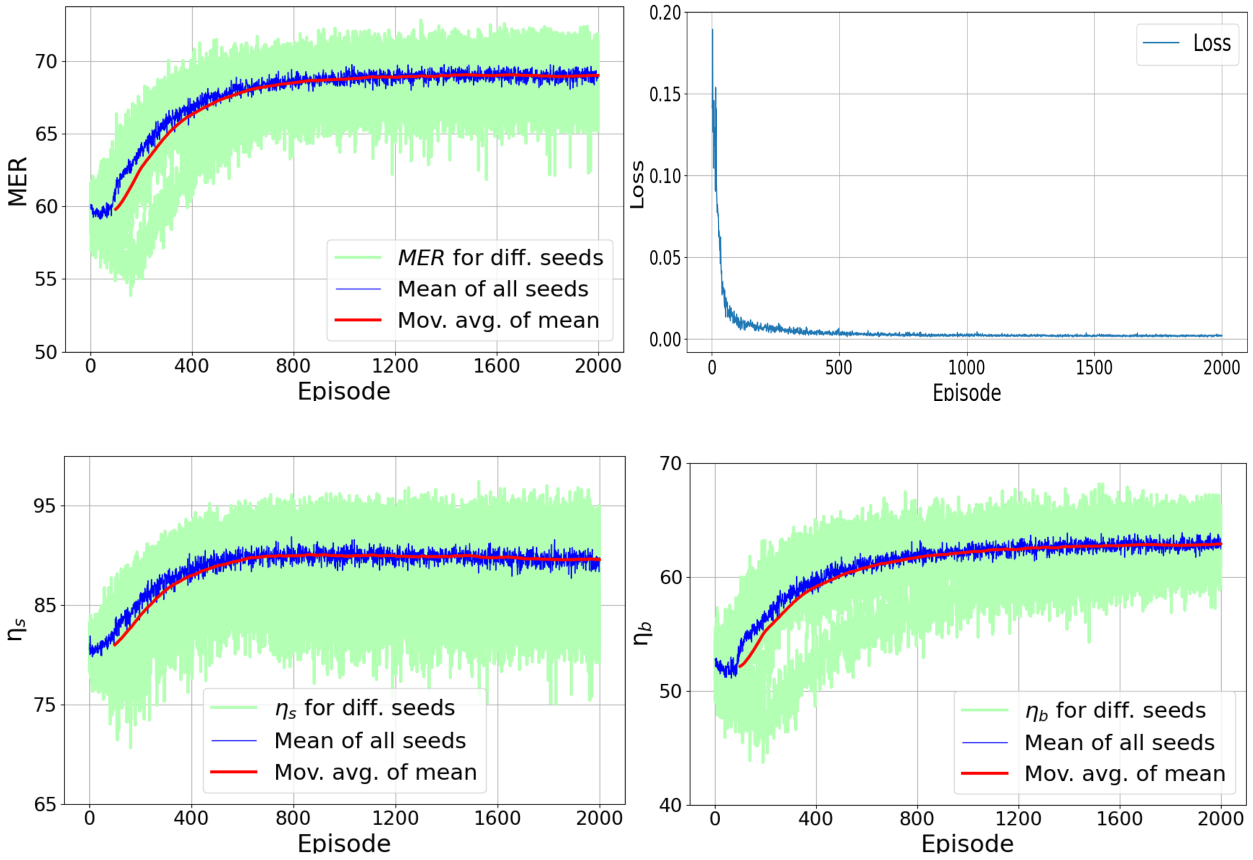

Figure 3: Training performance of $RL-SORT$.

4.4 Joint vs Individual Optimization

Table 1 shows the results of individual vs joint optimization. Individual optimization is carried out using two baselines: (a) $OP-MP$ which first uses OPTSORT for sorting and then MPackLite for bin-packing based on output from OPTSORT; and (b) $RL-ISO+BF$ which uses $RL-ISO$ for sorting followed by best-fit (BF) for packing. Since, sorting happens before bin packing, we follow the same chronology in creating these baselines. We use $RL-SORT$, $JOINTSORT$, and $HEURSORT$ as baselines for joint optimization. Each row represents the mean of 30 different data-sets. The $MILP$ based baselines needed significantly large computation times for larger loads and hence, these results are unavailable. The computation time/box t_c is calculated by dividing the total time required for chute assignment of all the boxes processed in a wave by the total number of boxes processed. One can observe the following:

1. $OP-MP$ and $RL-ISO+BF$ achieve higher η_s than $JOINTSORT$, $RL-SORT$, and $HEURSORT$.
2. A significantly higher bin fill-rate η_b is achieved by joint optimization algorithms $JOINTSORT$, $RL-SORT$, and $HEURSORT$ compared to $OP-MP$ and $RL-ISO+BF$.
3. $RL-SORT$ achieves the highest η_b among all algorithms;
4. RL and heuristic based algorithms maintain a low t_c and therefore, are amenable for deployment in real-world systems for large-scale scenarios in contrast to MILP based algorithms.

The first observation is primarily due to the chronology of process optimization (since sorting optimization is done first, a higher η_s and lower η_b is achieved for individual vs joint baselines). On the other hand, joint optimization remarkably increases the efficiency η_b (relative improvement of up to 60%) sacrificing marginal sorting efficiency in the process (up to 10%). This clearly reduces the number of RCs and hence trucks/transportation costs. Similar trends have been observed using other versions of $RL-SORT$ which use other bin packing algorithms such as first-fit (FF) and $MPackLite$ versus BF (the detailed tables are omitted due to space constraints).

Since RL and heuristic based algorithms solve the chute assignment problem for each parcel individually (dynamic allocation), they achieve a high computational speed (low t_c) in comparison to MILP based algorithm OPTSORT which solves the chute assignment problem for the entire wave simultaneously (this increases t_c, but also achieves higher η_s). Furthermore, η_b maxes out at around 64%. There are two primary reasons for this: first, the bin packing problem under consideration is a completely online one with no information about upcoming boxes and wide variations in box dimensions; and second, robotic packing constraints which rule out reshuffling of boxes. Further details can be found in (Ojha et al. 2021).

Table 1: Performances of baselines on independent and joint optimization.

	$OP-MP$		$JOINTSORT$		$RL-ISO+BF$			$RL-SORT$			$HEURSORT$		
Scale	η_s	η_b	η_s	η_b	η_s	η_b	t_c	η_s	η_b	t_c	η_s	η_b	t_c
1k	**98.95**	37.66	97.74	55.38	97.86	43.96	**0.02**	94	**60**	0.06	92.9	59.8	0.06
2k	**98.85**	36.28	98.42	54.65	97.71	44.25	**0.03**	94.81	**60.68**	0.07	90.73	60.67	0.07
3k	**98.84**	37.15	98.05	55.05	97.46	44.29	**0.04**	94.46	**61.18**	0.08	93.62	60.35	0.08
5k	-	-	-	-	**96.3**	44.24	**0.06**	93.15	**60.97**	0.11	92.95	60.05	0.1
10k	-	-	-	-	**94.4**	44.23	**0.05**	91.46	**61.92**	0.13	87.11	61.91	0.11
15k	-	-	-	-	**92.08**	44.02	**0.06**	88.67	**63.68**	0.17	84.75	62.57	0.15

4.5 *RL-SORT* vs *HEURSORT*

Table 2 shows the performance comparison of $RL-SORT$ and $HEURSORT$ for various destination densities and scale with uniform load profiles. The performances of both are comparable at small to medium scale, while for large-scale data-sets $RL-SORT$ outperforms $HEURSORT$ significantly in terms of sorting efficiency η_s. This is due to the fact that while $RL-SORT$ uses multiple states to evaluate chute assignments, $HEURSORT$ only uses the processing time-stamp. Furthermore, $RL-SORT$ clearly outperforms $HEURSORT$ when one considers the mean of all data-sets of any particular destination density (irrespective of scale). Exactly similar results were observed for non-uniform load profiles.

Table 2: Performance comparison of $RL-SORT$ and $HEURSORT$ across uniform data-sets.

		$RL-SORT$			$HEURSORT$		
Density (Parcel/Destn)	Scale	η_s	η_b	t_c	η_s	η_b	t_c
100	5k	**93.37**	**61.56**	**0.06**	92.78	60.29	**0.06**
	10k	**91.5**	60.58	**0.06**	83.95	**60.82**	**0.06**
	15k	**88.57**	60.43	**0.07**	82.37	**60.87**	**0.07**
	mean	**91.15**	**60.86**	**0.06**	86.37	60.66	**0.06**
125	5k	**93.71**	**64.49**	0.09	93.47	62.18	**0.08**
	10k	**91.42**	63.1	0.07	85.04	**63.46**	**0.06**
	15k	**89.04**	63.15	**0.06**	83.81	**63.23**	**0.06**
	mean	**91.39**	**63.58**	**0.07**	87.44	62.96	**0.07**
150	5k	93.9	**66.81**	0.1	**94.58**	63.69	**0.08**
	10k	**89.93**	64.85	0.08	84.66	**65.02**	**0.07**
	15k	**89.08**	**65.31**	0.08	84.73	65.12	**0.07**
	mean	**90.97**	**65.66**	0.09	87.99	64.61	**0.07**

4.6 Chute Overloading

Table 3 shows the comparison of $RL-SORT$ and $HEURSORT$ for situations where chute(s) are overloaded; i.e., multiple parcels arrive for a chute(s) in quick succession, thereby resulting in higher rates of rejection. The data-sets are classified in three categories depending on when overloading happens (beginning, middle, or end of the wave). $RL-SORT$ outperforms $HEURSORT$ in terms of both the efficiencies.

Table 3: Performance comparison of $RL-SORT$ and $HEURSORT$ in situations involving chute overloading.

		$RL-SORT$		$HEURSORT$	
Scale	cat	η_s	η_b	η_s	η_b
5k	I	**79.88**	58.96	76.83	**59.1**
	II	**81.08**	**59.85**	77.81	59.47
	III	**81.66**	**60.84**	77.19	60.68
10k	I	**73.03**	**61.16**	67.76	60.74
	II	**75.02**	**61.8**	70.44	61.42
	III	**74.78**	**63.27**	68.69	62.46

5 CONCLUSIONS

In this paper, we studied and formulated a scalable deep RL based algorithm $RL-SORT$ for *joint* optimization of two NP-hard processes in an automated SC. Through experimentation, we demonstrated that $RL-SORT$ is able to reduce transportation costs and improve SC throughput while maintaining a low computational burden. This makes $RL-SORT$ especially useful for deployment in real-world situations. The future work involve expanding the ambit of $RL-SORT$ to include scheduling and routing of vehicles to solve the integrated logistics problem; and increase its robustness against real-time uncertainties and variations.

REFERENCES

Balcou, C., and V. Yadavalli. 2017. “A supply chain management model to optimise the sorting capability of a third party logistics distribution centre”. *South African Journal of Business Management* 48(1):77–76.

Benzi, F., E. Bassi, F. Marabelli, N. Belloni, and M. Lombardi. 2019. “IIoT-based Motion Control Efficiency in Automated Warehouses”. In *2019 AEIT International Annual Conference (AEIT)*. September 18th-20th, Florence, Italy, 1–6.

Crainic, T. G., G. Perboli, and R. Tadei. 2008. “Extreme point-based heuristics for three-dimensional bin packing”. *Informs Journal on computing* 20(3):368–384.

Fedtke, S., and N. Boysen. 2017. “Layout planning of sortation conveyors in parcel distribution centers”. *Transportation Science* 51(1):3–18.

Ghosh, S., A. Pal, P. Kumar, A. Ojha, A. Paranjape, S. Barat, and H. Khadilkar. 2021. “A simulation driven optimization algorithm for scheduling sorting center operations”. In *Winter Simulation Conference*, edited by S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, 1–12. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gupta, A., S. Ghosh, and A. Dhara. 2022. “Deep Reinforcement Learning Algorithm for Fast Solutions to Vehicle Routing Problem with Time-Windows”. In *5th ACM Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, 236–240. Virtual.

Harrison, R. 2019. “Dynamically integrating manufacturing automation with logistics”. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation*. September 10th-13th, Zaragoza, Spain, 21–22.

Hubbs, C., C. Li, N. Sahinidis, I. Grossmann, and J. Wassick. 2020. “A deep reinforcement learning approach for chemical production scheduling”. *Computers and Chemical Engineering* 141(1):106982.

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”. *IISE Transactions* 53(12):1353–1368.

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.

McWilliams, D. L., P. M. Stanfield, and C. D. Geiger. 2005. “The parcel hub scheduling problem: A simulation-based solution approach”. *Computers & Industrial Engineering* 49(3):393–412.

Motor Intelligence 2022, April. “Courier, Express, and Parcel (CEP) Market - Growth, Trends, CoVID-19 Impact, and Forecasts (2022-2027)”. Technical report.

Ojha, A., M. Agarwal, A. Singhal, C. Sarkar, S. Ghosh, and R. Sinha. 2021. “A generalized algorithm and framework for online 3-dimensional bin packing in an automated sorting center”. In *Proceedings of the* 7^{th} *IEEE Indian Control Conference*. December 20^{th}-22^{nd}, Mumbai, India, 135–140.

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.

Zhao, H., Q. She, C. Zhu, Y. Yang, and K. Xu. 2021. “Online 3D Bin Packing with Constrained Deep Reinforcement Learning”. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 35(1), 741–749. Virtual.

AUTHOR BIOGRAPHIES

DEEPAK MOHAPATRA is a researcher with TCS Research. He earned his M. Tech in Aerospace Engineering from Indian Institute of Technology, Bombay. His research interests include optimization, reinforcement learning, and robotics. His email address is deepak.mohapatra1@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.

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 reinforcement learning and optimization. His email address is ojha.ankush@tcs.com.

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 reinforcement learning, control and optimization. His email address is supratim.ghosh2@tcs.com.

MARICHI AGARWAL is a Researcher at TCS Research. She earned her M. Tech degree in High Performance Computing from National Institute of Technology, Durgapur. Her research interests are in the robotics and machine learning. Her email address is marichi.agarwal@tcs.com.

CHAYAN SARKAR is a Research Scientist with TCS Research. He holds a PhD in Internet of Things from TU Delft. His research interests are in the areas of human-robot interaction and multi-robot systems. His email ID is sarkar.chayan@tcs.com.