A Deep Learning Approach to Address the Storage Location Assignment Problem

Paul Courtin, Jean-Baptiste Fasquel, Axel Grimault, Mehdi Lhommeau

▶ **To cite this version:**

Paul Courtin, Jean-Baptiste Fasquel, Axel Grimault, Mehdi Lhommeau. A Deep Learning Approach to Address the Storage Location Assignment Problem. 16th International Conference on Agents and Artificial Intelligence - Doctoral Consortium - ICAART 2024, SCITEVENTS, Feb 2024, Roma, Italy. ⟨hal-04775703⟩

HAL Id: hal-04775703

https://hal.science/hal-04775703v1

Submitted on 13 Nov 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

A Deep Learning Approach to Address the Storage Location Assignment Problem

Paul Courtin[1,2][a], Jean-Baptiste Fasquel[1][b], Mehdi Lhommeau[1][c] and Axel Grimault[1][d]
[1]*Université d'Angers, LARIS, SFR MASTIC, 62 Av. de Notre Dame du Lac, 49000 Angers, France*
[2]*Knapp France, 23 rue de la Maison Rouge, 77185 Lognes, France*
{*f_author, s_author*}*@univ-angers.fr, paul.courtin@knapp.com*

Abstract

Automated warehouses picking performance optimization is strongly influenced by a wise assignment of products into storage locations. The storage location assignment problem (SLAP), is usually solved with exact or approach methods. For this study we focus on shuttles-based storage and retrieval system (SBS/RS). In this article we present a new method to address the SLAP using Deep Learning. We exploit historical picking order data to feed a neural network model. This model returns an optimal allocation matrix for products into the SBS/RS based on past picking orders. We present our architecture and our preliminary results on a synthetic SBS/RS.

1 RESEARCH PROBLEM

A warehouse is an intermediary facility between suppliers and customers that plays an important role in daily supply chain operations. Warehouse activities typically encompass receiving, storing, order-picking, sorting, and shipping, among which order-picking is the most time and labor consuming operation (Zhang et al., 2019). Order-picking is the process of retrieving items from storage locations to fulfil customers orders. In this paper, we focus on an automated warehouse of type goods-to-man implementing a Shuttle-based Storage and Retrieval System (SBS/RS). SBS/RS are derivative of Automated Storage and Retrieval System (AS/RS) that operates shuttles for item retrieval and storage.

The SBS/RS is composed of one or more storage aisles, as shown in Figure 1. In the middle of aisle shelves, on each floor, runs a tier captive shuttle. These shuttles are responsible for handling storage totes and transporting them in the horizontal direction to and from the lifts. Lifts are positioned at the beginning of every storage aisles. Their function is to move the storage totes vertically, between the SBS/RS storage levels and the rest of the warehouse conveyor system. Buffer slots are located between lifts and shuttles, serving as temporary resting positions for storage totes. This arrangement allows for independent movement of shuttles and lifts.

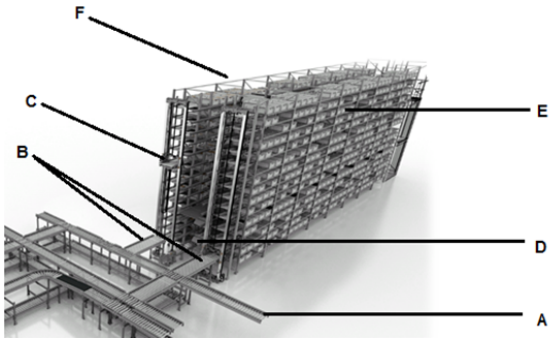

Figure 1: Drawing of an SBS/RS with input and output conveyor. (A) Warehouse conveyor; (B) Input and Output (I/O) conveyor section; (C) Lift moving vertically; (D) Racking level with shuttles moving horizontally; (E) storage location; (F) Aisle.

To increase the efficiency of warehouse operations, we focus on the order picking process. In goods-to-man automated warehouses, the system brings goods to human operators at picking stations. Storage totes are traveling through the warehouse, carried by conveyors. The SBS/RS holds storage totes and releases them on conveyor upon request from the Warehouse Management Software (WMS). The time spend by a storage tote to travel from its storage location to a picking station is the *retrieval time*. In our study, we focus only on the SBS/RS to maximize picking efficiency by reducing the retrieval time of storage totes. Retrieval time is defined as the time spent by a storage tote to travel from its storage location to a picking station. To reduce the retrieval time, we propose to optimize the allocation of storage totes

[a] https://orcid.org/0000-0002-7512-8178
[b] https://orcid.org/0000-0001-9183-0365
[c] https://orcid.org/0000-0001-5772-282X
[d] https://orcid.org/0000-0002-0816-0645

to storage locations within the SBS/RS. By strategically placing storage totes in optimal locations, we expect to minimize the travel time required for shuttles to retrieve them and deliver them to the input/output point (I/O) of the SBS/RS, which is the connection point between the conveyors and the SBS/RS. This optimization should significantly improve the overall efficiency of the order picking process. This problem of determining the optimal allocation of storage totes within a warehouse is known as the *Storage Location Assignment Problem* (SLAP). It consists of determining the most efficient assignment of items to locations in order to minimize or maximize one or more objectives. In this paper, we will seek to reduce the retrieval time by minimizing the travel time (or distance) of the SBS/RS shuttles.

2 OUTLINE OF OBJECTIVES

In this study, we propose a data-driven approach to solve the SLAP within a SBS/RS. For this, we consider a new dynamical decision model using Deep Learning to optimize storage totes allocations. In particular, our method is designed to tackle peak situations (sudden and high variation in requested pieces) occurring in some picking orders.
Objectives of this study are:

- Propose a new method to solve the SLAP using Deep Learning
- Evaluate how Deep Learning performs better than standard approaches

3 STATE OF THE ART

The SLAP was first formulated (Hausman et al., 1976) and was proven NP-hard (Frazelle, 1989). As the number of SKUs to store increases, the number of possible locations becomes very large.

Solutions to solve the SLAP include classic approaches such as exact methods: dynamic programming, integer mixed linear programming (Reyes et al., 2019), approximate methods: heuristics and metaheuristics (Talbi, 2016) and simulations. Dedicated storage strategies and policies can also be considered, including particular rules and formulation to deal with the SLAP (Kofler, 2014). The most common strategies are:

- *Class-Based* (CB): the products are separated into several classes, and a storage area is dedicated to each class. The most popular is the assignment in 3 classes, `ABC` where the SKUs are distributed according to a criterion such as the *turnover* (e.g. the most frequently ordered products are placed in the class `A`, associated with the area closest to the I/O point), introduced by (Hausman et al., 1976). Within each class, the products are arranged with a simple rule (e.g. the closest available position to the I/O point). Other classifications may be considered. For example, the classification `XYZ` is based on order fluctuations for a product (Nowotyńska, 2013; Stojanović and Regodić, 2017) ;
- Duration-of-Stay (DoS): SKUs are assigned to a location, where the distance from the I/O point is proportional to the time spend by the SKU in the warehouse. Under certain conditions (DoS known in advance and a balanced number of goods input-output), this rule is optimal (Goetschalckx and Ratliff, 1990) compared to other strategies, in terms of travel time and space occupation;
- *Cube-Order-per-Index* (COI): This metric defines the ratio between the space needed to store a SKU and its demand (Goetschalckx and Ratliff, 1990). Products with a low COI (i.e. small size and often in demand) are placed near the I/O point;
- *Random-Based*: This policy assigns SKU to storage location randomly. It's used in the industry and is used in academic study as comparison baseline.

Recent data-driven approaches to the Storage Location Assignment Problem (SLAP) in warehouses have explored various techniques to optimize storage tote allocation and retrieval strategies. One study (Antomarioni et al., 2021) considers association rule mining to pair SKUs (stock-keeping units) with storage totes within storage locations.

Another deals with shuttle storage location assignment and retrieval sequence scheduling with a storage policy using a hybrid solution with ant colony algorithm and adaptive large neighborhood search (Kazemi et al., 2019). Another uses a Gaussian process surrogate model (Park et al., 2023).

Previous work addressing the SLAP targeted the objective of decreasing the time required for completing a single order or a set of picking orders (Bolaños Zuñiga et al., 2020) or, more generally speaking, the minimization of the travel distance (Schenone et al., 2020).

To the best of the authors knowledge, no other study as tried to address and solve the SLAP using Deep Learning method (Zarinchang et al., 2023). We propose such Deep-Learning-based approach to dynamical allocate SKU to storage locations within an automated goods-to-man warehouse, in order to mini-

mize the retrieval time. Except few preliminary works (Li et al., 2019; Zhang et al., 2019), mainly based on Machine Learning, there is no study, in our sense, dealing with such an approach. In particular, (Li et al., 2019) uses a recurrent network to predict the duration of stay of pallets in the AVS/RS in order to optimize warehouse storage assignment. Other related works involving forecasting strategies exist in different application domains, such as energy building consumption (Cai et al., 2019).

4 METHODOLOGY

The proposed methodology using Deep Learning aims at minimizing the retrieval time of storage totes from an SBS/RS. For sake of simplicity, we consider that the SBS/RS can store only one SKU per tote.

4.1 Notation

For the remainder of the paper, we introduce the following notation:

- S: Set of SKUs ;
- M: Height of the SBS/RS, expressed in number of levels ;
- N: Width of the SBS/RS, expressed in number of channels ;
- $A(t)$: Set of allocations at timestamp t, where SKUs are allocated.

4.2 Approach

Figure 2 provides an overview of the proposed approach. The input data for the Deep Learning architecture consist of the number of pieces picked for each SKU on past days. The model generates an allocation matrix for the next day, where each SKU is assigned to one or more storage locations depending on the number of pieces to be picked. Some locations may remain empty.

The proposed architecture includes a part dedicated to handle historical time series data. The historical data is processed by specialized module, such as an LSTM (Long Short-Term Memory) network, similar to approaches considered in (Cai et al., 2019). Each LSTM is responsible to process historical data for one SKU (see LSTM modules in figure 2). Assuming that we have knowledge of past picking orders of SKUs, the output of each SKU's module is connected to the input of a second neural layer, the Dense layer shown in Figure 2. This part of the architecture is responsible for proposing the allocation. The result is a complete end-to-end Deep Learning architecture that aims to propose an assignment from past picking orders.

According to a past set of picking orders, the deep neural network provides at t, a tensor of size ($M \times N \times S$), as illustrated in Figure 2.

For each SKU $s \in S$, a probability of being assigned to each storage position is computed. The final assignment matrix $A(t)$ of size $M \times N$ is constructed by selecting the allocations of the SKUs associated with the highest probabilities (indicated by the *argmax* operation in Figure 2).

5 EXPECTED OUTCOME

1. Avoid operation research based approaches which can be very computational due to its combinatorial nature. By opposition, one expects that deep learning model are fast at inference
2. Data-based integration of past picking orders to decide about the appropriate SKU allocation
3. Minimize the retrieval time within SBS/RS scope

6 STAGE OF THE RESEARCH

We implemented the proposed Deep Learning-based architecture in Python using the PyTorch package. In this section, we present preliminary experiments on a synthetic SBS/RS. We begin by describing the SBS/RS and the experimental protocol, followed by the presentation of our results.

6.1 Synthetic SBS/RS

We consider 3 SKUs ($S = 3$) and a SBS/RS with a storage location matrix of 12 levels ($M = 12$) and 30 channels ($N = 30$), leading to 360 storage locations.

The input data consists of the number of pieces to pick for a list of SKUs on a daily basis. The output of the model is an allocation matrix $A(t)$. .

The synthetic dataset consists of generated picking orders that mimic real situations, such as the one illustrated by Figure 3 (Retail Data Analytics Challenge). Such data can be classified with demand variation, known as `XYZ` analysis (Nowotyńska, 2013; Stojanović and Regodić, 2017). Class `X` includes SKUs with stable, uniform, and continuous demand over time, exhibiting little variation. Class `Y` regroups SKUs with more fluctuating demand, particularly influenced by seasonality. Finally, Class `Z` consists of

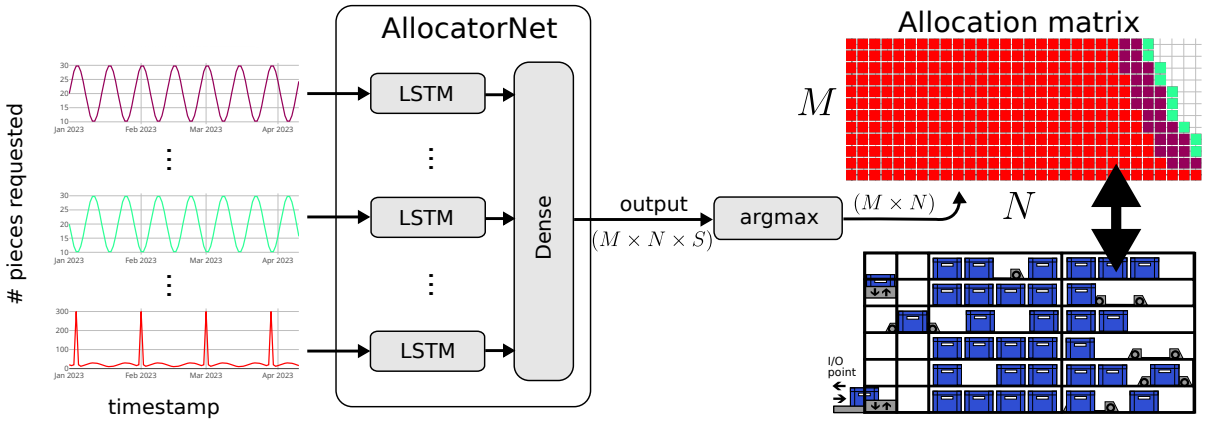

Figure 2: Overview of the proposed, using historical picking order data, the Model will allocate SKUs to storage locations. (right) historical picking order holds pieces requested per SKU. (middle) Our Deep Learning model, (left) the produced allocation matrix. and a representation of the SBS/RS side view.

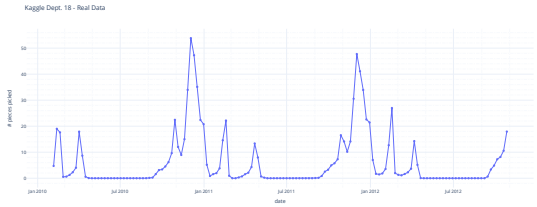

Figure 3: Real dataset example.

SKUs with sporadic, irregular, or even abnormal demand quantities compared to other products. Variations in demand for `Z` SKUs are deemed to be not easily foreseeable nor predictable. During our study we will focus on SKUs on type `Y` and especially on class `Z` as they represents the challenge to handle peaks situations.

Picking order signals ranging over 1460 days (4 years) contains the number of pieces to pick for each SKU per day. Consider one SKU that is regularly ordered (seasonal data class `Y`), represented by a sinusoid generated using the following formula :

$$y(t) = A.\sin(\frac{2\pi}{T}t + \varphi) \quad (1)$$

where the amplitude $A = 10$, the period $T = 18$, and the phase $\varphi = 0$. Another picking order signal consists of the same sinusoid with a phase shift of $\varphi = \pi$. A third picking order signal combines regular orders with peak demand periods (class `Z`) every 28 days (see Figure 4-left).

Input data are generated using the formula described above. Output data for the dataset are generated by allocating SKUs closer to I/O point based on the number of pieces requested. An example of allocation produced by this method is presented in Figure 4 right.

6.2 Protocol

6.2.1 Neural Model Training

We will enforce a supervised training for our model. Input data X are requested pieces per SKU $s \in S$ and output/target data Y are optimal SKUs allocations. Optimal allocation matrices $A(t)$ will be generated by a third-party optimization algorithm using Operational Research model under constraints. In our preliminary experiments, the considered algorithm, allocated SKUs sorted by demand as close as possible to the I/O point.

In order to fit our neural network weights, we uses Mean Squared Error (MSE) as loss function.

For gradient descent we utilize Adamax algorithm, a derived from Adam algorithm based on infinity norm. Adam provides a gradient-based optimization of stochastic objective functions (Kingma and Ba, 2017). We used with the following settings: learning rate $\gamma = 0.002$, betas $\beta_1 = 0.9, \beta_2 = 0.999$ and epsilon $\varepsilon = 1e^{-08}$, with objectives to minimize the gradient and with no weight decay. We use a reduce learning rate technique with a patience of 10 epochs, factor 0.1 in minimizing mode.

We perform no specific pre-processing to our data (i.e. no scaling, normalization nor standardization of the data fed to our deep learning model).

The dataset is splited into 50% for training, for 10%validation and 40% for testing. Training runs are configured to run at least 300 epoch.

6.2.2 Allocation Methods

We propose to compare the allocation of SKUs into storage locations produced by our approach at any timestamp t against four allocation method: Ideal,

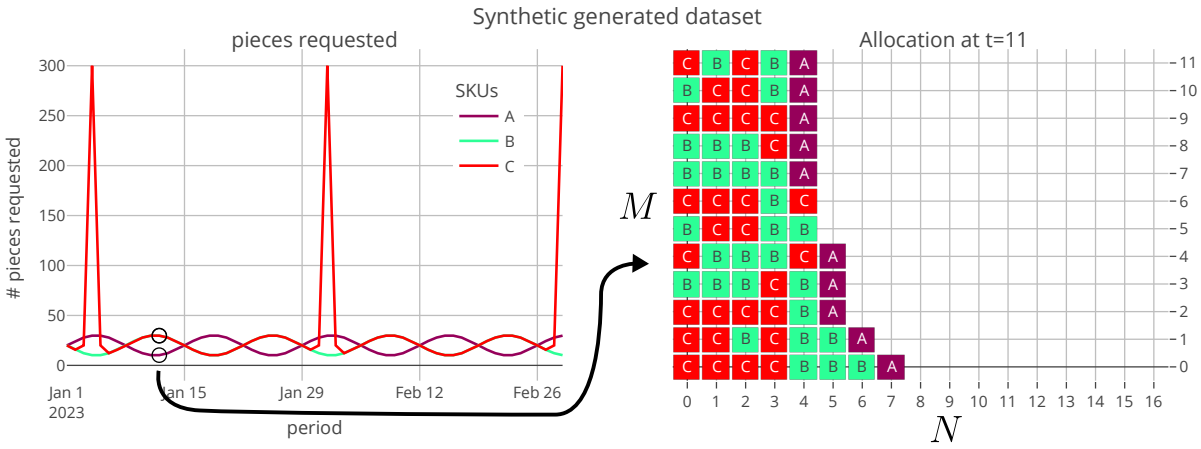

Figure 4: Synthetic dataset overview, input, and output data. (left) First 60 days displayed. SKU A: A perfect sinus generated with a fixed period of 14 days; SKU B, same sinus with $\varphi = \pi$. SKU C Sinus with peaks: same as the Sinus, but with peak (higher demand quantity) appearing every 28 days. (right) Example of SKUs allocation to storage location for $t = 11$.

Mean, Naive and Random.

One trivial storage allocation strategy is considered, which involves assigning frequently picked products to storage locations closer to the SBS/RS's I/O point. SKUs with high demand are placed into forwards locations while slow-moving SKUs are placed at the back. The optimal assignment is computed for each period t using data from the dataset (not inferred data). In the rest of the paper, this ideal allocation is named **Ideal**.

One also proposes to compare our approach with the **Mean** method. This approach consists in assigning SKU using the mean number of past pieces to pick over a given period (using sliding windows pattern). SKUs with higher average demand are placed into forwards locations. This approach is considered in many similar industrial applications.

Another approach is the **Naive** method. For this approach, the dataset is shifted by one period. The input data (number of pieces to pick per SKU) is set to the last observed value. Both method Naive and Random serves as baseline comparisons. Our proposed approach should provide better results to be considered effective.

Eventually we use the **Random** storage allocation policy. This method knows the number of pieces to pick for timestamp t +1, SKUs are randomly assigned to storage locations following uniform distribution.

The Ideal and Random methods always have knowledge of the number of pieces requested per SKU at $t + 1$. The Naive and Mean methods, on the other hand, must infer these values.

6.2.3 Evaluation metrics

To evaluate the performance of different methods, we will use three metrics.

First, we introduce the *Bad Allocation Rate* (BAR) metric to measure the difference in the number of storage locations allocated by each method compared to the Ideal allocation. To calculate BAR, we assign a value of 1 if SKU $s \in S$ is allocated to a location and 0 otherwise. A lower BAR score (close to 0) indicates that the method has allocated storage locations similarly to the Ideal allocation. Conversely, a higher BAR score indicates either under-allocation or over-allocation of storage locations.

The second metric is *Retrieval Time* (RT), which measures the total time in seconds required to retrieve all needed storage totes from SBS/RS storage locations to the I/O point. The retrieval cost for each location is calculated using equations of motion and considers the velocity and acceleration specifications of shuttles and lifts. RT is computed by summing the travel timed of shuttles and the vertical translation time of lifts (see Figure 5). This metric quantifies the impact of the allocation method on the time needed to retrieve SKUs from the SBS/RS. It is important to note that, for our method and Mean, Naive and Random, the RT value is expected to be higher than the value provided by Ideal method.

In some cases, there may not be enough storage locations assigned to pick all requested pieces for a particular SKU. In such cases, when computing the retrieval time, extra storage locations are assigned to that SKU. Moreover, a penalty is applied to these extra storage locations, which is equivalent to double their retrieval time. This penalty aims at mimic the

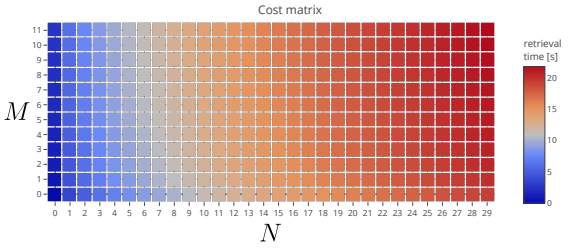

Figure 5: Cost Matrix example. Retrieval time in seconds for each storage location. I/O point is set at level 0 and channel 0. Storage locations closer to this point have the smallest retrieval time.

additional time required to replenish the SBS/RS with needed items, during picking operations.

The last metrics emphasize *Retrieval Time for Peaks* (RTP) timestamp. We compute the retrieval time only for timestamps with peak in demand. As we aim to propose a method capable of handling case of article of cases of class Z SKUs (sudden demand), this metric allows us to illustrate whether the method can detect and react to peaks effectively. It will illustrate the delay between a peak and its forecast, as too few allocations will result in poor metric value.

6.3 Results

In this section we compare the results obtained from the synthetic dataset using four different methods, alongside the retrieval time of the baseline Ideal. The Ideal method is used as a baseline for comparison because it has knowledge of the exact number of pieces to pick at any given time. Other methods being considered must infer this quantity, which introduces uncertainty and potentially leads to worse performance results compared to the Ideal method as the number of pieces to pick is not known in advance.

Table 1 provides the mean BAR value per day, the mean retrieval time (with penalty), and the retrieval time at peak timestamps for each allocation method.

We observe that our approach yields the highest scores for each metrics. A low BAR value indicates the accurate assignment of the right number of locations to SKUs. In terms of retrieval time for both the entire dataset and peak timestamps, our approach consistently outperforms other methods, demonstrating retrieval times that are closer to the ideal allocation than any of the alternative approaches under consideration.

Figure 6 represents an example of allocation provide by the 4 methods for 4 periods. on top, the number of pieces requested for 90 days for each SKU is displayed. Then for 4 periods the allocation provided by the 4 methods are displayed. One in a "normal" daily operation. One just before a peak in demand.

Table 1: Results on synthetic dataset for $N = 3$ SKUs. Bad Allocation Rate (BAR), Retrieval Time (RT) and Retrieval time on peaks (RTP) for each allocation method.

Method	BAR	RT [s]	RTP [s]
Random	0.270	991	4863
Mean	0.060	772	9294
Naive	0.062	721	9158
Our approach	**0.003**	**513**	**4721**
Ideal Allocation	–	505	4721

Another within a peak. Eventually, another represent allocations one day after a peak. This figure illustrates well the peak detection problem by some methods. For the third period considered, $t = 78$ only Random and our approach (using LSTM) provide enough storage location to fulfil the day demand. Mean and Naive are missing allocation. Theses 2 methods do not "react" in time for peak. As well, for the fourth period considered $t = 108$, Mean and Naive methods provided both too many allocations.

In terms of bad allocation rate, the Random method yields a mean BAR of 0.270, indicating incorrect SKU placements in the Storage and Retrieval System (SBS/RS) in 27% of cases. The Mean and Naive methods exhibit closely comparable results with mean BAR values of 0.060 and 0.062, respectively, implying erroneous allocations in the SBS/RS for 6% of cases.

In contrast, our proposed approach delivers superior performance, achieving a mean BAR of 0.003, corresponding to incorrect SKU placements in the SBS/RS in only 0.3% of cases. As illustrated in Figure 6, a specific instance of allocation error for our approach is evident at $t = 49$, where our method allocates two locations less than the ideal allocation.

Concerning retrieval time. Random provide the worst result with 991 seconds in average, that is almost double the ideal retrieval time (505 seconds). Then Mean and Naive have average values of 772 seconds and 721 seconds, more than Ideal. Our approach has a retrieval time of 513 seconds in average, very close from the Ideal 505 seconds. Naive and Mean methods suffer from penalty added to retrieval time. Because they often allocate to few locations, a penalty is applied to the retrieval time of the missing storage locations. Random always allocated the right number of storage location, but because of the dispersion of such allocation (Random picks storage location anywhere on the available SBS/RS locations matrix) the retrieval of retrieval is very high compare to other methods

Concerning retrieval time at peaks timestamp, Mean and Naive methods perform significantly worse than both random and our strategies, with respect to

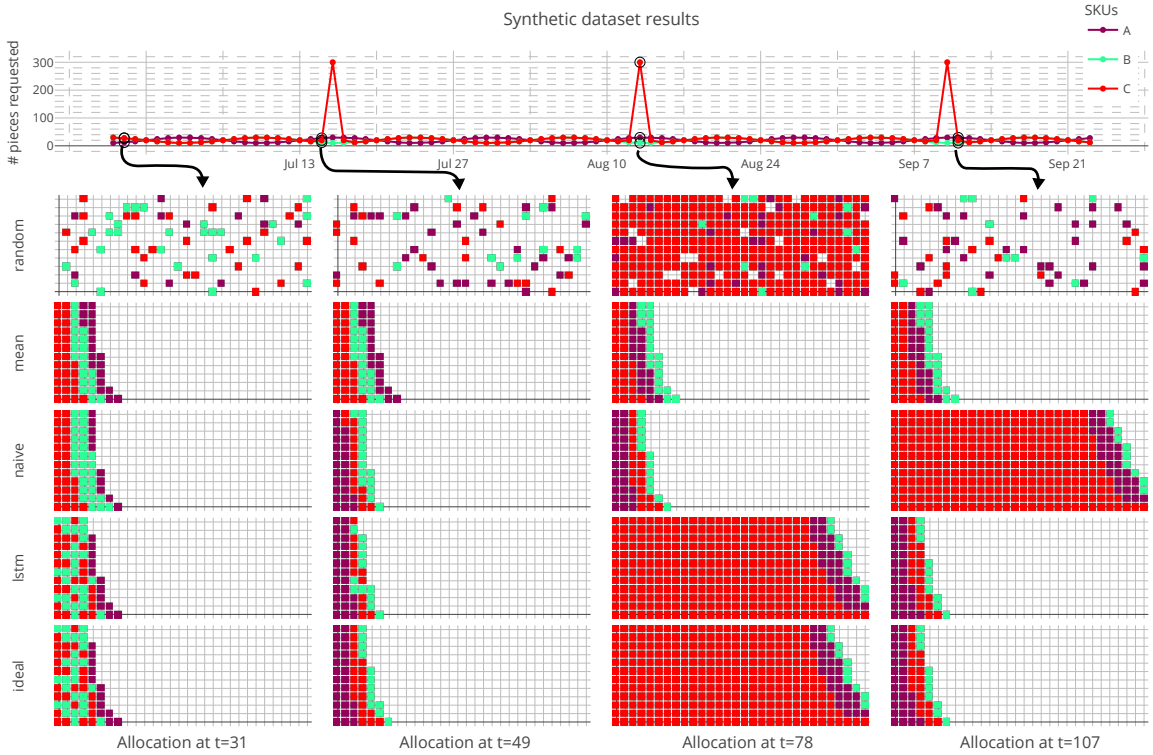

Figure 6: Synthetic dataset results.(top) Number of pieces requested for 90 days for each SKU.(bottom) Allocation from each method and ideal allocation for 4 timestamp. One in a "normal" daily operation. One just before a peak in demand. Another within a peak. Eventually, one day after a peak.

the Ideal allocation. Their mean values are 9286 seconds and 9156 seconds respectively. They perform badly because of missing allocations, see timestamp $t = 78$ in figure 6, they suffer from penalties added to their retrieval time. the Theses 2 methods provide worst results than Random method. Random performs better, its mean value is 4863 seconds. Because it always allocated the right number of storage locations, its computed retrieval time is not impacted by penalties. Our approach as a value of 4721 seconds, matching the Ideal value, this is the best method considering the retrieval time.

Experiments conducted on synthetic datasets have demonstrated that our Deep Learning method, utilizing LSTM, exhibits fewer errors in allocation and achieves superior retrieval times, particularly during peak demand periods, when compared to alternative methods under consideration. Ideal method is logically best suited if one has the knowledge of the number pieces to pick a any timestamp (this being not the case in practice). When this quantity must be inferred our method provided better results than the other considered methods.

Consequently, it is reasonable to assert that this method is well-suited to address the challenges posed by the SLAP. Moreover, this approach is expected to effectively handle the challenge of responding to peaks in demand when processing picking orders.

7 CONCLUSION

We have proposed a new data-driven approach to address the Storage Location Assignment Problem (SLAP) using a Deep Learning model. Our proposed method generates assignments of SKUs to storage locations based on historical picking orders. Preliminary results obtained on a synthetic dataset are promising, although they are limited by the small number of SKUs and storage locations considered.

In our future work, we plan to utilize Operations Research techniques to generate assignment matrices in more complex scenarios involving operational constraints such as weight distribution, balancing of stock between levels, security distance between flammable and combustible SKUs, and specific storage policies for hazardous , luxurious or fragile goods.

Additionally, we aim to scale up the number of SKUs handled by our model. Ultimately, we will

evaluate the performance of our Deep Learning-based allocation model using real-world data to demonstrate its effectiveness in practical applications.

ACKNOWLEDGEMENTS

Funding: All research in this study was funded by KNAPP France. There was no external funding. Competing interests: This work was done in the course of employment at KNAPP France, with no other competing financial interests. Data and materials availability: This work used publicly available data from Kaggle

REFERENCES

Antomarioni, S., Lucantoni, L., Ciarapica, F. E., and Bevilacqua, M. (2021). Data-driven decision support system for managing item allocation in an asrs: A framework development and a case study. *Expert Systems with Applications*, 185:115622.

Bolaños Zuñiga, J., Saucedo Martínez, J. A., Salais Fierro, T. E., and Marmolejo Saucedo, J. A. (2020). Optimization of the storage location assignment and the picker-routing problem by using mathematical programming. *Applied Sciences*, 10(2):534.

Cai, M., Pipattanasomporn, M., and Rahman, S. (2019). Day-ahead building-level load forecasts using deep learning vs. traditional time-series techniques. *Applied Energy*, 236:1078–1088.

Frazelle, E. H. (1989). *Stock location assignment and order picking productivity*. PhD thesis, Georgia Institute of Technology.

Goetschalckx, M. and Ratliff, H. D. (1990). Shared storage policies based on the duration stay of unit loads. *Management Science*, 36(9):1120–1132.

Hausman, W., Schwarz, L., and Graves, S. (1976). Optimal Storage Assignment in Automatic Warehousing Systems. *Management Science*, 22(6).

Kazemi, M., Asef-vaziri, A., and Shojaei, T. (2019). Concurrent optimization of shared location assignment and storage/retrieval scheduling in multi-shuttle automated storage and retrieval systems. *IFAC-PapersOnLine*, 52(13):2531–2536.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization.

Kofler, M. (2014). *Optimising the Storage Location Assignment Problem Under Dynamic Conditions*. PhD thesis, Johannes Kepler Universität Linz.

Li, M. L., Wolf, E., and Wintz, D. (2019). Duration-of-stay storage assignment under uncertainty.

Nowotyńska, I. (2013). An application of XYZ analysis in company stock management. *Modern Management Review*.

Park, J., Park, C., and Hong, S. (2023). Gaussian process-based storage location assignments with risk assessments for progressive zone picking systems. *Computers & Industrial Engineering*, 185:109700.

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

Schenone, M., Mangano, G., Grimaldi, S., and Cagliano, A. C. (2020). An approach for computing as/r systems travel times in a class-based storage configuration. *Production & Manufacturing Research*, 8(1):273–290.

Stojanović, M. and Regodić, D. (2017). The significance of the integrated multicriteria ABC-XYZ method for the inventory management process. *Acta Polytechnica Hungarica*, 14(5):20.

Talbi, E.-G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research*, 240(1):171–215.

Zarinchang, A., Lee, K., Avazpour, I., Yang, J., Zhang, D., and Knopf, G. K. (2023). Adaptive warehouse storage location assignment with considerations to order-picking efficiency and worker safety. *Journal of Industrial and Production Engineering*, 0(0):1–20.

Zhang, R.-Q., Wang, M., and Pan, X. (2019). New model of the storage location assignment problem considering demand correlation pattern. *Computers & Industrial Engineering*, 129:210–219.