University of Louisville

ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

5-2019

Maximizing space utilization in unit-load warehouses.

Luis Felipe Cardona Olarte
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

Part of the [Industrial Engineering Commons](#)

Recommended Citation

Cardona Olarte, Luis Felipe, "Maximizing space utilization in unit-load warehouses." (2019). *Electronic Theses and Dissertations*. Paper 3140.
<https://doi.org/10.18297/etd/3140>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

MAXIMIZING SPACE UTILIZATION IN UNIT-LOAD WAREHOUSES

By

Luis Felipe Cardona Olarte
M.Eng., Universidad Icesi, Colombia, 2014
B.Eng., Universidad Icesi, Colombia, 2011

A Dissertation
Submitted to the Faculty of the
J. B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Industrial Engineering

Department of Industrial Engineering
University of Louisville
Louisville, Kentucky

May 2019

MAXIMIZING SPACE UTILIZATION IN UNIT-LOAD WAREHOUSES

By

Luis Felipe Cardona Olarte
M.Eng., Universidad Icesi, Colombia, 2014
B.Eng., Universidad Icesi, Colombia, 2011

A Dissertation submitted on

April 19, 2019

to the following Dissertation Committee:

Professor Kevin R. Gue, Dissertation Director

Professor Monica Gentili

Professor Lihui Bai

Professor Csaba Biro

ACKNOWLEDGEMENTS

I am grateful to Dr. Leonardo Rivera for showing me the first steps into being an academic, to Dr. Kevin Gue always pursue the best of me, and to Dr. John Bartholdi III for laying up the horizon for all of us.

Thank you to Dr. Monica Gentili, Dr. Lihui Bai, and Dr. Csaba Biro for serving as part of my committee. My most sincere thanks to Fulbright-Colombia and to the Administrative Department of Science, Technology and Innovation (Colciencias) of Colombia for the Fulbright-Colciencias 2013 scholarship that allow me to course my PhD studies. To the Logistics and Distribution Institute at the University of Louisville for their financial support.

PROLOGUE

MAXIMIZING SPACE UTILIZATION IN UNIT-LOAD WAREHOUSES

Luis Felipe Cardona Olarte

April 19, 2019

This dissertation compiles three academic papers. The first paper is published, and the second and third are under-review. The topic of these papers is cumulative with each paper building on the results of the previous one. Most of the content of the papers has been taken verbatim, but it has been edited for consistency. Moreover, the dissertation includes an abstract, introduction, and conclusions that explain how the papers are interrelated and connects the implications of the cumulative research to existing literature and industry practices.

ABSTRACT

MAXIMIZING SPACE UTILIZATION IN UNIT-LOAD WAREHOUSES

Luis Felipe Cardona Olarte

April 19, 2019

In a unit-load warehouse, products are stored and retrieved in pallet quantities. Examples include retail distribution centers (DC), third-party DCs, and transshipment hubs in freight transportation. Expenses related to space are a significant component of the operational cost of unit-load warehouses; therefore, maximizing space utilization is important. Moreover, the continuing revolution of retail e-commerce is changing the role and design of modern distribution centers (Boysen et al., 2018). An important trend with serious implications for design is the desire of many distributors to locate DCs in or near metropolitan areas in order to support same-day service or better (Thuermer, 2018). Land in these areas is very expensive, so there is a need to make the best use of existing space. The ability to store more products in the same space increases inventory availability and therefore service, and the ability to store the same inventory in a smaller footprint reduces costs.

In this dissertation, we propose two strategies to improve space utilization in unit-load warehouses. We aim to minimize what we called *loss of vertical space within slots* (LVS)—the mismatch between the height of the pallet and the height of the slot where it is stored. LVS is a significant problem because it is standard practice to design storage racks in unit-load warehouses with all slots of equal height

(maximum pallet height) such that every pallet can fit in every slot; however, pallet heights vary greatly.

- (i) We propose the use of storage racks with multiple slot heights so that slot heights can better match the distribution of pallet heights. We analyzed historic (forecasted) inventory levels and the pallet heights to determine a robust design that guarantees a desired storage service level. Our method addresses the new warehouse design decisions that arise when having multiple slot heights: How to arrange the different slot heights in the rack-bays? How to organize the layout? How to avoid storage shortages? How do different slot heights affect travel times? We found that using multiple slot heights in unit-load warehouses has significant benefits in terms of footprint, expected travel time, and racking cost. For a typical warehouse, we expect space savings of 25–35 percent, depending on the number of slot types, and savings of 15–25 percent in annual operating cost.
- (ii) Although using multiple slot heights significantly decreases the loss of vertical space within slots, it does not completely eliminate it, and in warehouses where inventory levels are highly variable or product mixes change rapidly, this wasted space can still be significant. Examples of this situation in practice include warehouses with correlated order profiles, demands with seasonal peaks, new product launches, and distribution network consolidations. For such business environments, we propose pallet racks with dynamic heights as a way to maximize space utilization. Contrary to traditional pallet racks, the uprights and beams of pallet racks with dynamic heights are equipped with a mechanism to adjust slot heights easily. We found that pallet racks with dynamic heights have expected space savings of 16–30 percent when compared to traditional pallet racks.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
PROLOGUE	iv
ABSTRACT	v
LIST OF TABLES	ix
LIST OF FIGURES	x
I INTRODUCTION	1
A Strategies to maximize space utilization	4
1 Increasing occupancy	4
2 Design of aisles	7
3 Unit-load design	8
4 Automated storage systems	10
B Minimizing the loss of vertical space within slots	12
C Purpose of the dissertation	15
II MULTIPLE SLOT HEIGHTS IN UNIT-LOAD WAREHOUSES	18
A Service level in a warehouse with multiple slot heights	19
1 Validating the service level estimate	22
B Slot profiling	26
1 Determining slot quantities	27
2 Determining slot heights	29
C Potential benefits of using multiple slot heights	35
1 Why some warehouses benefit more than others	36
D Conclusions	38
E Appendix	41

1	Normality of aggregated inventory levels	41
2	Slot blocking	42
F	Upper bound of the joint probability	43
III LAYOUT OF WAREHOUSES WITH MULTIPLE SLOT HEIGHTS		45
A	Design of Rack-bays	45
1	Tiers	47
2	Cutting stock	50
3	Tiers versus Cutting Stock	59
B	Warehouse Design	61
1	Layout	63
2	Assigning cutting patterns to layout locations	64
C	Case Study	69
1	Parameters	69
2	Designing the Rack-bays	72
3	Creating the warehouse designs	73
D	Conclusions	77
IV PALLET RACKS WITH DYNAMIC SLOT HEIGHTS IN UNIT-LOAD WAREHOUSES		80
A	Potential benefits of pallet racks with dynamic heights	81
1	Warehouses with stationary inventory levels $\bar{\Delta}_\tau$	82
2	Warehouses with non-stationary inventory levels $\bar{\Delta}_v$	92
B	Three mechanisms for pallet racks with dynamic slot heights	97
C	Conclusions	103
V CONCLUSIONS		105
REFERENCES		107
CURRICULUM VITAE		125

LIST OF TABLES

1	Factorial design to validate $G(x, N)$	24
2	Probability of slot blocking	43
3	Slot heights (inches) of the rack-bays designs.	72
4	Metrics of the warehouse designs using multiple slot types.	75
5	Space utilization of cutting stock design for case study.	77
6	Factors and treatments of the completely randomized experiment with non-stationary time series.	93

LIST OF FIGURES

1	General functional areas of a unit-load warehouse.	2
2	Bulk storage area of a unit-load warehouse that we visited.	3
3	Difference between occupancy and space utilization.	4
4	Example of inventory levels from real data.	5
5	Storage policy	6
6	Aisles impact on space utilization	8
7	Unit-load composition	9
8	Challenges in unit-load design for an effective use of space.	10
9	Strategies to minimize loss of vertical space in pallet racks	11
10	Excess of space within slots beyond the required clearance.	13
11	Examples of racks where all slots have the same height (source: flexe.com)	14
12	AS/RS system with modular cells.	15
13	Dissertation overview	16
14	Rack with multiple slot types	18
15	Example of the multivariate cumulative normal distribution.	21
16	Standard Error of our estimate of the service level in a warehouse with multiple slot heights.	25
17	Tear drop pallet rack, which makes the set of possible slot heights discrete (source www.shelving.com).	29
18	Running times of the solution approaches.	32
19	Deviation of the heuristic search over the optimal value.	32
20	Example of Conjecture 2.4 for four and five slot types.	33
21	Comparison of the running time of the exhaustive search versus the enhanced search.	34

22	Improvement of the enhanced search over NOMAD	35
23	Expected benefits of using multiple slot types when compared to one slot type.	37
24	Example of $D(h, q)$ — Top view	38
25	Values of the features for the 3,600 problems.	39
26	Influence of different parameters on expected benefit.	40
27	Distribution of inventory levels.	41
28	Normal probability plot	42
29	Two methods to arrange slots into rack-bays.	46
30	TiersProfile — an enumerate procedure to a generate rack-bays design by tiers.	49
31	CSP-Enumerate, an optimal procedure to a generate rack-bays design by cutting stock.	53
32	OuterApproximation algorithm to solve P (Model 17).	55
33	Iterative process of OuterApproximation.	57
34	Example of the iterative procedure of CSP-Heuristic to solve a problem with three slot types.	58
35	Percentage difference in the number of rack bays of designs by tiers and by cutting stock.	60
36	Running times of algorithms to generate rack-bays.	61
37	Quality of the solutions provided by CSP-Heuristic.	61
38	Notation of warehouse layout model (Top view).	62
39	Simulation model of the storage system.	65
40	Example of a DOS_m assignment in a design with four cutting patterns.	66
41	Additional assignment methods for cutting patterns to layout locations.	67
42	Performance comparison of the five methods to assign cutting patterns to layout locations.	68
43	Example of the picking density for a layout generated by DOS_m	69

44	Example of a pallet heights distribution.	70
45	Data used in the case study.	70
46	Rack-bay patterns using four slot types.	72
47	Number of rack-bays for the rack-back designs.	73
48	Top view of warehouse layouts by cutting stock.	74
49	Total Operational cost	76
50	Difference between the wasted space within selective pallet racks and dynamic heights.	81
51	Queueing model of storage system.	84
52	Space savings of using pallet racks with dynamic heights $\bar{\Delta}_\tau(F)$ when F follows an uniform distribution.	89
53	Space savings of using pallet racks with dynamic heights when compare to pallet racks with fixed beams based on industry data.	92
54	Synthetic time series with irregular timestamps for D_{kt} with a horizon of 600 days.	94
55	Inventory signals generated from inventory policy (R, T) over randomly generated demands D_{kt}	95
56	Space savings of pallet racks with dynamic heights versus fixed beams	96
57	Pallet racks with modular cells.	98
58	Pallet racks with transferable beams.	99
59	Example of a potential mechanism (rail guides) for pallet racks with sliding beams.	100
60	Expected space savings of pallet racks with dynamic heights when com- pared to pallet racks with fixed beams with five slot types.	102

CHAPTER I

INTRODUCTION

In unit-load warehouses, operations are commonly divided into three main functions: receiving, bulk storage, and shipping. The flow of products typically follows a well-defined path. Pallets arrive in truck loads at the receiving docks. From there, lift truck operators unload the pallets and transfer them to the receiving staging area. Then, the pallets are processed and stored in the bulk storage area, which consists of pallet racks arranged to form parallel picking aisles. When customer orders arrive, lift truck operators pick the pallets from the bulk storage area and move them to the shipping staging area. Finally, the pallets are prepared for shipping and loaded into the trucks. Figure 1 depicts the layout of a typical flow-through distribution center, where the loading docks are located on the left side and the shipping docks are on the right side.

From the areas of the warehouse, bulk storage generally represents most of its footprint (Mulcahy, 2007). In Figure 2, we can see all the uses of the cubic space, but the only one adding value from a storage point of view is the space occupied by the product. As such, practitioners consider space utilization among the top five warehouse challenges (Blanchard, 2013) and a key performance indicator (Gu et al., 2010). Because of its importance, this work focuses on strategies to maximize space utilization in the bulk storage area.

Figure 1. General functional areas of a unit-load warehouse.

Figure 2. Bulk storage area of a unit-load warehouse that we visited.

The term ‘space utilization’ is often used ambiguously in practice to refer either to occupancy (number of stored pallets divided by the number of storage locations) or cubic utilization (cubic feet of product divided by the total cubic feet available). Here, we adopt the latter use. For example, the rack in Figure 3(a) has three slots, with two pallet positions each. The rack has 100% occupancy, but only 39% space utilization. The rack in Figure 3(b) is also 100% occupied, but has a higher space utilization (55%) because it wastes less vertical space within the slots. In this example, space utilization is the volume occupied by the pallet loads divided by the external dimensions of the rack-bay. In addition to the loss vertical space within slots, sources of wasted space include the clearance around the loads, the pallets themselves, and the racks.

Figure 3. Difference between occupancy and space utilization.

A Strategies to maximize space utilization

1 Increasing occupancy

Much research has been done on increasing storage occupancy, for which storage policies and warehouse sizing are the most important design decisions (Accorsi et al., 2012; Goetschalckx, 2012).

1.1 Outsourcing space for operational peaks

Organizations sometimes supplement their own warehouses with leased space during peak periods (Hung and Fisk, 1984). Figure 4 shows an example of real aggregated inventory levels. In this example, a warehouse with 440 pallet positions will provide a 100% service level, but its occupancy will be very low. To minimize leasing costs, warehouse managers are inclined to choose a storage capacity around 250 pallet positions with a service level of 95% and find temporary solutions for the peaks.

Figure 4. Example of inventory levels from real data.

Given aggregated inventory levels and a storage policy, the warehouse sizing problem determines the optimal mix of owned and leased storage (Gu et al., 2010). The objective may be to minimize the operational cost (Cormier and Gunn, 1996; Roll et al., 1989; Rao and Rao, 1998) or to guarantee a desired service level (Francis and White, 1992; Lee and Elsayed, 2005), where the service level is the probability that on any operational shift (an hour, half-day, or a day) the system is able to store every incoming pallet.

1.2 Storage policy

If a warehouse manager knows the stream of arriving and departing pallets for the next day, he could formulate an optimization model to produce a schedule of the put-away and picking operations. Unfortunately, the resulting integer program for the operation of a typical distribution center would take prohibitive time to solve (Goetschalckx, 2012). Most importantly, companies do not know the exact times of arrivals and departures days in advance. Therefore, companies have opted to solve the problem one pallet at a time through a set of management rules called storage policies. A storage policy encompasses two rules (Figure 5): one to assign a storage

location to an arriving pallet and another to decide which pallet to retrieve.

Figure 5. Storage policy

Literature has widely studied storage policies with early results from the 60's (Heskett, 1963). There are three major categories of policies:

- **Dedicated:** Each product is assigned to a dedicated set of storage locations, i.e., even when those locations are empty, they cannot be used by any other product. In this way, as products have dedicated locations the storage policy rules are trivial.
- **Shared:** All products can be stored in any storage location. In this way, when a pallet arrives any empty slot is a possible candidate and there are several rules to determine a specific slot; the most common one being to choose the closest open location (Park and Lee, 2007).
- **Class-based:** It is a mix between dedicated and shared. Products are organized in classes and each class is assigned to a different set of storage locations. However, products within the same class can be stored in any of the storage locations assigned to that class.

Dedicated policies favor labor costs and are easier to administer, but they have lower occupancy, which is why shared policies are most commonly used in bulk storage areas (Bartholdi and Hackman, 2017).

2 Design of aisles

As we can see in Figure 2, aisles are one of the major uses of the cubic space of a typical unit-load warehouse. Therefore, determining the right number and width of the aisles is critical. These decisions depend on the type of racks and lift trucks being used. In the most popular setting, warehouses use single-deep selective racks (AK Equipment, 2017) and counterbalanced lift trucks (Inventory Operations Consulting LLC., 2015). This implies aisles 12' wide (Kay, 1999) and that every pallet position in the layout must be directly accessible from an aisle (Figure 6(a)).

To reduce the space occupied by the aisles, the layout can have fewer aisles and make them narrower. Deep lane storage systems are the most common example—double deep, drive-in, and push-back racks (Tompkins et al., 2010). Warehouses using double deep racks may accommodate 41% more pallets within the same footprint than single deep selective racking (Figure 6(b)), and the benefits are even greater for drive-in and push-back racks which can be up to 15 pallets deep (Huang et al., 2010). Mobile racks achieve an even higher storage density (Boysen et al., 2017). Entire rows of racks are mounted on a rail system upon which they slide, leaving only one or two open aisles at the time. Similar to drive-in racking, pallet shuttle systems use deep lane storage, but instead of forklifts driving into each lane, a shuttle carries the pallets into deeper storage locations. This allows the system to be as deep as required (Tappia et al., 2019). Deep lane storage systems provide a high storage density by decreasing required aisle space; however, the additional storage locations are less accessible, resulting in a lower throughput (Reilly et al., 2017). To reduce aisle width, the warehouse can use narrow-aisle lift trucks, but they are more expensive, and narrow aisles can increase congestion (Gue et al., 2006), and, in turn, labor costs.

Figure 6. Aisles impact on space utilization

3 Unit-load design

In unit-load warehouses products are handled, transported, and stored in pallets, which are composed of cartons that hold the skus (Figure 7) — a stock keeping unit (sku) is the smallest unit of a product handled in a warehouse (Bartholdi and Hackman, 2017).

Pallets are a convenient way to handle products because they reduce processing times — instead of unloading, labeling, and scanning 300 water bottles, one only has to process one pallet. In exchange, one incurs unitizing costs and, more importantly, space utilization decreases, because products have different shapes, which are challenging to unitize (Figure 8). Determining the ideal unit-load is a widely studied problem (Tanchoco, 1983). One must find the ideal carton for each product and the best packing pattern for cartons into pallets (Trevino and Daboub, 1991) to minimize material handling, storage, and transportation costs.

In recent years, distribution centers have seen a growing number of skus (Berman, 2011) with harder-to-predict demands (Flynn et al., 2016), generating a rapidly changing mix of product sizes to administer. Gamberini et al. (2008)

Figure 7. Unit-load composition

propose a concurrent design of cartons, pallets, and racks for all products in a warehouse, with the objective of maximizing space utilization. They present an optimization model for how items are collected in cartons, cartons are collected in pallets, and pallets are stored in racks, as well as the appropriate dimensions at each stage. In their case study, they found that an integrated approach can increase the storage capacity of a rack system by up to 16.3%.

The study of the effects of packaging on the performance of the supply chain is known as “packaging logistics” (Saghir, 2004). It is a systemic approach for the design of unit-loads across the multiple stages in a supply chain (manufacturing, consolidation warehouses, distribution warehouses, transportation, and points of sale). When the supply chain is owned by a single company, where managers can influence every part of the unit-load design process, it is the most beneficial. “The truck is full, the pallet is full, the box is full, the shelves are full and the product is full” (García-Arca et al., 2014).

In the context of the Physical Internet, Lin et al. (2014) study cartons of standardized modular sizes in distribution centers with the goal of improving space

Figure 8. Challenges in unit-load design for an effective use of space.

utilization at the pallet level. They propose to restrict carton sizes to a set of modular sizes that can be easily combined to form pallets — similar to the LEGO construction toys. They applied this method to a warehouse that initially had 1,057 different carton sizes and reduced it to 80 modular sizes based on the size of a standard pallet. Because the cartons sizes are not being tailored to the product they hold, the space utilization decreases at a carton level, but the modular carton sizes generate better palletization patterns, which results in a better space utilization at the pallet level.

4 Automated storage systems

Unit-load automated storage and retrieval systems (Figure 9(a)) store products in pallet racking with narrow aisles, along which an automated crane travels to pick and put pallets away (Boysen and Stephan, 2016). These systems excel at having a high storage density — decreasing space requirements by 40% on average (Unarco Material Handling, Inc., 2010) — and a high throughput as they are fully automated. To achieve a higher space utilization, compact storage systems are a combination of AS/RSs and deep lane storage (de Koster et al., 2008). As in an AS/RS, an automated crane transports pallets horizontally and vertically, but

pallets are stored in deep lanes; either gravity flow racking, powered conveyor based racking, or a pallet shuttle system (Yu and De Koster, 2012). There also are innovations in this type of systems with advances in robotics and smarter shuttles (Azadeh et al., 2018) that make the system more productive, but we do not present them here as they do not focus on maximizing space utilization. Unfortunately, these systems are considerably expensive and rather inflexible (Guo and Liu, 2010; Lee et al., 1999).

(a) AS/RS (source: dematic.com)

(b) Compact storage systems (source: de Koster et al. (2008))

(c) Puzzle based storage (source: Gue (2006))

(d) Live cube (source: Zaerpour et al. (2017c))

Figure 9. Strategies to minimize loss of vertical space in pallet racks

Gue (2006) introduces another approach to reduce aisle space. Very high density storage systems store pallets in a compact grid as in the famous 15-puzzle game. In this system, unit-loads are placed on top of conveyor based units that can move pallets in the four cardinal directions. As they eliminate aisles all together, puzzle based systems achieve storage densities of least 90% (Gue and Kim, 2007) as a 2D system. Live-Cube is a fully automated compact storage systems that extends the puzzle based storage to multiple levels by integrating vertical shuttles (Zaerpour et al., 2017c). There are already commercially available options (ODTH First Class Logistics, 2017) that provide footprint savings of 40% to 50% (Zaerpour et al., 2017a). Furthermore, the system provides a high throughput, because the conveyor grid systems operate independently of lifts or shuttles, minimizing down times (Zaerpour et al., 2017b).

B Minimizing the loss of vertical space within slots

The purpose of this dissertation is to minimize this wasted space generated by the mismatch of pallet and slot heights (Figure 10), which we call loss of vertical space within slots (LVS).

LVS is significant because it is standard practice to design storage racks in unit-load warehouses with all slots of equal height (maximum pallet height) such that every pallet can fit in every slot. “All too often we see facilities where every pallet location is the same size” (F. Curties Barry & Company, 2016). Figure 11 shows four examples of warehouses where we can observe that all slots in the storage area have the same size and the loss of vertical space within slots that this creates.

To the best of our knowledge, only three papers have studied the use of unequal slots in storage areas to maximize space utilization. Lee et al. (1999) were the first to propose the use of storage location of unequal heights in AS/RS, but focused only on developing analytical expressions to estimate travel times of the shuttles in such a system. They call the system an AS/RS with modular cells. Lee

Figure 10. Excess of space within slots beyond the required clearance.

et al. (2005) further develop this idea with design guidelines for an AS/RS with modular cells. In this system, there are no beams, but the upright poles have lateral supports on top of which the pallets are placed (Figure 12). In this way, LVS is bounded by the height of the modular cell, which is expected to be far less than the slot height in a traditional rack system.

There are three downsides of adopting such a system: a) pallets need to be palletized leaving enough clearance at both sides for the lateral supports of the rack. In case pallets do not arrive with the proper clearance they will need to be repalletized; b) rack-bays can only hold one pallet width-wise, significantly increasing the number of uprights required and, in turn, the racking costs as the uprights are the most representative cost; and c) in the same ways as computer hard disk drives get fragmented, the picking and put-away activities can create scattered small spaces throughout the racks that are not suitable for storage.

Gallego et al. (2012) propose the use of unequal slots in pallet racks in work-in-process storage areas. The authors use a simulation-optimization approach for the design of the racks. First, they develop a discrete event simulation to model

(a) Montgomeryville, PA

(b) Boca Raton, Florida

(c) Carson, California

(d) Itasca, Illinois

Figure 11. Examples of racks where all slots have the same height (source: flexe.com)

the arrival and departure of pallets of different heights according to user-defined density functions. Then, a greedy heuristic uses the simulation model to search for the minimum number of rack-bays and the heights of slots for each bay to provide a desired service level. Our methods differ from theirs in three ways:

1. Their model takes as input the height H of a rack-bay and produces slots of sizes that evenly divide the rack: $H, H/2, H/3$ and so on, such that all slots within a column are the same size. However, there are a lot of ways in which a rack-bay can be partitioned in three slots, while their model only considers $H/3$. Our model explores the combinatorial problem exhaustively. Our model

Figure 12. AS/RS system with modular cells.

produces slot sizes directly from the data and then arranges them into columns following several patterns.

2. Their model assumes a stationary distribution of interarrival and departure times, whereas our method is based on historical, empirical data, and therefore reflects peak periods.
3. Their work addresses stochasticity with simulation, whereas we work directly with an empirical distribution from historical order data. This distinction allows us to address warehouses with thousands of skus rather than the dozens addressed in their paper.

C Purpose of the dissertation

In this dissertation, we propose two strategies to minimize the loss of vertical space within slots in unit-load warehouses. First, we propose the use of storage racks with multiple slot heights (Figure 13(b)) such that we can better match the slot heights to the distribution of pallet heights. Then, we propose a pallet rack

system with dynamic heights for warehouses in rapidly changing business environments (Figure 13(c)).

Figure 13. Dissertation overview

In Chapter II, we propose a two-phase procedure to determine the best profile of slots (heights and quantities) for a storage rack with multiple slot heights. First, we use a stochastic optimization model to find slot quantities given fixed values of slot heights to provide a desired service level. Next, we propose an enumeration procedure to find the best set of slot heights.

In Chapter III, we describe a method to generate layouts for unit-load warehouses that use multiple slot heights. The problem has two parts: slots must be arranged into rack-bays, and rack-bays must be arranged into a layout. For the first sub-problem, we describe two methods, depending on whether or not the warehouse has directed picking and put-away. For the second, we describe a simulation model and a greedy procedure based on the Duration-of-Stay storage policy.

In Chapter IV, we propose pallet racks with dynamic heights as a way to maximize space utilization in unit-load warehouses in rapidly changing business

environments. Contrary to traditional pallet racks, the uprights and beams of pallet racks with dynamic heights are equipped with a mechanism to easily adjust slot heights. First, we model a simplified version of the storage system using asymptotic methods of probabilistic analysis in the framework of the online bin packing problem. Next, we use simulation to evaluate the performance of three pallet rack systems with dynamic heights.

CHAPTER II

MULTIPLE SLOT HEIGHTS IN UNIT-LOAD WAREHOUSES

Storage racks in a unit-load warehouse typically have slots of equal height, whereas the unit-loads themselves have heights that vary significantly. The result of this mismatch is unused vertical space and storage spaces larger than they otherwise could be (Figure 14). We propose the use of storage racks with multiple slot heights to better match the distribution of pallet heights. The slot profile design problem seeks the best set of slot heights and their corresponding quantities such that a desired service level is met, where service level is the probability that all pallets present in a period can be stored.

Figure 14. Rack with multiple slot types

A Service level in a warehouse with multiple slot heights

In practice, service level in a unit-load warehouse is the ability to store an arriving pallet. The fundamental principle is that a pallet can be stored in any slot of height greater than its own, but cannot be stored in a slot of lesser height. For modeling purposes, *service level is the probability that all pallets present during an operational shift can be stored*, which is similar to service level Type I in inventory theory (Chu and Shen, 2010).

We assume discrete time periods during which a storage requirement is known or forecast. We call this period an “operational shift,” which could be a day, an 8-hour shift, or a 10-minute interval. All pallets in the warehouse during an operational shift are “served” if they can be assigned to slots; otherwise, there is a failure to serve or, in inventory terms, a stockout. We also assume an arriving pallet is stored in the smallest slot available and that pallets of a requested sku are picked from the largest slot in which they are stored. This policy is referred to as “best fit” in the online bin packing literature (Shor, 1991). In the context of warehousing, a best-fit policy should be easy to implement because it is a shared storage policy, and shared storage is commonly used in unit-load warehouses (Bartholdi and Hackman, 2017). It is worth noting that using a best fit policy could lead to higher travel times. For example, a typical shared storage policy for carton picking environments seeks to store pallets in the bulk (reserve) area in slots nearest the picking location, whereas the best fit policy seeks to maximize the probability that a future arriving pallet can be stored. Moreover, best fit is incompatible with first-in-first-out and other age-based selection policies, and might be difficult to integrate with hybrid or dedicated storage policies. Warehouses with these requirements can still benefit from using multiple slot heights, but would require different methods than the one we present here.

A *slot type* is a storage location of a specified height. To determine whether or not a pallet can be stored, we must consider not only its “preferred” slot type,

which is the smallest slot that can contain it, but also all larger slot types. The complementary view of this pallet-to-slot matching problem is that slots of a specific type (e.g., all slots of height 36") can receive any pallet with height less than or equal to that type.

Let a *slot profile* be the tuple $\langle x, N \rangle$, where $x = (x_1, x_2, \dots, x_L)$ are the slot heights and $N = (N_1, N_2, \dots, N_L)$ the corresponding slot quantities in a warehouse with L slot types. Let I_{kt} be the maximum inventory level of sku (stock keeping unit) k during operational period t , and h_k the pallet height of sku k . We use the terms ‘skus’ and ‘products’ interchangeably. To determine whether or not an inventory of pallets can be served, we must consider skus in groups according to height. Let C_i be the set of skus with pallet heights greater than slot height x_{i-1} ($x_0 = 0$ by convention). The aggregated inventory,

$$\mathcal{I}_{it} = \sum_{k \in C_i} I_{kt}, \quad (1)$$

must be stored in slot types corresponding to heights $(x_i, x_{i+1}, \dots, x_L)$. Said another way, those pallets cannot be assigned to slot types with heights $(x_1, \dots, x_{i-1})$.

The mathematical image of $\mathcal{I}_{it}$ is a discrete random variable (a function $f : X \rightarrow Y$ is said to have image Y) and we denote it by ξ_i . In Appendix 1, we show that despite ξ_i being a discrete random variable, it is reasonable to assume that it is normally distributed given the large number of skus handled in a typical distribution center and assuming the inventory levels of each sku are stationary.

Let $G(x, N)$ be the service level of a storage area with slot profile $\langle x, N \rangle$, and let $\mathcal{N}_i = \sum_{j=i}^L N_j$ be the cumulative number of slots of heights greater than or equal to x_{i-1} and $\mathcal{N} = (\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_L)$. Then, $G(x, N)$ is the probability that $\xi_i \leq N_i$ for all i ,

$$G(x, N) = \mathbb{P}(\xi \leq \mathcal{N}).$$

The random variables $\xi = \xi_1, \xi_2, \dots, \xi_L$ are not statistically independent because the aggregated inventory levels $\mathcal{I}_{it}$ are correlated. Therefore, we must use a

joint probability distribution to represent ξ and, given that ξ_i is normally distributed, this is the multivariate normal distribution. Consequently, the service level is

$$G(x, N) = F_{\mu, S}(\mathcal{N}),$$

where $F_{\mu, S}$ is the multivariate normal cumulative distribution function (Anderson et al., 1958), $\mu = (\mu_1, \mu_2, \dots, \mu_L)$ are the means of ξ , S is the covariance matrix of ξ , and ξ depends on the vector x . In explicit notation,

$$F_{\mu, S}(\mathcal{N}) = \int_{-\infty}^{\mathcal{N}_L} \dots \int_{-\infty}^{\mathcal{N}_2} \int_{-\infty}^{\mathcal{N}_1} \frac{1}{\sqrt{(2\pi)^L |S|}} \exp \left[-\frac{1}{2} (y - \mu)^T S^{-1} (y - \mu) \right] dy_1 dy_2 \dots dy_L. \quad (2)$$

Figure 15 shows a plot of $F_{\mu, S}(\mathcal{N})$ for an example with only two slot heights based of the aggregated historic inventory levels of a single company.

Figure 15. Example of the multivariate cumulative normal distribution.

Given a set of slots and an inventory history or forecast, we compute the service level using Algorithm 1.

Algorithm 1 Calculate service level $(h_k, I_{kt}, x, \mathcal{N})$

```
1:  $L \leftarrow \text{length}(x)$ 
2: for  $i \leftarrow 1$  to  $L$  do
3:    $\mathcal{I}_{it} \leftarrow \text{groupInventories}(x_i, I_{kt})$  ▷ Use Eq. 1
4: end for
5:  $\mu \leftarrow \text{averages}(\mathcal{I}_{it})$ 
6:  $S \leftarrow \text{covarianceMatrix}(\mathcal{I}_{it})$ 
7:  $p \leftarrow \text{Probability}(\mu, S, \mathcal{N})$  ▷ Use Eq. 2
8: return  $p$ 
```

1 Validating the service level estimate

In this section, we validate that $G(x, N)$ is a good estimate of the real service level $\Lambda(x, N)$ in a warehouse with multiple slot heights, where $\Lambda(x, N)$ represents what would happen if a warehouse with inventory levels I_{kt} had slot profile (x, N) . In statistical terms, this is a paired t-test to show that the mean difference between $\Lambda(x, N)$ and $G(x, N)$ is zero.

We ran a full factorial design experiment, where each experimental unit is a problem instance $P_s = \langle I_{kt}, h_k, x, \mathcal{N} \rangle$ generated from a synthetic, stationary time series. We generated each P_s in three steps: inventory levels I_{kt} , pallet heights h_k , and slot profile $\langle x, \mathcal{N} \rangle$.

- **Inventory levels:** We first generated synthetic time series to represent sku demands with the Python package TimeSynth (Maat and Malali, 2017). Each time series was generated based on: i) number of skus, ii) average, variance, minimum, and maximum of the number of periods with orders, and iii) average, variance, minimum, and maximum of the maximum quantity ordered per sku. The number of periods with orders and the order quantities were drawn from a truncated Gaussian distribution. We used signal types that can be parameterized to generate stationary time series, such as harmonic and Gaussian Matérn. The signal type for the time series of each specific sku was chosen randomly. We validate that the synthetic time series are stationary

using the Augmented Dickey-Fuller (ADF) test (Dickey and Fuller, 1979) implemented in the Python package statsmodels (Seabold and Perktold, 2010). Although the ADF test is one of the most popular tests for stationarity (Horváth et al., 2014), it must be used with caution as it has performed poorly in some simulation studies (Yung and Bentler, 1994). Therefore, we made a visual check when obtaining irregularities in the results.

After generating the sku demands, we developed a periodic review inventory policy simulator that transforms the demand of a sku k into I_{kt} based on a review cycle, a lead time, and the service level of the inventory policy (this service level should not be confused with the storage service level α). We used a horizon of 730 periods for all problems, equivalent to a year of a warehouse operation with two shifts. We randomly chose an initial inventory based on the maximum order quantity of each sku, but this leads to an initial chaotic behavior of the signal, for which we needed to specify a warm up period. To choose the length of the warm up period, we started at 20 periods and incrementally increased it until the ADF test suggested that the aggregated inventory levels were stationary.

- Pallet heights: We drew the pallet heights from one of the following distributions : uniform, truncated Gaussian, and Gaussian mixture (multimodal). We randomly chose the distribution and drew pallet heights based on a given minimum and maximum height. The specific parameters of the distribution such as mean and variance were also chosen randomly.
- Slot profile: We chose L slot heights evenly distributed in $[0, H]$, where H is the maximum pallet height. Then, given the desired storage service level α , we calculated optimal slot quantities N using Model 5.

We classified the parameters into factors and nuisance variables. The experiment has 9 factors (first half of Table 1) and 8 nuisance variables (second half

of Table 1). To choose realistic treatments for each parameter, we used a data set of real world distribution centers or our experience visiting DCs for some operational parameters. The experiment contained 3,840 experimental units with 5 replications each, resulting in 19,200 problem instances.

Parameter	Treatments
Number of skus	[1000, 5000, 8000]
Average periods with orders	[40, 120]
Variance of periods with orders	[400, 10000]
Average maximum quantity ordered	[2,8]
Variance of maximum quantities ordered	[4, 20]
Minimum pallet height	[10, 20]
Maximum pallet height	[40, 50]
Desired storage service level α	[0.85, 0.9, 0.95, 0.98]
Number of slot types L	[2,3,4,5,6]
Minimum periods with orders	10
Maximum periods with orders	730
Minimum maximum quantity ordered per sku	1
Maximum maximum quantity ordered per sku	20
Potential review cycles	[2,3]
Potential lead times	[3,5,9]
Service level of the inventory policy	[0.95, 0.98]
Pallet heights dist.	[uniform, trunc Gaussian, multimodal]

TABLE 1

Factorial design to validate $G(x, N)$

For each problem, we performed a paired t-test for the difference between $\Lambda(x, N)$ and $G(x, N)$. The p -value of the test was 0.742 greater than any meaningful critical value; consequently, there is no statistically significant difference between $\Lambda(x, N)$ and $G(x, N)$. Figure 16 plots a histogram of the difference between $\Lambda(x, N)$ and $G(x, N)$. Seventy one percent of the problem instances had an error within

$(-1\%, 1\%)$, and 97% of the problem instances had an error within $(-2\%, 2\%)$, which we found to be a reasonable standard error for our estimate.

Figure 16. Standard Error of our estimate of the service level in a warehouse with multiple slot heights.

We have one last detail to cover: We have assumed to this point that successful storage of all pallets in an operational shift is independent of the requirement during other shifts. However, this is not the case in practice. Consider a warehouse with one small slot and one large slot. On shift 1, two small pallets arrive, each containing a different sku. One is stored in the small slot, and one in the large slot. On shift 2, the sku in the small slot leaves to satisfy an order, and a large pallet arrives during shift 3. The pallets present in shift 3 could be successfully stored according to our model, but the actual sequence of arrivals and departures would have led to a stockout in practice, assuming there is no re-shuffling allowed. We call such an event “slot blocking.”

The likelihood that slot blocking occurs depends on the storage policy: in which slot should an arriving pallet be stored, and from which slot should a requested pallet be picked? The storage policy assumed in our method is “best fit.” In Appendix 2, we show that when using the best fit policy, slot blocking is negligible. Therefore, we calculate the service level directly from $G(x, N)$.

B Slot profiling

In this section, we determine the slot profile $\langle x, N \rangle$ that minimizes the sum of the heights of all slots, which is a surrogate for minimizing total space in the layout. In practice, the total space consumed in a layout is the sum of the space occupied by the rack-bays, into which a slot profile must be assigned. We have left the integrated problem beyond the scope of this paper and chosen instead to minimize the space required for the slot profile itself.

The problem has two constraints: the warehouse has to provide a desired service level α , and for modeling purposes only, we require the slot heights to be sorted in non-decreasing order. The data needed for the model are the historic maximum inventory levels I_{kt} of each sku k during period t and the pallet heights h_k . We know that the largest slot type must accommodate the largest sku, so we fix x_L to be the maximum pallet height H . The model is,

$$\begin{aligned}
& \min \quad \sum_{i=1}^L x_i N_i \\
& \text{s.t.} \quad G(x, N) \geq \alpha \\
& \quad \quad x_i \geq x_{i-1} \quad \quad \quad 2 \leq i \leq L-1 \\
& \quad \quad x_L = H \\
& \quad \quad x_i \in \mathbb{R}^+, N_i \in \mathbb{Z}^+ \quad \quad \quad \forall i.
\end{aligned} \tag{3}$$

Model 3 is a mixed-integer nonlinear optimization problem with a nonlinear, non-smooth constraint. Using randomly generated problems from a database gathered from industry data (see Section 2.3 for details), we tested nonlinear solvers like fmincon (The MathWorks, Inc., 2016), Ipopt (Wächter and Biegler, 2006), Nlopt (Johnson, 2017), Knitro (Artelys, 2017), Snopt (TOMLAB, 2017) and SCIP (Achterberg, 2009), but none was able to solve the model for $L > 3$. The computation of $G(x, N)$ introduces numerical instability that solvers cannot overcome. Therefore, we separate the problem into two parts. The first model

(Section 1) determines the optimal slot quantities N , given fixed values of slot heights x . The second model (Section 2) finds the best slot heights x using the first model to calculate the optimal N at each step of the search.

1 Determining slot quantities

Given a vector of slot heights x , the following model determines the optimal numbers of each slot type:

$$\begin{aligned}
& \min \quad x^T N \\
& \text{s.t.} \quad F_{\mu,S}(\mathcal{N}) \geq \alpha \\
& \quad \mathcal{N}_i = \sum_{j=i}^L N_j \quad \forall i \\
& \quad N_i \in \mathbb{Z}^+ \quad \forall i;
\end{aligned} \tag{4}$$

however, solvers could not overcome the numerical instability produced during the computation of $F_{\mu,S}$. Therefore, we solve the problem in terms of the vector z instead of N , where $\mathcal{N}_i = \mu_i + z_i \sigma_i$, μ_i is the mean of ξ_i , and σ_i is the standard deviation of ξ_i . Additionally, we normalize $F_{\mu,S}(\mathcal{N})$, so that μ is an all zeros vector, and we replace S by the correlation matrix ρ . We also normalize the vector of coefficients in the objective function. Because the multivariate normal distribution is logconcave (Kall and Mayer, 2011), we replace the service level constraint by its logarithm to make it concave. Finally, in Appendix F, we show that $\text{norminv}(\alpha)$ is a lower bound of z_i ; therefore, we include it as a boundary to facilitate the search.

We reformulate Model 4 as

$$\begin{aligned}
& \min \quad c^T z \\
& \text{s.t.} \quad \log(F_\rho(z)) \geq \log(\alpha) \\
& \quad \mu_i + z_i \sigma_i \geq \mu_{i+1} + z_{i+1} \sigma_{i+1} \quad i < L \\
& \quad z_i \geq \text{norminv}(\alpha) \quad \forall i,
\end{aligned} \tag{5}$$

where $c = d / \|d\|$, $d = x_1 \sigma_1, (x_i - x_{i-1}) \sigma_i, \dots, (H - x_{L-1}) \sigma_L$. The first constraint guarantees a desired service level. The second constraint guarantees that

$\mathcal{N}_1 \geq \mathcal{N}_2 \geq \dots \geq \mathcal{N}_L$, otherwise, we can obtain solutions that represent negative slot quantities.

Model 5 is a stochastic optimization problem with a linear objective and a joint chance constraint (Kall and Mayer, 2011). It is a convex problem, because the objective function is linear and the constraint is concave. The main challenge is computing $F_\rho(z)$, which can only be approximated by Monte Carlo methods for $L > 4$ (Mayer, 2000). Here, we use the implementation of the statistical module of SciPy (Jones et al., 2001) in Python calculate $F_\rho(z)$. Furthermore, we implemented the method proposed by Prékopa (1995) for the calculation of ∇F , so the solvers do not have to perform ∇F by finite series approximation.

Prékopa (1995) establishes that computing ∇F is equivalent to computing the conditional probability of F after applying a special transformation on the parameters, and given that we already have efficient algorithms to compute F , we take advantage of this relation to compute ∇F . The method is based on the relation

$$\nabla F_i = \tilde{F}_i(\tilde{z}) f_i(z_i), \quad (6)$$

where $f_i(z_i)$ is the univariate standard normal probability density function of ξ_i ; $\tilde{F}_i(\tilde{z})$ is the standard multivariate normal distribution of the correlation matrix $\tilde{\rho}$; and where

$$\tilde{\rho}_{jk} = \frac{\rho_{jk} - \rho_{ji}\rho_{ki}}{\sqrt{1 - \rho_{ji}^2}\sqrt{1 - \rho_{ki}^2}}, \quad \tilde{z}_k = \frac{z_k - \rho_{ki}z_i}{\sqrt{1 - \rho_{ki}^2}}, \quad j, k = 1, \dots, L; j \neq i, k \neq i.$$

After these modifications, we could solve Model 5 using the Python interface of Nlopt (Johnson, 2017). The starting point z_0 is a vector with all entries equal to $\text{norminv}(\alpha^{1/L})$. Choosing a good starting point for chance constrained stochastic problems is critical for convergence (Mayer, 2000), because as can be seen in Figure 2, the multivariate normal distribution has very flat regions for big and small values of $\mathcal{N}$. In these regions, the gradient is almost zero, and consequently solvers get lost. Once we have solved Model 5, we calculate the aggregate slot quantities vector $\mathcal{N}$ from $\mathcal{N} = \lceil \mu + z\sigma \rceil$, and the slot quantities vector N from $\mathcal{N}$.

2 Determining slot heights

Let $N(x)$ be the optimal slot quantities from Model 5. The model to find the optimal slot heights is

$$\begin{aligned} \min \quad & x^T N(x) \\ \text{s.t.} \quad & x_i \geq x_{i-1} \quad 2 \leq i \leq L-1 \\ & 0 \leq x_i \leq H \quad \forall i. \end{aligned} \tag{7}$$

The slot heights are continuous variables in Model 7; however, the beams of storage racks typically can only be adjusted in increments of two inches (Figure 17); therefore, we define the slot heights in a discrete domain, which limits the size of the feasible region and allows an exhaustive search for up to seven slot types. For more slot types, exhaustive search takes prohibitive time; therefore, for more than seven slot types, we use a Mesh Adaptive Direct Search (Le Digabel, 2011), which provides a near optimal solution in reasonable time.

Figure 17. Tear drop pallet rack, which makes the set of possible slot heights discrete (source www.shelving.com).

2.1 Exhaustive search

Let $x = (x_1, x_2, \dots, x_{L-1})$ be a point in solution space; then, the exhaustive search consists of nested loops that evaluate all x in the Cartesian power B^{L-1} such that $x_1 < x_2 < \dots < x_{L-1}$, where B is the set of even numbers between the minimum

and maximum pallet height. If B has n elements, the number of elements in the solution space is equal to the binomial coefficient $\binom{n}{L-1}$. In the case of five slot types and a maximum pallet height of five feet, the algorithm solves 1,107,568 stochastic optimization problems to find the optimal heights.

2.2 Mesh adaptive direct search (MADS)

MADS is a heuristic for black-box optimization that starts at an initial point provided by the user (Audet and Dennis, Jr., 2006). Then, the algorithm looks in the neighborhood (mesh) of the current point using a so-called polling method. Next, it performs the optional step of local search to improve the current solution. Afterwards, it varies the mesh size and the way that the local search and polling method interact for the next iteration. The algorithm continues iterating until a maximum of iterations is reached or no improvement is achieved through a predefined number of iterations.

We use NOMAD (Audet et al., 2017) through its Python interface and integrate it with the scientific library for parallel optimization Pygmo (Izzo, 2012). The algorithm starts from a population of 10 randomly generated feasible solutions in 8 parallel processes. It evolves using NOMAD with a mesh having twice as many elements as the number of variables and an extreme barrier approach to deal with the constraints (Audet et al., 2010). The algorithm stops when no further improvement can be obtained through iterations with a tolerance of $1e-8$.

2.3 Comparing procedures

Here, we compare the performance of the exhaustive search versus the heuristic search in terms of running time and the quality of solutions. We gathered inventory history and pallet heights data from several companies. From three distribution centers we obtained the pallet heights, from one, we obtained data of inventory history and pallet heights, and from another only the inventory history.

Unfortunately, no single company could provide the inventory history and pallet heights for every sku. The final database contained pallet heights for 54,888 skus and inventory history for 9,321 skus.

We used this database to generate industry-based problem instances to test both procedures. A problem instance is generated based on a scenario. A scenario is the tuple $\langle n, \alpha, L \rangle$, where n is the number of skus, α the desired service level, and L the number of slot types. To generate a problem instance from a scenario, we randomly choose n skus and take their pallet heights h_k and inventory history I_{kt} from the database. A problem instance is denoted by the tuple $\langle h_k, I_{kt}, \alpha, L \rangle$.

To study the running time of the exhaustive search, we generate scenarios using a general full factorial design for up to 6 slot types. The factors and corresponding levels were slot types $L = (2, 3, 4, 5, 6)$, desired service level $\alpha = (0.7, 0.8, 0.98)$, and number of skus $n = (100, 700, 2000, 8000)$. The experiment had 60 scenarios and we ran 5 replications, resulting in 300 problems. For 7 slot types, we ran fewer scenarios given the high running times. We ran 10 replications with $\alpha = 0.8$ and $n = 8000$. To analyze the running times of the heuristic search, we use the same general full factorial design, but for up to 20 slot types.

Using Minitab (2010) to process the results, we determined that the only significant factor that affects the running times is L . Figure 18(a) and 18(b) show the running times of the exhaustive and the heuristic search respectively. We ran the scenarios on a desktop computer running Ubuntu 16.04 with an Intel(R) Core(TM) i7 3.5 GHz with 8 cores and 16GB RAM. Both algorithms were implemented in Python 3.6. We noticed that the variability of running times for 11 slot types was unexpectedly high. After running the experiment for this case several times and obtaining the same result, we cannot explain this observation. Nevertheless, the running times are within a reasonable range (20 to 50 minutes).

To study the quality of the solutions provided by the heuristic search, we use it to solve each of the 310 problems. Given that the heuristic search works with a

Figure 18. Running times of the solution approaches.

continuous relaxation of the problem, we truncated its solution to make it discrete and comparable with the solution provided by the exhaustive search. Figure 19 shows the deviation over the optimum value of the heuristic search. On average, the solution found by the heuristic is within 1.3% of the optimal value and in 95% of the problem instances, the truncated heuristic search found a solution within 2.6% of optimal.

Figure 19. Deviation of the heuristic search over the optimal value.

2.4 A faster heuristic

We noticed a pattern in the optimal slot heights for every problem instance we solved, which we can illustrate with an example. For a warehouse with five slot types, it was always the case that,

$$\begin{aligned} x_{15} &\leq x_{14} \\ x_{14} &\leq x_{25} \leq x_{24} \\ x_{24} &\leq x_{35} \leq x_{34} \\ x_{34} &\leq x_{45} \end{aligned}$$

See Figure 20 for an illustration. We formalize this observation as a conjecture.

Figure 20. Example of Conjecture 2.4 for four and five slot types.

Conjecture. Let be $x_L = (x_{1L}, x_{2L}, \dots, x_{LL})$ the optimal heights of a warehouse with L slot types. Then, $x_{i-1,L-1} \leq x_{iL} \leq x_{i,L-1}$, for $i=1,2,\dots,L$.

We were unable to prove this conjecture, which is complicated because there are no analytical methods to find the optimal slot heights.

We can modify the exhaustive search (Section 2.1) to evaluate only solutions that conform to this constraint using a recursive algorithm that we call enhanced

search. To solve a problem for L slot types, the enhanced search starts by using an exhaustive search to solve the problem for two slot types. Then, based on the value of x_{12} , the enhanced search solves the problem for three slot types, evaluating only values for x_{13} that are less than or equal to x_{12} , and only values for x_{23} that are greater than or equal to x_{12} , and so forth. Figure 21 shows the running time of the enhanced search versus that of the exhaustive search. The enhanced search is significantly faster, at least 10 times for seven slot types on average. As a result, the conjecture allows us to solve problems with eight and nine slot heights, which cannot be solved by exhaustive search.

Figure 21. Comparison of the running time of the exhaustive search versus the enhanced search.

We used the enhanced search to solve each of the 310 problems described in Section 2.3 (all problems are up to 7 slot types), and it found an optimal solution for every problem. Furthermore, we compared the solutions of the enhanced search versus NOMAD (Section 2.2) for 100 random problems with eight slot types. In every problem, the solution found by the enhanced search is better. Figure 22 shows the deviation of the solutions found by the heuristic search over the solutions found by the enhanced search. We did not test the quality of solutions of the enhanced search for 9 slot types given that it takes around 51 hours to solve a problem.

Figure 22. Improvement of the enhanced search over NOMAD

C Potential benefits of using multiple slot heights

Here, we used problem instances randomly generated from a database of industry data (as in Section 2.3) to compare the space requirements of a slot profile with one slot height to that of slot profiles with up to six slot heights. We chose six slot types because in preliminary studies, we noticed that for more slot types the expected benefit flattens in most cases. In the authors experience, the size of our database is sufficient to represent a large portion of warehouse environments – with around 10.000 skus; however, there are warehouses with a significantly higher number of skus. In those cases, our methodology can still be used; although, our predictions on the expected benefits of using multiple slot heights will not apply directly.

We generated problem instances based on scenarios. A problem instance is defined as $\langle h_k, I_{kt}, \alpha \rangle$ and a scenario as $\langle n, \alpha \rangle$, where h_k are the pallet heights of sku k , I_{kt} is the maximum inventory level of sku k at period t , α is the desired service level, and n is the number of skus in the warehouse. Then, we solved each problem instance using the exhaustive search described on Section 2.1 for one up to six slot types.

To find a sample size, we did a preliminary run considering all combinations of $n = [100, 500, 1000, 2000, 3000]$ and $\alpha = [0.8, 0.9, 0.95]$, and 50 replications for each combination, resulting in 750 problems. Using ANOVA, we found that α did not have a statistically significant effect on the expected benefit. We also found that the largest number of skus we should consider is 1,000, because the problems with 2,000 and 3,000 had features too similar to each other, given that we have inventory data for only 9,321 skus. Finally, we ran an experiment with random factors n and α (using the values explored in the preliminary run) and features of interest r , cv_h , and cv_q . A power analysis for this experiment with a significance level of 0.05, a power of 0.8, and a small effect $f^2=0.02$ recommends a sample size of 550 (Faul et al., 2007). We ran the experiment for 3,600 random problem instances to get a better estimate.

Figure 23 shows a box plot — as well as the mean values — of the expected benefit for all problem instances by number of slot types. According to our experiment, the expected benefit when using two slot types instead of one is between 18.1% and 32.1%, with a mean of 26.7%. For six slot types, this benefit is between 29.6% and 45.5%, with a mean of 40.0%.

1 Why some warehouses benefit more than others

To determine the features of a problem instance that affect the expected benefit, we first model $\langle h_k, I_{kt} \rangle$ by means of $D(h, q)$, which calculates the probability that there are q pallets of height h on an arbitrary day in the warehouse. Figure 24 shows a top view of $D(h, q)$ for an arbitrary problem instance, where a darker color indicates a higher probability. In this example, the expected quantities of the pallet heights follow a Gaussian distribution. We call this distribution $Q(h)$. Intuitively, $Q(h)$ is the expected distribution of pallet heights for a typical day in the warehouse.

In practice, there are many characteristics of a warehouse that could affect the potential benefits of using multiple slot heights. In preliminary experiments, we

Figure 23. Expected benefits of using multiple slot types when compared to one slot type.

found three features had the most significant effect: the range of pallet heights r , which is the difference between the tallest pallet and the shortest one; the coefficient of variation of $Q(h)$, cv_h ; and the average coefficient of variation of pallet quantities for each pallet height, cv_q . The features r and cv_h quantify the dispersion of the pallet heights, and cv_q records the variability of inventory levels through time.

Figure 25 shows histograms for the values of r , cv_h , and cv_q of the 3,600 problem instances in the experiment.

We use a general linear model to analyze how the features of the problem affect the expected space savings. The p -values of the effect of the three features (r , cv_h , and cv_q) were below 0.00004 implying that all features are statistically significant. The pallet heights range r has a positive correlation with the expected benefit (Figure 26(a)), i.e., the larger the range, the higher the benefit of using multiple slot types. Figure 26(b) shows that the higher variability of the pallet heights (cv_h), the higher the expected benefit. To summarize, the more dispersed the distribution of pallet heights, the more benefits the use of multiple slot heights

Figure 24. Example of $D(h, q)$ — Top view

will bring. Finally, Figure 26(c) shows that the variability of the inventory levels has a negative correlation with expected benefits, because the more variable the inventory levels of skus, the larger the number of slots required to provide a service level, which offsets some of the space savings.

We should point out that Figure 26 should not be used to estimate the potential benefit for a specific problem instance, which would depend on the details of the problem at hand.

D Conclusions

Warehouse managers consider lack of space to be among their top challenges (HK Systems, Inc., 2016). By using racks with multiple slot heights, warehouses can store more product within the same available space or can be made smaller while still meeting operational requirements.

We evaluated the potential space savings of using storage racks with multiple slot heights using nominal problem instances generated with data from several distribution centers. We generated 3,600 problem instances with pallet heights

Figure 25. Values of the features for the 3,600 problems.

ranging from 52.5 inches to 67.5 inches, a coefficient of variation of the expected pallet heights on an average day ranging from 0.28 to 0.43, and an average coefficient of variation of historic inventory levels ranging from 0.15 to 0.4. Our results suggest that the expected benefit of using six slot types instead of one is between 29% and 45%, with a mean of 40%.

If a distribution center is already operating using only one slot type, adding a second slot has the potential to increase space utilization by approximately 26%. In this case, the company would have to balance the labor costs of taking pallets out of racks, re-configuring rack beams, and putting the pallets back again against the space savings.

Our experiments indicate significant potential benefits of using multiple slot

(a) Expected benefit versus range of pallet heights. (b) Expected benefit versus average coefficient of variation of pallet heights (cv_h).

(c) Expected benefit versus average coefficient of variation of inventory levels (cv_q).

Figure 26. Influence of different parameters on expected benefit.

heights; however, there are situations in which they may not be ideal. If the range of pallet heights is small, then using multiple slot heights is unlikely to bring much benefit. A warehouse with very few skus is not a good candidate for our models because the Gaussian approximation of aggregate inventory levels could be far from what happens in practice. Finally, distribution environments with unpredictable mixes of skus are also not good candidates, including warehouses with highly variable inventory levels, or many skus with strongly non-stationary or seasonal demands.

Despite these limitations, multiple slot profiles offer significant benefits to warehouses that meet, to a reasonable extent, the assumptions of the model.

E Appendix

1 Normality of aggregated inventory levels

The inventory level of a single sku with stationary and deterministic, demand and deterministic lead time is a periodic function that passes through every value of the range exactly once in each period (Figure 27(a)). This results in a uniform distribution for the inventory level.

Figure 27. Distribution of inventory levels.

The aggregated inventory level of a group of skus is the sum of their inventory levels. Assuming statistical independence between the inventory levels, the distribution of the aggregated inventory level is the convolution of the distributions of the inventory levels of the skus in the group. By the central limit theorem (Feller, 1945), for a large number of skus the aggregated inventory level will approach a normal distribution. Nevertheless, we need further analysis to determine for how many skus it is reasonable to assume normality.

Olds (1952) developed an analytical expression for the distribution of the sum of n non-identically distributed uniform variables. It is not the same density as the normal distribution, but from $n = 4$ on, the distribution has almost the same shape. Potuschak and Müller (2009) study the quality of fit of the normal approximation for this distribution and found it to be a reasonable approximation for n equal to 6, 9, and 12. Moreover, Murakami (2014) use numerical methods for n up to 12 obtaining a good fit.

The number of skus of the distribution centers from which we obtained data were on the order of thousands. This results in the aggregation of the inventory levels of hundreds of skus, far beyond the threshold of $n = 12$ found in theory. Consequently, we contend that the normal distribution is a suitable approximation.

Figure 27(b) shows an example of the aggregated inventory level over time of a random group of 30 skus of a data set industry data, where the inventory levels are well approximated by a normal distribution. We conducted the Anderson-Darling test for normality (Figure 28), where the p -value was 0.013, indicating a good fit.

Figure 28. Normal probability plot

2 Slot blocking

Formally, blocking of a pallet q happens when by the time that pallet q arrives:

1. all slots where q fits are full;
2. there exists a pallet p lower than pallet q that is stored in a slot where q fits;
3. there is at least one slot empty where p fits.

In this way, for slot blocking to happen, it is required that the above three events happen simultaneously, which is highly improbable. To illustrate this, we simulated 3,000 random scenarios from the data set used in Section 2.1 with up to 8 slot types for different service levels. We tracked each event of slot blocking and estimated the probability of slot blocking happening on a day depending on the number of slot types and the service level (Table 2). We did not observe any slot blocking in the case of two slot types; therefore, we did not include it in Table 2. We can see that in any case, the probability that slot blocking happens on a day is at most in the order of 10^{-4} , which we consider negligible.

service level	slot types					
	3	4	5	6	7	8
0.60	$7.33 \cdot 10^{-5}$	$1.00 \cdot 10^{-4}$	$1.53 \cdot 10^{-4}$	$1.53 \cdot 10^{-4}$	$1.47 \cdot 10^{-4}$	$1.80 \cdot 10^{-4}$
0.70	$6.00 \cdot 10^{-5}$	$6.67 \cdot 10^{-5}$	$1.40 \cdot 10^{-4}$	$1.73 \cdot 10^{-4}$	$1.87 \cdot 10^{-4}$	$2.07 \cdot 10^{-4}$
0.80	$6.00 \cdot 10^{-5}$	$6.67 \cdot 10^{-5}$	$7.33 \cdot 10^{-5}$	$8.00 \cdot 10^{-5}$	$1.13 \cdot 10^{-4}$	$1.27 \cdot 10^{-4}$
0.85	$4.67 \cdot 10^{-5}$	$4.67 \cdot 10^{-5}$	$5.33 \cdot 10^{-5}$	$8.00 \cdot 10^{-5}$	$1.00 \cdot 10^{-4}$	$1.27 \cdot 10^{-4}$
0.90	$2.00 \cdot 10^{-5}$	$5.33 \cdot 10^{-5}$	$6.00 \cdot 10^{-5}$	$9.33 \cdot 10^{-5}$	$1.00 \cdot 10^{-4}$	$1.00 \cdot 10^{-4}$
0.95	$2.00 \cdot 10^{-5}$	$1.33 \cdot 10^{-5}$	$6.67 \cdot 10^{-6}$	$4.33 \cdot 10^{-5}$	$4.00 \cdot 10^{-5}$	$5.33 \cdot 10^{-5}$

TABLE 2

Probability of slot blocking

F Upper bound of the joint probability

Let $F(x)$ be a multivariate joint probability distribution with L variables, density $f(x)$, and $F_i(x_i)$ the marginal probabilities. Here, we prove that $F_i(x_i)$ is an upper bound of $F(x)$.

The marginal probability is

$$F_i(x_i) = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{x_i} \dots \int_{-\infty}^{\infty} f(x) dx_1 \dots dx_i \dots dx_L.$$

Now, let $\Omega = \Omega_1, \Omega_2, \dots, \Omega_n$ be any partition of $\mathbb{R}^{L-1}$ such that $\Omega_1 = [-\infty, x_1] \times [-\infty, x_{i-1}] \times [-\infty, x_{i+1}] \dots \times [-\infty, x_L]$ and let

$$\omega_j(x_i) = \int_{-\infty}^{x_i} \left(\int \dots \int \int_{\Omega_i} f(x) dx_1 dx_{i-1} dx_{i+1} dx_L \right) dx_i,$$

then, we say that $F(x) = \omega_1(x_i)$ and we can rewrite $F_i(x_i)$ as

$$F_i(x_i) = F(x) + \sum_{j=2}^n \omega_j(x_i).$$

Given that $\omega_j(x_i)$ is strictly positive, we obtain that $F_i(x_i) > F(x)$.

CHAPTER III

LAYOUT OF WAREHOUSES WITH MULTIPLE SLOT HEIGHTS

We describe a method to generate layouts for unit-load warehouses that use multiple slot heights as a way to maximize warehouse space utilization. The problem has two parts: slots must be arranged into rack-bays (Section A), and rack-bays must be arranged into a layout (Section B). Furthermore, we develop a discrete event simulation model and use industry data to investigate the impact of using multiple slot heights for the warehouse footprint, expected travel time, racking cost, storage service level, and space utilization (Section).

A Design of Rack-bays

In selective pallet racks a rack-bay is a vertical column contained within two uprights. It is divided by beams to create slots, which, in turn, can have multiple pallet positions. In this section, we propose two methods to design rack-bays that provide a desired service level α and that minimize the required number of rack-bays—a surrogate for the warehouse footprint.

Following the notation of Cardona and Gue (2019a), let a slot profile be the tuple $\langle x, N \rangle$, where $x = (x_1, x_2, \dots, x_L)$ are the slot heights and $N = (N_1, N_2, \dots, N_L)$ the corresponding quantities in a design with L slot types. The objective is to minimize the sum of all slot heights while providing a desired service level α . The model is,

$$\begin{aligned}
& \text{Minimize} && \sum_{i=1}^L x_i N_i \\
& \text{s.t.} && G(x, N) \geq \alpha \\
& && x_i \in \mathbb{R}^+, N_i \in \mathbb{Z}^+ \quad \forall i,
\end{aligned} \tag{8}$$

where $G(x, N)$ is the probability that all pallets arriving during an operational shift can be put away. $G(x, N)$ is calculated using the multivariate normal distribution based on the pallet heights and the historic (forecasted) inventory levels of the skus (Cardona and Gue, 2019a). $G(x, N)$ makes the problem challenging because it has no closed form; it can only be approximated by simulation for $L > 3$ (Genz and Bretz, 2009).

Here, we extend Model 8 to generate a slot profile and its arrangement into rack-bays as a joint problem. We propose two ways to pack slots into rack-bays: In the first, we require rack-bays to have identical *tiers* (Figure 29(a)). In the second we assign slots to rack-bays as a form of the cutting stock problem, in which a collection of objects (slots) must be cut from a set of larger objects (rack-bays). The result of the cutting stock approach is a wide variety of rack-bay configurations (Figure 29(b)).

Figure 29. Two methods to arrange slots into rack-bays.

Finally, we adopt an online storage policy in which pallets are picked and put away one at time without information about the next assignment. This is in

contrast to the work of Quintanilla et al. (2015), where the information of all incoming pallets is available at the beginning of the operational shift. We called our policy *best fit, closest open location* with the objective to maximize space utilization. No relocation of pallets is allowed; once a pallet is stowed, it remains in the allocated slot until it is picked.

1 Tiers

Because in an arrangement by tiers all rack-bays are the same, a design is completely specified by the configuration of one rack-bay and the number of copies. The decision variables are the slot heights $x = (x_1, x_2, \dots, x_L)$, the slot quantities in one rack-bay $n = (n_1, n_2, \dots, n_L)$, and the number of rack-bays $\mathcal{Q}$ in the design. The objective is to minimize the number of rack-bays $\mathcal{Q}$ while guaranteeing a desired service level α . The model has five constraints: i) the sum of the heights of all slots in a rack-bay must be equal to the height of the rack-bay $\mathcal{H}$; ii) the slot profile must provide at least a desired service level α ; iii) defines the slot quantities N_i ; iv) heights x must be in ascending order; and v) the largest slot must be able to store the largest pallet H . The model is,

$$\begin{aligned}
& \text{Minimize} && \mathcal{Q} \\
& \text{s.t.} && \sum_i x_i n_i = \mathcal{H} \\
& && G(x, N) \geq \alpha \\
& && N_i = n_i \mathcal{Q} && \forall i \\
& && x_i \geq x_{i-1} && 2 \leq i \leq L \\
& && x_L = H \\
& && x_i, n_i, \mathcal{Q} \in \mathbb{Z}^+ && \forall i.
\end{aligned} \tag{9}$$

We restrict the domain of x_i to $\mathbb{Z}^+$ because in practice, slot heights in pallet racks can only be adjusted in discrete increments—the space between the tear drop

openings (Figure 17). We assume that the slot heights x , the maximum slot height H , and the rack-bays height $\mathcal{H}$ are scaled in terms of these increments.

Model 9 is challenging to solve for the following reasons: i) there are non-linear constraints; ii) the second constraint is a joint chance constraint; iii) it is non-convex with respect with x ; and iv) there are integer variables. Methods to deal with joint chance constraint programs are based on nonlinear relaxations of the problem (Ahmed and Shapiro, 2008). However, because Model 9 is also non-convex and has integer variables, this approach would require solving an intractable non-convex integer problem. Therefore, we propose an enumeration procedure called TiersProfile (Figure 30). The algorithm starts by generating all feasible values for slot heights and quantities (Algorithm 2), then it solves a univariate convex model to find the minimum number of rack-bays $\mathcal{Q}$ for each feasible solution (Model 14). Finally, it returns the solution with minimum $\mathcal{Q}$.

Algorithm 2 generates all feasible values for (x, n) based on the first constraint of Model 9, which in explicit notation is,

$$x_1 n_1 + x_2 n_2 + \cdots + x_L n_L = \mathcal{H}. \quad (10)$$

If x were a variable and n a parameter, Equation 10 would be a linear diophantine equation (Mordell, 1969), which can be solved using the extended Euclidean algorithm (Ramachandran, 2006). Algorithm 2 is based on this algorithm but considers that x and n are both variables and that $x, n \in \mathbb{Z}^+$. The inputs of the algorithm are the number of slot types L , the minimum pallet height h_0 , the maximum pallet height H , and the rack-bay height $\mathcal{H}$. The algorithm starts on slot type L . It makes $x_L = H$, then it iterates over 1 to ub_L to determine n_L , where ub_L (Equation 11) is the number of slots of height H that fit in a rack-bay if there were only one slot of all other slot types in the rack-bay.

$$ub_L = \left\lceil \frac{\mathcal{H} - (L-1)h_0 - (L-2)}{H} \right\rceil \quad (11)$$

Figure 30. TiersProfile — an enumerate procedure to generate rack-bays design by tiers.

Next, for each (x_L, n_L) , Algorithm 2 calls a recursive algorithm (Algorithm 3) that considers the remaining slot types, one at a time. The inputs of Algorithm 3 are the slot type l , the minimum pallet height h_0 , the height of the previous slot type x_{l+1} , and the remaining height h in the rack-bay after the height and quantity of slot types greater than l have been set. Algorithm 3 starts by calculating upper bounds for x_l and n_l ; then, it iterates over all possible combinations of (x_l, n_l) defined by these upper bounds calling itself recursively for slot type $l - 1$. The upper bounds of x_l and n_l are given by Equations 12 and 13 respectively. The upper bound ub_x is the minimum value between the height of slot type $l + 1$ and the maximum number of slots of type l that can fit in the remaining rack-bay's height h . The upper-bound ub_n is the maximum number of slots of type l that would fit in the rack-bay if $x_l = h_0$ and $n_l = 1$ for the remaining slot types.

Algorithm 2 GenerateSolutions($L, h_0, H, \mathcal{H}$)

```

1:  $x_L \leftarrow H$ 
2:  $ub_L \leftarrow \text{CalculateUpperBoundL}(\mathcal{H}, L, h_0)$  ▷ Eq. 11
3: for  $n_L$  from 1 to  $ub_L$  do
4:    $sols \leftarrow \text{SolsRecursive}(L-1, h_0, H, \mathcal{H} - H n_L)$  ▷ Algorithm 3
5:   for  $sol$  in  $sols$  do
6:     yield  $sol + (x_L, n_L)$ 
7:   end for
8: end for
9: return  $sols$ 

```

$$ub_x = \min \left(x_{l+1}, \frac{h - (l-1)h_0 - (l-2)}{n_l} \right) \quad (12)$$

$$ub_n = \frac{h - (l-1)h_0 - (l-2)}{h_0 + (l-1)} \quad (13)$$

For the smallest slot type ($l = 1$), Algorithm 3 explores all integer solutions of $x_1 n_1 = h$ such that $h_0 < x_1 < x_2$ using the method proposed in Lenstra (1987) to generate divisors.

Once Algorithm 2 has generated the set of all feasible values for (x, n) , TiersProfile (Figure 30) uses Model 14 to calculate the minimum number of rack-bays that provide service level α for each feasible value. Given the problem has only one variable $\mathcal{Q}$ and $G(x, N)$ is convex with respect to $\mathcal{Q}$ (i.e., the more rack-bays, the higher the service level), we use a standard bisection search to solve the model (Waeber et al., 2013).

$$\begin{aligned}
& \min \quad \mathcal{Q} \\
& \text{s.t.} \quad G(x, N) \geq \alpha \\
& \quad \quad N_i = n_i \mathcal{Q} \quad \forall i \\
& \quad \quad \mathcal{Q} \in \mathbb{Z}^+
\end{aligned} \quad (14)$$

2 Cutting stock

Here, we present two methods to design rack-bays by solving a cutting stock problem: an optimal algorithm (CSP-Enumerate) that works for up to seven slot

Algorithm 3 SolsRecursive(l, h_0, x_{l+1}, h)

```
1: if  $l = 1$  then
2:    $sols \leftarrow \text{DivisorsBounded}(h, h_0, x_2)$   $\triangleright$  find divisors of  $h$  using Lenstra (1987)
3:   for  $x_1$  in  $sols$  do
4:      $n_1 \leftarrow h/x_1$ 
5:     yield  $(x_1, n_1)$ 
6:   end for
7: else
8:    $ub_n \leftarrow \text{CalculateUpperBoundn}(h, l, h_0)$   $\triangleright$  Eq.13
9:   for  $n_l$  from 1 to  $ub_n$  do
10:     $ub_x \leftarrow \text{CalculateUpperBoundx}(h, l, h_0)$   $\triangleright$  Eq.13
11:    for  $x_l$  from  $h_0$  to  $ub_x$  do
12:       $sols \leftarrow \text{SolsRecursive}(l-1, h_0, x_l, h - x_l n_l)$ 
13:      for  $sol$  in  $sols$  do
14:        yield  $sol + (x_l, n_l)$ 
15:      end for
16:    end for
17:  end for
18: end if
19: return  $sols$ 
```

types and a heuristic (CSP-Heuristic) for designs with more slot types.

The objective again is to minimize the number of rack-bays in the design while providing a desired service level α . However, to “pack” the slots into rack-bays we follow the framework of the Cutting Stock Problem (CSP), which is the problem of cutting large pieces (metal rolls, wood logs, etc) of equal size into smaller pieces to meet a demand (Kantorovich, 1960). In our problem, the large pieces are the rack-bays and the smaller pieces are the slots. If a slot profile $\langle x, N \rangle$ were given, the following model would generate a design as a CSP:

$$\begin{aligned} \min \quad & \sum_j q_j \\ \text{s.t.} \quad & \sum_j n_{ij} q_j \geq N_i \quad \forall i \\ & q_j \in \mathbb{Z}^+ \quad \forall j, \end{aligned} \tag{15}$$

where n_{ij} represents the “cutting patterns” (the number of slots of height x_i in

cutting pattern j) and q_j is the number of rack-bays with cutting pattern j in the design. The cutting patterns are generated beforehand by an auxiliary method (Suliman, 2001) based on the height of the rack-bay $\mathcal{H}$ and the slot heights x . Model 16 extends Model 15 to determine the slot profile and its arrangement into rack-bays as a joint problem. The first, fifth, and sixth constraints are the same as in Model 9. The second constraint defines the slot quantities N_i in terms of the cutting patterns n_{ij} and patterns quantities q_j . The third and fourth constraints guarantee that only feasible patterns are used depending on the slot heights and the height of the rack bay, where M is an arbitrary large number. Additionally, because x is not known in advance, the cutting patterns n_{ij} will include the patterns for any possible value of x .

$$\begin{aligned}
\min \quad & \sum_j q_j \\
\text{s.t.} \quad & G(x, N) \geq \alpha \\
& N_i = \sum_j n_{ij} q_j & \forall i \\
& \sum_i x_i n_{ij} \leq \mathcal{H} y_j & \forall j \\
& q_j \leq M y_j & \forall j \\
& x_i \geq x_{i-1} & 2 \leq i \leq L \\
& x_L = H \\
& x_i, q_j \in \mathbb{Z}^+ & \forall i, j \\
& y_j \in \{0, 1\} & \forall j
\end{aligned} \tag{16}$$

Unfortunately, the number of cutting patterns (columns of n_{ij}) grows exponentially with the number of slot types L . Let K be the maximum number of slots in a rack-bay, then there will be K^L cutting patterns. This makes Model 16 computationally intractable. Therefore, we propose CSP-Enumerate as an enumeration procedure to generate cutting patterns in a more efficient way.

CSP-Enumerate

The algorithm (Figure 31) starts by enumerating all possible values for the slot heights $x = (x_1, x_2, \dots, x_L)$. We call this set X and define it as all combinations of L slot heights from the range of integers between the minimum h_0 and maximum pallet height H in lexicographic order (Payne and Ives, 1979). In this way, X has $\binom{H-h_0}{L}$ elements.

Figure 31. CSP-Enumerate, an optimal procedure to a generate rack-bays design by cutting stock.

For each $x \in X$, we generate the set of cutting patterns η depending on the height of the rack-bays $\mathcal{H}$. For this purpose, we find all solutions of Equation 10 in terms of $n = (n_1, n_2, \dots, n_L)$ for nonnegative integers. Each solution is a cutting

pattern of $\mathcal{H}$, and we denote the set of all solutions by $\eta = (n_{ij})$. In combinatorics, this problem is known as the money changing problem (Beck and Robins, 2007). To solve it, we use the algorithm presented in Bocker and Liptak (2007). Then, for each pair of slot heights and cutting patterns (x, η) , the algorithm determines pattern quantities $Q = (q_j)$ — number of rack-bays with each cutting pattern j in the design — that minimizes the number of rack-bays while providing a desired service level α . For that purpose, we resort to Model 17, which is a stochastic cutting stock problem.

$$\begin{aligned}
& \min \quad \sum_j q_j \\
& \text{s.t.} \quad G(x, N) \geq \alpha \\
& \quad \quad N_i = \sum_j n_{ij} q_j \quad \forall i \\
& \quad \quad N_i, q_j \in \mathbb{Z}^+ \quad \forall i, j
\end{aligned} \tag{17}$$

There is little literature on the Stochastic CSP (Alem et al., 2010); approaches assume either that the random variables are independent (Scull, 1981), use discrete distributions (Beraldi et al., 2009; Sarper and Jaksic, 2018), or use heuristics (Hähnel et al., 2018). To the best of our knowledge there is no method to solve a Stochastic CSP with chance constraints as in Model 17. Standard optimization software for convex integer programming (Bonami et al., 2008) could not overcome the numerical instability introduced by the change constraint. Therefore, we propose a customized procedure called OuterApproximation to solve it (Figure 32).

Let P be the optimization problem defined by Model 17. Our approach to solve P is based on outer approximation procedures Bonami et al. (2012). OuterApproximation starts by finding the optimal solution $N^0 = (q_0, N_0)$ of $\bar{P}$, the continuous relaxation of P . For that purpose, it uses the method-of-moving-asymptotes (MMA) algorithm (Svanberg, 2002).

Then, the algorithm starts an iterative procedure that builds an every time tighter polyhedral approximation of $G(x, N) = \alpha$. The polyhedral approximation

Figure 32. OuterApproximation algorithm to solve P (Model 17).

consists of a set of “cuts,” where a cut is defined by a point N^k in the vicinity of $G(x, N) = \alpha$ and the gradient $\nabla G(x, N)$ at N^k . In this way, the $G(x, N) \geq \alpha$ is replaced by a set of linear constraints making the problem tractable.

In the first iteration, the algorithm initializes the set of cuts $\mathcal{C} = \{N^0\}$ and solves the polyhedral approximation version of P (Model 18). Figure 33(a) shows an example of a polyhedral approximation of $G(x, N) \geq \alpha$.

$$\begin{aligned}
\min \quad & \sum_j q_j \\
\text{s.t.} \quad & \sum_i \left(\nabla G_i(x, N^k) (N_i - N_i^k) \right) + G(x, N^k) \geq \alpha \quad \forall k \\
& N_i = \sum_j n_{ij} q_j \quad \forall i \\
& N_i, q_j \in \mathbb{Z}^+ \quad \forall i, j.
\end{aligned} \tag{18}$$

Let $(\bar{q}^k, \bar{N}^k)$ be the solution of Model 18. If $(\bar{q}^k, \bar{N}^k)$ is feasible for P , then OuterApproximation terminates. If $(\bar{q}^k, \bar{N}^k)$ is infeasible, the algorithm uses Model 19 to find the closest point to $(\bar{q}^k, \bar{N}^k)$ in the feasible region of P and adds this point to the set of cuts $\mathcal{C}$ for the next iteration. Figure 33(b) shows an example of the feasibility problem, where $\bar{N}^k$ is outside the feasible region of P and $\bar{N}^F$ is the solution found by Model 19 based on $\bar{N}^k$. OuterApproximation iterates until it finds a solution that is optimal to the polyhedral approximation problem and feasible for P .

$$\begin{aligned}
\min \quad & \sum_j \delta_j \\
\text{s.t.} \quad & G(x, N) \geq \alpha \\
& N_i = \sum_j n_{ij} (q_j + \delta_j) \quad \forall i \\
& \delta_j \in \mathbb{R}^+ \quad \forall j
\end{aligned} \tag{19}$$

Figure 33. Iterative process of OuterApproximation.

CSP-Heuristic

In our computational experiments, the set of all feasible solutions for the slot heights x was too large for more than seven slot types, making the problem intractable. Hence, we propose a heuristic (CSP-Heuristic) based on a Mesh Adaptive Direct Search (MADS) (Audet and Dennis, 2006). The main difference between CSP-Enumerate and CSP-Heuristic is that CSP-Enumerate explores all possible values for the slot heights x , while CSP-Heuristic uses MADS (Audet and Dennis, 2006) to find a near optimal value for x .

Let $f(x)$ be the optimal value of Model 17. Then, CSP-Heuristic addresses the following non-convex optimization problem:

$$\begin{aligned}
 \min \quad & f(x) \\
 \text{s.t.} \quad & x_{i+1} \geq x_i \quad 1 \leq i \leq L-1 \\
 & x \in \mathbb{R}^+ \quad \forall i.
 \end{aligned}$$

MADS is a derivative-free method for non-convex, non-smooth optimization.

It is an iterative stochastic search that can be parameterized and combined with other heuristics to exploit characteristics of the problem. We called CSP-Heuristic to the specific configuration of MADS to solve Model 17.

Figure 34 shows an example of the iterative process of CSP-Heuristic for a randomly generated problem instance with three slot types. Because the tallest slot x_3 is fixed to be H , only x_1 and x_2 are variables. The algorithm starts at $(45, 55)$ and iterates until it converges to $(36, 53)$, which in this case happens to be optimal.

Figure 34. Example of the iterative procedure of CSP-Heuristic to solve a problem with three slot types.

CSP-Heuristic starts at $\bar{x}_0$, where $\bar{x}_0$ are the slot heights from a slot profile generated by the method proposed by Cardona and Gue (2019a). At each iteration, it defines the incumbent solution $\hat{x}$ as the best feasible solution available. Then, it generates a subset S of points from the frontier of a hyper-cube centered at $\hat{x}$ with side of length Δ . This step is called SEARCH, and there are multiple heuristics to choose S ; here we use a Latin Hypercube Sampling (Liefvendahl and Stocki, 2006). The aim of SEARCH is to find a better incumbent point within S . Next, it executes a procedure called POLLING, where CSP-Heuristic varies Δ and samples the hyper-cube centered at $\hat{x}$ within the whole region, not just the frontier. If the algorithm stalls over a local optima, it will use a variable neighbor search (Audet

et al., 2008) to explore a distant neighborhood. CSP-Heuristic iterates until no improvement is reached after 20 iterations.

3 Tiers versus Cutting Stock

We compare our methods to generate rack-bays (by tiers and by cutting stock) in terms of the number of rack-bays (a surrogate for space requirement) and the running time of algorithms. For that purpose, we use computer simulations on randomly generated problem instances.

We implemented the methods in Python 3.6 using Cython (Behnel et al., 2011) and used the implementation of the statistical module of SciPy (Jones et al., 2001) in Python to calculate the Multinormal Gaussian Distribution $G(x, N)$. To estimate $\nabla G(x, N)$, we implemented the method proposed by Prékopa (1995). We used Cython (Behnel et al., 2011) for the enumerative procedure and Gurobi Optimization, Inc. (2017) for the MIP solver. To solve non-linear problems, we used the NLOpt framework (Johnson, 2017). Finally, for the MADS, we used the NOMAD implementation (Audet et al., 2017). We used a desktop computer running Ubuntu 16.04 with an Intel(R) Core(TM) i7 3.5 GHz with 8 cores and 16GB RAM.

We use a set of 1200 randomly generated problem instances, where a problem instance is the tuple of pallet heights, inventory levels, height of rack-bays, desired service level, and number of slot types $\langle h_k, I_k(t), \mathcal{H}, \alpha, L \rangle$. We use the Python package TimeSynth (Maat and Malali, 2017) to generate random time series for the inventory levels and standard random generators for the pallet heights from multi-modal Gaussian distributions. The service level was fixed to 85% and the height of the rack-bays to 400".

For each problem instance, we created both designs, one by tiers and one by cutting stock, and compared the number of rack-bays required. Because they consider more possible cutting patterns for the rack-bays, cutting stock designs have fewer or the same number of rack-bays than designs with tiers. Figure 35 shows the

percentage difference in number of rack-bays of designs by cutting stock when compared to designs by tiers. On average, cutting stock designs have between 5% and 20% fewer rack-bays than designs by tiers.

Figure 35. Percentage difference in the number of rack bays of designs by tiers and by cutting stock.

Figure 36 shows the running times. TiersProfile can solve up to 5 slot types in a few minutes, but for 9 slot types it can take around 20 hours. For more slot types, the running time actually decreases because there are fewer patterns in which the rack-bay can be partitioned into slots. CSP-Enumerate can solve problems with up to five slot types within minutes, but it could take more than 19 hours for seven slot types, beyond which the algorithm is computationally infeasible. To solve problems for more slot types with an arrangement by cutting stock, we use CSP-Heuristic. Rack-bay designs by cutting stock are more challenging to solve because the number of feasible cutting patterns for the rack-bays is significantly higher than rack-bay designs by tiers.

Finally, we studied the quality of solutions provided by CSP-Heuristic, because optimality is not guaranteed. For up to seven slot types, we compared the solutions of both algorithms for each problem instance. Figure 37 shows that in 90% of problem instances, the solution provided by CSP-Heuristic is within 2 percent of the optimal value.

Figure 36. Running times of algorithms to generate rack-bays.

Figure 37. Quality of the solutions provided by CSP-Heuristic.

B Warehouse Design

In this section, we generate a warehouse design given a set of rack-bays. Let $\mathcal{R} = \langle x, \eta, Q \rangle$ be a set of rack-bays with slot heights x , cutting patterns $\eta = (n_{ij})$, and pattern quantities $Q = (q_j)$. Also let $\mathcal{W} = \{w_{rc}\}$ be a warehouse layout defined as a set of layout locations w_{rc} , where r indicates the row of rack-bays and c the column of rack-bays within the layout (Figure 38). A warehouse design is then the tuple $\langle \mathcal{R}, \mathcal{W}, Y \rangle$, where Y is the assignment of cutting patterns to layout locations. Y is only relevant for sets of rack-bays determined by cutting stock, because all rack-bays are equal in designs with tiers.

Figure 38. Notation of warehouse layout model (Top view).

Assume the picking and put-away is done one pallet at a time using forklifts. There is one Pickup and Deposit (P&D) point centrally located along the bottom side of the warehouse. The slots are e_w wide and e_l in length, and the aisles are a_p wide. The linear and vertical speeds of the forklifts are v_l and v_z respectively. We assume that forklifts are identical and able to reach the highest slot while carrying the heaviest pallet. We assume that pallets are stored in single-deep selective racks, that all rack-bays have the same height, and that the slots in a rack-bay are arranged with smaller slots at the bottom. All other things being equal, placing smaller slots near the bottom reduces the expected vertical travel of a forklift operation.

The annual operational cost is the sum of the leasing cost and the material handling cost. The leasing cost is the product of warehouse area and the annual cost per square foot c_1 . The material handling cost is the product of the expected travel time and c_2 , which is the cost of one minute of travel multiplied by the expected number of trips per year.

The storage policy we assume for the operation is *best fit, closest open location* (Cardona and Gue, 2019a), which assigns an arriving pallet to the smallest slot that can contain it, with ties broken by distance to the P&D point. The policy picks pallets from the largest slot containing the required sku, again breaking ties by

distance to the P&D point.

To generate a warehouse design given a set of rack-bays, we first generate a layout based on the desired number of rack-bays (Section 1). Next, if rack-bays were determined by cutting stock, we assign the cutting patterns to the layout locations (Section 2); for a design by tiers this step is not necessary because all rack-bays are equal.

1 Layout

The model to generate a layout has two decision variables, the number of double rack-bay columns C and the number of rack-bay rows R (in Figure 38, $C = 3$ and $R = 7$). The objective is to minimize the operational cost, and the layout must have at least a desired number of layout locations $\mathcal{Q}$ given by the set of rack-bays R because $\mathcal{Q} = \sum_j q_j$. The model is,

$$\begin{aligned} \min \quad & Z = c_1 \text{Area} + 2c_2 E[T] \\ \text{s.t.} \quad & 2R(C+1) \geq \mathcal{Q} \\ & R, C \in \mathbb{Z}^+, \end{aligned} \tag{20}$$

where Area (Equation 21) is the warehouse footprint and $E[T]$ is the expected travel time of a one-way trip to a slot.

$$\text{Area} = (a_p + 2e_l)(C+1)(2a_p + Re_w) \tag{21}$$

The expected travel time $E[T]$ depends on the location of each slot in the warehouse and its probability to be visited. We assume pallets are picked from the closest picking point to the slot where they are stored. The blue dots in Figure 38 represent the picking points at the bottom tier. We denote each picking point by OP_{dr} that specifies the aisle d and row r where it is located. The number of aisles in the layout is $C+1$. Equations 22-23 calculate the location (X_{dr}, Y_{dr}) of the picking point OP_{dr} considering the point $(0,0)$ as the left-bottom corner of the layout.

$$X_{dr} = \left(e_l + \frac{a_p}{2}\right)(2d - 1) \quad (22)$$

$$Y_{dr} = \left(a_p + \frac{e_w}{2}\right) + e_w(d - 1) \quad (23)$$

To pick up a pallet from the picking point OP_{dr} , a forklift starts at the P&D point and travels through the aisles with the fork at ground level and speed equal to v_l until it reaches the rack-bay of the desired slot. Then, it lifts the fork up to the tier of the desired slot with a speed v_z . We assume that every slot is equally likely to be picked. In turn, this means that each rack-bay has the same probability to be visited and that we can approximate the length of the vertical travel to visit a slot in a rack-bay by $\mathcal{H}/2$. The expected travel time is,

$$E[T] = \frac{1}{(C+1)R} \sum_{d=1}^{C+1} \sum_{r=1}^R \left[\frac{1}{v_l} \left(\left| X_{dr} - \frac{(a_p + 2e_l)(C+1)}{2} \right| + Y_{dr} \right) + \frac{\mathcal{H}}{2v_z} \right].$$

Model 20 is challenging to solve because it is a non-convex, non-smooth integer program. However, the problem has a small enough feasible space to use a simple enumeration procedure to solve it. First, we replace R by the expression in Equation 24 given there is no benefit to having more rack-bays than $\mathcal{Q}$. Then, we explore all values from 1 to ub_C (Equation 25) and choose the layout with minimum operational cost.

$$R = \left\lceil \frac{\mathcal{Q}}{2(C+1)} \right\rceil \quad (24)$$

$$ub_C = \left\lceil \frac{\mathcal{Q}e_w}{4a_p} - 1 \right\rceil \quad (25)$$

2 Assigning cutting patterns to layout locations

We propose a method (called DOS_m) to arrange the cutting patterns in a layout based on a common storage policy called Duration-of-Stay (Ratliff and Goetschalckx, 1990), with the objective to minimize the expected travel time of the warehouse.

In Duration-of-Stay, pallets with shortest stays are assigned to slots closer to the P&D point. The length of the stay of an incoming pallet depends on the demand of the sku and the number of pallets already in stock. For example, if in one day a pallet of sku k arrives, the average demand for k is 1 pallet per day, and there are 5 pallets in stock already, it is expected that the arriving pallet stays for 6 days in the warehouse. In our problem, instead of assigning pallets to slots, we assign cutting patterns to layout locations.

DOS_m sorts all layout locations in W by increasing travel time (Eqs. 22–23) and the rack-bays in $\mathcal{R}$ by the expected duration of stay of a pallet in the rack-bay. Finally, DOS_m assigns the rack-bays with the shortest expected stay of a pallet to the layout locations with the shortest travel time. To estimate the expected duration of stay of a pallet in a rack-bay, we propose a simulation model (Figure 39).

Figure 39. Simulation model of the storage system.

The simulation model splits the arrival stream of pallets into J queues (one for each cutting pattern) and represents each rack-bay as a server. Let RB_{kj} be the k -th rack-bay with cutting pattern j in a design with J cutting patterns. The rack-bays of the same cutting pattern are grouped and sorted by their index j . The put-away rule is as follows: for an arriving pallet, all empty slots of the type where the pallet fits best are identified. From these set of slots, the pallet is put-away in

the slot with a rack-bay with minimum index j . The picking rule is similar, but instead of the slot where it fits best, it selects the slot in which it fits worst. We simulate the model using historical (forecast) data of arrivals and departures of pallets of each sku and record the average duration of stay of a pallet in each rack-bay RB_{kj} . Figure 40(a) shows results of the simulation for an example based on industry data, and Figure 40(b) shows the assignment of cutting patterns to layout locations based on DOS_m based on the result of the simulation. Most of the rack-bays of the first two most active cutting patterns are allocated close to the P&D, then the cutting patterns are intercalated as their durations of stay become similar. Finally, because W can have more layout locations than $\mathcal{R}$ has rack-bays, we assigned the surplus layout locations to the most active cutting pattern.

Figure 40. Example of a DOS_m assignment in a design with four cutting patterns.

We measure the performance of DOS_m by comparing it to four other methods: by rows, by columns, greedy, and random. In the first three methods, all rack-bays of the same cutting pattern are located in sequence according to the respective rule; then, the most active cutting patterns are assigned to the layout location with shortest travel time. In the random assignment, cutting patterns are assigned to layout locations randomly. Figure 41 shows the assignment following these four methods for the same example of Figure 40.

Figure 41. Additional assignment methods for cutting patterns to layout locations.

To estimate the expected travel time of a warehouse design, we developed a discrete event simulation model of the operation of a distribution center. The inputs of the simulation model are a warehouse design $\langle \mathcal{R}, \mathcal{W}, Y \rangle$, the pallet heights h_k , and the inventory levels $I_k(t)$. From the inventory levels, the simulation model needs the inventory levels for the first day (I_1) as the number of pallets for each sku i stored in the warehouse at the initial state. It also calculates the picking and put-away activity from $I_k(t)$, where every subtraction from the inventory is counted as a pick and every addition as a put-away.

To set up the initial state, the model allocates the pallets in I_1 to the slots using the best fit, closest open location policy. To start the simulation, the put-aways and the picks are merged into a list. Then, the trips are sorted in ascending order by date-time and processed one-by-one using the best fit, closest open location policy.

To compare the five methods, we use the data set of 1200 problem instances as in Section 3. For each problem instance, we used CSP-Enumerate to generate a set of rack-bays $\mathcal{R}$; then, for each $\mathcal{R}$, we generated five warehouse designs, one for each method to assign cutting patterns to layout locations. Next, we used the discrete event simulation model to calculate the expected travel time of each warehouse design. Figure 42 shows the results of the computational experiment, where the expected travel time $E[T]$ for each warehouse design is normalized based on the method with minimum $E[T]$ for the corresponding problem instance. DOS_m and Greedy had the best performance with no statistically significant difference between them. The deviation over the optimum of using an arrangement by rows or columns is no more than 2 percent. Finally, the random method had the worst performance.

Figure 42. Performance comparison of the five methods to assign cutting patterns to layout locations.

Another way to test DOS_m is to plot the picking density resulted from the simulation. Figure 43 shows the picking density for the layout of Figure 40(b). As intended, we can see that the most visited rack-bays are assigned to the layout locations with lowest travel time.

It is worth mentioning that we performed further simulations with other storage policies that favor travel time over space utilization such as *closest open*

Figure 43. Example of the picking density for a layout generated by DOS_m .

location and observed that the service level can be significantly affected. Therefore, we strongly recommend to *best fit, closest open location* as the storage policy for warehouses using multiple slot heights. We did not include all details of these simulations for conciseness.

C Case Study

To illustrate the benefits of using multiple slot heights in unit-load warehouses, we used a nominal, industry-based case study. It includes data from two independent distribution centers. From one DC, we gathered the sku dimensional data for 7,102 skus. From the other DC, we gathered the history of inventory levels for 409 days of all active skus. Unfortunately, we did not have a complete data set (inventory levels and sku dimensions) from either DC.

1 Parameters

The data set contained the dimensional data of 7,102 skus with pallet heights between 6 and 71 inches (h_k) and the inventory level of these skus I_{kt} for 409 days. Figure 44 shows the distribution of pallet heights at an arbitrary point in time (this includes the height of the wooden pallet itself).

Figure 44. Example of a pallet heights distribution.

Figure 45(a) shows the aggregated inventory level for all skus for the 409 days; the average number of pallets in inventory was 33,710.4 and the maximum was 35,807. From I_{kt} , we also calculated the number of forklift trips (Figure 45(b)) per week considering every subtraction from the inventory as a pick and every addition as a put-away. Unfortunately, we do not have enough information about the demand to explain the spikes. The average number of trips per week was 15,410.7, which resulted in an estimated 770,535 trips per year.

Figure 45. Data used in the case study.

We assumed an operation of a typical unit-load warehouse in the greater area of Chicago, Illinois, USA. The DC receives only full pallets, and the forklifts pick

only full pallets. The pallets (40" by 48") are stored in single deep selective racks with beams 96" wide (capable of holding two pallets side-by-side). The rack uprights are 400" high, 42" deep, and have 4" by 4" posts. When these racks are back to back to form a double rack-bay column, there should be a 6" flue space between them for fire protection (AK Material Handling Systems, 2015).

Considering the uprights' posts and flue space, the slot exterior dimensions (e_w , e_l) are 100" and 48" respectively. The beam height for this equipment is 5" and the vertical clearance within slots is 4". The estimated price for a beam is 31.69 USD (Cisco-Eagle, 2016). The estimated price for an upright 200" high is 237.42 USD (Cisco-Eagle, 2016). We assumed that uprights 400" cost twice as much.

To calculate the material handling cost, we assume the DC operates 245 days per year, in 8-hour shifts and 1 shift per day with a 30 minute break for lunch and two breaks of 15 minutes. The operators in forklifts travel at an average speed of 5.5 mph (V_l) and their fork lifting speed is 65 ft/min (Hyster-Company, 2012). We consider an aisle width (a_p) of 12' according to the recommendations of the Occupational Safety and Health Administration (2008). The estimated price for the forklift is between 35,000 USD and 55,000 USD (Equipment-Trader, 2016); so, we assumed a price of 45,000 USD. The maintenance of the forklift cost is an estimated 2,000 USD/year. The efficiency of the operator is assumed to be 85% and the reliability of the equipment 99.4%. The salary of the operator is estimated at 27,000 USD/year for the greater area of Chicago, IL (Glassdoor, 2017), and we considered an overhead of 20% for taxes and benefits (CNN-Money, 2013). Based on these parameters, we calculated the cost of a minute of travel to be 0.45158 USD/min. Finally, we multiplied the cost of a minute of travel by the average number of trips per year (770,535 trips/year) and obtained c_2 equals to 347,958.49 (USD $\times$ trips)/(min $\times$ year).

We estimated the unitary leasing cost c_1 to be 4.95 USD/sq.ft per year for the greater area of Chicago, IL (Loopnet, 2016). Finally, we assume a discount rate

of 4% for the cost analysis.

2 Designing the Rack-bays

We generated rack-bays with up to five slot types using both arrangements by tiers and by cutting stock with a desired service level (α) of 85%. Table 3 shows the optimal slot heights of the ten rack-bay designs. Figure 46 shows the rack-bay patterns for four slot types. When following a tiers arrangement, all rack-bays in the warehouse have the same pattern (Figure 46(a)); however, when following a cutting stock arrangement, the warehouse contains a combination of five different patterns (Figure 46(b)).

No. Slot Types	Tiers	Cutting Stock
Two	[48, 80]	[50, 80]
Three	[40, 60, 80]	[44, 56, 80]
Four	[20, 44, 56, 80]	[38, 48, 60, 80]
Five	[38, 44, 50, 56, 80]	[40, 48, 54, 62, 80]

TABLE 3

Slot heights (inches) of the rack-bays designs.

Figure 46. Rack-bay patterns using four slot types.

Figure 47 shows the number of rack-bays for the designs generated and the

percentage improvement over the previous number of slot types. The number of rack-bays serves as a surrogate for the warehouse footprint, as in general a warehouse with fewer rack-bays will have a smaller footprint. We observed savings of 22.2% and 36.2% in the number of rack-bays required for rack-bays designs with slot types by tiers and by cutting stock respectively. Additionally, we see that rack-bay designs by cutting stock are significantly better than designs by tiers for up to four slot types; however, for five slot types they are very similar. Overall, a tier-based designs with 5 slot types have 36.1% fewer rack-bays (38% for a design by cutting stock) than a design with one slot type.

Figure 47. Number of rack-bays for the rack-back designs.

3 Creating the warehouse designs

We generated a warehouse design for each rack-bay design of Section 2 and evaluated its performance in terms of footprint, travel time, total operational cost, storage service level, and racking cost. For the designs by cutting stock, we use DOS_m to arrange the different rack-bay patterns in the layout.

Moreover, we account for slot blocking—a storage stock out that occurs even if there are enough slot quantities to serve the current inventory because pallets are being stored in slots taller than their “preferred” height (Cardona and Gue, 2019a).

Using the discrete event simulation model presented in Section 2 and the data set of 1200 randomly generated problem instances of Section 3, we calculated that, in most cases, slot blocking can be offset by adding an extra one percent more rack-bays to the designs.

Figure 48 shows the layouts for one, two, and five slot types for warehouse layouts by cutting stock. The layouts are plotted in a 720' by 372' frame, where each small rectangle represents a rack-bay. The white area within the frame of Figures 48(b)-48(d) represents the savings footprint. The layout with five slot types is 35.4% smaller than the layout with one slot type. When following a tiers arrangement, the footprint savings are 33.7 percent.

Figure 48. Top view of warehouse layouts by cutting stock.

We use the discrete event simulation model presented in Section 2 to calculate the expected travel time for all warehouse designs (Table 4). Using multiple slot types affects the travel time of a warehouse in two ways. First, the

storage policy must consider the compatibility of pallets and the different slot heights. Second, because warehouses using multiple slot types are smaller, pickers do not travel as far. Overall, we observe savings of 14.1% and 16.6% in the expected travel time, for tiers and cutting stock designs respectively.

	Slot types	Area (sq. ft)	E[T] (minutes)	Racking cost (USD)	Service Level (%)
Tiers	1	267,840	1.367	3,852,648	96.3
	2	220,320	1.215	3,250,145	96.0
	3	193,392	1.217	2,903,891	94.6
	4	193,392	1.218	2,907,155	99.2
	5	177,408	1.173	2,703,858	96.8
Cutting Stock	1	267,840	1.367	3,852,648	96.3
	2	198,288	1.207	3,042,877	97.3
	3	182,952	1.185	2,766,372	96.8
	4	177,408	1.170	2,771,453	99.2
	5	172,800	1.164	2,659,462	97.5

TABLE 4

Metrics of the warehouse designs using multiple slot types.

Using the cost parameters described in Section 1, we calculated the annual operational cost (material handling plus leasing cost). Figure 49 shows the annual operational cost for each warehouse design and the percentage improvement over the previous number of slot types. The most significant savings are for two and three slot types as in Figure 47. The annual operational costs of using five slot types are 24.3% and 25% for designs by tiers and by cutting stock respectively.

Furthermore, we studied the effect on the racking cost. We observe that warehouses using multiple slot heights require fewer rack-bays and in turn less racking. In the case study, the savings on the racking cost were 29.8% and 30.9% for designs by tiers and by cutting stock respectively.

We also tracked storage shortages during the simulation. Although the

Figure 49. Total Operational cost

required service level is 85%, we find that our method to generate warehouse designs is conservative as all designs recorded service level of more than 94%. This is mainly because when generating the warehouse layout, only full rows or full columns of rack-bays can be added.

Finally, we investigate the space utilization of the designs. During the simulation, we tracked the use of space at each moment in time through following four indicators:

- TUS is the fraction of total warehouse volume that can be used to store pallet loads.
- $ES(t)$ is the fraction of the total warehouse volume that can be used to store pallet loads but is empty at time t .
- $SU(t)$ is the fraction of the total warehouse volume occupied by the pallet loads at time t . It is what we define as space utilization.
- $LVS(t)$ is the fraction of the total warehouse volume that is being wasted within occupied slots at time t because of the mismatch of pallet heights and slot heights. For unoccupied slots, we say that loss of vertical space is zero.

The total usable space equals to the sum of empty slots space, the loss of vertical space, and space occupied by the pallet loads at every time t , i.e. $TUS = ES(t) + SU(t) + LVS(t)$. Table 5 shows TUS and the time averages of $LVS(t)$ and $SU(t)$ for the five warehouse designs generated by cutting stock. Having multiple slot types reduces the loss of vertical space within slots, which decreased from 12.67% for one slot type to 2.57% for five slot types. Because of this reduction of wasted space, the warehouses have a higher space utilization, which increased from 13.49% for one slot type to 21.07% for five slot types.

Slot types	TUS	$\overline{LVS}$	$\overline{SU}$
1	27.72%	12.67%	13.49%
2	25.63%	6.37%	18.51%
3	24.77%	4.15%	19.47%
4	24.34%	3.61%	19.97%
5	24.82%	2.57%	21.07%

TABLE 5

Space utilization of cutting stock design for case study.

D Conclusions

Using multiple slot heights in unit-load warehouses has significant benefits in terms of footprint, expected travel time, and racking cost. In our case study, we observed reductions in footprint of 25.9%, 31.6%, and 35.4% for two, three, and five slot types respectively. In terms of the expected travel time, the benefits were 11.6%, 13.3%, and 14.8% for two, three, and five slot types respectively. Using cost parameters of a typical unit-load warehouse in the greater area of Chicago, USA, the expected savings on the annual operational cost (leasing plus material handling cost) are 14.5%, 19.6%, and 24.3% for two, three, and five slot types respectively.

We explored two methods to generate designs for rack-bays—tiers and cutting stock. In general, designs by cutting stock consume less space; however, they are

more difficult to operate for warehouse associates. Therefore, we recommend designs by tiers for distribution centers with paper based picking and put-away operations, and designs by cutting stock for distribution centers with directed picking and put-away. Furthermore, we observed that the space benefits of cutting stock designs are significant for up to four slot types, but beyond this point, they are very similar.

When using rack-bays by cutting stock, there are multiple rack-bay patterns that must be arranged into a layout. We consider five methods to assign patterns to layout locations (Rows, Columns, *DOSm*, Greedy, and Random). Out of them, *DOSm* and Greedy presented the best performance. However, Rows and Columns deviated from the best known performance by no more than 2 percent.

A secondary benefit of using multiple slot heights is lower racking cost. Layouts with multiple slot heights occupy less area and, in turn, need less racking. Therefore, we observed savings of 15% to 30% on the racking cost depending on the number slot types.

Furthermore, we strongly recommend using *best fit, closest open location* as the storage policy for warehouses using multiple slot heights. Our simulations showed that with this storage policy, warehouses using multiple slot heights have a lower footprint and travel time without compromising the storage service level. However, with other storage policies such as closest open location, the storage service level can be significantly affected. Moreover, slot blocking can also affect the storage service level; however, we observed that adding 1 percent more rack-bays to the layout offsets this effect.

Finally, warehouses using multiple slot types require additional procedures for the put-away and picking activities to make sure that pallets are assigned to feasible slots and are retrieved from the most convenient ones. Determining the best number of slot types requires a trade-off between a better space utilization and the additional administrative efforts. In practice, we do not expect more than ten slot types to be used in warehouse designs because the benefits of using multiple slot

types flattens after five slot types; and the complexity of such designs is higher.

CHAPTER IV

PALLET RACKS WITH DYNAMIC SLOT HEIGHTS IN UNIT-LOAD WAREHOUSES

Although using multiple slot heights significantly decreases the loss of vertical space within slots, it does not completely eliminate it, and in warehouses where inventory levels are highly variable or product mixes change rapidly, this waste can still be significant. Examples of this situation in practice include warehouses with correlated order profiles, demands with seasonal peaks, product launches, and distribution network consolidations (Forger, 2018). Outsourced warehousing is, most often, the way that companies provide the extra storage capacity needed during peaks (Hariga, 2011), because redesigning the storage racks is generally considered a major project. Alternatively, we propose to add mechanisms to the beams and uprights of pallet racks to allow slot heights to be easily adjusted such that the loss of vertical space within slots is minimized.

The purpose of this paper is to study the potential benefits of pallet racks with dynamic slot heights for unit-load warehouses. We begin by investigating a theoretical storage system rather than focusing on a specific mechanism. Then, we compare the space requirements of three storage systems with dynamic slot heights: pallet racks with “modular cell”, pallet racks with transferable beams, and pallet racks with semi-automated, sliding beams.

A Potential benefits of pallet racks with dynamic heights

We compare the space requirements (number of rack-bays) B_f of selective pallet racks and requirements B_d for pallet racks with dynamic slot heights. The space requirements of both systems differ because of the ways wasted space accumulates within the rack-bays. In selective pallet racks, pallets are not placed on top of each other within slots, and a slot can only store pallets lower than itself. This creates two sources of wasted space: empty slots and loss of vertical space within slots (Figure 50(a)). In pallet racks with dynamic slot heights, however, all empty space within a rack-bay can be aggregated at the top (Figure 50(b)), increasing its potential for storage.

Figure 50. Difference between the wasted space within selective pallet racks and dynamic heights.

In selective pallet racks, slot heights can be adjusted, but it takes so much effort that companies rarely do it. Therefore, we assume slot heights are fixed in selective pallet racking and refer to them as pallet racks with fixed beams.

Following the notation of Cardona and Gue (2019a), we denote a system of pallet racks with fixed beams by the tuple $\langle x, N, \eta \rangle$, where $x = (x_1, x_2, \dots, x_L)$ are the slot heights, $N = (N_1, N_2, \dots, N_L)$ the slot quantities, and η the assignment of slots to rack-bays. We assume $\langle x, N \rangle$ can be packed into $\mathcal{Q}$ rack-bays of height $\mathcal{H}$ without

leaving any remaining space; therefore, we denote a system of pallet racks with fixed beams by $\langle x, N, \mathcal{H} \rangle$. In contrast, in pallet racks with dynamic slot heights pallets can be stowed within any open space. We denote a system of pallet racks with dynamic slot heights by $\langle \mathcal{Q}, \mathcal{H} \rangle$.

Let h_k be the height of pallets of sku k and I_{kt} the historic (forecasted) inventory levels, then a problem instance is given by the tuple $P = \langle h_k, I_{kt} \rangle$. In this way, we define the space savings of pallet racks with dynamic slot heights for a problem instance P as the normalized difference between the space requirements of pallet racks with fixed beams $B_f(P)$ and with dynamic slot heights $B_d(P)$,

$$\Delta(P) = \frac{B_f(P) - B_d(P)}{B_f(P)}.$$

Because $\Delta(P)$ depends on P , we must study a statistic of $\Delta(P)$ rather than $\Delta(P)$ itself. The most common statistics are the maximum and the average — worst case and average case in the analysis of algorithms (Greene and Knuth, 2007). We choose the average because it has been shown to be a better predictor in practice (Szpankowski, 2011). Let $\mathcal{P}$ be the population of problem instances, then the expected space savings of pallet racks with dynamic slot heights is

$$\bar{\Delta} = \frac{1}{|\mathcal{P}|} \sum_{P \in \mathcal{P}} \left(\frac{B_f(P) - B_d(P)}{B_f(P)} \right),$$

where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$.

In preliminary studies, we observed that $\Delta(P)$ varies greatly depending on whether or not the inventory signals I_{kt} of P are stationary—i.e., the mean, variance, and autocovariance of I_{kt} are approximately constant over time. Therefore, we split the analysis of $\bar{\Delta}$ into $\bar{\Delta}_\tau$ and $\bar{\Delta}_v$ for problem instances with stationary and non-stationary inventory signals, respectively.

1 Warehouses with stationary inventory levels $\bar{\Delta}_\tau$

We develop a probabilistic analysis of $\bar{\Delta}_\tau$ assuming the arrivals and departures of pallets for each sku k follow a Markovian process (Section 1.1). Then,

we use a simulation model to estimate $\bar{\Delta}_\tau$ over randomly generated problem instances based on industry data (Section 1.2).

1.1 Asymptotic analysis of $\bar{\Delta}_\tau$

We use an open queueing network model (Bitran and Dasu, 1992) to show that the steady state distribution of pallet heights F converges when arrivals and departures follow a Markovian process. Then, we use asymptotic methods (G. et al., 1988) to study the “packing” of F into rack-bays. For pallet racks with fixed beams, we pack pallets into slots, then slots into rack-bays. For pallet racks with dynamic slot heights, we pack pallets directly into rack-bays.

For the queueing model (Figure 51), we assume pallets of sku k arrive according to a Poisson process with rate λ_k and stay in the system for an exponentially distributed amount of time with mean $1/\mu_k$. Arrival and service times are independent. Now, imagine there is an infinite number of rack-bays, or equivalently, that the storage system always has sufficient space to store required pallets. We can model this system as an $M_K/M_K/\infty$ queue, in which K classes of customers, each having a (potentially) unique exponential service rate, arrive to a system with an infinite number of servers (therefore, there is no waiting). For our storage system, skus are classes of customers and rack-bays are servers.

Let X_k be the random variable that describes the number of pallets of sku k in the system in steady state and $\tilde{p}_k = \lambda_k/\mu_k$ its expectation (Kleinrock, 1975, section 3.4). Let p_k be the normalized values of $\tilde{p}_k$ (i.e. $\sum_k p_k = 1$) and h_k be an ordered set (if in practice there are multiple skus with the same pallet height, we aggregate them). Then, $\langle h_k, p_k \rangle$ describes the discrete distribution of pallet heights of the storage system in steady state. Now, let F be the cumulative distribution function of the continuous approximation of $\langle h_k, p_k \rangle$. In this way, we denote the expected space savings of using pallet racks with dynamic slot heights by $\bar{\Delta}_\tau(F)$.

To study $\bar{\Delta}_\tau(F)$, we appeal to the bin packing literature, where a similar

Figure 51. Queueing model of storage system.

problem arises. In the stochastic online bin packing problem (online BPP), one-dimensional items are drawn from an arbitrary distribution F and packed immediately into bins of fixed size without information of future arrivals with the objective to minimize the number of bins (Hoffmann, 1982). In this analogy, items are pallets and bins are rack-bays. There is no known optimal algorithm to solve the online bin packing (Seiden, 2002); therefore, most of the literature focuses on approximation algorithms. In this context, the number of bins $B_A^n(F)$ necessary to pack n items drawn from F using algorithm A is used as a performance metric of A . The online BPP has been studied extensively in computer science and our probabilistic analysis of $\bar{\Delta}_\tau(F)$ is based on this body of work.

Following the notation of Coffman et al. (1997) as closely as possible, let $S^n(F)$ be the sum of the heights of n items drawn from F and $W_A^n(F)$ the expected wasted space in the bins left after packing those items, then

$$B_A^n(F) = S^n(F) + W_A^n(F). \quad (26)$$

Because we want to compare the space requirements of both storage systems (fixed beams and dynamic heights) under optimal conditions, we assume A is optimal and called it OPT (by optimal algorithm, we mean an algorithm that finds optimal solutions). From now on, we simply write $B^n(F)$ and $W^n(F)$ to denote

$B_{OPT}^n(F)$ and $W_{OPT}^n(F)$. Furthermore, we want to study the system in steady state, not at a specific n . In the online BPP literature, this is achieved by calculating the metric of interest when $n \rightarrow \infty$. In this way, we can express the expected space savings of using pallet racks with dynamic slot heights $\bar{\Delta}_\tau(F)$ in terms of $B_f^n(F)$ and $B_d^n(F)$, which are the space requirements of selective pallet racks with fixed beams and dynamic heights, respectively.

$$\bar{\Delta}_\tau(F) = \lim_{n \rightarrow \infty} \frac{B_f^n(F) - B_d^n(F)}{B_f^n(F)}$$

For notational convenience, we introduce the ratio

$$\begin{aligned} R^n(F) &= \frac{S^n(F)}{B^n(F)} \\ &= \frac{S^n(F)}{S^n(F) + W^n(F)}, \end{aligned} \tag{27}$$

which is the expected space utilization of the bins. Then, we express $\bar{\Delta}_\tau(F)$ in terms of this ratio.

$$\begin{aligned} \bar{\Delta}_\tau(F) &= \lim_{n \rightarrow \infty} \left[\frac{R_f^n(F)}{S^n(F)} \left(\frac{S^n(F)}{R_f^n(F)} - \frac{S^n(F)}{R_d^n(F)} \right) \right] \\ &= 1 - \lim_{n \rightarrow \infty} \frac{R_f^n(F)}{R_d^n(F)} \end{aligned} \tag{28}$$

Now, we must choose a distribution F that represents a typical scenario, which is the result of reasonable values for λ_k and μ_k . We use the uniform distribution for our average case analysis because: i) The uniform distribution is supported on a bounded interval (there are no pallets of infinite height), and in the absence of historical data, it is the best representation of F (Guiasu and Shenitzer, 1985)—i.e., out of all continuous distributions, the uniform distribution has the highest entropy; ii) The results of Cardona and Gue (2019a) suggest that $\bar{\Delta}_\tau(U)$ is an upper bound for $\bar{\Delta}_\tau(F)$ in practice. Using industry data, Cardona and Gue (2019a) observed that the normal distribution $\mathcal{N}$ is a good approximation of F and that the wasted space within the slots is directly proportional to the variance σ^2 of

$\mathcal{N}$, and given σ^2 is proportional to the entropy of $\mathcal{N}$ (Misra et al., 2005), this suggests that $\bar{\Delta}_\tau(U)$ is an upper bound of $\bar{\Delta}_\tau(\mathcal{N})$. Unfortunately, we are unable to prove this conjecture.

We develop the asymptotic analysis of $\bar{\Delta}_\tau(F)$ when F follows the uniform distribution $U(0, u]$ for $u \leq 1$ and $u = 1/p$, where $p \in \mathbb{Z}^+$ and $p > 1$. That is, the tallest pallet is at most $\mathcal{H}/2$ high. Karmarkar (1982) proved that there exists a “perfect” packing of $U(0, u]$ for $0 < u \leq 1$. In our context, this means that n items drawn from $U(0, u]$ can be packed into bins of size 1 leaving negligible wasted space when $n \rightarrow \infty$; consequently, $W_d^\infty(U(0, u]) = 0$. By Equation 27, this means that $R_d^\infty(U(0, u]) = 1$. By substituting $R_d^\infty(U(0, u]) = 1$ into Equation 28, we obtain

$$\bar{\Delta}_\tau(U(0, u]) = 1 - \lim_{n \rightarrow \infty} R_f^n(U(0, u]).$$

Now, by Equations 26–27,

$$\begin{aligned} 1 - R^n(F) &= 1 - \frac{S^n(F)}{B^n(F)} \\ &= \frac{W^n(F)}{B^n(F)}, \end{aligned}$$

therefore,

$$\bar{\Delta}_\tau(U(0, u]) = \lim_{n \rightarrow \infty} \frac{W_f^n(U(0, u])}{B_f^n(U(0, u])}. \quad (29)$$

For pallet racks with fixed beams, the analysis has two parts. First, pallets are packed into slots with expected waste $W_s^n(F)$. Then, slots are packed into rack-bays with expected waste $W_r^n(F)$. In this way, $W_f^n(F) = W_s^n(F) + W_r^n(F)$. We are interested in analyzing the system with optimal slot heights; so we first determine the vector of slot heights x^* . Then, we calculate $W_s^n(F)$ and $W_r^n(F)$ under x^* , which we denote by $\bar{W}_s^n(F)$ and $\bar{W}_r^n(F)$, respectively.

To calculate $\bar{W}_s^n(F)$, we first estimate $\bar{B}_s^n(F)$, the sum of the height of all slots necessary to pack n items drawn from F . Then, we use Equation 26 to obtain $\bar{W}_s^n(F)$. Let $\bar{B}_s^1(F)$ be the normalized value of $\bar{B}_s^n(F)$, i.e., $\bar{B}_s^1(F) = \bar{B}_s^n(F)/n$ for n

items drawn from $U(0, u]$. Then, $\bar{B}_s^1(F) = x^* \cdot N^*$, where $x^* = (x_1, x_2, \dots, x_L)$ are the optimal slot heights, $N^* = (N_1, N_2, \dots, N_L)$ the optimal slot quantities, and $\sum_i N_i = 1$. Because we study the storage system with an optimal dispatching discipline (i.e., $A = OPT$), we assume items of height $(0, x_1]$ are stored only in slots of height x_1 , items of height $(x_1, x_2]$ are stored only in slots of height x_2 , and so on. Therefore,

$$\begin{aligned} N_1 &= F(x_1) \\ N_i &= F(x_i) - F(x_{i-1}) \quad 2 \leq i \leq L. \end{aligned} \tag{30}$$

Using Equation 30, we write $\bar{B}_s^1(F)$ as a function of x^*

$$B_s^1(F) = x_1 F(x_1) + \sum_{i=2}^{L-1} x_i (F(x_i) - F(x_{i-1})) + x_L (1 - F(x_{L-1})),$$

and

$$B_s^1(U(0, u]) = x_1^2 + \sum_{i=2}^{L-1} x_i (x_i - x_{i-1}) + u (1 - x_{L-1}). \tag{31}$$

To find x^* , we solve $\nabla B_s^1(U(0, u]) = 0$, which results in a linear system of equations $\mathcal{A}x = b$ with

$$\mathcal{A} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & . & . & 0 \\ 0 & -1 & 2 & -1 & . & . \\ . & . & -1 & 2 & -1 & . \\ 0 & . & . & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 0 \\ 0 \\ . \\ . \\ . \\ u \end{pmatrix}.$$

Because $\mathcal{A}$ is a square matrix and has linearly independent rows, it is invertible (Artin, 1991). This implies that x^* exists and is unique. We compute $x^* = \mathcal{A}^{-1}b$ for $1 \leq L \leq 10$ and find that

$$x_i^* = \frac{i}{L} u \quad \text{and} \quad N_i^* = \frac{1}{L} \quad 1 \leq i \leq L. \tag{32}$$

Substituting these x_i^* values into Equation 31,

$$\bar{B}_s^n(U(0, u]) = \frac{un}{2} \left(\frac{L+1}{L} \right). \quad (33)$$

Consequently, the expected wasted space within slots is

$$\begin{aligned} \bar{W}_s^n(U(0, u]) &= \bar{B}_s^n(U(0, u]) - nE[U(0, u)] \\ &= \left(\frac{u}{2} \left(\frac{L+1}{L} \right) - \frac{u}{2} \right) n \\ &= \frac{un}{2L}. \end{aligned} \quad (34)$$

Once we have packed $U(0, u]$ into a slot profile $\langle x^*, N^* \rangle$, we must pack $\langle x^*, N^* \rangle$ into rack-bays. The slot profile $\langle x^*, N^* \rangle$ represents a discrete uniform distribution with support x^* and probabilities N^* . Lemma 1 states that we can map $\langle x^*, N^* \rangle$ into a special discrete uniform distribution denoted by $U\{j, k\}$, for which we have a wasted space analysis available.

Lemma 1. *For every $\langle x^*, N^* \rangle$ generated from $U(0, u]$, there is a mapping of $\langle x^*, N^* \rangle$ into $U\{j, k\}$, where $U\{j, k\}$ is a discrete uniform distribution with support $1, 2, \dots, j$, for any $j, k \in \mathbb{Z}^+$ with $j < k - 1$.*

Proof. Given that u in $U(0, u]$ is of the form $1/p$ for a $p \in \mathbb{Z}^+$, we write x^* (Equation 32) as $i/(Lp)$. In this way, $(Lp)x^*$ is equivalent to $1, 2, \dots, j$. Also $p > 1$, then $L < Lp$. Consequently, $\langle (Lp)x^*, N^* \rangle$ is equivalent to $U\{L, Lp\}$, which is of the form $U\{j, k\}$ as defined in Lemma 1. $\square$

E. et al. (2002) show that the expected wasted space of packing $U\{j, k\}$ is $O(1)$. This means that n items drawn from $U\{j, k\}$ can be packed into bins of size k with constant expected wasted space; in turn, this means that $\bar{W}_r^\infty(U\{j, k\}) = 0$. Because $\langle x^*, N^* \rangle$ can be mapped into $U\{j, k\}$ (Lemma 1), the expected wasted space of $\langle x^*, N^* \rangle$ is also $O(1)$, which in turn means that $\bar{W}_r^\infty(\langle x^*, N^* \rangle) = 0$; therefore, $B_f^n(U(0, u]) = \bar{B}_s^n(U(0, u])$. Finally, we use Equations 33 and 34 to simplify Equation 29.

$$\begin{aligned}\bar{\Delta}_\tau(U(0, u]) &= \lim_{n \rightarrow \infty} \frac{\bar{W}_s^n(U(0, u])}{\bar{B}_s^n(U(0, u])} \\ &= \frac{1}{L+1}.\end{aligned}\tag{35}$$

Figure 52 plots Equation 35 for different number of slot types. We see that the expected space savings of pallet racks with dynamic slot heights is 33%, when compared to pallet racks with fixed beams for two slot types $L = 2$. For $L = 5$, the expected space savings are 16.6%, and for $L = 10$, around 9%.

Figure 52. Space savings of using pallet racks with dynamic heights $\bar{\Delta}_\tau(F)$ when F follows an uniform distribution.

1.2 Estimating $\bar{\Delta}_\tau$ based on industry data

In addition to our asymptotic analysis, we estimate the potential benefits of pallet racks with dynamic slot heights using problem instances randomly generated from data of pallet heights and historic inventory signals from five distribution centers. The database contains inventory history of around 10,000 skus for about 18 months and pallet heights of around 55,000 skus. We generated a problem instance $\langle h_k, I_{kt} \rangle$ by randomly selecting n skus from the database. We generated problem instances based on a factorial design with 20 replications with factors and treatments: $n = (1000, 2000, 3000)$ number of skus, $L = (2, 3, 4, 5)$ number of slot types, and $\alpha = (0.8, 0.9, 0.5)$ the desired service level. We evaluated up to five slot

types, because Cardona and Gue (2019a) observed that the space savings flattens for more slot types. In this way, the experiment contained 720 problem instances.

For each problem instance $\langle h_k, I_{kt} \rangle$, we find the optimal slot profile $\langle x^*, N^*, \mathcal{H} \rangle$ following the procedure of Cardona and Gue (2019b). Then, we use a discrete event simulation model $\mathcal{S}$ to find the service level Λ provided by $\langle x^*, N^*, \mathcal{H} \rangle$. The simulation model $\mathcal{S}$ receives $\langle h_k, I_{kt} \rangle$ and starts from an optimal packing of I_{k1} into the rack-bays by solving a cutting stock problem (Cardona and Gue, 2019b). Then, $\mathcal{S}$ uses a dispatching policy A to determine the specific slots to which pallets are assigned to and retrieved from for each period t according to I_{kt} . In warehousing, A is known as the storage policy. The most common storage policies in the literature are random pick (Hausman et al., 1976), closest open location (Park and Lee, 2007), Duration-of-Stay (Goetschalckx and Ratliff, 1990), and robust storage assignment (Ang et al., 2012). These storage policies aim to minimize travel time, which compromises space utilization. In contrast, Quintanilla et al. (2015) proposed a storage policy focused on space utilization. However, it assumes that the information of all incoming pallets is available at the beginning of the operational shift, which does not apply to our context. Instead, we resort to approximation algorithms for the online BPP to define A .

There are two main families of algorithms for the online BPP: Fit and Harmonic. Examples of Fit algorithms are First-fit, Next-fit, and Best-fit (Sgall, 2014). A common way to evaluate their performance is the asymptotic worst-case ratio — the ratio between the number of bins found by an algorithm and the optimum in the worst case (Epstein et al., 2012). First-fit and Best-fit have an asymptotic worst-case ratio of 1.7, while for Next-fit it is 2 (Seiden, 2002). To improve this performance, Lee and Lee (1985) introduce Harmonic with an asymptotic worst-case ratio of 1.69. Several variants of Harmonic were developed afterwards. The best known asymptotic worst-case ratio is 1.58 (Seiden, 2002). Another approach to evaluate the performance of online algorithms is the

average-case analysis that focuses on a typical scenario (Shor, 1991), where First-Fit and Best-fit present the best performance (Kamali and López-Ortiz, 2015)—the asymptotic average-case performance ratio of First-fit and Best-fit is 1 while that of the Harmonic is 1.29 (Shor, 1986).

The performance analysis of these algorithms is sensitive to what is defined as a “typical” case. In general, there is no agreement on which performance measure is better. The most promising approaches are Best-fit (Johnson, 1974), Harmonic- k (Lee and Lee, 1985), Sum of Squares (Csirik et al., 2006), Lagrangian-based (Gupta and Radovanovic, 2015), and CHAMP (Asta et al., 2016). Nevertheless, Best-fit seems to have the best performance in practice (Ghaderi et al., 2014; Rajagopal, 2016; Kamali and López-Ortiz, 2015). Therefore, we define Best-fit as the dispatching policy A of our simulation model $\mathcal{S}$. Specifically, to put a pallet of height h away, A identifies all the rack-bays where it fits. Then, it chooses the rack-bay where the sum of the height of the pallets within is maximal. Ties are broken randomly. Analogously, to pick a pallet of sku k , A chooses the rack-bay where the sum of the height of the pallets within is minimal out of all rack-bays that contain a pallet of sku k .

To find a system of pallet racks with dynamic slot heights $\langle \mathcal{Q}, \mathcal{H} \rangle$ equivalent to $\langle x^*, N^*, \mathcal{H} \rangle$ for the problem instance $P = \langle h_k, I_{kt} \rangle$, we use a simulation-based optimization model to find the minimum number of rack-bays $\mathcal{Q}$ that provides a service level Λ for a system with pallet racks with dynamic slot heights $\langle \mathcal{Q}, \mathcal{H} \rangle$. This model is based on the simulation model S and a bisection search (Waeber et al., 2013) given that Λ is monotone with respect to $\mathcal{Q}$, i.e, the more rack-bays, the higher the service level.

Finally, we define $\Delta_\tau(P)$ as the normalized difference between the number of rack-bays of $\langle x^*, N^*, \mathcal{H} \rangle$ and $\langle \mathcal{Q}, \mathcal{H} \rangle$ for a problem instance P . Figure 53 shows the space savings $\bar{\Delta}_\tau$ for the 720 problem instances by number slot types. We observed that for two slot types, the expected space savings are between 23% and 27% in

most cases, and for five slot types, they are between 9% and 12%. These results are consistent with the asymptotic analysis of $\bar{\Delta}_\tau(U(0, u])$ presented in Section A. which provides empirical evidence to the conjecture that $\bar{\Delta}_\tau(U(0, u])$ is an upper bound for $\bar{\Delta}_\tau(F)$.

Figure 53. Space savings of using pallet racks with dynamic heights when compare to pallet racks with fixed beams based on industry data.

2 Warehouses with non-stationary inventory levels $\bar{\Delta}_v$

To this point we have studied the potential benefits of pallet racks with dynamic racks for warehouses with stationary inventory levels. In this section, we study $\bar{\Delta}_v$ for warehouses with rapidly changing business environments or correlated order profiles, where the assumption of stationarity is no longer valid.

We run a completely randomized design with 1,000 problem instances generated from synthetic, non-stationary time series. We generated each problem instance $P = \langle h_k, I_{kt} \rangle$ based on a set $\mathcal{F}$ of features (Table 6). Then, we use the same procedure of Section 1.2 to generate a system of pallet racks with fixed beams $\langle x, N, \mathcal{H} \rangle$ and one with dynamic slot heights $\langle \mathcal{Q}, \mathcal{H} \rangle$. For each P , we use the procedure to find pallet racks with fixed beams with two up to five slot types $L = (2, 3, 4, 5)$ and with a desired service level $\alpha = 0.95$. Finally, we calculate $\Delta_v(P)$ as the normalized difference between the number of rack-bays of both systems for P .

We are interested in estimating the expected value $\bar{\Delta}_v$ of $\Delta_v(P)$ and whether or not there are factors in $\mathcal{F}$ that have a statistically significant effects on $\bar{\Delta}_v$.

Feature	Treatments
Number of skus	[1000, 3000, 8000]
Distribution of h_k	[uniform, truncated normal, or truncated bimodal normal]
Range of h_k	[40, 50, 80]
Variance of h_k	[10, 20, 30]
Demand signal type	[random walks, harmonic, Gaussian Linear, Gaussian Matérn, Gaussian Exponential]
Average days with orders	[40, 120]
Variance of days with orders	[400, 10000]
Average maximum order quantity	[5,10]
Range of lead times	[1, 10]

TABLE 6

Factors and treatments of the completely randomized experiment with non-stationary time series.

To generate a problem instance P , we use the Python package TimeSynth (Maat and Malali, 2017) to generate the synthetic time series and Numpy (Oliphant, 2006) as a random number generator. To generate the pallet heights h_k for a P , we use uniform, truncated normal, and truncated bimodal normal probability distributions, based on a given variance and range; the mean was chosen such that the distribution is symmetrical.

To generate I_{kt} , we first generate synthetic, non-stationary time series for the sku demands D_{kt} . Then we simulate a periodic-review inventory policy (R, T) (Rao, 2003) over D_{kt} . We consider three types of signals (random walks, harmonic, and Gaussian processes), which can be parameterized to reproduce several kinds of non-stationary processes such as growing variances, trends, and seasonality. The random walk (Spitzer, 1970) had no drift and was generated using Numpy; the

harmonic signal was sinusoidal (Cattani, 2010); and for the Gaussian processes, we included linear, Matérn, and exponential kernels (Roberts et al., 2013). The demand signals D_{kt} were generated using irregular time samples with an order frequency (number of days with orders) according to a mean and variance defined in $\mathcal{F}$. Figure 54 plots two examples of the demand signals generated using this process— a Gaussian process with a Matérn kernel and a random walk.

(a) Gaussian process with Matérn kernel, 71 days with orders.

(b) Random walk without drift, 100 days with orders.

Figure 54. Synthetic time series with irregular timestamps for D_{kt} with a horizon of 600 days.

For the inventory policy (R, T) , we use a dynamic T with variable lead times to create trends in the inventory signals, i.e., the mean of the lead time changes over time. The interaction of the different signal types and the dynamic T results in four types of inventory signals (Figure 55): positive trend, negative trend, seasonal, and stationary. We use a vector of probabilities $Y = (y_1, y_2, y_3, y_4)$ to determine the type of inventory signal of each sku k in the problem instance P . We generate Y from a random partition of the interval $[0, 1]$ (Drobot, 1981).

Finally, $\mathcal{F}$ contains 9 factors that define a scenario, out of which a problem instance is generated (Table 6). We did a preliminary run of the experiment with 200 problem instances and use the method describe by Simpson et al. (2002) to

(a) Positive trend

(b) Negative trend

(c) Seasonality

(d) Stationary

Figure 55. Inventory signals generated from inventory policy (R, T) over randomly generated demands D_{kt} .

determine a sample size, which resulted in approximately 1,000 problem instances.

Using ANOVA to analyze the results, we found that L and the coefficient of variation of the sum of the height of all pallets in the system CV have statistically significant effects on $\bar{\Delta}_v$. Figure 56(a) plots the space savings $\bar{\Delta}_v$ of pallet racks with dynamic slot heights compared to pallet racks with fixed beams by number of slot types L . These results indicate that $\bar{\Delta}_v$ is between 20% and 35% in most cases when the pallet racks with fixed beams have two slot types. For three slot types, they are between 22% and 30%; and for five slot types, between 21% and 24%. We also see that the expected space savings for two slot types is about 33%, and for five slot types, it is 22%. Comparing Figures 53–56(a), we observed that warehouses with non-stationary inventory levels ($\bar{\Delta}_v$) have significantly higher expected space savings of using pallet racks with dynamic slot heights than warehouses with stationary inventory levels ($\bar{\Delta}_\tau$). For example, for five slot types $\bar{\Delta}_\tau$ is 9–12%, whereas $\bar{\Delta}_v$ is 17–26%.

(a) Space savings depending on the number of slot types.

(b) Effect of the coefficient of variation of the sum of pallet heights on the space savings.

Figure 56. Space savings of pallet racks with dynamic heights versus fixed beams

Moreover, we study the effect of the coefficient of variation CV of the space requirements (sum of the height of all pallets) through time on $\bar{\Delta}_v$. Figure 56(b)

shows the negative correlation between CV and $\bar{\Delta}_v$ for five slot types, i.e., the higher the variability of the space requirements, the lower the expected space savings. Pallet racks with dynamic slot heights minimize one kind of wasted space, the loss of vertical space within slots. However, if the space requirements are highly variable through time, there will be unavoidable empty space in the storage system during valley periods. In those cases, pallet racks with dynamic slot heights add little value.

B Three mechanisms for pallet racks with dynamic slot heights

Up to now, we have studied a theoretical system that represents pallet racks with dynamic slot heights as a *storage system class* regardless of the specific mechanism that allows the slot heights to be adjusted. In this section, we study three potential mechanisms for pallet racks with dynamic slot heights—modular cells, transferable beams, and sliding beams. Then, we use a simulation-based optimization procedure similar to the one presented in Section 2 to compare their space requirements.

Pallet racks with modular cells

Lee et al. (1999) introduce pallet racking with modular cells. In this system, there are no beams, but the upright’s poles have lateral supports, upon which forklifts place the pallets (Figure 57(a)). The lateral supports are located every δ inches along the height of the upright’s poles (Figure 57(b)) similar to drive-in pallet racks. Let a slot be the set of δ modular cells that a pallet occupies. Then, the loss of vertical space within a slot is bounded by δ . Therefore, the smaller the δ , the greater the space utilization, but also the greater the cost of racking. Lee et al. (2005) introduce mathematical models to find the δ that minimizes the total cost depending on the pallet heights and quantities. Instead, we choose δ such that space utilization is maximized.

Figure 57. Pallet racks with modular cells.

There are three drawbacks of such a system: a) before being stored, pallets need to be repalletized, because pallets in modular cells have a lateral clearance (Figure 57(b)), whereas it is industry practice to palletize using the whole pallet area. This is extra handling increases labor costs; b) The lateral clearance is wasted space; and c) in the same way computer hard disk drives get fragmented (Moore and Hancock, 2003), the picking and put-away activities create scattered empty spaces throughout the racks that are not suitable for storage, which is wasted space—we call this fragmentation.

Pallet racks with transferable beams

In pallet racks with transferable beams, pallet loads are placed on top of a skid. Then, the forklifts assemble the skid-pallet into the rack-bays. The uprights have integrated hooks every ε inches along their height, where the angles of the skid located at its corners (Figure 1) are engaged (Figure 58(a)). Additionally, skids have two holes in their cross section, where the forks of the material handling equipment

engage. We assume the pallets enter and exit the storage system with the same skid.

Figure 58. Pallet racks with transferable beams.

There is already a system of this kind on the material handling market—STAK adjustable racking system (Stanley Vidmar, 2017). In this system, the picking and put-away of pallets is done by a worker on foot using a captive lifting and handling device with a rail system on top of the racks along which the captive lifting device travels. STAK is designed to store heavy-duty products (2–4 tons) in relatively small storage areas of raw materials in manufacturing facilities. In this context, forklifts are a scarce resource, so STAK offers the advantage that the handling of pallets does not require a forklift. However, it is not suitable for distribution centers (DC), where retrieval times are critical. In a DC, the captive lifting and handling device with an operator on foot creates a bottleneck for the system. Despite this disadvantage, we believe STAK serves as a proof of concept for pallet racks with transferable beams, and we assume it can be mechanically modified such that skids are handled by the forklifts. We also assume ε is small enough such that the loss of vertical of space within slots is negligible. In this way,

fragmentation accounts for all the wasted space within the system.

Pallet racks with sliding beams

In both systems (modular cells and transferable beams), once a pallet is put away, it stays in that place until it is retrieved. We propose a semi-automated system where the beams slide along the height of the uprights—pallet racks with sliding beams. In this system, forklifts stow and retrieve pallets from the rack-bays, whereas the movement of the sliding beams occurs automatically. We envision a mechanism where the uprights have rail guides that allow the beams to slide (Figure 59), with or without a pallet load, and are transferable among rack-bays. Additionally, this system could be equipped with a motorized mechanism that moves the beams, a latching mechanism to secure static pallets, and an electronic control system that coordinates the movement of beams.

Figure 59. Example of a potential mechanism (rail guides) for pallet racks with sliding beams.

Currently, there is no such a system on the material handling market. There are mechanical systems with rail guides in other equipment (Septimatech Group Inc, 2018) and automated systems with electronic control components (Gue et al.,

2014) that could serve as reference for the design of pallet racks with sliding beams. We leave these challenges beyond the scope of this paper and focus on evaluating its space requirements. For that purpose, we assume the time of the movement of the beams is negligible compared to the travel times of the forklifts. Also, we assume idle beams can be removed from a rack-bay if necessary to stow pallets.

Comparing performance

To compare the space requirements of the three systems (modular cells, transferable beams, sliding beams), we use the same set of 1,000 problem instances $P = \langle h_k, I_{kt} \rangle$ of Section 2 generated from synthetic, non-stationary time series. For each P , we use a similar simulation-based optimization procedure as in Section 2, but we modified the simulation model S according to the specifics of each storage system. Then, we compare $\Delta_v(P)$ for each of the three systems for the same service level Λ , using pallet racks with fixed beams and five slot types.

For pallet racks with modular cells, we assume the lateral supports are located every 10". We use $\delta = 10$ as the double of the beam thickness—a beam is commonly 4–6" thick (Cisco-Eagle, 2016). To account for the lateral clearance within pallets, we scale the height of each pallet by ϑ such that the volume of the load of a pallet using the whole area is the same as the volume of a pallet with the clearance. The standard pallet size is 40" $\times$ 48". Based on specifications of drive-in racks (Cisco-Eagle, 2016), we assume the clearance within the pallet for modular cells is 4" at each side. Therefore, we assume $\vartheta = 1.2$.

For modular cells and transferable beams, we modify S to consider that once a pallet is stowed in a rack-bay at height z , it remains at z until it is picked to fulfill a customer order—in S all empty space within a rack-bay is aggregated at the top. We also use a different dispatching discipline A in the simulation model for modular cells and transferable beams. To put a pallet of height h away, A uses the Best-fit rule, but ties are broken by the lowest z within the rack-bay. Analogously, to pick a

pallet of sku k , A chooses the rack-bay where a pallet of sku k is located at the highest z . Our goal is that empty spaces accumulate at the top of the rack-bay.

Figure 60 plots the results of the simulation. The expected space savings were 10.2%, 22.3%, and 23.2% for modular cells, transferable beams, and sliding beams, respectively, when compared to pallet racks with fixed beams ($L = 5$). In 90% of the problem instances, the space savings of modular cells were between 1.4% and 19.0%; for transferable beams, they were between 15.1% and 29.5%; and for sliding beams, they were between 16.0% and 30.5%.

Figure 60. Expected space savings of pallet racks with dynamic heights when compared to pallet racks with fixed beams with five slot types.

Additionally, the difference between the expected space savings of pallet racks with transferable (Figure 60(b)) and sliding beams (Figure 60(c)) is not statistically significant. This means that fragmentation (empty spaces scattered around the

rack-bays) is not a significant fraction of the total wasted space. For the mechanical design, this implies that the ability to move beams with a pallet on top does not bring significant benefits.

The space requirements of the three storage systems are different because of the way wasted space accumulates within. We identify three kinds of wasted space in these systems: loss of vertical space within slots, fragmentation, and the remaining height at the top of the rack-bays. Pallet racks with modular cells have the three kinds of wasted space. In transferable beams, there is no loss of vertical space, whereas in sliding beams, there is only the remaining height at the top of the rack-bays.

Throughout this paper, we have used the number of rack-bays as the surrogate for the warehouse footprint—i.e., we expect the more rack-bays a warehouse has, the bigger its footprint. However, selective pallet racking are commonly set up for two pallet positions per slot, whereas pallet racks with dynamic slot heights are designed for one pallet position per slot. Therefore, pallet racking with dynamic slot heights will require more uprights for the same number of rack-bays, which increases the space requirements. We leave this analysis out scope of this paper, because we believe despite this limitation, the number of rack-bays serves as a good estimate of the space requirements.

C Conclusions

Pallet racks with dynamic slot heights provide significant space savings to unit-load warehouses when compared to pallet racks with fixed beams.

For warehouses with stationary inventory signals, our analytical derivation suggests that the space savings are 9–33 percent, depending on the number of slot types of the pallet racks with fixed beams. Using industry data, we observed that for two slot types, the expected space savings are 23–27 percent in most cases, and for five slot types, they are 9–12 percent.

For warehouses in rapidly changing business environments, we use a set of 1,000 randomly generated problem instances based on synthetic, non-stationary inventory time series. Our results suggest that the expected space savings of pallet racks with dynamic slot heights are 21–24 percent, when compared to pallet racks with fixed beams and five slot types. In most problem instances (95 percent), the space savings were 14–35 percent. Moreover, we found that the variability of the sum of pallet heights (space requirements) is negatively correlated to the space savings of pallet racks with dynamic slot heights. Consequently, pallet racks with dynamic slot heights provide the most benefit to warehouses with highly variable inventory levels and little variability of the space requirements through time.

Finally, we compare the performance of three mechanisms of pallet racks with dynamic slot heights—modular cells, transferable beams, and sliding beams. The expected space savings of modular cells is 2–19 percent; for transferable beams, it is 15–29 percent; and for sliding beams, it is 16–30 percent. Additionally, we found that fragmentation (empty spaces scattered around the rack-bays) is not significant in pallet racks with dynamic slot heights, when using a Best-fit storage policy. This implies that the ability to move beams with a pallet on top does not bring significant space savings.

Although there is no system of pallet racks with dynamic slot heights on the material handling market for unit-load distribution centers, the technology for modular cells and transferable beams is already available. However, pallet racks with sliding beams is a semi-automated system that requires further development in terms of the mechanical mechanism to slide the beams and the electronic controls to coordinate the beam movements. We expect these systems to be more expensive than traditional racking. Nevertheless, we hope that the space savings they provide, encourage their development. We believe these systems will find their space in the material handling market as, in recent years, automation and robotics continue to change the industry.

CHAPTER V

CONCLUSIONS

We present two strategies to maximize space utilization in unit load warehouses that aim to minimize what we call the *loss of vertical space within slots*—the mismatch between the height of the pallet and the height of the slot where it is stored. We found that what sounds like a pedestrian source of waste can lead to significant benefits in terms of footprint, expected travel time, and racking cost.

We introduce the use of multiple slot heights in unit-load warehouses to better match slot heights to the pallet heights distribution and its variation through time. We found that a warehouse with a proper slot profile should expect footprint savings are 20 to 40 percent, depending on number of slot types and variability of inventory levels. Moreover, because of its smaller footprint, it requires less racking—between 15 to 30 percent lower racking costs— and forklift drivers travel less—we observe savings of about 15 percent in the expected travel time. Overall, we expect savings between 15 to 25 percent on the total operating cost.

For practical numbers of slot types (less than 20), we found that the greater the number of slot types, the greater the benefit. However, to implement multiple slot heights, warehouse managers will have to adopt additional procedures for the put-away and picking activities to make sure that pallets are assigned to feasible slots and are retrieved from the most convenient ones. Determining the best number of slot types is a trade-off between a better space utilization and the additional administrative efforts. Furthermore, we notice that some warehouses benefit more than others. Using nominal problem instances randomly generated with data from

several distribution centers, we found that warehouses that benefit the most from using multiple slot heights have large ranges and variances of the distribution of pallet heights.

Using multiple slot heights in unit-load warehouses significantly decreases the loss of vertical space within slots. Nevertheless, it does not completely eliminate it and in some cases, this wasted space can still be significant. For this situation, we propose pallet racks with dynamic heights that are equipped with extra mechanisms to easily adjust the slot heights. There are multiple mechanical ways in which the mechanism can be implemented. However, we study the system as a “class” of pallet racks rather than focusing on a specific mechanism. Our probabilistic analysis indicates that pallet racks with dynamic heights have expected footprint savings of 16 percent when compared to traditional pallet racks (with five slot types and optimal slot profile) for warehouses with stationary and statistically independent order profiles. Moreover, we found that for warehouses where inventory levels are highly variable or product mixes change rapidly, pallet racks with dynamic heights bring even greater footprint savings. Using randomly generated problem instances based on non-stationary, synthetic time series, we found that the expected savings of pallet racks with dynamic heights are 16–30 percent for warehouses in rapidly changing business environments.

Furthermore, we study three systems of pallet racks with dynamic heights and compare their performance—pallets racks with modular cells, transferable beams, and slidable beams. The expected space savings of pallet racks with modular cells is 2 to 19 percent when compared to traditional pallet racks (with five slot types and optimal slot profile). For transferable beams and slidable beams, the space savings are not statistically different and are between 15 to 30 percent. Given its significant benefits, we believe that pallet racks with dynamic heights will find a space in the material handling market as warehouse real estate prices keep rising, and the cost of equipment and automation keeps declining.

REFERENCES

- Septimatech Group Inc. Guide Rail Adjustment, 2018.
<https://septimatech.com/product/guide-rail-adjustment/> [10 March 2019].
- Riccardo Accorsi, Riccardo Manzini, and Marco Bortolini. A Hierarchical Procedure for Storage Allocation and Assignment within an Order-Picking System. A Case Study. *International Journal of Logistics Research and Applications*, 15(6): 351–364, 2012.
- Tobias Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009. ISSN 1867-2949.
- Shabbir Ahmed and Alexander Shapiro. *Solving Chance-Constrained Stochastic Programs via Sampling and Integer Programming*, chapter 12, pages 261–269. Informss, 2008.
- AK Equipment. Pallet Rack Guide, 2017.
<https://www.akequipment.com/pallet-rack-guide/> [10 February 2018].
- AK Material Handling Systems. Pallet Rack & Fire Protection Systems, 2015.
<http://www.akequipment.com/pallet-rack-fire-protection-systems-2/> [12 October 2017].
- Douglas José Alem, Pedro Augusto Munari, Marcos Nereu Arenales, and Paulo Augusto Valente Ferreira. On the cutting stock problem under stochastic demand. *Annals of Operations Research*, 179(1):169–186, 2010. ISSN 02545330.
- Theodore Wilbur Anderson, Theodore Wilbur Anderson, Theodore Wilbur Anderson, Theodore Wilbur Anderson, and Etats-Unis Mathématicien. *An Introduction to Multivariate Statistical Analysis*, volume 2. Wiley New York, 1958.

- Marcus Ang, Yun Fong Lim, and Melvyn Sim. Robust Storage Assignment in Unit-Load Warehouses. *Management Science*, 58(11):2114–2130, 2012.
- Artelys. Artelys Knitro - Nonlinear optimization solver, 2017.
<https://www.artelys.com/en/optimization-tools/knitro>.
- Michael Artin. *Algebra*. Pearson; 1 Edition, 1991.
- Shahriar Asta, Ender Özcan, and Andrew J. Parkes. CHAMP: Creating heuristics via many parameters for online bin packing. *Expert Systems with Applications*, 63:208–221, 2016. ISSN 09574174.
- C. Audet and J. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- C. Audet and J.E. Dennis, Jr. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- C. Audet, S. Le Digabel, G. Couture, and V. Rochon Montplaisir. The NOMAD project, 2017. Software available at <https://www.gerad.ca/nomad/> [20 September 2017].
- Charles Audet, Vincent Béchar, and Sébastien Le Digabel. Nonsmooth optimization through mesh adaptive direct search and variable neighborhood search. *Journal of Global Optimization*, 41(2):299–318, Jun 2008.
- Charles Audet, J. E. Dennis, and Sébastien Le Digabel. Globalization Strategies for Mesh Adaptive Direct Search. *Computational Optimization and Applications*, 46(2):193–215, Jun 2010.
- K. Azadeh, M. B. M. De Koster, and D. Roy. Robotized warehouse systems: Developments and research opportunities. *Transportation Science (In press)*, 2018. Available at doi:10.2139/ssrn.2977779.

- John J Bartholdi and Steven T Hackman. *Warehouse & Distribution Science: Release 0.98*. Atlanta, GA, The Supply Chain and Logistics Institute, School of Industrial and Systems Engineering, Georgia Institute of Technology, 2017.
- Matthias Beck and Sinai Robins. Stationary and non-stationary time series . In H.M. Mader, S.G. Coles, C.B. Connor, and L.J. Connor, editors, *Statistics in Volcanology, Special Publications for IAVCEI No. 1*, pages 129–143. Geological Society, London, 2006.
- Matthias Beck and Sinai Robins. The coin-exchange problem of frobenius. In Matthias Beck and Sinai Robins, editors, *Computing the Continuous Discretely: Integer-Point Enumeration in Polyhedra*, pages 3–24. Springer-Verlag New York, 2007.
- S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39, 2011.
- P. Beraldi, M.E. Bruni, and D. Conforti. The stochastic trim-loss problem. *European Journal of Operational Research*, 197(1):42 – 49, 2009. ISSN 0377-2217.
- Barry Berman. Strategies to Reduce Product Proliferation. *Business Horizons*, 54(6):551 – 561, 2011.
- Gabriel R Bitran and Sriram Dasu. A review of open queueing network models of manufacturing systems. *Queueing systems*, 12(1-2):95–133, 1992.
- Dave Blanchard. Top 5 Warehouse Challenges and How to Overcome Them—Industry Week/IW, 2013.
<http://www.industryweek.com/warehousing-and-distribution/supply-chain-logistics-top-5-warehouse-challenges-and-how-overcome-them> [15 Feb 2018].
- Sebastian Bocker and Zsuzsanna Liptak. A fast and simple algorithm for the money changing problem. *Algorithmica*, 48(4):413–432, Aug 2007. ISSN 1432-0541.

Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2):186 – 204, 2008. ISSN 1572-5286. In Memory of George B. Dantzig.

Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. Algorithms and Software for Convex Mixed Integer Nonlinear Programs. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 1–39. Springer New York, 2012.

Nils Boysen and Konrad Stephan. A survey on single crane scheduling in automated storage/retrieval systems. *European Journal of Operational Research*, 254(3):691 – 704, 2016. ISSN 0377-2217.

Nils Boysen, Dirk Briskorn, and Simon Emde. Sequencing of picking orders in mobile rack warehouses. *European Journal of Operational Research*, 259(1):293 – 307, 2017. ISSN 0377-2217.

Nils Boysen, René de Koster, and Felix Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 2018. ISSN 0377-2217.

Luis F. Cardona and Kevin R. Gue. How to determine slot sizes in a unit-load warehouse. *IIE Transactions*, 51(4):355–367, 2019a.

Luis F. Cardona and Kevin R. Gue. Layout of unit-load warehouses with multiple slot heights. *Transportation Science (In review)*, 2019b.

Carlo Cattani. Harmonic wavelet approximation of random, fractal and high frequency signals. *Telecommunication Systems*, 43(3):207–217, Apr 2010.

Leon Yang Chu and Zuo-Jun Max Shen. A Power-of-Two Ordering Policy for

- One-Warehouse Multiretailer Systems with Stochastic Demand. *Operations Research*, 58(2):492–502, 2010.
- Cisco-Eagle. Pallet Storage: Selective, Pushback, Drive-in, Pallet Flow - and More, 2016. <http://www.cisco-eagle.com/catalog/c-1288-pallet-rack.aspx>.
- CNN-Money. You make \$70k but cost your boss \$88k, 2013. <http://money.cnn.com/2013/02/28/smallbusiness/salary-benefits/>.
- E. Coffman, M. Garey, and D. Johnson. *Approximation Algorithms for Bin Packing: A Survey*. PWS Publishers, Boston, 1997.
- G. Cormier and E.A. Gunn. On Coordinating Warehouse Sizing, Leasing and Inventory Policy. *IIE Transactions*, 28(2):149–154, 1996.
- Janos Csirik, David S. Johnson, Claire Kenyon, James B. Orlin, Peter W. Shor, and Richard R. Weber. On the sum-of-squares algorithm for bin packing. *J. ACM*, 53(1):1–65, January 2006. ISSN 0004-5411.
- R. (M.) B. M. de Koster, T. Le-Duc, and Y. Yugang. Optimal Storage Rack Design for a 3-dimensional Compact AS/RS. *International Journal of Production Research*, 46(6):1495–1514, 2008.
- David A Dickey and Wayne A Fuller. Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *Journal of the American Statistical Association*, 74(366a):427–431, 1979.
- Vladimir Drobot. Uniform Partitions of an Interval. *Transactions of the American Mathematical Society*, 268(1):151–160, 1981. ISSN 00029947.
- Coffman E., Courcoubetis C., Garey M., Johnson D., Shor P., Weber R., and Yannakakis M. Perfect packing theorems and the average-case behavior of optimal and online bin packing. *SIAM Review*, 44(1):95–108, 2002.

- Leah Epstein, Lene M. Favrholt, and Jens S. Kohrt. Comparing online algorithms for bin packing problems. *Journal of Scheduling*, 15(1):13–21, 2012.
- Equipment-Trader. Lifting-Equipment/Forklifts-For-Sale, 2016.
<http://www.equipmenttraderonline.com/Forklifts-For-Sale/search-results?category=Forklifts>.
- F. Curties Barry & Company. Operational Assessment Will Identify How to Better Utilize Your Space, 2016. <https://www.fcbsco.com/blog/operational-assessments-identify-how-to-better-utilize-your-space> [12 January 2018].
- Franz Faul, Edgar Erdfelder, Albert-Georg Lang, and Axel Buchner. G*power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods*, 39(2):175–191, May 2007.
- William Feller. The Fundamental Limit Theorems in Probability. *Bulletin of the American Mathematical Society*, 51(11):800–832, 1945.
- Barbara B. Flynn, Xenophon Koufteros, and Guanyi Lu. On Theory in Supply Chain Uncertainty and its Implications for Supply Chain Integration. *Journal of Supply Chain Management*, 52(3):3–27, 2016.
- Gary Forger. On demand warehousing prepares for takeoff, 2018. Modern Material Handling https://www.mmh.com/archive/magissue/mmh_december_2018 [16 January 2018].
- L.F. Francis, R.L. and McGinnis Jr. and J.A. White. *Facility Layout and Location: An Analytical Approach*. Englewood Cliffs: Prentice-Hall, 1992.
- Coffman E. G., Lueker G. S., and Rinnooy Kan A. H. G. Asymptotic Methods in the Probabilistic Analysis of Sequencing and Packing Heuristics. *Management Science*, 34(3):266–290, March 1988. ISSN 0025-1909.

- Mario C Vélez Gallego, Diego A Valencia Ramírez, and Carlos A Castro Zuluaga. A Simulation-Optimization Heuristic for Configuring a Selective Pallet Rack System. *Ingeniare: Revista Chilena de Ingeniería*, 20(1):17, 2012.
- Rita Gamberini, Andrea Grassi, Cristina Mora, and Bianca Rimini. An Innovative Approach for Optimizing Warehouse Capacity Utilization. *International Journal of Logistics Research and Applications: A Leading Journal of Supply Chain Management*, 11(2), 2008.
- Jesús García-Arca, J Carlos Prado-Prado, and a Trinidad Gonzalez-Portela Garrido. Packaging Logistics: Promoting Sustainable Efficiency in Supply Chains. *International Journal of Physical Distribution & Logistics Management*, 44(4): 325–346, 2014.
- Alan Genz and Frank Bretz. *Computation of multivariate normal and t probabilities*, volume 195. Springer Science & Business Media, 2009.
- Javad Ghaderi, Yuan Zhong, and R. Srikant. Asymptotic optimality of best fit for stochastic bin packing. *ACM SIGMETRICS Performance Evaluation Review*, 42(2):64–66, September 2014. ISSN 0163-5999.
- Glassdoor. Salaries: Forklift Operators, 2017.
<https://www.glassdoor.com/Salaries/index.htm>.
- Marc Goetschalckx. Storage Systems and Policies. In Riccardo Manzini, editor, *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*, pages 31–51. Springer London, London, 2012.
- Marc Goetschalckx and H Donald Ratliff. Shared Storage Policies Based on the Duration Stay of Unit Loads. *Management Science*, 36(9):1120–1132, 1990. ISSN 0025-1909.

- Daniel H Greene and Donald E Knuth. *Mathematics for the Analysis of Algorithms*. Springer Science & Business Media, 2007.
- Jinxiang Gu, Marc Goetschalckx, and Leon F. McGinnis. Research on Warehouse Design and Performance Evaluation: A Comprehensive Review. *European Journal of Operational Research*, 203(3):539 – 549, 2010.
- K. R. Gue and R. D. Meller. A constructive aisle design model for unit-load warehouses with multiple pickup and deposit points. *European Journal of Operational Research*, 236(1), 2014. ISSN 03772217.
- K. R. Gue, K. Furmans, Z. Seibold, and O. Uludağ. GridStore: A Puzzle-Based Storage System With Decentralized Control. *IEEE Transactions on Automation Science and Engineering*, 11(2):429–438, April 2014. ISSN 1545-5955.
- Kevin R Gue. Very High Density Storage Systems. *IIE Transactions*, 38(1):79–90, 2006.
- Kevin R Gue and Byung Soo Kim. Puzzle-based storage systems. *Naval Research Logistics (NRL)*, 54(5):556–567, 2007.
- Kevin R Gue, Russell D Meller, and Joseph D Skufca. The Effects of Pick Density on Order Picking Areas with Narrow Aisles. *IIE transactions*, 38(10):859–868, 2006.
- Silviu Guiasu and Abe Shenitzer. The Principle of Maximum entropy. *The Mathematical Intelligencer*, 7(1):42–48, 1985.
- S. Guo and T. Liu. An evaluation of storage assignment policies for twin shuttle as/rs. In *2010 IEEE International Conference on Management of Innovation Technology*, pages 197–202, 2010.
- Varun Gupta and Ana Radovanovic. Lagrangian-based online stochastic bin packing. *SIGMETRICS Perform. Eval. Rev.*, 43(1):467–468, June 2015.

- Gurobi Optimization, Inc. Gurobi optimizer reference manual, 2017.
 =<http://www.gurobi.com> [12 April 2018].
- M. Hähnel, J. Martinovic, G. Scheithauer, A. Fischer, A. Schill, and W. Dargie.
 Extending the Cutting Stock Problem for Consolidating Services with Stochastic Workloads. *IEEE Transactions on Parallel and Distributed Systems*, 29(11): 2478–2488, 2018. ISSN 10459219.
- Moncer A. Hariga. Inventory models for multi-warehouse systems under fixed and flexible space leasing contracts. *Computers & Industrial Engineering*, 61(3):744 – 751, 2011.
- Warren H. Hausman, Leroy B. Schwarz, and Stephen C. Graves. Optimal storage assignment in automatic warehousing systems. *Management Science*, 22(6): 629–638, 1976.
- J.L. Heskett. Cube-per-order index - a key to warehouse stock location. *Transportation and Distribution Management*, 3:27–31, 1963.
- Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research, 9th Ed.* McGraw-Hill Higher Education, 2009.
- HK Systems, Inc. Determining Your Optimal Storage Density. *An educational series developed by HK Systems - Maximum Experience, Minimal risk*, 8(1):1–2, 2016.
 URL <http://www.mhi.org/media/members/14202/128589775518881716.pdf>.
- Ulrich Hoffmann. A class of simple stochastic online bin packing algorithms. *Computing*, 29(3):227–239, 1982.
- Lajos Horváth, Piotr Kokoszka, and Gregory Rice. Testing stationarity of functional time series. *Journal of Econometrics*, 179(1):66 – 82, 2014. ISSN 0304-4076.
- Weiping Huang, Zhili Zhou, and Qiyuan Sun. *Throughput Analysis of an Automated Warehouse with Pallet Shuttle*, pages 1509–1514. ASCE Library, 2010.

- MS Hung and John C Fisk. Economic Sizing of Warehouses: A Linear Programming Approach. *Computers & Operations Research*, 11(1):13–18, 1984.
- Hyster-Company. H135-155FT Series, 2012.
<http://www.hyster.com/uploadedFiles/Hyster/PDFs/North-America/Products/H135-155FT-BTG.pdf>.
- Inventory Operations Consulting LLC. The aisle width decision, 2015.
<http://www.inventoryops.com/Aisle%20Width.htm> [14 June 2018].
- Dario Izzo. Pygmo and Pykep: Open source tools for massively parallel optimization in astrodynamics (the case of interplanetary trajectory optimization). In *Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT*, 2012.
- David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272 – 314, 1974. ISSN 0022-0000.
- S.G. Johnson. NLOPT, 2017. <http://ab-initio.mit.edu/nlopt> [20 September 2017].
- Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. <http://www.scipy.org/> [15 Jun 2018].
- P. Kall and J. Mayer. *Stochastic Linear Programming. Models, Theory, and Computation*, volume 156 of *International Series in Operations Research & Management Science*. Springer US, 2 edition, 2011.
- Shahin Kamali and Alejandro López-Ortiz. All-around near-optimal solutions for the online bin packing problem. In *International Symposium on Algorithms and Computation*, pages 727–739. Springer, 2015.
- Leonid Vitalevich Kantorovich. Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422, 1960.

- N. Karmarkar. Probabilistic analysis of some bin-packing problems. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 107–111, Nov 1982.
- Michael G. Kay. Material handling equipment (MHE) taxonomy, 1999.
<http://www.mhi.org/cicmhe/resources/taxonomy> [12 Mar 2018].
- Leonard Kleinrock. *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, New York, NY, USA, 1975.
- Sébastien Le Digabel. Algorithm 909: NOMAD Nonlinear Optimization with the MADS Algorithm. *ACM Transactions on Mathematical Software*, 37(4): 44:1–44:15, February 2011.
- C. C. Lee and D. T. Lee. A simple on-line bin-packing algorithm. *J. ACM*, 32(3): 562–572, July 1985. ISSN 0004-5411.
- 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.
- Young Hae Lee, J.M.A. Tanchoco, and Sung Jin. Performance Estimation Models for AS/RS with Unequal Sized Cells. *International Journal of Production Economics*, 37(18):4197–4216, 1999.
- Young Hae Lee, Moon Hwan Lee, and Sun Hur. Optimal Design of Rack Structure with Modular Cell in AS/RS. *International Journal of Production Economics*, 98(2):172–178, 2005.
- H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126(3):649–673, 1987.

- Mattias Liefvendahl and Rafał Stocki. A study on algorithms for optimization of latin hypercubes. *Journal of statistical planning and inference*, 136(9):3231–3247, 2006.
- Yen-Hung Lin, Russell D. Meller, Kimberly P. Ellis, Lisa M. Thomas, and Barbara J. Lombardi. A Decomposition-based Approach for the Selection of Standardized Modular Containers. *International Journal of Production Research*, 52(15):4660–4672, 2014.
- Loopnet. Warehouse Industrial for Lease, 2016.
<http://www.loopnet.com/Listing/17467286/804-E-Park-Avenue-Libertyville-IL/>.
- Reinier Maat and Abhishek Malali. A Multipurpose Library for Synthetic Time Series in Python, 2017. <https://github.com/TimeSynth/TimeSynth> [10 February 2018].
- J Mayer. On the Numerical Solution of Jointly Chance Constrained Problems. In *Probabilistic Constrained Optimization*, pages 220–235. Springer, 2000.
- Mike Futch. The Dawn of the Automated Warehouse, 2018. <https://www.mhlnews.com/technology-automation/dawn-automated-warehouse> [10 February 2019].
- Inc. Minitab. Minitab 17 statistical software [computer software], 2010. URL www.minitab.com.
- Neeraj Misra, Harshinder Singh, and Eugene Demchuk. Estimation of the entropy of a multivariate normal distribution. *Journal of Multivariate Analysis*, 92(2):324 – 342, 2005.
- Martin Moore and Steven Hancock. File Systems. In Martin Moore and Steven Hancock, editors, *Tru64 UNIX Troubleshooting*, HP Technologies, pages 227 – 249. Digital Press, Burlington, 2003.

- Louis Joel Mordell. *Diophantine equations*, volume 30. Academic Press, 1969.
- David E Mulcahy. Order-Fulfilment Objectives and Their Impact on Company Profit and Customer Service. In *Eaches or Pieces Order Fulfillment, Design, and Operations Handbook*, chapter 1, pages 1–60. CRC Press, 2007.
- Hidetoshi Murakami. A Saddlepoint Approximation to the Distribution of the Sum of Independent Non-Identically Uniform Random Variables. *Statistica Neerlandica*, 68(4):267–275, 2014.
- Occupational Safety and Health Administration. *Powered Industrial Trucks*. United States Department of Labor, 2008. URL <https://www.osha.gov/SLTC/etools/pit/workplacehazards/narrowaisles.html>.
- ODTH First Class Logistics. The magic black box, 2017. Available at <http://odth.be/eng/eng-black-box/> [20 Febraury 2019].
- Edwin G Olds. A Note on the Convolution of Uniform Distributions. *The Annals of Mathematical Statistics*, 23(2):282–285, 1952.
- Travis Oliphant. NumPy: A guide to NumPy. USA: Trelgol Publishing, 2006. <http://www.numpy.org/> [March 10, 2019].
- B.-C. Park and M.-K. Lee. Closest Open Location Rule Under Stochastic Demand. *International Journal of Production Research*, 45(7):1695–1705, 2007.
- W. H. Payne and F. M. Ives. Combination generators. *ACM Transactions on Mathematical Software*, 5(2):163–172, June 1979. ISSN 0098-3500.
- Heinrich Potuschak and Werner G. Müller. More on the Distribution of the Sum of Uniform Random Variables. *Statistical Papers*, 50(1):177–183, 2009. ISSN 1613-9798.

- András Prékopa. Bounding and Approximation of Probabilities. In *Stochastic Programming*, pages 179–217. Springer, 1995.
- Sacramento Quintanilla, Ángeles Pérez, Francisco Ballestín, and Pilar Lino. Heuristic algorithms for a storage location assignment problem in a chaotic warehouse. *Engineering Optimization*, 47(10):1405–1422, 2015.
- Kalpana Rajagopal. Simulation of online bin packing in practice. *UNLV Theses, Dissertations, Professional Papers, and Capstones*, page 67, 2016. Available at <https://digitalscholarship.unlv.edu/thesesdissertations/2894>.
- Parthasarathy Ramachandran. Use of extended euclidean algorithm in solving a system of linear diophantine equations with bounded variables. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Algorithmic Number Theory*, pages 182–192, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- AK Rao and MR Rao. Solution Procedures for Sizing of Warehouses. *European Journal of Operational Research*, 108(1):16–25, 1998.
- Uday S Rao. Properties of the periodic review (r, t) inventory control policy for stationary, stochastic demand. *Manufacturing & Service Operations Management*, 5(1):37–53, 2003.
- H. Donald Ratliff and Marc Goetschalckx. Shared Storage Policies Based on the Duration Stay of Unit Loads. *Management Science*, 36(9):1120–1132, 1990.
- Patrick J. Reilly, Jennifer A. Pazour, and Kellie R. Schneider. Propagation of unit location uncertainty in dense storage environments. *International Journal of Production Research*, 55(18):5435–5449, 2017.
- Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Philosophical*

- Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110550, 2013.
- Y. Roll, M.J. Rosenblatt, and D. Kadosh. Determining the Size of a Warehouse Container. *The International Journal of Production Research*, 27(10):1693–1704, 1989.
- Mazen Saghir. The concept of packaging logistics. In *Proceedings of the Fifteenth Annual POMS Conference, Cancun, April, 2004*.
- Hüseyin Sarper and Nebojsa I. Jaksic. Evaluation of procurement scenarios in one-dimensional cutting stock problem with a random demand mix. *Procedia Manufacturing*, 17:827–834, 2018. ISSN 23519789.
- D. Scull. A Stochastic Cutting Stock Procedure: Cutting Rolls of Insulating Tape. *Management Science*, 27(8):946–952, 1981.
- Skipper Seabold and Josef Perktold. Statsmodels: Econometric and Statistical Modeling with Python. In *9th Python in Science Conference*, 2010.
- Steven S Seiden. On the online bin packing problem. *Journal of the ACM (JACM)*, 49(5):640–671, 2002.
- Jiří Sgall. Online bin packing: Old algorithms and new results. In Arnold Beckmann, Erzsébet Csuhaj-Varjú, and Klaus Meer, editors, *Language, Life, Limits: 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings*, pages 362–372. Springer International Publishing, Cham, 2014.
- P. W. Shor. How to pack better than best fit: tight bounds for average-case online bin packing. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 752–759, Oct 1991.

- Peter W. Shor. The average-case analysis of some on-line algorithms for bin packing. *Combinatorica*, 6(2):179–200, Jun 1986.
- T. Simpson, L. Dennis, and W. Chen. Sampling Strategies for Computer Experiments: Design and Analysis. *International Journal of Reliability and Application*, 2:209–240, 2002.
- Frank Spitzer. *Principles of Random Walk*. Springer New York, 1970.
- Stanley Vidmar. STAK Adjustable Racking System, 2017.
<http://www.stanleyvidmar.com/products/adjustable-racking/stak-system>.
- Saad M.A. Suliman. Pattern generating procedure for the cutting stock problem. *International Journal of Production Economics*, 74(1):293 – 301, 2001. ISSN 0925-5273.
- Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, pages 555–573, 2002.
- Wojciech Szpankowski. *Average case analysis of algorithms on sequences*, volume 50. John Wiley & Sons, 2011.
- J. M. A. Tanchoco. Unit load design - a key element in integrating material movement and storage. In *Presented at the MHI Advanced Institute for Material Handling Teachers*, June 1983.
- Elena Tappia, Debjit Roy, Marco Melacini, and René De Koster. Integrated storage-order picking systems: Technology, performance models, and design insights. *European Journal of Operational Research*, 274(3):947 – 965, 2019.
- The MathWorks, Inc. MATLAB and Optimization Toolbox Release 2016b, 2016.
<https://www.mathworks.com/help/stats/mvncdf.html>.

- Karen E. Thuermer. State of Industrial Real Estate: Its crunch time, 2018.
https://www.logisticsmgmt.com/article/state_of_industrial_real_estate_its_crunch_time [1 Feb 2019].
- TOMLAB. TOMLAB /SNOPT toolbox, 2017.
http://tomopt.com/docs/TOMLAB_SNOPT.pdf.
- James A. Tompkins, John A. White, Yavuz A. Bozer, and J. M .A. Tanchoco. *Facilities Planning*. John Wiley & Sons, 2010.
- Jaime Trevino and Juan J. Daboub. *Computer Aided Design of Unit Loads: A Design and Selection Procedure*, pages 143–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 1991.
- Unarco Material Handling, Inc. Automated storage & retrieval systems details, 2010. <http://www.unarcorack.com/asrs-systems-details/> [6 Jul 2018].
- A. Wächter and L. T. Biegler. On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006.
- R. Waeber, P. Frazier, and S. Henderson. Bisection search with noisy responses. *SIAM Journal on Control and Optimization*, 51(3):2261–2279, 2013.
- Yugang Yu and René BM De Koster. Sequencing heuristics for storing and retrieving unit loads in 3d compact automated warehousing systems. *IIE Transactions*, 44(2):69–87, 2012.
- Yiu-Fai Yung and Peter M Bentler. Bootstrap-corrected ADF test statistics in covariance structure analysis. *British Journal of Mathematical and Statistical Psychology*, 47(1):63–84, 1994.

Nima Zaerpour, Yugang Yu, and René de Koster. Small is beautiful: A framework for evaluating and optimizing live-cube compact storage systems. *Transportation Science*, 51(1):34–51, 2017a.

Nima Zaerpour, Yugang Yu, and René B. M. de Koster. Response time analysis of a live-cube compact storage system with two storage classes. *IIE Transactions*, 49(5):461–480, 2017b.

Nima Zaerpour, Yugang Yu, and René B.M. de Koster. Optimal two-class-based storage in a live-cube compact storage system. *IIE Transactions*, 49(7):653–668, 2017c.

CURRICULUM VITAE

NAME: Luis Felipe Cardona Olarte

ADDRESS: Department of Industrial Engineering
University of Louisville
Louisville, KY, 40292

EDUCATION: B.S. Industrial Engineering
Universidad Icesi, Colombia
2011
M.Eng. Industrial Engineering
Universidad Icesi, Colombia
2013

PREVIOUS

RESEARCH: Warehousing

TEACHING: Facility Design

AWARDS: Fulbright-Colciencias Colombia Scholarship