CIRRELT
1971 – 2021

CIRRELT-2021-22

Estimating Optimal ABC Zone Sizes in Manual Warehouses

Allyson Silva
Kees Jan Roodbergen
Leandro C. Coelho
Maryam Darvish

May 2021

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2021-007.

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

Estimating Optimal ABC Zone Sizes in Manual Warehouses

Allyson Silva[1], Kees Jan Roodbergen[2], Leandro C. Coelho[1,2,3], Maryam Darvish[1]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, Québec, Canada
2. Department of Operations, Faculty of Economics and Business, University of Groningen, the Netherlands
3. Canada Research Chair in Integrated Logistics, Université Laval, Québec, Canada

Abstract. The ABC storage policy is the most popular method used to determine storage location assignment in warehouses. It consists of dividing the storage area into three zones, and the products to be stored into three classes, such that the most demanded class of products is assigned to the best-located storage zone. Despite the method's popularity, zone sizes are still majorly determined by simple rules-of-thumb, which can lead to major efficiency losses in many common warehouse settings. In this work, we investigate how several factors, such as the warehouse layout, the pick list characteristics, and the policies used to decide the way zones are arranged and to create the routes, can impact the optimal solution for the zone-sizing problem in a multi-block warehouse. We simulate different warehouse settings to obtain the best zone sizes using an extensive grid search and analyzing the data. We show that the best sizes found vary significantly under different input factors. Since extensive simulations are too costly to be done in practice, we train four different machine learning techniques – ordinary least squares, decision tree, random forest, and multilayer perceptron – to predict the optimal zone sizes from the data generated in the simulations. The results show a trade-off between the attributes considered when deciding which model to use, notably related to their applicability and performance. Regardless of the choice made, the use of any of these models leads to a significant improvement in order-picking efficiency, both in the average and worst-case scenarios, when compared to the zone sizes commonly used in practice, including the one-zone system (random policy) and the two-zone system, where classes are split using the 20/80 rule.

Keywords: Storage location, zone sizing, order picking, warehouse, machine learning computing.

Acknowledgements. The authors thank Compute Canada for providing high-performance parallel computing facilities. This work was partly supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2019-00094 and 2020-00401. This support is gratefully acknowledged.

---

* Corresponding author: allyson.fernandes-da-costa-silva.1@ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2021

1. Introduction

When products from a supplier arrive at a warehouse, they are often temporarily stored in a storage area, for example, in pallet racks or on shelves. The products reside in this storage area until they are retrieved in response to customers' orders. Methods for *storage location assignment* decide which location of the storage area to assign to which product [42]. Typically, a warehouse receives products from a supplier in bulk (for example, a full pallet), while shipments to customers are in significantly smaller quantities (for example, in cardboard boxes). Hence, the retrieval of products from storage, called *order-picking*, tends to be much more labor-intensive than their storage. Generally, order-picking is the most time-consuming activity of the entire warehouse [14]. Furthermore, the order-picking process is often time critical, for example, due to strict deadlines for meeting same-day or next-day delivery requirements in the e-commerce [4]. It is for these reasons that the design of good policies for storage location assignment is vital for achieving an efficient order-picking process in warehouses [9, 38].

If the storage and order-picking processes are performed by workers who walk or drive between locations in the storage area, this is commonly referred to as a *manual warehouse* or a picker-to-parts warehouse [9]. Of course, there is significant attention to automated warehouse systems in the literature [4]. However, manual warehouses can provide better flexibility for accommodating demand fluctuations, and have lower investment costs [6]. Flexibility and scalability are especially valuable in warehouses of online retailers [48]. These emphasize the need for continued process improvements for manual warehouses. The storage area of a typical manual warehouse has a rectangular shape consisting of parallel pick aisles. The area is divided into blocks and pickers use cross aisles between pairs of blocks to change aisles. Pick routes start and end at an I/O point located at one corner of the storage area. A graphical display of such layout is provided in Silva et al. [50, Figure 3].

Policies for storage location assignment match products with locations based on properties of both the products and the locations. The 'most important' products should be assigned to the 'best' locations, and the 'least important' products to the 'worst' locations. For deciding which products are 'most important', the criterion of order frequency is commonly used, i.e., how often a product occurs on a customer's order [e.g., 16, 21, 37]. However, order frequency is not stable over time, which may induce the need for repositioning products when their demand has changed [35]. To mitigate the need for repositioning, while simultaneously maintaining the advantages of assigning 'most important' products to 'best' locations, the concept of *class-based storage* is widely deployed in practice and studied in the

literature [e.g., 8, 28]. In the class-based storage, products are grouped into classes, based on order frequencies, and each class is subsequently assigned to a dedicated *zone* of the warehouse. Within a zone, the assignment is random. The class of the fastest moving products is generally called class A, the next fastest-moving class is called class B, and so on.

To implement class-based storage, three decision problems are to be addressed: (i) determining the number of classes, (ii) determining the size of each zone for each class, and (iii) determining the positioning of each zone in the layout. From the literature on decision problem (i), it is known that additional classes tend to increase efficiency, albeit at a decreasing rate [10]. Since having a large number of classes defeats the purpose of hedging against demand frequency changes, a common compromise between the two objectives is to use three classes [see, e.g., 17, 43, 27], which is often referred to as *ABC storage*. Also decision problem (iii) is studied extensively [see, e.g., 8, 20, 38]. In contrast, decision problem (ii) has received only little attention in the literature. In Section 2, we review relevant literature in more detail. Note that methods for addressing decision problem (iii) are often referred to as *storage policies* [37, 50]. Even though this wording may appear more encompassing than it actually is, we adhere to this convention.

In this paper, we develop and compare several methods for determining zone sizes for class-based ABC storage in manual warehouses. The objective is to minimize the average order-picking time. The first challenge in addressing this problem is that the best zone sizes depend on a multitude of factors, including the warehouse layout, and other operating policies that impact the process. Notably, relevant operating policies to consider are storage policies (how the zones are positioned in the layout) and routing policies (how a route through the warehouse is determined to retrieve a customer's order). The second challenge is that there are no closed-form mathematical expressions to determine average order-picking time for every possible configuration, which leaves only simulation as a potential method for a consistent performance analysis. The third challenge is that simulation requires relatively high computation times, which makes it intractable to compare very large number of solutions.

Our approach is to deploy four machine learning (ML) regression methods to estimate zone sizes when using the class-based storage policy. Specifically, we develop models using ordinary least squares, decision tree, random forest, and multilayer perceptron. As independent variables, we use five layout properties, demand frequency of products, the number of products in customers' orders, four storage policies, and four routing policies. As dependent variables, we use the best zone sizes for each considered configuration of the independent variables. We determine these best zone sizes by obtaining

the average order-picking time through a simulation. For every combination of zone sizes and every configuration of independent variables, we subsequently select the zone sizes with the lowest average order-picking. For this purpose a tailor-made simulation model is implemented. The number of possible configurations is infinite, therefore we used as input for our models a limited set of configurations, randomly selected from warehouse settings commonly found in practice. We used 16,000 configurations as input data for our methods, for each of which we simulated 4350 zone sizing options to determine the best zone sizes. In total the calculation time amounted to about 12,000 CPU hours. By deploying ML techniques on this data set, we aim at developing models that can quickly and easily determine proper zone sizes, without future need for building a simulation model and running it for thousands of hours.

Computational experiments attest the effectiveness of our methods by a comparison against the most common zone sizes approaches from practice, including the use of a random policy, a two-zone policy, and a classical 20/30/50 policy. We show that there is a trade-off between simplicity and performance among the ML models used, so that the most adequate one depends on the conditions available during the decision making process. This paper also analyses and quantifies the impact of different features on the predicted zone sizes. This can help better understand the zone sizing problem and interpret the solutions.

The remainder of this paper is structured as follows. Section 2 gives an overview of related literature. In Section 3, we describe our simulation model, including a detailed description of the four storage policies and four routing policies considered. Section 4 contains an overview of the four ML techniques used. Section 5 presents the results of the computational experiments performed, with an analysis of the best zone sizes found in the simulations and the importance of the features to the models, the parameters set, and a performance comparison between all models and the arbitrary zone sizes commonly used in practice. In Section 6, we present analysis of the results, including a qualitative analysis of each method attribute and how the methods perform for different policies. Finally, the concluding remarks of this paper are found in Section 7.

2. Literature review

Zone shapes and sizes are two problems usually investigated together in the class-based storage literature. For warehouses with single-command cycles, in which storage and retrieval operations are performed carrying a single load, found mainly in automated systems and unit-load warehouses, these

problems can be solved analytically. Hausman et al. [17] analyze the optimal partitioning point for a two-zone system in terms of the demand skewness, showing that the first zone should be smaller when demands are more skewed. For the ABC system, the best partitioning points are determined numerically using grid search. The results show that the best combination of classes A and B tends to be smaller when demands are more skewed and, consequently, class C tends to be larger. Eynan and Rosenblatt [13] extend the Hausman et al. [17] procedure to determine zone boundaries optimally for n classes. Rao and Adil [40] also develop analytical models to find optimal partitioning point in a two-zone system, but considering that storage location assignments are determined using a heuristic storage policy. Similar procedures are also presented for different systems, such as for a 3D compact automated system [56], a live-cube compact system [57], and a warehouse with diagonal cross aisles [3]. Still for single picks, Van den Berg [54] solves the zone sizing problem using a dynamic programming algorithm that simultaneously assigns locations and products to classes.

Different than for single-command cycles, there is no firm strategy on how to define class partitions when pickers are required to perform multiple picks in each picking route [9]. The boundaries established for the zones are usually determined in the warehousing literature from simulation experiments that compare the average route length (ARL) traveled by pickers under different warehouse settings. However, building and running simulation models are time-consuming and expensive for warehouses. Therefore, in practice or mainly due to its simplicity, the adoption of arbitrary zone sizes is a common practice. In this paper, we consider a setting where multiple picks are performed in each route.

For a specific case of a warehouse with a single-block layout, Petersen et al. [38] analyze how to set up the class-based policy for zones shaped following simple rules commonly used in practice in a system with two, three and four zones. A few different zone sizes are tested only for the two-zone case, which they concluded that either a 30/70 or 40/60 partition is the best option for the specific layout considered, depending on the number of picks to be done in the routes. For the three-zone case, they arbitrarily considered sizes as 20/30/50. In fact, for many studies, the size of each zone is an input parameter, not a decision variable [11, 24, 45, 51]. Many of them consider that demands follow a 20/80 curve (20% of products account for 80% of the total demand) and, therefore, define zones using the 20/80 partition for two classes or the 20/30/50 for three classes. Using these zone sizes for a skewed demand scenario in an ABC system, Le-Duc and De Koster [20] investigate zone shapes in a two-block warehouse with the I/O located in the head of the cross aisle that separates them. A mathematical model is developed to determine the partial length of each aisle used for storing

each class. They show that the shape of the optimal zones depends largely on the demand skewness, the number of picks in each route, the storage assignment policy and the warehouse length/width ratio. For the same two-block layout, Rao and Adil [39] develop analytical models to simultaneously determine the number of classes, class boundaries, pick list size and number of aisles. Dijkstra and Roodbergen [11] show that the optimal class boundaries in a single-block warehouse where pickers follow a routing policy and a single cross aisle should be used must be non-increasing as a function of the aisle number. Chan and Chan [8] present a case study of a single-block, multi-level rack warehouse with an ABC system where different combinations of storage policies to determine the zone shapes and routing policies are simulated for various pick densities. Roodbergen [43] provides insights on the interactions between layout, routing and storage policies in an ABC system, showing that the best zone sizes change according to the policies in use. No study attempted to develop tools to estimate optimal zone sizes precisely under different settings, including many more layouts, than those tested in the papers discussed without requiring to run their simulators again, which is one of the main contributions of this paper.

3. Multi-block warehouse characteristics and simulator

The routes followed by pickers to retrieve products demanded are influenced by several factors. Le-Duc and De Koster [20] and Petersen [37] summarize them to the layout of the warehouse, the demand pattern, the storage strategy, the batching method, and the routing method. Each of these features are discussed next.

3.1. Layout factors

In a standard multi-block warehouse, there are five layout factors to be considered: number of aisles, number of cross aisles, aisle length, aisle width, and cross aisle width.

The number of aisles influences travel distances since more aisles containing picks increase the distance to be travelled in a route. Longer aisles may also increase the route length since pickers spend more time travelling in a single aisle before a new one can be entered. Every warehouse has at least two cross aisles, one at the front and one at the back of the storage area. Additional cross aisles may reduce route length depending on the pick locations. Finally, wider aisles and cross aisles also increase the travel distance. We ignore the distance between the left and right side racks in an aisle and consider that pickers walk in the middle of aisles and cross aisles. The total storage capacity of the warehouse

is twice the number of aisles multiplied by the aisle length. This is a continuous representation of storage capacity, which generalizes the representation of capacity in terms of total number of slots.

3.2. Demand distribution

The only factor related to the demand is its distribution. Typically, relatively a few products account for the vast majority of the demand volume. Some related works [20, 38] consider that the demands are represented by a step function where a certain percentage of products account for a certain percentage of the total demand, usually 20% of products accounting for 80% of demand. Caron et al. [7] present an analytical function to describe this characteristic using a continuous function known as ABC curve. Namely,

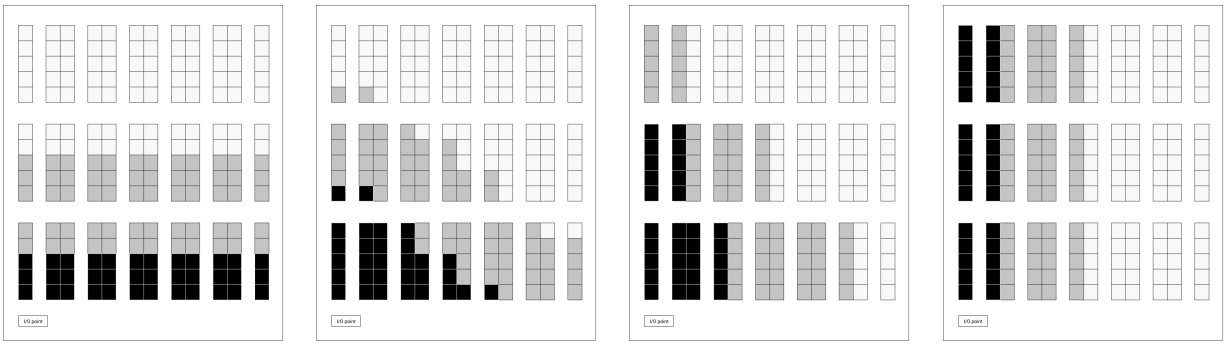
$$F(x) = \frac{(1+s)x}{s+x}, \quad 0 \leq x \leq 1, s \geq 0, s + x \neq 0, \tag{1}$$

where x indicates the zone size corresponding to the items whose order frequency represent a fraction $F(x)$ of total warehouse activity. The parameter s indicates the skewness of the demand. For example, for $s = 0.067$ it holds that 80% of the picks are generated by 20% of the products, reducing to 70% for $s = 0.12$, 60% for $s = 0.2$ and 50% for $s = 0.333$, which are common skewness values found in the step functions of related works.

3.3. Storage strategy

The storage strategy accounts for how to assign products within the storage area. In this work, we consider that an ABC storage policy is used. While our main objective is to estimate the optimal combination of zone sizes, the zone shapes are determined by following one of the four methods for the ABC storage policy shown in Figure 1. In the figure, different colors define the zones, with the class A being represented in black, class B in grey, and class C in white. *Across-aisle* locates the A products in the front-most locations of each pick aisle. *Nearest-location* ranks each storage location by their distance to the I/O point and, then, locates the A products in the closest locations. Nearest-location is closely related to the method of diagonal storage [37], which defines boundaries following a diagonal shape. For single product orders, nearest-location minimizes the expected ARL. *Nearest-subaisle* ranks subaisles according to the distance of their heads to the I/O point and, then, the A products are assigned to the closest subaisles. In case more than one zone is assigned to the same subaisle, the products of the best class are assigned to the best locations within this subaisle first. Finally, *within-aisle* ranks aisles according to their distance to the I/O point and assigns the

A products to the best aisles. Different studies show that each of these policies perform well under different warehouse settings and all of them can actually be found in real cases [19, 20, 37, 43].



(a) Across-aisle **(b)** Nearest-location **(c)** Nearest-subaisle **(d)** Within-aisle

Figure 1: Storage policies

3.4. Batching policy

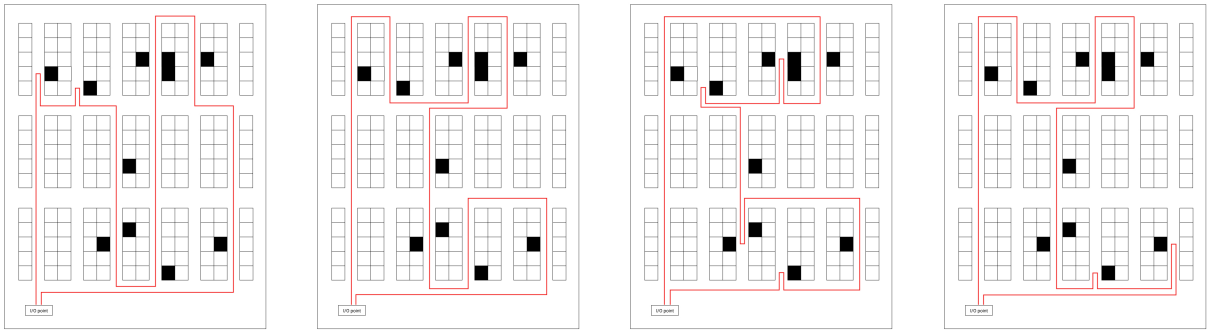
Batching accounts for how the demanded products are grouped to be retrieved by a single picking tour. Several batching policies exist [9]. In this work, we consider the pick-by-order batching policy, meaning that each order is picked individually. This is a common strategy when orders are fairly large. Thus, we measure *picks per route*, i.e., the number of locations to visit in a single route, as another input factor that influences the ARL.

3.5. Routing policy

Given a pick list and the location of the products contained in it, the order-picking problem (OPP) consists in determining the route to be followed by the picker in order to retrieve them with the objective of minimizing the total distance traveled [33, 47]. Although exact methods can be found to solve this problem [33, 41, 46, 52], it is faster to generate routes using simple heuristic policies, which are also more likely to be accepted by the pickers since they are more intuitive [15].

We consider the four routing policies shown in Figure 2. *Aisle-by-aisle* considers that each aisle containing at least one pick is visited once. The best cross-aisle to move to the next aisle is determined using dynamic programming. In *S-shape*, pickers traverse the left-most aisle that contains picks to the back of the warehouse and then return to the front picking products one block at a time. Any subaisle containing a pick is traversed. After the last pick in a block, the picker returns to the front of that block and continues to the next. *Largest gap* also starts with the picker moving to the back

of the warehouse, then picking products one block at a time. Whenever a subaisle contains a pick, instead of entirely crossing it, the picker avoids the "largest gap" and returns to leave it from the same side entered. A gap represents the distance between two adjacent picks, or between the middle of a cross aisle and the nearest pick. The last subaisle of a block is traversed entirely to allow the picker to enter the subaisles from the other side of the block. The last considered policy is the *combined*. It follows the same logic of performing all picks from the back of the warehouse to the front, block by block. However, whenever a subaisle is entered, the picker can choose between traversing it or making a return when all picks within it are done. The choice is made using dynamic programming by always looking one subaisle ahead in order to be in a better starting point for the next subaisle. A detailed description of these policies is found in Roodbergen and De Koster [44].



(a) Aisle-by-aisle **(b)** S-shape **(c)** Largest gap **(d)** Combined

Figure 2: Routing policies

3.6. Warehouse simulator

Given the five layout factors, the skewness parameter, the number of picks per route, the combination of storage and routing (S/R) policies, and the zone sizes, we use a simulator to estimate the expected ARL. Although an extensive literature exists on route length estimation using analytical formulas [2, 18, 20, 32, 34], they are not available for all situations, notably for most multi-block layouts. Also, simulators can provide the exact solutions that these methods are estimating.

In the simulator, the warehouse layout is created and the zones are established, considering the storage policy used and the zone sizes. Then, pick lists are sampled using the ABC curve. They basically contain in which zones each pick will be performed. Then, a random location within that zone is chosen for each pick. The simulator computes the route traveled by the picker to retrieve all products demanded following the routing policy used. This procedure is repeated several times. Each time the

route length is found for a new pick list sampled, the estimated ARL and its two-sided confidence interval C are updated as:

$$C_{\alpha/2} = \bar{x} \pm z_{\alpha/2}\frac{s}{\sqrt{n}}, \tag{2}$$

where $\bar{x}$ is the estimated ARL, s is the standard deviation of the route lengths estimated from the n pick lists sampled, and $z_{\alpha/2}$ is a value that follows from the cumulative normal distribution function for a chosen degree of certainty $(1 - \alpha/2)$. The simulations stop when the maximum accepted half-width ϵ is attained, i.e., when (3) holds.

$$\epsilon \geq z_{\alpha/2}\frac{s}{\bar{x}\sqrt{n}} \tag{3}$$

4. Machine learning regression methods

In the context of regression analysis, ML methods are used to predict a set of continuous output values (targets) from a set of input variables (features). These methods are trained using real or synthetic data in order to find a model that best fits the dataset that they are "learning" by finding the correlations between features and targets.

The use of ML regression methods in the warehousing literature is very limited, with most applications found in demand forecasting [31, 49], but also in rental prices estimation [23], forklifts engagement predictions [26], and the development of a dynamic routing system for automated guided vehicles [30].

We test several ML regression methods to estimate the ABC zone sizes that minimize the ARL to perform picks. The objective is to derive a model that learns the unknown function used by the warehouse simulator, previously described, that indirectly leads to the optimal zone sizes through the minimization of the ARL. If we were fitting a model to predict the route length of a single route, the ML methods would be learning the set of instructions defined by the routing policy used. To predict the ARL, it is reasonable to think that the unknown function is also dependent on the storage policy used. This means that 16 different functions are to be learned considering the combinations of the four storage strategies (S) of Figure 1 combined with the four routing policies (R) of Figure 2. For this reason, each of the ML methods used are trained for each combination of S/R policies separately. The dataset used to train them is generated using the simulator considering its inputs (see Section 5.2.2) as features.

The ML regression methods used in this study are: ordinary least squares (OLS), regression tree (RT), random forest (RF), and multilayer perceptron (MLP).

In summary, OLS constructs linear functions combining the input features to predict each zone size [55]. RT is a multi output model composed of several binary decisions performed in a tree-like structure that is able to capture nonlinear relationships between the features and the targets [22]. RF is an ensemble method where multiple RTs are randomly created to perform independent predictions, then all predictions are averaged, thus increasing the model robustness [5]. Finally, MLP is a feedforward deep neural network, consisting of an input layer that receives problem features, one or more hidden layers where the values of these features are transformed using nonlinear functions, and an output layer that gives the predicted target values [1]. These methods represent four of the most commonly used categories of supervised learning methods for regression analysis, i.e., linear regression, decision trees, ensemble methods, and neural networks. Other methods were also tested in a preliminary round of experiments, such as ridge and lasso regressions, AdaBoost, gradient boosting, and support vector machines, but only the most promising ones were selected.

4.1. Ordinary least squares

OLS is one of the most popular statistical techniques used for regression analysis. It fits a linear function using the values of the input features to predict the targets. The linear model is such that the sum of the squared deviation is minimized.

Linear regression models assume that target values are unbounded, i.e., they can assume any real number. In the case of ABC zone sizes, they are bounded by zero and the maximum number of storage locations in a warehouse. Multivariate fractional regression models [29] can be an alternative to use in this situation, however, these models are more appropriate when there are many observations at the upper and/or lower bounds. For ABC zone sizing, it is known that optimal zone sizes lie not too close to the bounds of the unit interval. Otherwise, it would be more advantageous to reduce the number of classes in the solution. In this case, we ignore the bounded nature of the targets and use a regular OLS regression.

Since OLS supports the prediction of only one target, for the ABC zone sizing, three linear functions are modeled, one for each zone size. Advantages of OLS compared to the other models used here are that it provides simple functions that are easy to apply in real cases and to interpret, since they show which features contribute the most to the predicted values. However, OLS usually does not perform well when the relationships between features and targets are nonlinear.

4.2. Regression tree

RT is a non-parametric model that can be used for regression analysis. It is based on a hierarchical decision scheme using a tree like structure. The tree contains a root node containing all data, a set of internal nodes and a set of terminal nodes (leaves). At each node, a binary decision is made using a condition associated to one of the input features, until a leaf node is reached containing a combination of the estimated target values [22].

Construction of an RT starts from the root node, where its prediction is made based on the target values of the training samples. Then, it searches over all features for a split that minimizes the sum of squared errors for the two new nodes generated. This process continues for each new node generated until a user-defined stopping criteria is reached. An RT is easy to understand and to interpret, since it brakes complex decisions into several simpler ones, hence it can be visualized. Moreover, it can be used to deal with complex nonlinear relationships. More on regression trees is found in Loh [22].

Since the tree is built from training samples, it may not generalize the data well, which results in a low accuracy when applied to unseen data (overfitting). Commonly, pruning is used to reduce overfitting. Methods used to stop early the tree building process are known as pre-pruning [12]. Pre-pruning avoids growing an overly complex tree. The pre-pruning technique considered here determines a minimum number of training samples (*min samples leaf*) required to be at a leaf node. So, a node will only be split if it results in two other nodes containing at least this number of samples. In Section 5.4.1, we assess several choices for *min samples leaf* to obtain a robust RT to estimate optimal ABC zone sizes.

4.3. Random forest

RF [5] is a machine learning algorithm that belongs to the class of the ensemble methods. In ensemble learning, several, usually simple, base estimators are combined such that each estimator provides a prediction and, then, all predictions are combined, improving the robustness over single estimators. RF combines RTs such that each tree is built from a set of randomly sampled data with randomly sampled features with the same distribution for all trees in the forest (bootstrap sample). As the number of trees increases, the error for the forest tends to converge, decreasing the variance of the RF model, and reducing overfitting. Due to the randomness behind it, RF is robust with respect to outliers.

Two parameters are important when fitting an RF model: the number of trees and the maximum number of features. The number of trees in the ensemble should be as high as possible considering

the total training time. The maximum number of features represents the number of features chosen randomly at each tree node. When it is set low, trees become more complex and diverse. Otherwise, if set to be close to the number of features, the trees will tend to be very similar. In Section 5.4.2, we test different combinations of these parameters to set the RF model to estimate optimal ABC zone sizes.

4.4. Multilayer perceptron

Artificial neural networks are machine learning algorithms inspired by how the human brain processes information and are used to approximate complex functions including nonlinear relationships that depend on several features. MLP is an artificial neural network composed of a structured network containing one input layer, one or multiple hidden layers, and one output layer. The input layer consists of one node for each feature considered, while the output layer consists of one node for each target to be predicted. Each hidden layer is composed of a number of nodes where the values from the previous (input or hidden) layer are input, transformed using a nonlinear activation function, and sent to the next (hidden or output) layer in the forward direction (feedforward architecture). A loss function, usually minimizing the sum of squared errors in regression problems, is optimized in the output layer via backpropagation [1].

MLPs are scale sensitive; if one input feature has a larger range than another, and both have a similar variance, then the MLP has a tendency to be more sensitive to the larger one. Therefore, feature scaling is recommended when setting an MLP [1]. Standardization is a common way to scale features and is done as:

$$\hat{X} = \frac{X - \bar{X}}{\sigma}, \tag{4}$$

where X represents the feature samples for a feature with mean $\bar{X}$ and standard deviation σ, transforming all features to have a mean of zero and unit variance.

The MLP has been demonstrated to be a very powerful tool since it does not require a high level of abstraction about the data domain and it self-organizes its complexity by adding or removing neurons according to the available data or computational power [1]. Its hidden layers have a non-convex loss function, such that different initializations can lead to different validation accuracy. We detail in Section 5.4.3 how we set the number of hidden layers and the number of neurons.

4.5. Model evaluation

The objective of all models is to estimate the ABC zone sizes such that the ARL is minimized. Due to the long time required to run simulations with a high level of confidence, we do not use this metric directly when setting up the models. Instead, we use a common score metric for evaluating the model's performance: Mean Squared Error (MSE). The formula for MSE is shown in equation (5), where n is the sample size, Y_i is the i-th observed value for the feature and $\hat{Y}_i$ is the value predicted.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{5}$$

A model is considered to be better for a combination of parameters that result in a lower MSE. We show in Section 5 that this metric predicts well the model performance for the original objective of minimizing ARL.

5. Computational experiments

In this section, we use each of the four supervised learning models to predict optimal ABC zone sizes. Each model has its most relevant parameters set using the data generated in the simulations, and then the performances of each model with the best settings are compared.

The simulations were run on a parallel cluster of machines with an Intel Gold 6148 Skylake with 2.4 GHz at each node with up to 1,000 nodes allowed to be used in parallel. The ML models were implemented using the Python package Scikit-learn [36]. We demonstrate the easiness to set the models by performing all training and testing on a personal computer with a six-core Intel Core i5-9400 with 2.9 GHz.

5.1. Dataset generation

The dataset used to train our models was generated using the warehouse simulator described in Section 3 [43]. We generate 1000 samples to represent different warehouse settings. Each sample is generated using common values found in practice for the features passed to the simulator. The values of each feature were selected using a uniform distribution considering lower and upper bounds. All values generated are integer, except for the skewness. The bounds used are:

- Number of aisles = $[6, 18]$;

- Number of cross aisles = $[2, 6]$;
- Aisle length (in meters) = $[10, 30]$;
- Aisle width (in meters) = $[2, 6]$;
- Cross aisle width (in meters) = $[2, 6]$;
- Picks per route = $[5, 25]$;
- Skewness = $[0.05, 0.35]$.

The storage assignment was determined using across-aisle (Aa), nearest-location (Nl), nearest-subaisle (Ns), and within-aisle (Wa), and the routes were created using aisle-by-aisle (Aba), S-shape (Sh), largest gap (Lg), and combined (Co). Therefore, a total of 16,000 instances were generated.

For each instance, we estimate the ARL by performing a grid search simulating the combinations of $z_A = \{1\%, 2\%, \ldots, 75\%\}$, $z_B = \{1\%, 2\%, \ldots, 75\%\}$, and $z_C = 100\% - z_A - z_B$. This results in 4350 feasible combinations for each instance. Pick lists were generated until $\epsilon = 0.25\%$ for a 99% of degree of certainty. The result is a 5 GB data file containing the expected ARL for each combination of zone sizes and instances. Total simulation time took around 12,000 CPU hours, which was only possible to run due to the parallel cluster of machines available.

5.2. Data analysis

For all models, we split the 1000 sampled warehouses arbitrarily into 67% as a training set and 33% as a test set. For the training set, we filter the solutions that result in an ARL gap up to δ percent to the ARL of the best known solution (BKS) found in the grid search. For the simpler techniques (OLS and RT), we use $\delta = 0\%$, such that only the BKSs are used for training. For the "black-box" methods (RF and MLP), δ has to be determined, since more training data may allow the method to learn more about the unknown function inside the simulator.

For the test set, we only use the BKSs, since we want to compare the predicted values with the best ones.

5.2.1. Targets analysis

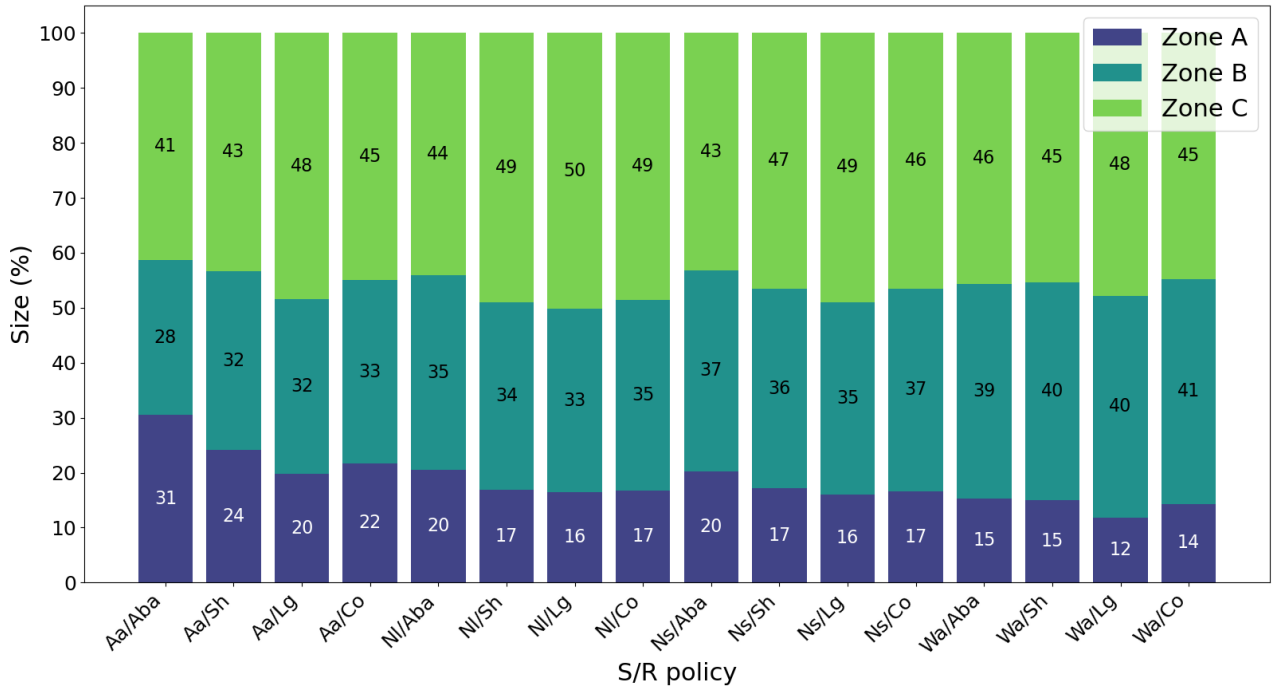
We first analyze the data obtained from the simulator regarding the best combinations of ABC zone sizes. A summary of this data is presented in Table 1. It shows that the overall average best zone

sizes found are $z_A = 18.33\%$, $z_B = 35.49\%$, and $z_C = 46.18\%$, which slightly deviates from the commonly used sizes of 20/30/50. The high standard deviation and the high range between minimum and maximum sizes observed indicate that the use of fixed zone sizes as a rule-of-thumb may lead to a solution numerically distant from the optimal one in several cases. Differences are more noticeable when analyzing specific S/R combinations, as shown in Figure 3. For instance, the overall optimal zone A size is more than 2.5 times higher when using across-aisle storage with aisle-by-aisle routing (Aa/Aba), than when using within-aisle with largest gap (Wa/Lg). We highlight that the figure conceals actual variation in performance within each bar. Later in this paper (see Figure 5), we detail the solutions found when using the average best zone sizes. We show that although the average performance is better than 20/30/50, the variation is still very high.

Table 1: Statistics for the best combinations of zone sizes found in the simulations

Attributes	Zone A size	Zone B size	Zone C size
Mean	18.33	35.49	46.18
Standard deviation	8.77	8.07	10.22
Minimum	2	1	4
Lower quartile	12	30	39
Median	17	35	46
Upper quartile	23	41	52
Maximum	55	62	81

Figure 3: Average best zone sizes per S/R policies found in the simulations



A large deviation from optimal zone sizes does not necessarily imply a large gap to the optimal ARL. Another metric used to justify the need for having accurate models is to observe how many combinations of zone sizes simulated are within a gap δ from the BKS. For $\delta = 0\%$, we have that the number of combinations that satisfy this condition is approximately one per sample, since ties rarely occur. Due to the uncertainty when estimating ARL using the simulator, defined at $\epsilon = 0.25\%$, we are interested in finding a combination that leads to an ARL below twice the value of ϵ from the best solution. If this is the case, we cannot reject the hypothesis that the solution found is equal to the BKS.

Some combinations of S/R policies are naturally easier to estimate good zone sizes since their performance do not change significantly when changing the solutions marginally. It is the case of Aa/Sh, which places high demanded products in as many aisles as possible with a routing policy that prefers to travel a few aisles entirely. This is a poor combination in practice since bringing high demanded products to the front of the aisles does not lead to a reduction in the number of aisles visited. The opposite is expected for Wa/Sh, in which placing a single high demanded product in a new aisle can significantly increase the ARL. This implies that a model to predict optimal zone sizes for the Aa/Sh policies is not required to be as accurate as a model for the Wa/Lg policies to generate good solutions. For this matter, different models are applied for each S/R policy combination.

5.2.2. Features analysis

Feature engineering is a process of using domain knowledge to create/extract new features from a given dataset [53]. We generated new features using nonlinear operations between the original ones and those meaningful in the warehousing context. They are:

- *Number of subaisles*: number of aisles / (number of cross aisles − 1)
- *Subaisle length*: aisle length / (number of cross aisles − 1)
- *Warehouse length*: aisle length + cross aisle width × number of cross aisles
- *Warehouse width*: number of aisles × aisle width
- *Dimensions ratio*: warehouse length / warehouse width
- *Picks per aisle*: picks per route / number of aisles
- *Picks per subaisle*: picks per route / number of subaisles

- *Pick density*: picks per route / (aisle length × number of aisles)

These eight "artificial" features and the seven original ones constitute the preliminary set of input features used in the models.

We performed univariate linear regressions between each of the 15 features and each of the three targets for each S/R policy. The obtained coefficient of determination (R^2) is presented in Table 2 with those above 0.1 highlighted. Observe that *skewness* (7) has the highest R^2 values in most policy combinations, which means that this feature shares a higher percentage of the variance with the targets than the others. When nearest-location or nearest-subaisle storage policies are used, *skewness* is almost always the only feature with a relatively high R^2. For the across-aisle policy – except when used with S-shape – *number of cross aisles* (2), *number of subaisles* (8), *subaisle length* (9) and *picks per subaisle* (14) are the most linearly correlated with the best zone sizes found. Note that all these features are related to each other since the three artificial ones directly or indirectly contain the number of cross aisles. Finally, when using within-aisle – except with largest gap – several features have a relatively high R^2. We highlight that none of the correlations observed are absolutely high (above 0.5), which is an indication that either the features are not correlated at all with the targets or they have a nonlinear correlation.

Table 2: R^2 for the univariate linear regression for the train set

S/R policy	*(1)*	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*	*(8)*	*(9)*	*(10)*	*(11)*	*(12)*	*(13)*	*(14)*	*(15)*
Aa/Aba	0.00	**0.25**	0.00	0.00	0.01	0.01	**0.20**	**0.14**	**0.25**	0.05	0.00	0.02	0.01	**0.10**	0.02
Aa/Sh	0.01	0.04	0.01	0.00	0.01	0.01	0.05	0.01	0.04	0.03	0.01	0.03	0.00	0.02	0.00
Aa/Lg	0.00	**0.33**	0.00	0.00	0.00	0.02	0.03	**0.19**	**0.37**	**0.10**	0.00	0.03	0.00	**0.22**	0.00
Aa/Co	0.01	**0.16**	0.00	0.01	0.01	0.01	0.08	0.08	**0.19**	0.04	0.01	0.03	0.00	**0.10**	0.00
Nl/Aba	0.03	0.04	0.03	0.05	0.01	0.00	**0.37**	0.03	0.04	0.01	0.08	0.08	0.01	0.02	0.01
Nl/Sh	0.00	0.02	0.02	0.02	0.00	0.00	**0.24**	0.01	0.02	0.02	0.03	0.01	0.00	0.04	0.01
Nl/Lg	0.02	0.02	0.03	0.05	0.00	0.01	**0.21**	0.03	0.06	0.00	0.08	0.03	0.01	0.04	0.00
Nl/Co	0.01	0.03	0.01	0.03	0.01	0.00	**0.23**	0.02	0.02	0.02	0.04	0.02	0.00	0.04	0.00
Ns/Aba	0.02	0.01	0.02	0.03	0.00	0.02	**0.29**	0.00	0.01	0.02	0.05	0.04	0.02	0.01	0.03
Ns/Sh	0.01	0.07	0.01	0.02	0.01	0.02	**0.14**	0.05	0.08	0.03	0.02	0.01	0.02	**0.13**	0.02
Ns/Lg	0.02	0.03	0.02	0.04	0.01	0.01	**0.23**	0.04	0.04	0.02	0.06	0.03	0.02	0.02	0.00
Ns/Co	0.01	0.02	0.01	0.01	0.01	0.01	**0.18**	0.02	0.03	0.02	0.01	0.00	0.01	0.06	0.02
Wa/Aba	0.07	0.07	0.01	0.06	0.01	0.06	**0.18**	0.06	0.05	0.06	**0.10**	**0.14**	0.06	**0.11**	0.04
Wa/Sh	0.01	**0.13**	0.01	0.03	0.00	0.04	**0.19**	**0.10**	**0.13**	0.05	0.03	0.03	0.03	**0.17**	0.03
Wa/Lg	0.00	0.02	0.01	0.03	0.01	0.05	**0.25**	0.01	0.03	0.01	0.03	0.02	0.01	0.01	0.02
Wa/Co	0.02	**0.10**	0.02	0.04	0.00	0.05	**0.21**	0.09	0.08	0.06	0.02	0.04	0.03	**0.11**	0.04

*(1): Number of aisles, (2): Number of cross aisles, (3): Aisle length, (4): Aisle width, (5): Cross aisle width, (6): Picks per route, (7): Skewness, (8): Number of subaisles, (9): Subaisle length, (10): Warehouse length, (11): Warehouse width, (12): Dimensions ratio, (13): Picks per aisle, (14): Picks per subaisle, (15): Pick density

5.3. Dimensionality reduction

The addition of highly correlated features may lead to some undesirable conditions, such as overfitting, overly complex models, and more computational time to fit. Dimensionality reduction is the process of transforming a dataset such that only the relevant features are used. We use the backward search to observe which features can be removed from the linear (OLS) and nonlinear (RT, RF, MLP) models in order to improve their performance. Backward search iteratively removes one feature at a time considering the best removal strategy, i.e., the feature that degrades the least the model performance [25]. For OLS and RT, fewer features lead to cleaner models, which are easier to interpret and implement in practice. From the experiments, we observe that the use of the same features selected for RT in the other nonlinear models leads them to perform better as well, besides being trained faster. The results of the backward search for the linear and nonlinear models with default parameters used in Scikit-learn are presented in Table 3. It shows the feature removed in each iteration of the search, and the MSE values for all models fit for both train and test sets. Since no results are obtained after removing the last feature, nothing is shown in the last row of the table.

Table 3: Results for the backward search feature selection method (which feature is removed next)

OLS			*RT*		
Feature	MSE train	MSE test	Feature	MSE train	MSE test
All features	36.27	43.63	All features	1.03	45.38
Pick density	36.42	37.66	Pick density	1.03	38.46
Cross aisle width	36.51	**37.66**	Cross aisle width	1.03	38.77
Warehouse width	36.69	37.69	Warehouse width	1.03	38.34
Number of subaisles	37.00	37.96	Number of subaisles	1.03	38.46
Aisle length	37.59	38.35	Picks per subaisle	1.03	37.85
Number of cross aisles	38.18	38.89	Subaisle length	1.03	38.01
Picks per aisle	38.93	39.67	Warehouse length	1.03	37.46
Dimensions ratio	39.83	40.48	Aisle length	1.03	36.65
Warehouse length	40.79	41.26	Aisle width	1.03	36.13
Number of aisles	42.51	42.73	Number of aisles	1.03	**36.08**
Aisle width	44.54	44.33	Picks per route	1.03	38.09
Picks per route	46.13	46.02	Picks per aisle	1.13	52.43
Picks per subaisle	47.55	47.60	Dimensions ratio	29.96	47.84
Subaisle length	56.31	55.89	Number of cross aisles	53.03	56.29
Skewness	–	–	Skewness	–	–

From the results shown in Table 3, we observe that a maximum performance for the test set is obtained when 13 features are used in OLS and 5 features in RT. The most important features are read from the bottom to the top of the table. Unsurprisingly, *skewness* is the most important feature overall for

both methods. Another relevant feature is *picks per route*. *Number of cross aisles* and *dimensions ratio* seem to be very relevant in RT, but not so much in OLS, which could be explained by these features having a considerable nonlinear correlation with the targets, but a very low linear one. It is interesting to see that most of the artifical features contain significant information to improve the OLS performance, but for RT some of them are actually used to substitute some of the original features in the models. For example, combining *aisle length*, *number of aisles*, *cross aisle width* and *aisle width* with *number of cross aisles* seems to generate a new feature (*dimensions ratio*) that contains more significant information for the problem than considering the first four as separate features. For the remainder of our experiments, we use OLS with the best 13 features (the last 13 features in Table 3). RT, RF and MLP are trained with the best five features (the last five features in Table 3). Next, we detail how the parameters of these models were set in this study.

5.4. Parameter setting

All methods except for the OLS require some parameter tuning to improve performance. We show next how these parameters were set.

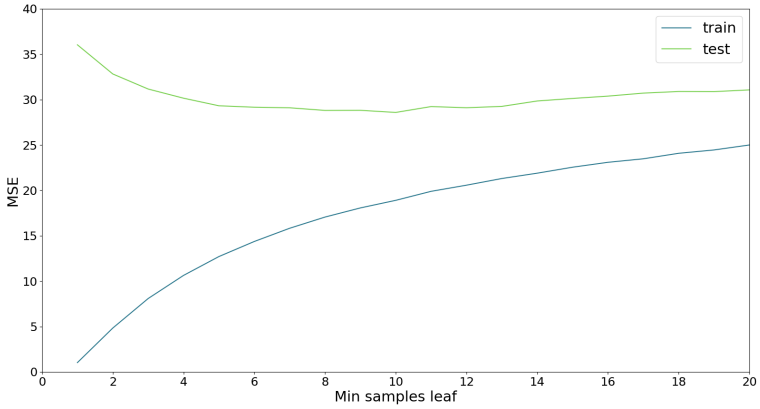
5.4.1. Regression tree

The results for the feature selection presented in Section 5.3 show a very high difference in performance between the train and test sets in the RT model. This indicates that when RT is trained using the default parameters, it might overfit. In RTs, pre-pruning is used to avoid the tree growing too complex by controlling the *min samples leaf* parameter, as previously explained. The choices for this parameter ranged from 1 to 20. As shown in Figure 4, overfitting is observed when the value is too low. As the value increases, the curves representing the MSE for the train and test sets converge. Past the point where MSE for the test set is minimum (*min samples leaf* $= 10$), both curves start to increase together, leading to an underfitted model. For a better performance, we set the *min samples leaf* parameter to 10 for the remaining experiments.

5.4.2. Random forest

The first parameter set for the RF estimator is related to the dataset used for training. As previously stated, the parameter δ determines which data from those generated by the simulator are used to train a model. A higher δ can help the model learn more about the simulator. However, it also means that

Figure 4: Results for the setting of the *min samples leaf* parameter in the RT model



significantly more data are used to train, which may lead to an exponential increase in the training time. We train the RF with default parameters for $\delta = 0.0\%$ to 0.5%. For that, we have added a new feature to the dataset called *gap*. This feature measures the distance of the observed data point to the best solution obtained for the same problem. This way, *gap* in the training set varies between zero and δ, while for the test set *gap* is always zero, since we are interested in predicting the optimal solution. We show the results of training the RF with different δ values in Table 4. From them, we observe an increase in the MSE when δ increases. Although the model is highly fitted to the test set for lower δ values, the peak score for the test set is obtained with $\delta = 0.0$. Since the goal of the RF is to maximize performance, we use $\delta = 0.0\%$ for the remaining experiments.

Table 4: Results for the setting of the δ parameter in the RF model

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE train	3.64	5.89	12.06	19.19	26.64	34.06
MSE test	**20.02**	20.99	22.03	22.10	23.18	23.61

Random forests depend primarily on two parameters: the number of estimators and the maximum number of features. The number of estimators represents the number of trees in the ensemble. Overall, more trees reduce variance at the cost of increased computation time. After preliminary tests, we keep the number of estimators at the Scikit-learn default value of 100. The maximum number of features represents the number of features randomly chosen at each base estimator. More features make the trees more similar, while fewer features can make trees more diverse. However, too few features may turn the trees too biased. We run another batch of experiments to set the maximum number of features, setting it from one to all six features, i.e., the five chosen using the backward search and *gap*. The results show that the MSE for the test set is at its lowest point (MSE = 19.88) when the

maximum number of features is four. We use this value for the remaining experiments using RF.

5.4.3. Multilayer perceptron

The parameters to tune in the MLP are the number of hidden layers and the number of neurons in each hidden layer. Before tuning them, we perfomed experiments for different δ values to observe how MLP behaves when more information about the gaps are used. The values tested for δ are the same as those used in the RF. Table 5 shows the results obtained for an MLP with a single hidden layer with 10 neurons and the maximum number of iterations set to 5000 to allow convergence in the optimization process. The remaining parameters are as default in Scikit-learn. All features were scaled. Unlike the RF trained, overfitting is not a concern in the MLP. The performance for the test set increases with more data used for training, but peak performance is obtained with $\delta = 0.1$. Following the goal of maximizing performance, we train the MLP in the remaining experiments using $\delta = 0.1$.

Table 5: Results for the setting of the δ parameter in the MLP model

δ	0.0	0.1	0.2	0.3	0.4	0.5
MSE train	32.17	27.78	32.75	38.97	45.45	52.11
MSE test	34.11	**28.03**	28.18	29.65	30.17	29.89

As the results in Table 5 show, the performance of the MLP using the parameters described is worse than RF. In order to improve it, the two parameters are set using a grid search. We tested the number of hidden layers from one to three with 20, 50, 100, 200 and 300 neurons each. The maximum number of iterations is set to 5000. Table 6 shows the results obtained in the grid search. The best setting found is the use of two layers with 200 neurons each. The MSE for the test set improved from 28.03 to 19.63 when compared to using a single layer with 10 neurons.

Table 6: Results for the setting of the number of hidden layers and neurons parameters in the MLP model

Neurons per layer	*1 layer*		*2 layers*		*3 layers*	
	MSE train	MSE test	MSE train	MSE test	MSE train	MSE test
20	24.01	25.26	19.96	22.14	18.10	21.37
50	20.62	22.81	16.45	20.30	15.06	20.40
100	18.59	21.26	15.43	19.92	13.67	19.88
200	17.81	20.77	15.06	**19.63**	13.33	20.03
300	17.37	21.08	14.61	19.84	12.83	19.77

5.5. Performance comparison

In this section, we compare the performance of each model. First, we recap how each model performed using the MSE metric and the training time (Table 7). The results show that OLS scores worse than the other models for the test set, while MLP generates solutions closer to the best solutions found in the simulations. However, the time to fit the MLP is significantly higher than the time to fit the RF, and the score improvement is marginal.

Table 7: MSE among different models

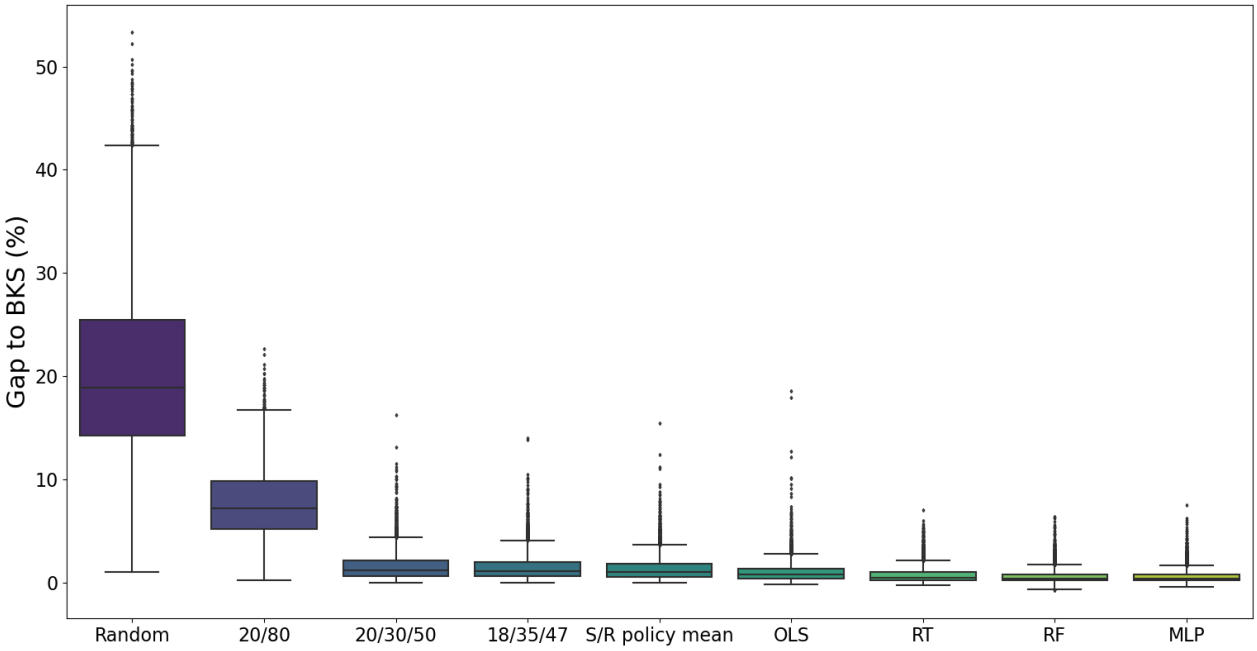
Method	*MSE train*	*MSE test*	*Time to fit (s)*
Ordinary least squares	36.51	37.65	5.8
Decision tree	18.90	28.58	5.8
Random forest	3.60	19.88	9.2
Multilayer perceptron	15.06	19.63	173.1

A model that performs well in the MSE metric will not necessarily do well for the zone sizing problem. In order to measure the real model performances, we run the simulator using the predicted zone sizes for the test set and compared against the BKS found by the grid search done using the simulator as reference. In Figure 5, we compare the quality of the predictions made by each model against a *random* policy, which is equivalent to the use of a single zone, a *20/80* policy representing the most common partition for a two zones system, a *20/30/50* policy representing the most common sizes used for an ABC system, a *18/35/47* policy representing the average target values shown in Table 1, and *S/R policy mean* representing the average target values derived for each S/R policy (Figure 3).

The conclusions drawn from Figure 5 and the previous experiments indicate that the use of three zones performs significantly better on average than a random storage or a two zone system, regardless of the method used to define zone boundaries. Moreover, although 20/30/50 is the most common combination used in practice, actually 18/35/47 has on average a better performance (1.61% versus 1.50%), and if fixed sizes are picked from the list derived for each S/R policy, the solutions are even better on average (1.37%).

Regarding the first regression method, although the linear relationship between features and targets is weak, OLS is capable of improving the average gap to the reference solutions compared to the fixed sizes (1.05%), but the variance in performance remains high. RT, which is a competing method against these due to its easy interpretability and replicability, has an even better average performance (0.76%) and a much lower variance.

Figure 5: Methods gaps to the ARL of the best combination of zone sizes found in the simulations



Finally, as expected, RF and MLP show the best average performances among all methods (0.65% both). Also, variance is low with a slight advantage to MLP. On the other hand, RF finds better solutions in the best cases, sometimes even finding solutions with a negative gap to the reference solutions.

These conclusions attest that the models that performed better for the MSE metric also performed better for the real zone sizing problem, even though this metric is not contained in the objective function of the problem.

6. Managerial analysis

In this section, we provide a discussion to highligth the pros and cons of using each modeling strategy to solve the zone sizing problem in practice. We also deepen in the model performance analysis for the S/R policy combinations.

6.1. Methods comparison

Whenever developing models to solve optimization problems, it is usual to give all attention to their performance and overlook their applicability. In practice, there are many variables that are usually

disregarded when modeling the problem. They may play a vital role when deciding whether a better performing method is of practical use. For the zone sizing problem, we present a set of seven important characteristics a method must have in order to be applicable. All methods presented in this work are evaluated according to these attributes. The first one is, obviously, the method's *performance* considering the average quality of the solutions obtained. Another desirable attribute is the *variance* of solutions. Ideally, a method provides similar solutions in terms of quality regardless of the warehouse setting. Infeasible solutions, such as negative zone sizes or sizes that sum up more than 100%, can be generated from models so that the *infallibility* is another point to be considered. An infallible method can always provide feasible solutions, disregarding its performance, even for unusual features. The fourth point considered is the *interpretability*. Methods that result in interpretable models have a decision process easy to understand and, therefore, to be accepted by practitioners. This can reduce the resistance for its adoption, especially in the cases that the result are unusual zone boundaries. Another characteristic desired is the *quickness* to train and run the method until satisfactory solutions are obtained. Specifically for the zone sizing problem, it is important to evaluate whether a method requires a demand *forecasting accuracy* in order to make good predictions. Some methods are more sensible to the precision of the real skewness feature than others and, therefore, accuracy in the demand forecasting is required. Finally, the implementation *cost* of the method should be considered. The cost can be measured based on the effort required for its implementation.

Table 8 provides a summary of how each method is classified among these seven characteristics. Arbitrary sizes consider zone boundaries established from commonly used arbitrary values or the reference values for the S/R policy combinations. For OLS and RT, we consider the adoption of the models provided in this work, which are ready to be adopted and very easy to understand since they consist of simple arithmetic operations in the OLS case or a tree with binary decisions in the RT case. For RF and MLP, we understand that the models trained here are hardly replicable. Therefore, new models would have to be trained, for example, using real data from the warehouse operations. Finally, for the simulator method, we consider that the warehouse would have to implement their own tailor-made simulator and perform a grid search to find the best combination of zone sizes.

From a practical perspective, no single method absolutely outperforms others for all attributes. Overall, arbitrary sizes are better applicable in cases where no demand forecasts are available. This is hardly the case for existing warehouses. The only advantage of OLS over RT is its slightly easier interpretability. However, RT has many more advantages and is more recommended to be used in the

Table 8: Qualitative comparison between different methods to solve the zone sizing problem

Method	*Performance*	*Variance*	*Infallibility*	*Interpretability*	*Quickness*	*Forecasting accuracy*	*Cost*
Arbitrary sizes	Poor	Very high	Yes	Very easy	No run required	None	None
OLS	Fair	High	Can fail	Very easy	No run required	Required	None
RT	Good	Average	Yes	Easy	No run required	Desirable	None
RF	Very good	Average	Yes	Hard	Fast train and test	Desirable	Low*
MLP	Very good	Low	Can fail	Very hard	Slow train, fast test	Required	Low*
Simulator	Excellent	Very low	Yes	Average	Very slow	Required	High

*Considering that data for training is readily available. Otherwise, gathering data for the warehouse performance under different zone sizes can be significantly more expensive than building a simulator

same conditions. RF and MLP are better options when the slightly better performance they provide can result in significant savings, for example, in companies with large scale operations and many warehouses. Therefore, it is reasonable to assume that data is available to train them. The simulator is recommended in case an even better performance is desirable and enough resources are available to implement it.

6.2. S/R policy combination analysis

All results presented so far are for the ensemble of models trained. Another way to evaluate their performance is by looking at each of the 16 models individually. Table 9 shows the average and maximum percentage deviation for the predictions obtained with each model to the ARL of the BKS for the samples in the test set. The results show that solutions are better for some S/R policy combinations than others. The solutions for the combinations with the across-aisle and the nearest-location are closer to optimum than those for the nearest-subaisle and, even worse, the within-aisle. However, regardless of the policies, all average solutions are improved when using a trained ML method. This is true even for a bad performing method, such as OLS.

As explained in Section 5.2.1, some combinations are naturally easier to find good estimations for the optimal zone sizes. The combinations involving the nearest-subaisle and the across-aisle policies (except Aa/Lg) are those for which the models approximate the BKS better. The hardest problems are mostly those involving a storage policy linked to the aisles (within-aisle) and subaisles (nearest-subaisle). After investigating the data from the simulations, we noted a trend for these two policies that the BKS found usually have zone boundaries overlapping the point where aisles or subaisles end. In order to investigate that, we rounded all predictions found for the zones A and B to the percentage representing the nearest number of full subaisles that these zones should cover. For example, instead of predicting that z_A covers 14.7 subaisles, it is rounded such that 15 subaisles belong to it. In Table 10, we show the new average and maximum results for each S/R policy combination using this rounding

Table 9: Average and maximum percentage deviations to the ARL of the BKS found in the simulations for each S/R policy combination (best values in boldface)

S/R Policy	*18/35/47*		*S/R policy mean*		*OLS*		*RT*		*RF*		*MLP*	
	Avg (%)	Max (%)	Avg (%)	Max (%)	Avg (%)	Max (%)	Avg (%)	Max (%)	Avg (%)	Max (%)	Avg (%)	Max (%)
Aa/Aba	1.55	4.30	1.44	8.58	0.63	6.75	0.34	3.41	**0.27**	**1.42**	0.28	2.02
Aa/Sh	0.99	4.35	0.85	6.45	0.75	5.52	**0.23**	**2.59**	0.29	5.44	0.37	6.21
Aa/Lg	2.08	5.47	1.81	6.08	1.60	5.46	**0.53**	5.33	0.66	**4.74**	0.61	5.17
Aa/Co	1.16	3.77	1.22	4.29	0.94	3.64	0.42	5.50	**0.30**	**2.57**	0.36	2.74
Nl/Aba	0.76	5.22	0.81	6.88	0.49	3.21	0.44	2.28	**0.35**	**2.28**	0.38	2.93
Nl/Sh	0.96	5.24	0.91	4.90	0.69	4.01	0.61	2.83	0.46	2.74	**0.45**	**2.73**
Nl/Lg	0.98	5.97	0.87	4.22	0.62	3.45	0.58	**3.44**	**0.47**	3.63	0.49	3.52
Nl/Co	0.93	5.79	0.87	4.94	0.63	3.70	0.48	2.43	**0.38**	**1.90**	0.39	2.58
Ns/Aba	1.31	7.64	1.45	9.55	1.03	6.15	1.04	**4.65**	**0.85**	5.62	0.91	5.64
Ns/Sh	1.56	8.75	1.47	8.13	1.19	5.42	1.22	5.23	1.01	**4.18**	**0.96**	4.58
Ns/Lg	1.57	7.26	1.41	6.71	1.32	6.64	1.20	4.73	**1.06**	5.25	1.08	**4.29**
Ns/Co	1.47	6.53	1.40	5.96	1.23	5.68	1.15	5.64	**0.91**	**4.77**	1.02	6.12
Wa/Aba	2.24	14.01	2.14	15.43	1.64	10.17	1.15	**5.98**	**1.01**	6.36	1.10	7.52
Wa/Sh	1.88	10.00	1.73	7.92	1.20	7.39	0.96	5.21	0.85	3.85	**0.66**	**3.48**
Wa/Lg	2.51	10.11	1.87	11.22	1.60	18.60	0.89	7.07	0.69	6.30	**0.60**	**3.13**
Wa/Co	1.97	10.50	1.71	6.84	1.30	17.94	0.91	4.83	0.80	4.98	**0.70**	**4.68**
Average	1.50	7.18	1.37	7.38	1.05	7.11	0.76	4.45	**0.65**	**4.13**	**0.65**	4.21

to the nearest subaisle method for the predictions done by the RF. The average performance of the predicted solutions improves from 0.65% to 0.50%, but the gains are most notable in the predictions for the nearest-subaisle and within-aisle policies. This indicates that full subaisles are generally good candidates to constitute an optimal solution for these policies.

Table 10: Comparison between the original predictions by RF against the predictions rounded to the nearest subaisle

S/R Policy	*Original predictions*		*Rounded predictions*	
	Avg (%)	Max (%)	Avg (%)	Max (%)
Aa/Aba	**0.27**	**1.42**	0.28	1.71
Aa/Sh	0.29	**5.44**	**0.24**	6.01
Aa/Lg	0.66	**4.74**	**0.65**	5.14
Aa/Co	**0.30**	**2.57**	**0.30**	3.07
Nl/Aba	**0.35**	**2.28**	0.39	2.32
Nl/Sh	**0.46**	**2.74**	0.52	3.76
Nl/Lg	**0.47**	**3.63**	0.50	3.75
Nl/Co	**0.38**	**1.90**	0.43	2.24
Ns/Aba	0.85	5.62	**0.59**	**2.50**
Ns/Sh	1.01	4.18	**0.64**	**3.99**
Ns/Lg	1.06	5.25	**0.71**	**3.39**
Ns/Co	0.91	**4.77**	**0.66**	5.84
Wa/Aba	1.01	**6.36**	**0.79**	6.45
Wa/Sh	0.85	3.85	**0.46**	**3.54**
Wa/Lg	0.69	6.30	**0.39**	**3.44**
Wa/Co	0.80	**4.98**	**0.50**	5.34
Average	0.65	4.13	**0.50**	**3.91**

The stopping criterion used for the simulations was a maximum half-width of 0.25%. In a previous analysis done in Section 5.2.1, we already demonstrated that this criterion leads to not rejecting the hypothesis that a solution with a gap of 0.5% to the BKS is equal to it, due to the overlap of

the confidence intervals of the best known and predicted solutions. With the predictions rounded to the nearest subaisle method, the average gap of the predicted solutions reached this threshold, which means further improvements past this point are statistically meaningless. In fact, 67.9% of the predictions made by RF after rounding are below a 0.5% gap from the BKS. Before rounding, this metric was only at 41.6%. Gains with rounding to the nearest subaisle are observed in all models used for the nearest-subaisle and within-aisle storage policies. This is an important observation to be considered as a simple guideline for practitioners to improve zone sizing solutions.

7. Conclusions

This paper investigates the ABC zone sizing problem, which arises from the class-based policy used to solve the storage location assignment problem. In this problem, a multi-block warehouse is divided into three zones of sizes to be determined. The set of products to be assigned to storage locations are divided into three classes according to their demands, and, the best classes are assigned to the best zones. Although the zone sizing is a common problem faced in manual warehouses, the literature about it is still scarce.

We have generated detailed synthetic data to represent real warehouses and orders using a simulator to estimate the average route length. The simulator is fed with a set of inputs related to the warehouse layout, the pick list characteristics, and the storage and routing policies used. An analysis of the data obtained from an extensive batch of simulations revealed that good zone sizes vary significantly within a large range. Only half of the optimal zone sizes for the common warehouse settings analyzed fall within a range of 12% to 23% for zone A, 30% to 41% for zone B, and 39% to 52% for zone C. We have also shown that some combinations of storage and routing policies are harder to approximate good solutions than others. We observed that all considered features have weak linear correlations with the zone sizes, and that the demand skewness is the most correlated feature for most storage and routing policies combinations.

We have then used the data from the simulations to train several machine learning methods to learn on how to predict the optimal zone sizes. This was done by showing to them what are the best solutions found for different settings. The main advantage of this approach is that it provides a faster way to solve the problem without the need to implement a simulator, which can be too time-consuming for warehouses. The models that can capture nonlinear relationships between features and targets performed better than the linear regression. We have shown that a set of only five features – demand

skewness, number of cross aisles, dimensions ratio, picks per aisle, and picks per route – is enough to obtain a good performance using these nonlinear models.

There is a trade-off between performance and applicability among the models. Arbitrary sizes are easy to remember and significantly outperform alternative methods such as the random storage or a two-zone system. However, they do not fully benefit from the performance potential of using a three-zone system. The linear functions and decision trees are interpretable models that consider the problem features to improve performance. Even though linear correlations are weak, the linear regression model outperformed the more primitive solutions. The decision tree trained predicts solutions on average 0.76% from the best ones found in the simulations and it is a method that can be easily adopted using the trees generated here. For even better results, an ensemble method, such as the random forest, or a neural network, such as the multilayer perceptron, can be used. We used a random forest to show that solutions with an average gap below 0.5% to the best found in the simulations are achievable. At this point, the loss of solution's quality is statistically insignificant, such that most of the solutions predicted are no different than those provided by the simulations.

References

[1] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, Berlin, 2018.

[2] D. Battini, M. Calzavara, A. Persona, and F. Sgarbossa. Order picking system design: the storage assignment and travel distance estimation (SA&TDE) joint method. *International Journal of Production Research*, 53(4):1077–1093, 2015.

[3] M. Bortolini, M. Faccio, E. Ferrari, M. Gamberi, and F. Pilati. Design of diagonal cross-aisle warehouses with class-based storage assignment strategy. *The International Journal of Advanced Manufacturing Technology*, 100(9-12):2521–2536, 2019.

[4] N. Boysen, R. B. M. de Koster, and F. Weidinger. Warehousing in the e-commerce era: A survey. *European Journal of Operational Research*, 277(2):396–411, 2019.

[5] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[6] M. Calzavara, F. Sgarbossa, and A. Persona. Vertical lift modules for small items order picking: an economic evaluation. *International Journal of Production Economics*, 210:199–210, 2019.

[7] F. Caron, G. Marchet, and A. Perego. Routing policies and coi-based storage policies in picker-to-part systems. *International Journal of Production Research*, 36(3):713–732, 1998.

[8] F. T. S. Chan and H. K. Chan. Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications*, 38(3):2686–2700, 2011.

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

[10] R. B. M. De Koster, T. Le-Duc, and N. Zaerpour. Determining the number of zones in a pick-and-sort order picking system. *International Journal of Production Research*, 50(3):757–771, 2012.

[11] A. S. Dijkstra and K. J. Roodbergen. Exact route-length formulas and a storage location assignment heuristic for picker-to-parts warehouses. *Transportation Research Part E: Logistics and Transportation Review*, 102:38–59, 2017.

[12] F. Esposito, D. Malerba, G. Semeraro, and J. Kay. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):476–491, 1997.

[13] A. Eynan and M. J. Rosenblatt. Establishing zones in single-command class-based rectangular AS/RS. *IIE Transactions*, 26(1):38–46, 1994.

[14] E. Frazelle. *World-Class Warehousing and Material Handling*. McGraw-Hill, New York, 2nd edition, 2016.

[15] E. H. Grosse, S. M. Dixon, W. P. Neumann, and C. H. Glock. Using qualitative interviewing to examine human factors in warehouse order picking. *International Journal of Logistics Systems and Management*, 23(4):499–518, 2016.

[16] X. Guo, Y. Yu, and R. B. M. De Koster. Impact of required storage space on storage policy performance in a unit-load warehouse. *International Journal of Production Research*, 54(8): 2405–2418, 2016.

[17] W. H. Hausman, L. B. Schwarz, and S. C. Graves. Optimal storage assignment in automatic warehousing systems. *Management Science*, 22(6):629–638, 1976.

[18] H. Hwang, Y. H. Oh, and Y. K. Lee. An evaluation of routing policies for order-picking operations in low-level picker-to-part system. *International Journal of Production Research*, 42(18):3873–3889, 2004.

[19] J. M. Jarvis and E. D. McDowell. Optimal product layout in an order picking warehouse. *IIE Transactions*, 23(1):93–102, 1991.

[20] T. Le-Duc and R. B. M. De Koster. Travel distance estimation and storage zone optimization in a 2-block class-based storage strategy warehouse. *International Journal of Production Research*, 43(17):3561–3581, 2005.

[21] 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.

[22] W.-Y. Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82 (3):329–348, 2014.

[23] Y. Ma, Z. Zhang, A. Ihler, and B. Pan. Estimating warehouse rental price using machine learning techniques. *International Journal of Computers, Communications & Control*, 13(2), 2018.

[24] R. Manzini, R. Accorsi, M. Gamberi, and S. Penazzi. Modeling class-based storage assignment over life cycle picking patterns. *International Journal of Production Economics*, 170:790–800, 2015.

[25] T. Marill and D. Green. On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, 9(1):11–17, 1963.

[26] D. Mirčetić, N. Ralević, S. Nikoličić, M. Maslarić, and D. Stojanović. Expert system models for forecasting forklifts engagement in a warehouse loading operation: A case study. *Promet – Traffic&Transportation*, 28(4):393–401, 2016.

[27] M. Mirzaei, N. Zaerpour, and R. B. M. de Koster. The impact of integrated cluster-based storage allocation on parts-to-picker warehouse performance. *Transportation Research Part E: Logistics and Transportation Review*, 146:102207, 2021.

[28] V. R. Muppani and G. K. Adil. Efficient formation of storage classes for warehouse storage location assignment: a simulated annealing approach. *Omega*, 36(4):609–618, 2008.

[29] J. M. R. Murteira and J. J. S. Ramalho. Regression analysis of multivariate fractional data. *Econometric Reviews*, 35(4):515–552, 2016.

[30] D. Nguyen Duc, T. Tran Huu, and N. Nananukul. A dynamic route-planning system based on industry 4.0 technology. *Algorithms*, 13(12):308, 2020.

[31] K. I. Nikolopoulos, M. Z. Babai, and K. Bozos. Forecasting supply chain sporadic demand with nearest neighbor approaches. *International Journal of Production Economics*, 177:139–148, 2016.

[32] S. G. Ozden, A. E. Smith, and K. R. Gue. A novel approach for modeling order picking paths. *Naval Research Logistics*, 68(4):471–484, 2020.

[33] L. Pansart, N. Catusse, and H. Cambazard. Exact algorithms for the order picking problem. *Computers & Operations Research*, 100:117–127, 2018.

[34] P. J. Parikh and R. D. Meller. A travel-time model for a person-onboard order picking system. *European Journal of Operational Research*, 200(2):385–394, 2010.

[35] J. A Pazour and H. J. Carlo. Warehouse reshuffling: Insights and optimization. *Transportation Research Part E: Logistics and Transportation Review*, 73:207–226, 2015.

[36] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[37] C. G. Petersen. The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations & Production Management*, 19(10):1053–1064, 1999.

[38] C. G. Petersen, G. R. Aase, and D. R. Heiser. Improving order-picking performance through the implementation of class-based storage. *International Journal of Physical Distribution & Logistics Management*, 34(7):534–544, 2004.

[39] S. S. Rao and G. K. Adil. Optimal class boundaries, number of aisles, and pick list size for low-level order picking systems. *IIE Transactions*, 45(12):1309–1321, 2013.

[40] S. S. Rao and G. K. Adil. Analytical models for a new turnover-based hybrid storage policy in unit-load warehouses. *International Journal of Production Research*, 55(2):327–346, 2017.

[41] H. D. Ratliff and A. S. Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521, 1983.

[42] J. Reyes, E. Solano-Charris, and J. Montoya-Torres. The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, 10(2):199–224, 2019.

[43] K. J. Roodbergen. Storage assignment for order picking in multiple-block warehouses. In R. Manzini, editor, *Warehousing in the Global Supply Chain: Advanced Models, Tools and Applications for Storage Systems*, pages 139–155. Springer, 2012.

[44] K. J. Roodbergen and R. B. M. De Koster. Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9):1865–1883, 2001.

[45] K. J. Roodbergen, I. F. A. Vis, and G. D. Taylor Jr. Simultaneous determination of warehouse layout and control policies. *International Journal of Production Research*, 53(11):3306–3326, 2015.

[46] A. Scholz and G. Wäscher. Order batching and picker routing in manual order picking systems: The benefits of integrated routing. *Central European Journal of Operations Research*, 25(2): 491–520, 2017.

[47] A. Scholz, S. Henn, M. Stuhlmann, and G. Wäscher. A new mathematical programming formulation for the single-picker routing problem. *European Journal of Operational Research*, 253(1): 68–84, 2016.

[48] A. H. Schrotenboer, S. Wruck, K. J. Roodbergen, M. Veenstra, and A. S. Dijkstra. Order picker routing with product returns and interaction delays. *International Journal of Production Research*, 55(21):6394–6406, 2017.

[49] Y. Shi, T. Wang, and L. C. Alwan. Analytics for cross-border e-commerce: Inventory risk management of an online fashion retailer. *Decision Sciences*, 2020.

[50] A. Silva, L. C. Coelho, M. Darvish, and J. Renaud. Integrating storage location and order picking problems in warehouse planning. *Transportation Research Part E: Logistics and Transportation Review*, 140:102003, 2020.

[51] N. Sooksaksun, V. Kachitvichyanukul, and D.-C. Gong. A class-based storage warehouse design using a particle swarm optimisation algorithm. *International Journal of Operational Research*, 13(2):219–237, 2012.

[52] C. Theys, O. Bräysy, W. Dullaert, and B. Raa. Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, 200(3):755–763, 2010.

[53] C. R. Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf. A conceptual basis for feature engineering. *Journal of Systems and Software*, 49(1):3–15, 1999.

[54] J. P. Van den Berg. Class-based storage allocation in a single-command warehouse with space requirement constraints. *International Journal of Industrial Engineering*, 3:21–28, 1996.

[55] S. Weisberg. *Applied Linear Regression*, volume 528. John Wiley & Sons, Hoboken, NJ, 2013.

[56] Y. Yu and R. B. M. De Koster. Optimal zone boundaries for two-class-based compact three-dimensional automated storage and retrieval systems. *IIE Transactions*, 41(3):194–208, 2009.

[57] N. Zaerpour, Y. Yu, and R. B. M. De Koster. Optimal two-class-based storage in a live-cube compact storage system. *IISE Transactions*, 49(7):653–668, 2017.