A Simulation Model to Evaluate the Layout for Block Stacking Warehouses

Shahab Derhami
Auburn University, Alabama USA

Jeffrey S. Smith
Auburn University, Alabama USA

Kevin R. Gue
University of Louisville, Kentucky USA

Abstract

Storing pallets of Stock Keeping Units (SKUs) on top of one another in lanes on a warehouse floor is known as block stacking. This storage system is widely used in manufacturing systems and distribution centers. The arrangement of lanes in the layout of this system significantly impacts utilization of the storage space and transportation costs. Existing research that studies the layout for this system focuses exclusively on determining the optimal lane depth with respect to space utilization and ignores transportation costs. In this study, we develop a simulation model that computes several performance metrics to evaluate both of these objectives for a warehouse layout. It aims to take the stochastic variations exist in the real world situation into account. Designing the layout based on the historical data distinguishes this model from the analytical models in the systems with high level of uncertainty, where determining the required parameters for analytical models are difficult due to the high variations. We verified the model using the existing analytical models and developed an experimental analysis to show the trade-off between the space utilization and transportation costs in the layout design problem.

1 Introduction

A block stacking warehouse is a unit load storage system where pallets of stock keeping units (SKUs) are stacked on top of one another in lanes on the warehouse floor. This storage system does not require any storage racks or other storage facilities and can be inexpensively implemented in any wide area. However, space planning is challenging in this inexpensive storage system. This system is widely operated under the *shared storage policy*. In this policy, empty lanes are available to all SKUs. However, to avoid blockage and relocation of pallets, an empty lane is entirely dedicated to a SKU once a pallet of that SKU is stored in the lane. This policy utilizes storage space more efficiently, though the order picking process might be generally less efficient in this system.

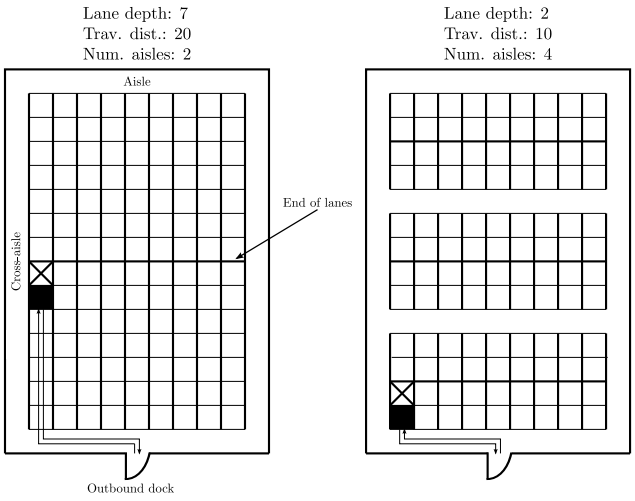

Figure 1: Transportation costs vs. space utilization with respect to the lane depth (lane depths and travel distances are in units of pallet).

This restriction leads to waste storage spaces in lanes when they are being filled or depleted. This is because there will be some unoccupied spaces in a lane that are only available to the assigned SKU. This effect is termed *honeycombing* and the waste associated with it is incurred to the system until a lane becomes entirely occupied or empty [2]. In addition to the honeycombing, aisles also contribute to the overall amount of wasted space. Aisles are not used for pallet storage but are required for access to the lanes. To utilize the storage space more efficiently, the layout must be designed such that these two types of waste are minimized.

The transportation costs are also impacted by the lane depths as well. Deep lanes create a layout that has few aisles and therefore utilizes the storage space more efficiently but suffers from higher transportation costs. The reverse is true for the shallow lanes. Hence, a trade-off between the space utilization and transportation costs must be taken into account in designing a warehouse layout. Figure 1 illustrates this trade-off for two extreme cases. The layout with deep lanes provides more storage locations (126 vs 108 pallet positions) but longer transportation distance (20 vs 10 pallet locations) to retrieve the same pallet position.

In this paper, we describe a discrete-event simulation model that evaluates both space utilization and transportation costs for a given warehouse layout. The proposed simulation model considers stochastic variations in the major production factors, like a real world situation, for the layout design problem. It evaluates a given warehouse layout with respect to several metrics pertinent to space utilization and transportation costs. This makes it a useful tool that can be used along with an optimization algorithm to find an optimal layout with respect to these two objectives.

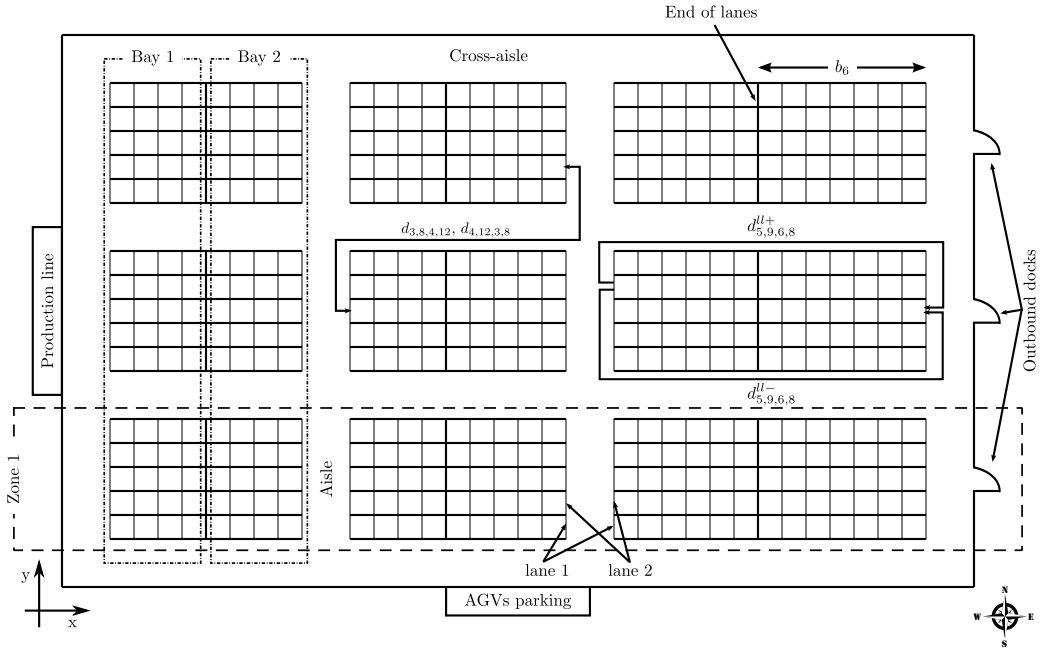

Figure 2: A typical warehouse layout considered in this paper.

The system simulated in this research is a manufacturing warehouse that consists of a production line, storage lanes, aisles, cross-aisles, outbound docks, AGVs (forklifts), and a parking for the AGVs (forklifts). The structure of the warehouse is given to the model. It is the definition, shape, and location of storage lanes, aisles, cross-aisles, the production line output queue, outbound docks, and the AGVs parking. The warehouse operations that are simulated in the model are generally of two types: production storage and outbound load. In a production storage operation, a pallet is produced and an AGV or forklift picks it up and delivers it to a floor stack. In an outbound load operation, a truck arrives at an outbound dock and an AGV is sent to pick up the requested pallet from the floor stack and deliver it to the truck. A sample warehouse layout considered in this research is shown in Figure 2.

This paper is organized as follows. First, we briefly review the related research on the warehouse layout design especially those that particularly studied block stacking. Next, we explain the proposed simulation model and its elements in section 3. Then, the experimental analysis is discussed in section 4 and finally, future research is described in section 5.

2 Related research

The research studied warehouse layout mostly considered it with respect to transportation costs [8, 16, 20, 13, 19, 15, 17, 18]. Few studies investigated designing the layout with respect to the other objectives such as storage costs [9], storage capacity [12, 21], and space utilization [7, 4]. Comprehensive reviews on the research about the warehouse layout can be found in [3, 6, 1]. Studies that particularly investigated block stacking are briefly reviewed in the following.

Kind [10] was the first scholar considered the trade-off between the lane depth and width to find the optimal lane depth that minimizes waste of storage space. However, he did not provide any derivations for his model. Later, Matson [14] proposed a more accurate model to approximate the optimal lane depth. Her model assumed instantaneous resupply (i.e., infinite production rate) like the warehouses that store products received from suppliers.

Goetschalckx and Ratliff [5] showed that if storage in multiple lane depths is allowed for a SKU, then the optimal lane depths follow a continuous triangular pattern. Larson et al. [11] proposed a class-based storage system to classify SKUs and allocate required storage space to the classes.

Derhami et al. [4] further developed Matson's model [14] and obtained the optimal lane depth that maximizes volume utilization under finite production rate constraint. They showed that using infinite production rate model in a finite production rate environment produces lane depths about twice as deep as they should be, but the resulting loss of space is not significant. That is because the space utilization curve, as a function of lane depth, is quite flat. However, shorter optimal lanes mean more aisles and therefore more flexibility with respect to travel for forklifts. They did not consider the effect of the lane depth on the transportation costs.

The aforementioned studies aimed to calculate the optimal lane depth that maximizes floor/volume utilization. Using these models, the warehouse layout can be designed only by dividing the warehouse floor into the lanes with depths equivalent to the optimal depth. However, there is another important factor significantly impacted by the warehouse layout, it is the transportation cost. Considering both space utilization and transportation costs simultaneously in designing the layout for a block stacking warehouse has not been adequately addressed in the literature.

In this study, we develop a simulation model that can be used as part of a simulation-based optimization model to find an optimal warehouse layout with respect to these two objectives. Our simulation model computes several performance metrics to evaluate a given layout with respect to these two objectives. Like a real world situation, it considers stochastic variations in the major production factors. This empowers it to achieve accurate results for an uncertain environment where existing analytical models fail due to the high variations exist in the main production factors and demand. In such a situation, the historical production and outbound

Algorithm 1 The overall pseudo-code for the simulation model.

```
call data-preparation
call distance-calculation
while the event list is not empty do
    event = the earliest event in the event list
    TimeNow = time of event
    call the corresponding procedure for the event
    remove the event from the event list
call performance-evaluation
```

load data can be used in the simulation model. This model can then be used as an evaluation tool along with an optimization algorithm to find the most effective layout with respect to both space utilization and transportation costs.

3 The simulation model

The proposed simulation model is an event-oriented simulation that simulates the production storage and outbound load operations in a warehouse while computing the desirable performance metrics. It consists of four modules: data preparation, distance calculation, main simulation, and performance evaluation.

The *data-preparation* procedure prepares the simulation input data. It generates the simulation event list from the real or randomly generated production and outbound load data. The *distance-calculation* procedure calculates the rectilinear distances between the locations of the warehouse where vehicles travel between them. All distances are stored and later used in the main simulation procedure.

The main simulation procedure consists of nine sub-procedures (events). These procedures simulate the entire warehouse operations. These operations are either a production storage operation where an AGV or a lift truck picks up a produced pallet from the production line and delivers it to a storage lane, or an outbound load operation where an AGV or a lift truck picks up a pallet from the storage area and delivers it to an outbound dock. The required parameters used to measure utilization of the storage volume and transportation costs are calculated while these operations are executed.

The *performance-evaluation* procedure is executed after all events in the simulation event list are executed. It calculates the performance matrices used to evaluate the layout in terms of space utilization and transportation costs. The overall pseudo-code of the simulation model is illustrated in Algorithm 1. The following assumptions, as presented in Figure 2, are presumed in the simulation model:

- Lanes are perpendicular to the left (west) side of the warehouse.
- For ease of navigation, all lanes in a bay have the same depth.

- Outbound docks are located on the right (east) side of the warehouse.
- One outbound dock exists in the middle of each zone.
- The production line is located on the left (west) side of the warehouse.
- The AGVs parking is located in the middle of the south side of the warehouse.
- All aisles and cross-aisles are bidirectional and wide enough to allow two AGVs cross each other (i.e. no AGV blockage)
- The AGVs traffic congestion is ignored.
- AGVs travel with a constant speed.
- Stochastic variations are added to the calculated travel times to incorporate the delays and variations caused by the traffic congestion and variable speeds of the vehicles on the turns and etc.

The elements of the model are explained in the following sections. The notation used in the main simulation procedure is described in Table 1.

3.1 *Data-preparation* procedure

The first step is to collect and prepare the simulation input data. Two types of data are collected:

- **Data related to the warehouse**, which includes warehouse dimensions, number of bays, number of lanes, number of cross-aisles, number of aisles, number of outbound docks, number of AGVs, AGVs average speed, bay depths, aisles and cross-aisles widths. This data mainly characterize the warehouse layout (See Figure 2).
- **Data related to SKUs,** which includes the number of SKUs, pallets heights, stackable heights, historical or randomly generated production and outbound load data, and initial inventory.

Unlike the traditional simulation models where the next event is scheduled after the current event of the same type is executed, our simulation model takes the simulation event list in advance. This provides more flexibility as either the historical (real) or randomly generated production and outbound load data can be used to build the entire simulation event list.

A row of the event list provided to the model at the start of the simulation contains the time of production or outbound load for a unit pallet of a SKU. So, if the historical production or outbound load data are in batch units, they must be expanded to pallet

Table 1: Table of notation.

n_a	number of aisles
n_c	number of cross-aisles
n_b	number of bays
n_l	number of lanes in a bay
n_A	number of AGVs
n_o	number of outbound docks
a^a	aisle width (in units of pallets)
a^c	cross-aisle width (in units of pallets)
s^h	warehouse height (in units of distance i.e., inch, cm)
s^l	warehouse length (in units of distance i.e., inch, cm)
s^w	warehouse width (in units of distance i.e., inch, cm)
A^s	AGV's average speed (in units of pallets/hour)
b_i	depth of bay i (in units of pallets)
T^s	total simulation time (in units of hours)
T^w	warm-up period time (in units of hours)
r^u	total distance traveled unloaded (in units of pallets)
r^l	total distance traveled loaded (in units of pallets)
t^u	total time that AGVs travel unloaded (in units of hours)
t^l	total time that AGVs travel loaded (in units of hours)
W_{ij}^H	honeycombing waste of storage space generated in lane j of bay i
O_{ij}	occupied space-time in lane j of bay i
E	simulation event list
L_w^p	pick-up waiting list for the production line
L_w^o	pick-up waiting list for outbound orders
t^p	time to load or unload a pallet to an AGV (in units of hours)
t_1^w	time that a pick-up request is added to either of the waiting lists (in units of hours)
t_2^w	time that a waiting pick-up request is assigned to an AGV (in units of hours)
ϵ_t	epsilon time unit

unit events. A batch of production can be expanded to the transactional events using random production times. This can be obtained by sampling production times from random distributions. We generate production times for each SKU from a symmetric triangular distribution whose most likely value is the average production time for the SKU and its lower and upper bounds are 20 percent lower and higher than the average production time.

For example, assume that the historical production data includes the production of a batch of three pallets of a SKU at time zero. If the average production time for this SKU is two hours and the three randomly generated processing times are 2, 1.8, and 2.1 hours, then three production events are added to the event list for this SKU at times 2, 3.8, and 5.9 hours. On the other hand, expanding a batch of an outbound load does not involve randomness and all pallets are scheduled at the same time. This

means, a historical outbound load of a batch of three pallets of a SKU at time zero, for example, is expanded to three outbound load events at time zero.

The other production events that must be scheduled in the event list are the production events to build the initial inventory (current state of the system). Hence, the initial inventory is built up in the model by producing and storing pallets of SKUs one by one like a regular production event. Then, the *warm-up* event is executed and resets all the variables used for performance evaluation to their initial values. The warm-up period must be scheduled once all pallets of the initial inventory are produced and stored – no matter how many AGVs work in the system. For this reason, the longest path that an AGV may travel to store a pallet and returns to the parking (a path from the AGVs parking to the production line and then from the production line to the furthest storage lane and then from there to the parking) is found and denoted as d^{max}. Then, the *pallet-production* events are scheduled with $d^{max}/(n_A A^s)$ hours time intervals for the entire inventory one by one. The warm-up time is then scheduled at

$$T^w > \frac{d^{max}(\sum_{i=1}^{n_s} v_i - 1)}{n_A A^s} + \epsilon_t \tag{1}$$

where n_s is the number of SKUs and v_i is the initial inventory of SKU i. This lower bound ensures that the *warm-up* event is scheduled after the initial inventory is entirely built up. The production and outbound load data are added to the event list starting from $(T^w + \epsilon_t)$.

For example, assume that a SKU has five pallets of initial inventory, d^{max} is 2000 pallets, A^s is 2500 pallets/hour, and two AGVs work in the warehouse. The *Pallet-production* events are scheduled for this SKU at times 0, 0.4, 0.8, 1.2, and 1.6 hours. The warm-up time can be set to any time after 1.6 hours.

3.2 *Distance-calculation* procedure

This procedure calculates rectilinear distances between the following locations:

- storage lanes.
- storage lanes and the production line.
- storage lanes and outbound docks.
- storage lanes and the AGVs parking.
- outbound docks and the production line.
- outbound docks and the AGVs parking.
- the production line and the AGVs parking.

This procedure is executed just once and computed distances are stored for later usage in the main simulation procedure. The following sections explain how these distances are calculated.

3.2.1 Distance between storage lanes

A rectilinear distance between two locations is obtained by summing distances traveled along the x-axis and y-axis. As it is shown in Figure 2, we used a Cartesian coordinate system and assumed the origin is located at the southwest corner of the layout. We numbered bays from left to right (i.e. west to east) and numbered lanes and cross-aisles from bottom to top (i.e. south to north). The x-coordinates for all lanes in a bay are identical (i.e., $x^l_{i,1} = ... = x^l_{i,j}$). For example, they are calculated for lanes in bays 1 to 3 in Figure 2 as

$$x^l_{1,1} = ... = x^l_{1,15} = a \tag{2}$$

$$x^l_{2,1} = ... = x^l_{2,15} = a + b_1 + b_2 \tag{3}$$

$$x^l_{3,1} = ... = x^l_{3,15} = 2a + b_1 + b_2 \tag{4}$$

The y-coordinates are calculated by defining zones. Lanes between two consecutive cross-aisles form a zone. That is, lanes located between cross-aisles i and $i+1$ create zone i. Zones determine how many cross-aisles are passed when traveling between two lanes. As demonstrated in Figure 2, lanes between the first and the second cross-aisles are assigned to zone 1, so $z^l_1 = ... = z^l_5 = 1$, $z^l_6 = ... = z^l_{10} = 2$, and so on. Zone assignment in Figure 2 started from the southern cross-aisle, but it can be done conversely. Since, we assumed that cross-aisles are equi-spaced, the number of lanes per zone for a bay is

$$n_{l/z} = \frac{n_l}{n_z} \tag{5}$$

where n_z is the number of zones. For the sake of simplicity, we assume that $n_{l/z}$ is an integer. Subsequently, the assigned zone for lane j is obtained by

$$z^l_j = \left\lceil \frac{j}{n_{l/z}} \right\rceil \tag{6}$$

The y-coordinates of the lanes are then calculated by

$$y^l_{ij} = j + a^c z^l_j \tag{7}$$

For the two lanes located in different zones, the distance is

$$d^{ll}_{ijkl} = \left|x^l_{ij} - x^l_{kl}\right| + \left|y^l_{ij} - y^l_{kl}\right| \tag{8}$$

but for the lanes located in an identical zone, it is the shortest path between the path that connects them from the cross-aisle located at the top of the lanes and the path

that passes from the cross-aisle located below them (see $d^{ll+}_{5,9,6,8}$ and $d^{ll-}_{5,9,6,8}$ in Figure 2). That is,

$$d^{ll-}_{ijkl} = \left|x^l_{ij} - x^l_{kl}\right| + (y^l_{ij} + y^l_{kl}) - 2(z^l_j(a^c + n_{l/z}) - n_{l/z}) \tag{9}$$

$$d^{ll+}_{ijkl} = \left|x^l_{ij} - x^l_{kl}\right| + 2(z^l_j(a^c + n_{l/z}) + 1) - (y^l_{ij} + y^l_{kl}) \tag{10}$$

$$d^{ll}_{ijkl} = d^{ll}_{klij} = min\{d^{ll-}_{ijkl}, d^{ll+}_{ijkl}\} \tag{11}$$

3.2.2 Distance between storage lanes and the production line

We assumed that the production line output queue located at the middle of the west side of the warehouse. So, its x-coordinate is zero and its y-coordinate is

$$y^p = \frac{s^w}{2} \tag{12}$$

The assigned zone to the production line is

$$z^p \approx \left\lceil \frac{y^p}{n_{l/z} + a^c} \right\rceil \tag{13}$$

The distance between the production line and the storage lanes located in the first bay or in the other bays but in different zones is

$$d^{lp}_{ij} = x^l_{ij} + \left|y^p - y^l_{ij}\right| \tag{14}$$

and for the lanes located in the same zone as the production line (except the lanes located in the first bay), it is calculated as

$$d^{lp-}_{ij} = x^l_{ij} + (y^l_{ij} + y^p) - 2(z^l_j(a^c + n_{l/z}) - n_{l/z}) \tag{15}$$

$$d^{lp+}_{ij} = x^l_{ij} + 2(z^l_j(a^c + n_{l/z}) + 1) - (y^l_{ij} + y^p) \tag{16}$$

$$d^{lp}_{ij} = min\{d^{lp-}_{ij}, d^{lp+}_{ij}\} \tag{17}$$

3.2.3 Distance between storage lanes and outbound docks

The x-coordinates of all outbound docks, x^o_i, are s^l and their y-coordinates are

$$y^o_i \approx \left\lfloor \frac{s^w}{n_o + 1} \right\rfloor i \tag{18}$$

So, their assigned zones are

$$z^o_i \approx \left\lceil \frac{y^o_i}{n_{l/z} + a^c} \right\rceil \tag{19}$$

The distance between dock k and the storage lanes located in the last bay or in the other bays but in different zones is

$$d^{lo}_{ijk} = \left|x^l_{ij} - x^o_k\right| + \left|y^l_{ij} - y^o_k\right| \tag{20}$$

and the distance to the lanes that are in the other bays but same zone as outbound dock k is obtained by

$$d^{lo-}_{ijk} = \left|x^l_{ij} - x^o_k\right| + (y^l_{ij} + y^o_k) - 2(z^l_j(a^c + n_{l/z}) - n_{l/z}) \tag{21}$$
$$d^{lo+}_{ijk} = \left|x^l_{ij} - x^o_k\right| + 2(z^l_j(a^c + n_{l/z}) + 1) - (y^l_{ij} + y^o_k) \tag{22}$$
$$d^{lo}_{ijk} = min\{d^{lo-}_{ijk}, d^{lo+}_{ijk}\} \tag{23}$$

3.2.4 Distance between storage lanes and AGVs parking

We assumed that the AGVs parking located in the middle of the south side of the layout. Thus, its x-coordinate, x^A, is $s^l/2$ and its y-coordinate is zero. The distance between the AGVs parking and all storage lanes is calculated by

$$d^{lA}_{ij} = \left|x^l_{ij} - x^A\right| + y^l_{ij} \tag{24}$$

3.2.5 Distance between outbound docks and the production line

The distance between the production line output queue and outbound dock k located in a different zone is

$$d^{po}_k = x^o_k + |y^p - y^o_k| \tag{25}$$

and distance to the outbound dock located in the same zone is calculated as

$$d^{po-}_k = x^o_k + (y^p + y^o_k) - 2(z^p(a^c + n_{l/z}) - n_{l/z}) \tag{26}$$
$$d^{po+}_k = x^o_k + 2(z^p(a^c + n_{l/z}) + 1) - (y^p + y^o_k) \tag{27}$$
$$d^{po}_k = d^{po}_k = min\{d^{po-}_k, d^{po+}_k\} \tag{28}$$

3.2.6 Distance between outbound docks and AGVs parking

The distance between outbound dock k and the AGVs parking is

$$d^{Ao}_k = \left|x^o_k - x^A\right| + y^o_k \tag{29}$$

Algorithm 2 Pseudo-code for *distance-calculation* procedure.

calculate $n_{l/z}$
set $B = 0,\ c = 0$
for all $(i \in \{1, .., n_b\})$ **do**
  if (i MOD 2 == 1) **then**
    $c = c + 1$
    $x^l_{i,1} = ... = x^l_{i,n_l} = B + a^a c$
  else
    $x^l_{i,1} = ... = x^l_{i,n_l} = B + a^a c + b_i$
  $B = B + b_i$
for all $(i \in \{1, .., n_b\}$ & $j \in \{1, .., n_l\})$ **do**
  calculate $z^l_j,\ y^l_{ij}$ using eqs. (6) and (7)
  for all $(l \in \{j + 1, .., n_l\})$ **do**
    calculate $d^{ll}_{ijil},\ d^{ll}_{ilij}$ using eq. (8)
  for all $(k \in \{i + 1, .., n_b\}$ & $l \in \{1, .., n_l\})$ **do**
    calculate $d^{ll}_{ijkl},\ d^{ll}_{klij}$ using eqs. (8)–(11)
calculate $y^p,\ z^p$ using eqs. (12) and (13)
for all $(i \in \{1, .., n_b\}$ & $j \in \{1, .., n_l\})$ **do**
  calculate d^{lp}_{ij} using eqs. (14)–(17)
for all $(k \in \{1, .., n_o\})$ **do**
  calculate $y^o_k,\ z^o_k$ using eqs. (18) and (19)
for all $(i \in \{1, .., n_b\}$ & $j \in \{1, .., n_l\}$ & $k \in \{1, .., n_o\})$ **do**
  calculate d^{lo}_{ijk} using eqs. (20)–(23)
$x^A = s^l/2$
for all $(i \in \{1, .., n_b\}$ & $j \in \{1, .., n_l\})$ **do**
  calculate d^{lA}_{ij} using eq. (24)
for all $(k \in \{1, .., n_o\})$ **do**
  calculate d^{po}_k using eqs. (25)–(28)
calculate d^{Ao}_k using eq. (29)
calculate d^{pA}_k using eq. (30)
report $\{d^{ll},\ d^{lp},\ d^{lo},\ d^{lA},\ d^{po},\ d^{Ao},\ d^{pA}\}$

3.2.7 Distance between production line and AGVs parking

This distance is obtained by

$$d^{pA} = y^p + x^A \tag{30}$$

Algorithm 2 presents the pseudo-code for the *distance-calculation* procedure. The traveling times are stochastic in the simulation model. We generate a random traveling time for each trip by sampling from a symmetric triangular distribution whose most likely value is the average traveling time between the selected locations and its lower and upper bounds are 20 percent lower and higher than the average time.

3.3 Main simulation procedure

The main simulation procedure consists of nine events (sub-procedures), which are executed until the simulation event list becomes empty. In each iteration, the earliest event is found and the simulation time is updated to the event time. Then, the procedure associated with that event is executed. Afterward, that event is removed from the event list and simulation proceeds by selecting the next earliest event. This procedure is shown in Algorithm 1.

Among the simulation events *pallet-production*, *outbound-pick-up*, and *warm-up* events, which represent production of a SKU, outbound request for a SKU, and the warm-up period, are scheduled in the event list in the *data-preparation* step. The rest of the events are scheduled in the simulation model according to their precedent events. The simulation events are described in the following.

3.3.1 *Pallet-production* event

In this step, a pallet of a SKU has been produced and sent to the production line output queue for storage. A pick-up request is issued. If there is at least one AGV available, a storage lane is assigned to this pallet and the nearest AGV to the production line is dispatched (not physically) to pick up the pallet. That is, the AGV resource is seized by setting the variable representing its availability to busy. Otherwise, if no AGV or empty space is available, the SKU is added to the pick-up waiting list to be picked up later when an AGV becomes available or a lane is freed up.

To avoid having multiple partially occupied lanes, we use the *unique storage lane* policy. In this policy, once an empty lane is assigned to a SKU, no other lane can be used to store that SKU until the assigned lane is fully occupied. Afterward, another empty lane is assigned to the SKU. We call this lane the *open storage lane* and model must keep track of these lanes for all SKUs. The same policy is used for depletion and the associated lane is called the *open depletion lane*. This means, at any simulation time, SKUs have at most one open lane for storage and one for depletion.

An empty lane can be assigned to a SKU randomly or by using a greedy algorithm to sort empty lanes with respect to the waste of storage space or total traveling distance. The waste of storage space for this purpose can be calculated using the models developed in [4].

Finally, r^u is updated by adding the distance between the current location of the assigned AGV and the production line. Similarly, t^u is updated by adding the time that it takes for the AGV to travel and pick-up the pallet. At the end, *pallet-pick-up* event is scheduled at ($TimeNow + t^p +$ travel time). Algorithm 3 presents the pseudo-code for this procedure.

Algorithm 3 The pseudo-code for *pallet-production*.

if an AGV or storage space is available **then**
  lane = the *open storage lane* for the produced SKU
  Capacity = capacity of the *lane* based on the SKU stackable height
  TotalInv = *on-the-way* pallets to the *lane* + its occupied positions
  if *TotalInv*==*Capacity* **then**
    mark *lane* as *fully* occupied
    find a new *lane* based on the selection criteria
  update *on-the-way* pallets to the *lane* by one
  find the nearest available AGV to the production line
  set the status of the selected AGV to busy
  TravDist=distance from the current location of the AGV to the production line
  $r^u = r^u + TravDist$
  $t^u = t^u + (TravDist/A^s) + t^p$
  add *pallet-pick-up* to E at $(TimeNow + (TravDist/A^s) + t^p)$
else
  add the SKU to L^p_w
  $t^w_1 = t^w_1 + TimeNow$

Algorithm 4 The pseudo-code for *pallet-pick-up*.

TravDist = distance between the production line and the *open storage lane*
TravInLane = lane depth - occupied floor positions in the *open storage lane*
$r^l = r^l + TravDist + TravInLane$
$t^l = t^l + (TravDist + TravInLane)/A^s + t^p$
$r^u = r^u + TravInLane$
$t^u = t^u + (TravInLane/A^s)$
add *storage-process* to E at $(TimeNow + t^p + (TravDist + 2 \times TravInLane)/A^s)$

3.3.2 *Pallet-pick-up* event

At this point, the assigned AGV has arrived at the production line and picked up the produced pallet. Since the storage location is known (from the *pallet-production* event), r^l can be updated by adding the distance between the production line and the *open storage lane*. Moreover, the distance that the AGV travels inside the lane must be calculated and added to r^l. It is equal to the depth of the *open storage lane* deducted by the occupied floor positions. t^l is updated by adding the time that it takes for the AGV to travel to and inside the assigned lane. The AGV unloads its load in the furthest available spot in the lane and then travels back to the entrance of the lane. So, this distance must be added to r^u and t^u. Finally, the *storage-process* event is scheduled at ($TimeNow + t^p$+ traveling time). The pseudo-code for this procedure is shown in Algorithm 4.

3.3.3 *Storage-process* event

At this point, the AGV has reached the storage lane, unloaded its load in the furthest available position, and moved back to the entrance of the lane. The honeycombing waste of storage space for this lane must be updated as one more pallet position of the lane has been occupied. It is updated for a lane j in bay i as follows.

$$W_{ij}^{H} = W_{ij}^{H} + (b_i s^h - k_{ij} h_s)(TimeNow - LaneLastEventTime) \tag{31}$$

where k_{ij} is the number of occupied pallet positions in the lane, h_s is the pallet height for SKU s (assigned SKU to this lane), and $LaneLastEventTime$ is the time of the last change in the inventory of the lane. The occupied space-time is updated as well

$$O_{ij} = O_{ij} + k_{ij} h_s (TimeNow - LaneLastEventTime) \tag{32}$$

Afterward, the variable representing the AGV's availability is set to *free*, indicating that it is available. Also, the location (status) of the AGV is updated to the location of the storage lane.

Since the AGV is released now, it can be assigned to the pallets waiting for pick-up at the production line or storage lanes (outbound requests). Hence, the pick-up waiting list for the production line is checked for any postponed pick-up request. If there are any requests, the *pallet-production* event is scheduled at $TimeNow$ for the SKU that has waited the most. Otherwise, the pick-up waiting list for the outbound requests is checked and the *outbound-pick-up* event is scheduled at $TimeNow$ for the SKU that has waited the most and has non-zero inventory.

If both queues are empty, the AGV must be sent to the AGVs parking, but first, we need to ensure that no coincident production or outbound event is going to seize it. Thus, the *release-AGV* event is scheduled at $(TimeNow + \epsilon_t)$ to ensure that this operation is pushed to the end of the processing sequence of all events scheduled at $TimeNow$. The pseudo-code for this procedure is described in Algorithm 5.

3.3.4 *Release-AGV* event

Once an AGV unloads its load, it is immediately sent to the AGVs parking if there is no pick-up request waiting. In the *release-AGV*, the AGV is seized again like if it physically started traveling to the parking. The distance between the AGV's current location to the parking is added to r^u and the traveling time is added to t^u. The *park-AGV* event is scheduled at ($TimeNow$ + traveling time). Algorithm 6 shows the pseudo-code for this procedure.

3.3.5 *Park-AGV* event

In this event, the released AGV arrives at the AGVs parking and waits for next assignment. Thus, the variable representing the AGV's status is updated to *free* and its current location is set to *parked at the parking*.

Algorithm 5 The pseudo-code for *storage-process*.

lane = the *open storage lane* for the storing SKU
update W_{ij}^H and O_{ij} for the *lane* using eqs. (31) and (32)
Set $LaneLastEventTime$ to $TimeNow$
increment inventory of the *lane* by one pallet
increment inventory of the SKU by one pallet
decrement *on-the-way* pallets to the *lane* by one pallet
update the status of the AGV to available
update the location of the AGV to the location of the *lane*
if L_w^p is not empty **then**
    add *pallet-production* to E at $TimeNow$ for the SKU waited the most
    $t_2^w = t_2^w + TimeNow$
    remove the SKU from L_w^p
else if L_w^o is not empty **then**
    find the SKU with the longest waiting time and non-zero inventory
    add *outbound-pick-up* to E at $TimeNow$
    $t_2^w = t_2^w + TimeNow$
    remove the SKU from L_w^o
else
    add *release-AGV* to E at $(TimeNow + \epsilon_t)$

Algorithm 6 The pseudo-code for *release-AGV*.

TravDist = distance between the AGV's current location and parking
$r^u = r^u + TravDist$
$t^u = t^u + (TravDist/A^s)$
update the status of the AGV to busy
add *park-AGV* to E at $(TimeNow + (TravDist/A^s))$

3.3.6 *Outbound-pick-up* event

In this event, a pick-up request has been placed to deliver a pallet from the storage area to an outbound dock. First, the status of the AGVs is checked to find a list of all available AGVs. If there is no AGV available or the inventory of the requested SKU is zero, the SKU is added to the pick-up waiting list and $TimeNow$ is added to t_1^w to keep the waiting times. But if at least one AGV is available, the depletion lane is selected and the nearest AGV to that lane is seized for delivery. So, the status of the selected AGV changes to *busy*, and r^u and t^u are updated accordingly.

To select the depletion lane at first, the *open depletion lane* for the requested SKU is checked. If it is not empty (considering *on-the-way* pick-ups), then it is used for depletion. Otherwise, one of the lanes that keep the requested SKU is selected as the *open depletion lane*. The selection can be random or by using a greedy algorithm that sorts lanes with respect to the waste of storage space or travel distance to the outbound area. The model proposed in [4] can be used to calculate waste of storage space for different bay depths. In this approach, the lane that generates the

Algorithm 7 The pseudo-code for *outbound-pick-up*.

if an AGV is available **and** the SKU inventory > 0 **then**
  lane = the *open depletion lane* for the requested SKU
  TotalInv = Inventory of the *lane* - sum of *on-the-way* pick-ups to the *lane*
  if *TotalInv* $== 0$ **then**
    mark *lane* as *empty*
    select a new *lane* with respect to the selection criteria
  find the nearest available AGV to the *lane*
  set the status of the selected AGV to busy
  increment *on-the-way* pick-ups to the *lane* by one unit
  TravDist=distance from the current location of the AGV to the *lane*
  TravInLane = *lane* depth - occupied floor positions in the *lane*
  $r^u = r^u + TravDist + TravInLane$
  $t^u = t^u + (TravDist + TravInLane)/A^s$
  add *depletion-process* to E at $(TimeNow + (TravDist + 2 \times TravInLane)/A^s + t^p)$
else
  add the SKU to L^o_w
  $t^w_1 = t^w_1 + TimeNow$

highest waste of space is depleted first. Finally, the *withdrawal-process* is scheduled at ($TimeNow$+ traveling time). Algorithm 7 presents this procedure in details.

3.3.7 *Depletion-process* event

At this step, the assigned AGV has picked up a pallet of the requested SKU and starts traveling to the assigned outbound dock. First, the inventory of the lane is decreased by one unit and the honeycombing waste of space and occupied space-time are updated for this lane using (31) and (32), respectively. If the inventory of the lane becomes zero, it is marked *empty* and becomes available to all SKUs. Then, distance between the storage lane and assigned outbound dock is added to r^l. Also t^l is updated by the time that the AGV requires to travel this distance and unloads the load. Afterward, the *truck-loading* event is scheduled at ($TimeNow$+ traveling time). The pseudo-code of this procedure is described in Algorithm 8.

3.3.8 *Truck-loading* event

In this procedure, the AGV unloads its load in a truck and becomes *available*. The location of the AGV is updated to the assigned outbound dock. Since one AGV becomes available, the pick-up waiting lists for the production line and outbound requests are checked for any waiting requests. If there are any requests waiting, the corresponding event is scheduled at $TimeNow$, otherwise the *release-AGV* event is scheduled at $(TimeNow + \epsilon_t)$. This procedure is presented in Algorithm 9.

Algorithm 8 The pseudo-code for *depletion-process*.

lane = the *open depletion lane*
update W_{ij}^H and O_{ij} for the *lane* using eqs. (31) and (32)
$LaneLastEventTime = TimeNow$
decrement *lane* inventory by one pallet
decrement SKU inventory by one pallet
decrement *on-the-way* pick-ups to the *lane* by one
TravDist=distance from the *lane* to the assigned outbound dock
TravInLane = *lane* depth - occupied floor positions in the *lane*
$r^l = r^l + TravDist + TravInLane$
$t^l = t^l + (TravDist + TravInLane)/A^s + 2t^p$
add *truck-loading* to E at $(TimeNow + (TravDist/A^s) + t^p)$

Algorithm 9 The pseudo-code for *truck-loading*.

update the status of the AGV to *available*
update the AGV's location to the assigned outbound dock
if L_w^p is not empty **then**
    add *pallet-production* to E at $TimeNow$ for the SKU waited the most
    $t_2^w = t_2^w + TimeNow$
    remove the SKU from L_w^p
else if L_w^o is not empty **then**
    find the SKU with the longest waiting time and non-zero inventory
    add *outbound-pick-up* to E at $TimeNow$
    $t_2^w = t_2^w + TimeNow$
    remove the SKU from L_w^o
else
    add *release-AGV* to E at $(TimeNow + \epsilon_t)$

3.3.9 *Warm-up* event

This event is executed just once and resets all variables used for performance evaluation (not the control variables) to zero. Algorithm 10 shows this procedure and lists variables impacted by this event.

3.4 *Performance-evaluation* event

The *performance-evaluation* procedure is executed after the main simulation procedure stops. The following performance metrics are calculated: average waste of storage volume ($\bar{W}^s$), utilization of storage volume ($\bar{U}^s$), percentage of wasted volume ($\bar{W}$), average AGVs utilization ($\bar{U}^A$), total travel distance (r), and average waiting time for pick-ups ($\bar{t}^w$). Among these metrics, $\bar{W}^s$, $\bar{U}^s$, and $\bar{W}$ are used to evaluate a layout in terms of space utilization. They show how well the warehouse space is utilized for storage. $\bar{U}^A$ and r evaluate the layout with respect to transportation costs. Finally, $\bar{t}^w$ and $\bar{U}^A$ can be used to determine the AGV fleet size in the system.

Algorithm 10 The pseudo-code for *warm-up*.

for all lanes **do**
  reset $LaneLastEventTime$ to $TimeNow$
  reset $W_{ij}^H,\ O_{ij}$ to zero
reset $t_1^w,\ t_2^w,\ r^u,\ r^l,\ t^u,\ t^l$ to zero

3.4.1 Average waste of storage volume

It is the average number of pallet positions wasted in the warehouse (in units of pallets $\times$ feet) and calculated as

$$\bar{W}^s = \frac{\sum_{i=1}^{n_b}\sum_{j=1}^{n_l} W_{ij}^H + s^h(n_a a^a s_e^w + n_c a^c s^l)(T^s - T^w)}{T^s - T^w} \tag{33}$$

where s_e^w is the effective warehouse width and is equal to $s^w - n_c a^c$.

3.4.2 Utilization of storage volume

It shows how well the warehouse volume is utilized for storage purpose.

$$\bar{U}^s = \frac{\sum_{i=1}^{n_b}\sum_{j=1}^{n_l} O_{ij}}{\sum_{i=1}^{n_b}\sum_{j=1}^{n_l}(O_{ij} + W_{ij}^H) + s^h(n_a a^a s_e^w + n_c a^c s^l)(T^s - T^w)} \tag{34}$$

3.4.3 Percentage of wasted volume

It shows what percentage of the warehouse volume wasted during the simulation period.

$$\bar{W} = \frac{\sum_{i=1}^{n_b}\sum_{j=1}^{n_l} W_{ij}^H + s^h(n_a a^a s_e^w + n_c a^c s^l)(T^s - T^w)}{s^h s^l s_e^w (T^s - T^w)} \tag{35}$$

3.4.4 Average AGV utilization

It shows the average utilization for AGVs.

$$\bar{U}^A = \frac{t^l + t^u}{n_A(T^s - T^w)} \tag{36}$$

3.4.5 Total travel distance

The total distance that AGVs traveled either loaded or unloaded.

$$r = r^u + r^l \tag{37}$$

Table 2: Characteristics of the SKUs in the test problem.

SKUs	Production rate (pallet/month)	Demand rate (pallet/month)	Production batch (pallet)	Stackable height (pallet)	Pallet height (ft)
SKU 1	20105	460	434	4	3
SKU 2	30785	187	275	4	3
SKU 3	21452	254	321	3	3
SKU 4	10063	2807	1248	4	4
SKU 5	27966	1262	728	4	4
SKU 6	29288	1385	763	4	4
SKU 7	9391	1911	980	2	4
SKU 8	30948	2941	1141	3	3
SKU 9	35027	2571	1054	2	5
SKU 10	31955	2867	1123	2	4

3.4.6 Average waiting time for pick-ups

The average time that pick-up requests had to wait for an AGV or empty space.

$$\bar{t}^w = \frac{t_2^w - t_1^w}{n_e} \tag{38}$$

where n_e is the total number of *pallet-production* and *outbound-pick-up* events in the simulation.

4 Experimental study

We tested our model on a test problem that consists of ten randomly generated SKUs. The average production and demand rates, production batch quantities, and pallets sizes for the SKUs are shown in Table 2. To use stochastic production and outbound load times in the simulation, we sampled these times from symmetric triangular distributions whose lower and upper bounds are 50% lower and higher than the average production and outbound load times presented in Table 2. The production batch quantities were obtained by the EOQ model. The warehouse specifications are listed in the following:

- $s^w = 1728$ ft
- $s^l = 3456$ ft
- $s^h = 16$ ft
- $n_c = 2$
- $n_A = 5$
- $n_o = 1$
- $a^c = 2$ pallets
- $a^a = 2$ pallets
- $A^s = 2$ miles/hour
- truck capacity: 20 pallets
- pallet size: 48×48 inches

We assumed that the number of requested pallets for a SKU in an outbound order varies between 1 to 5 pallets and they add up to 20 pallets for each order. Thus, they were randomly generated from a discrete uniform distribution with a mean of 3 pallets.

4.1 Model verification

The model was verified by comparing its results with the analytical model proposed by Derhami et al. [4]. To meet the assumption made for the analytical model, the test problem was simulated under deterministic rates for a layout with ten equally deep bays (i.e., 10 bays with 6 pallets deep, 7 aisles, and 2 cross-aisles). That is, the simulation was run with the deterministic and constant production and outbound rates shown in Table 2. Also, in order to comply with the unit load assumption presumed in the analytical model, the number of requested pallets for each SKU was set to one in the outbound orders. Moreover, the number of AGVs was increased to 100 to prevent any waiting time caused by the lack of AGVs. The simulation was run for 8 months and found the average waste of storage volume of 10985.9 yd^3. Using the same data, the average waste of storage volume obtained 10786.4 yd^3 by the analytical model developed in [4]. The 1.8% difference between the results is about the same estimation error reported in [4].

4.2 Simulation parameters

We tested different scenarios to find the best parameters for the initial inventory of the warehouse and warm-up period aiming to obtain accurate and reliable results while the computational cost is considered. We tested three different initial inventories: 0, 25, and 50 percent of the production batch quantities and four different warm-up periods: 0, 1, 2, and 3 months. So, totally 12 scenarios were simulated for 450 days and the space utilization was monitored. The space utilization converged after the same amount of simulation time in all scenarios regardless of the initial inventory of the warehouse. However, the zero initial inventory scenarios are more favorable in terms of computational cost as they do not require to build any initial inventory.

From the warm-up perspective, the space utilization converged approximately after 8 months in all scenarios except the zero warm-up scenarios, for which it took longer. However, the space utilization converged sooner in the one-month warm-up scenarios (almost after 3 months) allowing the model to collect more information in the remaining simulation period. Figure 3 presents the convergence of the space utilization for different warm-up scenarios with zero initial inventory. We chose zero initial inventory with one month warm-up period for our experimental study.

We tested different values for the number of replications and measured the Confidence Interval (CI) of the average space utilization within the replications using t-distribution. The CIs must be narrow such that the simulation provides statisti-

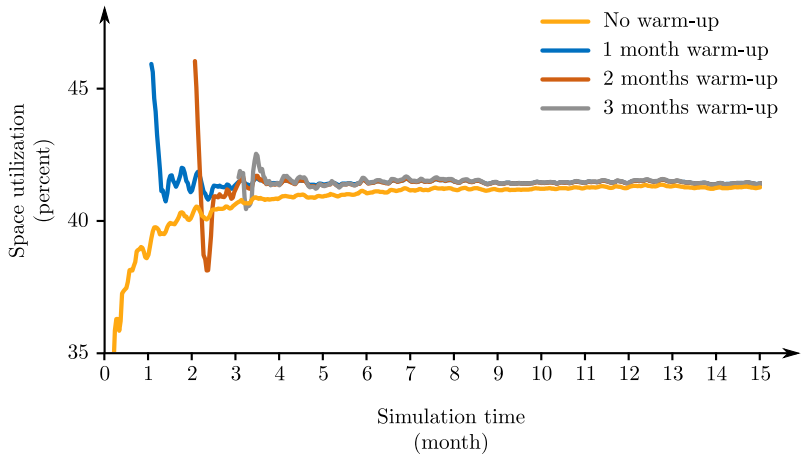

Figure 3: Space utilization for different warm-up scenarios.

cally valid results for the layout comparison. We arbitrarily chose a layout with 4 bays (2 bays with 16 pallets deep and 2 bays with 17 pallets deep) and simulated it for 2 to 24 replications. Figure 4 shows the CIs with respect to the number of replications. As the figure presents, the CI becomes smaller as the number of replications increases; however, the changes become smaller as the number of replications grows. We ran the simulation on a computer equipped with 3.5 GHz Intel Xeon processor with 8 cores and 16 GB of RAM memory. To better utilize the computational resources, we ran the simulation module in parallel on multiple processors. That means up to 8 replications could be handled simultaneously by the computer. Hence, the computational time between 1 to 8, 9 to 16, and 17 to 24 replications are almost the same. Figure 4 presents the computational times. The CI changes insignificantly from 8 to 24 replications while the computational time almost triples. Considering the run-times, we chose to replicate the model for 8 replications as it provided sufficiently narrow CI (0.001).

4.3 Experimental analysis

In this section, we describe our experiment with ten different layouts from shallow to deep bays aiming to illustrate the trade-off between the space utilization and transportation costs with respect to the bay depths. The layouts are obtained by dividing the warehouse effective length (warehouse length minus the sum of aisle widths) by the given number of bays. If one bay is much deeper than the others, the bay depths are adjusted such that they all become close to one another while the required number of bays is met.

For example, the layout with 10 bays is obtained by dividing the effective ware-

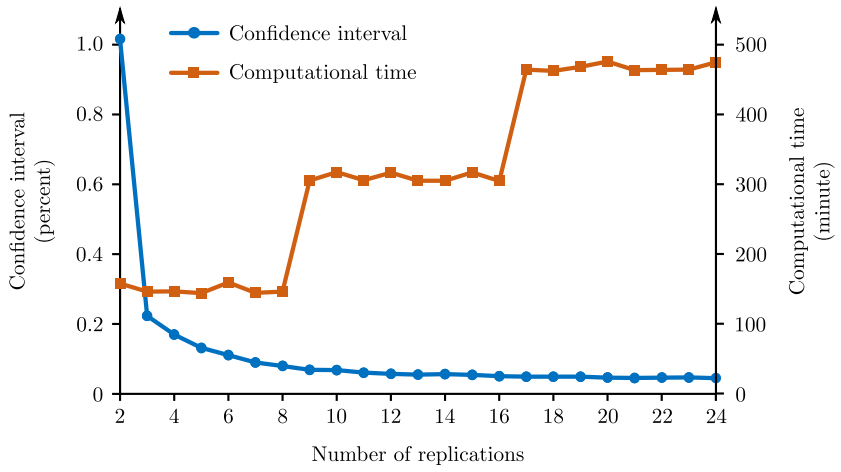

Figure 4: CI of average space utilization for different number of replications.

house length (72-(5+1)2 pallets) by 10, which yields 10 bays with 6 pallets deep. The layout with 20 bays is obtained by dividing (72-(10+1)2) by 20 which results in 19 bays with 2 pallets deep and 1 bay with 12 pallets deep. This arrangement can be adjusted to 10 bays with 2 pallets deep and 10 bays with 3 pallets deep. The adjustment aims to make the bay depths as close as possible. To perform a fair comparison, all other variables (including the simulation event list) were kept fixed in all ten experiments.

Table 3 shows the layouts and compares the proposed performance metrics for these layouts. Notice that we converted the units of $\bar{W}^s$ and r to yd^3 and mile using the pallet sizes. As the table presents, the space utilization improves as the number of bays increases from 2 to 4 (i.e., the bay depths decrease). Then, it diminishes as the bay depths become shallower. This is because the number of aisles increases and therefore, the storage space decreases. On the other hand, the average travel distance declines as the number of bays grows but, it becomes steady after 12 bays. As Figure 5 illustrates, there is a trade-off between the space utilization and total travel distance with respect to the lane depth. That is, the transportation costs improve at the cost of lower space utilization. However, this improvement is bounded and will be modest after a certain point. This shows optimizing just one of these factors without considering the other one results in poor performance of the other factor. Like the space utilization, $\bar{W}^s$ and $\bar{W}$ reach their best values for the layout with 4 bays and then they become worse as the number of bays grows.

The pick-up waiting times are incurred in two ways: waiting for an AGV or an empty storage space. Since the number of AGVs were kept fixed in all experiments, the growth in the $\bar{t}^w$ occurs as a result of reduction in the storage space. Increasing the number of bays results in having more aisles and therefore, less storage space

Table 3: Performance metrics for the layouts with different bay depths.

Number of bays	Bays depths* (pallet)	$\bar{W}^s$ (yd^3)	$\bar{U}^s$ (percent)	$\bar{W}$ (percent)	$\bar{U}^A$ (percent)	r (mile)	$\bar{t}^w$ (hour)	Run time (minute)
2	(34^2)	12376 ± 27.9	45.5 ± 0.07	50.4 ± 0.11	70.0 ± 0.24	33241 ± 126.1	5.5 ± 0.67	188
4	$(16^2, 17^2)$	11395 ± 33.5	47.7 ± 0.08	46.4 ± 0.14	64.4 ± 0.22	30317 ± 113.9	3.1 ± 0.50	196
6	$(11^5, 9)$	11460 ± 24.3	47.5 ± 0.09	46.6 ± 0.10	63.4 ± 0.19	29790 ± 99.4	3.0 ± 0.46	211
8	$(8^7, 6)$	11746 ± 26.9	46.8 ± 0.05	47.8 ± 0.11	62.6 ± 0.18	29346 ± 93.2	3.3 ± 0.52	209
10	(6^{10})	12130 ± 22.7	45.9 ± 0.08	49.4 ± 0.09	62.2 ± 0.18	29155 ± 90.2	3.9 ± 0.52	246
12	$(5^{11}, 3)$	12357 ± 35.4	44.3 ± 0.12	50.3 ± 0.14	61.4 ± 0.11	28737 ± 53.7	8.7 ± 0.68	234
14	(4^{14})	13045 ± 28.1	43.6 ± 0.09	53.1 ± 0.11	61.8 ± 0.14	28924 ± 70.5	6.1 ± 0.64	257
16	$(3^{10}, 4^6)$	13524 ± 29.2	42.4 ± 0.10	55.0 ± 0.12	61.6 ± 0.14	28827 ± 67.9	7.5 ± 0.65	294
18	$(3^{17}, 1)$	13977 ± 31.5	41.2 ± 0.11	56.9 ± 0.13	61.8 ± 0.11	28928 ± 50.6	9.2 ± 0.66	330
20	$(2^{10}, 3^{10})$	14419 ± 30.9	39.9 ± 0.11	58.7 ± 0.13	61.6 ± 0.08	28806 ± 34.4	11.1 ± 0.66	330

*(x^a, y^b) means the layout contains a bays with x pallets deep and b bays with y pallets deep.

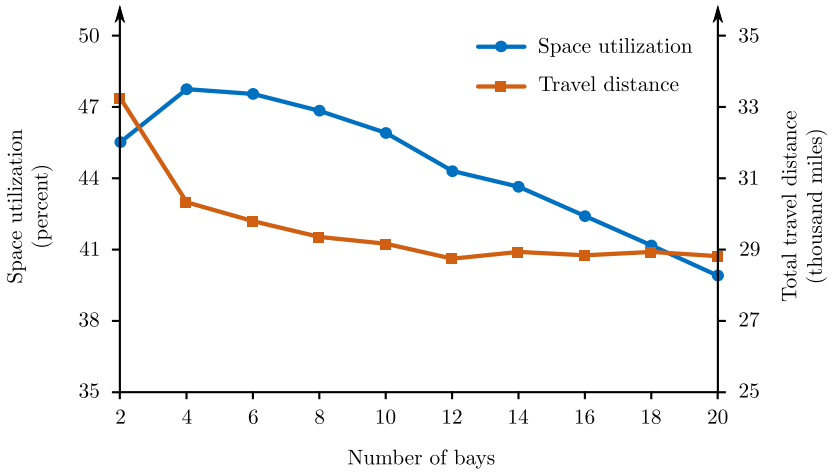

Figure 5: Average travel distance vs. space utilization for different bay depths.

remains in the warehouse. Hence, new pallets have to wait more for space to be freed up.

The AGVs utilization diminishes as the number of bays grows. This is because the total travel distance decreases as the number of bays increases and therefore, the AGVs are used less. The simulation run times have increased as the number of bays increased. This is because the storage space shrinks as the number of bays increases and the model has to reschedule (add to the waiting list) more *pallet-production* or *outbound-pick-up* events as the results of lack of storage space.

Another important finding of our experiment is that disregarding the stochastic variations in the model leads to a considerable error in estimating the performance of a layout in the practical cases. The average waste of storage volume for the 10 bays layout was obtained 10985.9 yd^3 under the deterministic conditions (see verification experiment in section 4.1) while it was reported 12130.3 yd^3 under stochastic condi-

tions (see Table 3). That means, ignoring the stochasticity of the problem resulted in approximately 10% underestimation of the waste of space in this case. This highlights the advantage of using the simulation model to design the warehouse layout.

5 Conclusions and future research

In this study, we developed a simulation model that computes performance metrics required to evaluate a warehouse layout with respect to space utilization and transportation costs. This model can also be used to find the optimal AGV (or lift truck) fleet size to operate the warehouse. The main advantage of this model is that it considers the stochastic variations and can work with the historical production and outbound load data. This is beneficial in the systems with high level of uncertainty where the main production and demand parameters change during the planning horizon. The existing analytical models fail to provide accurate results in such environments.

We tested our model on a test problem for various layouts and showed that there is a trade-off between utilization of storage volume and transportation costs with respect to the lane depth. A warehouse with deep lanes utilizes the storage volume better. However, shallow lanes incur less transportation costs. Our experiment shows that the utilization of storage space and transportation costs decrease as the bay depths decrease in the layout. However, the reduction in the transportation costs becomes modest at a certain lane depth. This finding can be used to find an efficient layout with respect to both of these two objectives. Our model can further be used as part of an optimization approach to seek for such a layout.

This paper is a part of a larger research that authors are conducting in designing the layout for block stacking systems. We aim to develop a simulation-based optimization approach to design the warehouse layout with respect to both space utilization and transportation costs. In that respect, our next step is to develop an optimization algorithm that works with the proposed simulation model to find an optimal layout. The optimization algorithm seeks for the best set of bay depths and the number of cross-aisles using the simulation model to evaluate candidate solutions with respect to both objective functions.

References

[1] P. Baker and M. Canessa. Warehouse design: A structured approach. *European Journal of Operational Research*, 193(2):425 – 436, 2009.

[2] J. J. Bartholdi III and S. T. Hackman. Warehouse & distribution science: release 0.96. *Atlanta, GA, The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology*, 2014.

[3] R. de Koster, T. Le-Duc, and K. J. Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481 – 501, 2007.

[4] S. Derhami, J. S. Smith, and K. R. Gue. Optimising space utilisation in block stacking warehouses. *International Journal of Production Research*, pages 1–17. doi: 10.1080/00207543.2016.1154216.

[5] M. Goetschalckx and D. H. Ratliff. Optimal lane depths for single and multiple products in block stacking storage systems. *IIE Transactions*, 23(3):245–258, 1991.

[6] J. Gu, M. Goetschalckx, and L. F. McGinnis. Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, 203(3):539 – 549, 2010.

[7] K. R. Gue. Very high density storage systems. *IIE Transactions*, 38(1):79–90, 2006.

[8] K. R. Gue and R. D. Meller. Aisle configurations for unit-load warehouses. *IIE Transactions*, 41(3):171–182, 2009.

[9] S. S. Heragu, L. Du, R. J. Mantel, and P. C. Schuur. Mathematical model for warehouse design and product allocation. *International Journal of Production Research*, 43(2):327–338, 2005.

[10] D. Kind. Elements of space utilization. *Transportation and Distribution Management*, 15:29–34, 1975.

[11] T. Larson, H. March, and A. Kusiak. A heuristic approach to warehouse layout with class-based storage. *IIE Transactions*, 29(4):337–348, 1997.

[12] M.-K. Lee and E. A. Elsayed. Optimization of warehouse storage capacity under a dedicated storage policy. *International Journal of Production Research*, 43(9): 1785–1805, 2005.

[13] W. Lu, D. McFarlane, V. Giannikas, and Q. Zhang. An algorithm for dynamic order-picking in warehouse operations. *European Journal of Operational Research*, 248(1):107 – 122, 2016.

[14] J. O. Matson. *The analysis of selected unit load storage systems*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, 1982.

[15] S. Önüt, U. R. Tuzkaya, and B. Doğaç. A particle swarm optimization algorithm for the multiple-level warehouse layout design problem. *Computers & Industrial Engineering*, 54(4):783 – 799, 2008.

[16] L. M. Pohl, R. D. Meller, and K. R. Gue. An analysis of dual-command operations in common warehouse designs. *Transportation Research Part E: Logistics and Transportation Review*, 45(3):367 – 379, 2009.

[17] F. Ramtin and J. A. Pazour. Analytical models for an automated storage and retrieval system with multiple in-the-aisle pick positions. *IIE Transactions*, 46 (9):968–986, 2014.

[18] F. Ramtin and J. A. Pazour. Product allocation problem for an AS/RS with multiple in-the-aisle pick positions. *IIE Transactions*, 47(12):1379–1396, 2015.

[19] K. J. Roodbergen and I. F. A. Vis. A model for warehouse layout. *IIE Transactions*, 38(10):799–811, 2006.

[20] K. J. Roodbergen, G. P. Sharp, and I. F. Vis. Designing the layout structure of manual order picking areas in warehouses. *IIE Transactions*, 40(11):1032–1045, 2008.

[21] S. Zhou, Y. Y. Gong, and R. de Koster. Designing self-storage warehouses with customer choice. *International Journal of Production Research*, 54(10):3080–3104, 2016.