On the Feasibility of using Neural Networks as High-level solvers for the Pod Allocation Problem within Robotic Mobile Fulfillment Systems

Maria Torcoroma Benavides-Robles, Jorge M. Cruz-Duarte,
Ivan Amaya, and José C. Ortiz-Bayliss
Tecnologico de Monterrey, School of Engineering and Sciences, Monterrey 64849, Mexico,
E-mails: {A00836554, jorge.cruz, iamaya2, jcobayliss}@tec.mx

***Abstract*—The increasing need for automation in various industrial processes has created opportunities for incorporating robots. In this regard, a Robotic Mobile Fulfillment System (RMFS) is a collaborative environment where robots deliver products to humans to fulfill orders. The RMFS is a complex problem that integrates various optimization scenarios. In this work, we analyze the feasibility of using the multi-layer perceptron to implement an algorithm selector to improve how we solve the Pod Allocation Problem, one of the challenging subproblems within the RMFS. Our work covers 208 RMFS instances, which are solved with six Pod Allocation Solvers and with the algorithm selectors produced through our approach. Our experiments indicate that this approach leads to competent algorithm selectors. Moreover, such algorithm selectors can rival the best low-level solvers under some particular conditions.**

***Index Terms*—RMFS, Robotic Mobile Fulfillment System, Pod Allocation Problem, RAWSim-O simulator.**

I. Introduction

Online ordering is an excellent alternative since people can buy products from anywhere. However, as products offered by a single seller (*e.g.*, Amazon) broaden, it becomes more complex to handle everything. This not only applies to being able to process more payments in the same time window. It also calls for bigger warehouses. In turn, the difficulty of handling and fulfilling orders increases. So, automated warehouses become paramount.

Although automated warehouses vary [1], in this work, we only consider those related to Robotic Mobile Fulfillment Systems (RMFS). In such systems, automated guided vehicles (AGVs) deliver products to packing stations (workstations) so humans can remain at the same location while serving the orders. Hence, AGVs are in charge of storing and removing products from their storage locations. Products are transported by lifting the pods containing them so that several products can be transported simultaneously. Wurman *et al*. proposed this system in 2008, and they called it the *Kiva system* [17]. Its popularity grew due to the increased warehouse efficiency, and it was eventually rebranded as RMFS.

Maria Benavides thanks the Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCyT) for the financial support provided through her fellowship 866896. The authors also thank CONAHCyT for the financial support given through grant 287479.

There are few works about RMFS. Most of them analyze the system's sensitivity to different parameters, including the influence of locating workstations at diverse places [4] and the effect of distributing products differently [7]. It also covers the sequencing of orders [15]. Other works on RMFS have targeted its interaction with AI techniques. In this sense, works relate to clustering orders and products based on similarity metrics [20] and reinforcement learning as a high-level solver [3]. Nonetheless, such a solver was developed for selecting among different collision-avoidance approaches.

As stated previously, research on RMFS is scarce. Moreover, the literature has not yet explored the feasibility of using simple neural networks as high-level solvers within this domain. Hence, this work aims to fill such a knowledge gap. We do so by focusing on the Pod Allocation Problem (PAP), for simplicity, and by formulating the PAP from an Algorithm Selection Problem (ASP) perspective [10], [11] Thus, we propose a high-level strategy based on the algorithm portfolio idea [18]. In this sense, we train a neural network to learn those patterns that match each instance configuration with the best available solver.

This document is organized as follows. Section II glimpses at the main concepts related to our work, as well as at the current literature. Section III describes our proposed solution model. We present the experiments and results in Section IV. Finally, Section V synthesizes the main insights from this work and lays out some paths for future research.

II. Background and Related Work

In this section, we provide a brief overview of the problem domain and the machine learning model we use as an algorithm selector.

A. Robotic Mobile Fulfillment Systems (RMFS)

These systems emerge from the complex interaction of diverse subproblems within an automated warehouse. Hence, simple variables like the number of robots, pods, products, and humans now interact across multiple problem domains. For example, the number of robots affects how their paths must be planned (more robots, more risk of collisions) and also

their scheduling to fulfill orders (more robots, more pickers available).

Considering the RMFS as a whole is, then, usually futile. To circumvent this, researchers usually tackle a component of the RMFS and simplify the remaining ones. In an overall sense, we may classify the subproblems within RMFS as follows:

- **Path planning**. Relates to all routing processes. This includes picking up pods, taking them to workstations, and returning them to storage. Note that it also covers the routing of a pod to a replenishment station for refilling its stock.
- **Zoning**. Relates to the layout of all the zones within a warehouse, including those for workstations, pods, and replenishment. It also covers the layout of waiting and charging zones for the AGVs.
- **Assignment**. Considers the distribution of all elements within the warehouse. Hence, the way in which products are distributed on pods, and the scheduling of orders into workstations are included in this subproblem.

Since this is a complex problem, solution approaches have also been somewhat convoluted. Some hybrid approaches include the combination of simple approaches with metaheuristics [13], [21]. Others relate to more complex endeavors [22]. Simulations within RMFS have analyzed the effects of changes in the storage layout [6] and warehouse configuration [16]. Authors have also considered Machine Learning algorithms [9], [20] and Mathematical Programming techniques [14].

Finally, it is worth commenting about available simulating environments in the literature. Nowadays, the only framework that remains supported is RAWSim-O [8]. This framework was developed by Merschformann *et al.*, seeking to extend a previous approach known as Alphabet Soup [5], which is no longer maintained. RAWSim-O offers many customization options and seeks to lessen the gap between simulation and reality.

B. Neural Networks

Neural Networks have recurrently appeared in the literature as a way to address various problems related to pattern recognition, forecasting, optimization, function approximation and control, to mention a few [2]. Although many types of neural networks exist [2], [19], in this work we focus on the Multi-Layer Perceptron (MLP).

An MLP comprises a set of interconnected layers in which the inputs lead to outputs using a series of weighted edges and nodes (neurons). The remaining (internal) layers are usually referred to as *hidden layers*. Within an MLP, all the nodes from one layer are connected to all nodes in the next layer through the weighted edges. Moreover, the output of each neuron is based on the *activation function* of such a neuron, which operates over the dot product between its inputs and the weights of the corresponding edges. Hence, the training process of an MLP consists on adjusting the weights of the edges based on the input data and the expected outputs.

III. Solution Approach

This work relies on the RAWSim-O framework [8] to simulate the warehouse and all the inner processes required for order completion. For fairness, we fixed the simulation time to five hours. As mentioned, the RMFS involves many variables easily defined within the framework, such as the layout distribution, the number of robots, the total SKUs, and the techniques used to solve each subproblem. Moreover, the framework internally generates all the products and orders with this information. Particularly, we focused on the PAP, part of the Path Planning category of subproblems within the RMFS. With this information, we could monitor the effect of selecting different techniques to solve such a subproblem.

Our solution model aims to take advantage of the difference in performance exhibited by different solvers for the PAP. The rationale is simple: if the solvers exhibit different performances when tested on different instances, we can improve the search by selecting the most suitable solver whenever we solve a particular instance. Our solution proposal uses MLPs as algorithm selectors. Through the training process, the neural networks learn to classify the input (the characterization of the problem state). In this context, the class returned by the network represents the solver to use. Then, once the network is trained, it can recommend one suitable solver for the PAP when a particular problem state occurs.

The following lines contain relevant information about various aspects of our solution model.

A. Problem Characterization

In this work, we characterize the instances using six straightforward features:

- **Replenishment stations**. It refers to the number of refueling stations. We assumed that they are located on the left side of the warehouse.
- **Workstations**. It represents the total number of workstations. We assumed that they are located on the right side of the warehouse.
- **Bots**. It represents the total number of robots available to transport pods.
- **Pods**. It indicates the total number of pods within the warehouse, which covers 85% of the available storage space (to allow for resorting of the warehouse).
- **Bots per station**. It represents the ratio between bots and stations (includes both, workstations and replenishment stations).
- **Horizontal aisles**. It refers to the number of horizontal aisles that the warehouse contains.

We rescaled all these features using the standard procedure defined by (1). Through this process, we guarantee that the values of such features lie in the range from 0 to 1.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{1}$$

B. Solvers

Among the options for solving the PAP within the RMFS, we have included the following solvers:

- **Random** places pods in random positions.
- **Fixed** stores pods at locked (fixed) positions.
- **Nearest** allocates pods based on distance information.
- **Station-based** stores pods close to workstations.
- **Cache** keeps pods that may be required in the near future, close to workstations.
- **Turnover** assigns positions to returning pods using item-frequency information.

For clarity, we will refer to them as low-level solvers since they work directly on the PAP to produce a solution.

C. RMFS Instances and Performance Metrics

Using the RAWSim-O framework [8], we generated and solved 208 RMFS instances with the six PAP solvers previously described. We randomly created the training and test sets from those instances, with 70% and 30% of the instances, respectively. All the networks were trained using the training set. Due to space restrictions, we only report the results for the test set.

Regarding the quality of the solutions, we have considered two performance metrics:

- **Throughput time**. It indicates how long an order takes to exit the system.
- **Orders processed**. It represents the number of orders completed within the five-hour simulation window.

D. MLP-based Algorithm Selector

Once we have described all the supporting elements, we can move on to the core of our proposal. We generated two algorithm selectors to test the proposed approach as described previously. Such algorithm selectors rely on MLPs. The algorithm selectors resulted from two independent processes, each focusing on one particular performance metric. As recommended when working with supervised learning tasks, we trained both networks exclusively on the training set. For simplicity, we used the default training configuration of the library `neuralnet` in R. Regarding the topology of the networks, they contain six neurons in each of the three layers (input, hidden, and output), and the neurons in the hidden and output layers also receive a bias as input. Although we consider that using some kind of hyper-parameter tuning strategy might improve the performance of the networks (and the algorithm selectors they represent, as a consequence), we have used a straightforward parameter setting based on our previous experience and some preliminary tests.

Figure 1 depicts the overall structure of the Multi-Layer Perceptrons used as algorithm selectors in this work. The six neurons in the input layer are used to read the problem state (defined by the six features described in Section III-A and the six neurons in the output layer indicate the low-level solver to use (see Section III-B for more details on the solvers). The neuron with the highest output value in the output layer is the one that determines the solver to use.

IV. Experiments and Results

As mentioned previously, we only report the results obtained on the test set.

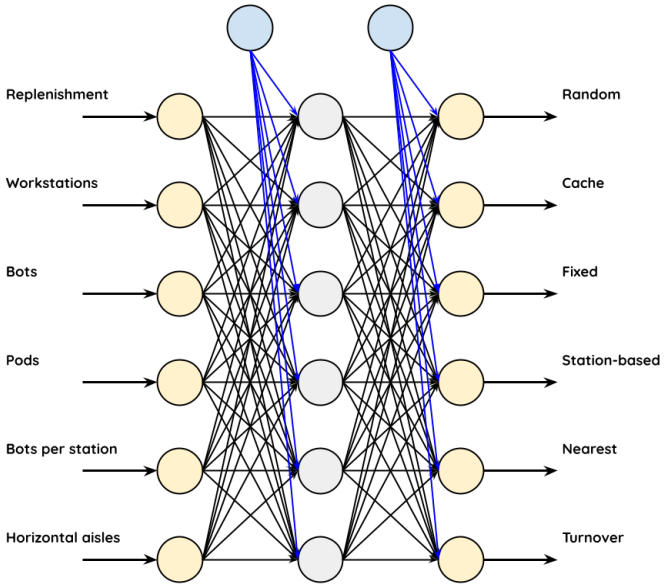

Fig. 1. An algorithm selector coded within a Multi-Layer Perceptron model. The network receives the features that characterize the problem and returns a suitable low-level solver for the PAP.

A. Analysis of the Throughput Time

The first algorithm selector, NNAS-T, uses a neural network to choose the low-level solver that best solves the instance according to the throughput time metric. We trained the neural network as a classifier capable of matching the features of the problem instance to one suitable low-level solver. Then, we used such a network as an algorithm selector to solve the instances in the test set. It is worth noting that NNAS-T ignores Cache and Station-based as available low-level solvers since it never uses them via the solving process.

We can observe the behavior of both Nearest and NNAS-T in the test set in Fig. 2. In such a figure, we can appreciate the cases where NNAS-T improves the throughput time obtained by Nearest. In fact, NNAS-T produces a throughput time shorter or equal to Nearest in 47.5% of the instances in the test set, representing a marginal saving of around 100-time units concerning Nearest (0.42%). To validate our results, we conducted a two-tailed non-parametric Wilcoxon test on the throughput times of NNAS-T and Nearest on the test set. In this test, the null hypothesis (H_0) states that the actual median of the throughput time is equal to the median of Nearest, and the alternative hypothesis (H_1) states that the actual median of the throughput time is different for NNAS-T and Nearest. With this test, we can state, with 5% of significance (a p-value of 0.7432), that the actual median time of NNAS-T equals the actual median time of Nearest. In other words, the two methods are statistically equivalent in performance when the throughput time is considered.

Regarding accuracy, which indicates the percentage of cases where NNAS-T obtains the best possible result among all the low-level solvers, NNAS-T obtains a modest 36.06%. Although the accuracy obtained by NNAS-T may seem small

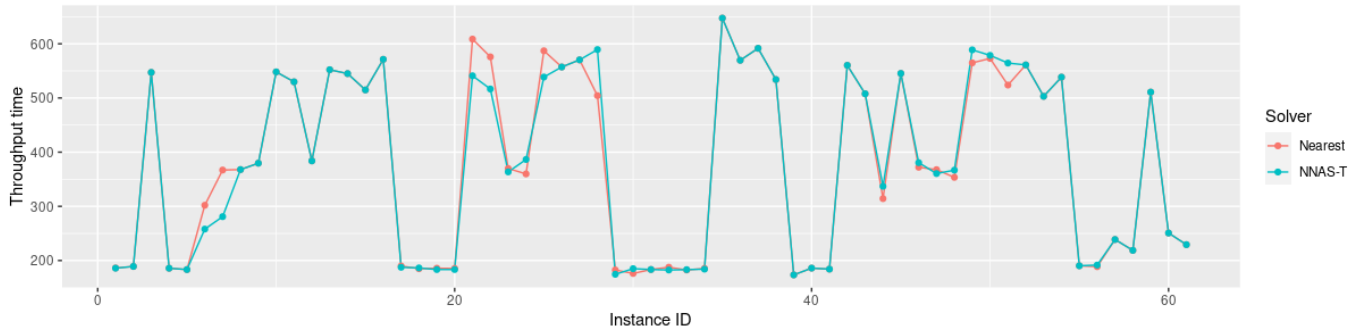

Fig. 2. Performance of NNAS-T on the test set when compared against Nearest, the best low-level solver in the test set when considering the throughput time as performance metric.

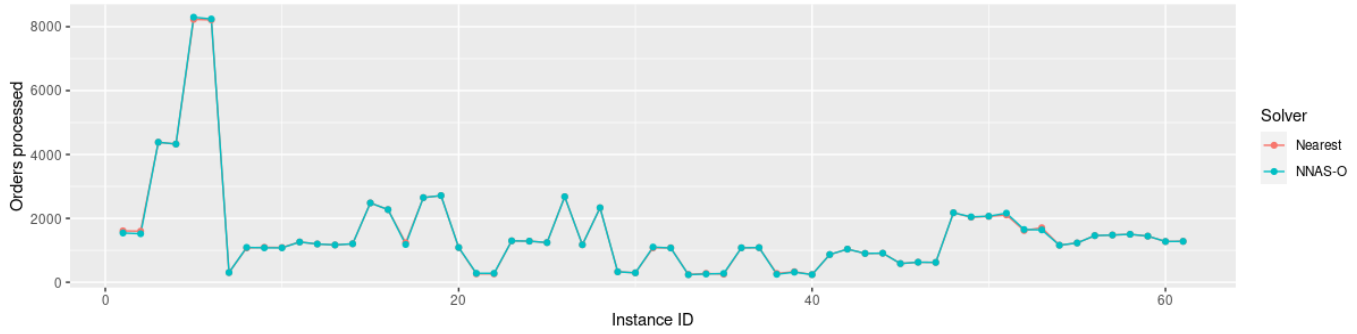

Fig. 3. Performance of NNAS-O on the test set when compared against Nearest, the best low-level solver in the test set when considering the number of orders processed as performance metric.

at first sight, it is enough to produce a competitive performance that rivals Nearest, the best low-level solver in terms of throughput time, as previously shown.

B. Analysis of the Orders Processed

In the case of the number of orders processed, we trained a new neural network, this time as a classifier that receives the features of the problem instance as inputs and returns a low-level solver, aiming at maximizing the number of orders processed. As we did for the throughput time metric, we used this network as an algorithm selector to solve the instances in the test set. The resulting algorithm selector, NNAS-O, excludes Fixed from the list of available low-level solvers (Fixed is never used by the selector). When we analyze the cost of using NNAS-O on the test set, it processes 68 orders less than Nearest, the best low-lever solver for the test set in terms of the number of orders processed. Regardless of this result, NNAS-O behaves at least as well as Nearest in 81.96% of the instances in the test set. Regarding its accuracy, NNAS-O chooses the best low-level solver in 47.54% of the cases.

Figure 3 depicts the behavior of both Nearest and NNAS-O in the test set. In such a figure, we can observe the various cases where NNAS-O improves the number of orders processed by Nearest. We conducted a two-tailed non-parametric Wilcoxon test on the number of orders processed of NNAS-O and Nearest on the test set. In this test, the null hypothesis (H_0) states that the actual median of the number of orders processed equals that of Nearest. The alternative hypothesis (H_1) states that the actual median of the number of orders processed differs for NNAS-O and Nearest. With this test, we can state, with 5% of significance (a p-value of 0.9959), that the actual median orders processed by NNAS-O and Nearest are the same. In other words, both methods are statistically equivalent in performance when the number of orders processed is considered.

V. Conclusion

In this work, we analyzed the feasibility of using a neural network as a high-level solver for improving the performance of Robotic Mobile Fulfillment Systems (RMFS). To this end, we produced two neural networks and used them as algorithm selectors for the Pod Allocation Problem (PAP) within RMFS. We tested such an approach on 208 problem instances.

Our data revealed that multi-layer perceptrons can learn the patterns in the problem states. Hence, they can recommend a suitable low-level solver for addressing this problem. In other words, these networks are suitable for creating a mapping between instance features and low-level solvers. Besides, the statistical evidence suggests that the algorithm selectors produced throughout our approach are equivalent in performance to the best solvers according to two different performance metrics: throughput time and number of orders processed.

Besides that, our proposed approach even manages to improve upon some of the instances.

Our work shall continue, as several paths lie before us. One relates to detailing the behavior of the neural network and its sensitivity. Another one is to consider more complex network architectures so that it may learn other kinds of patterns. For example, it is intriguing how the model would behave if recurrent or adversarial neural networks are used instead of the multi-layer perceptron. A third approach relates to extending the dataset by integrating more scenarios. Finally, our proposed approach can be extended. It may permeate to other subproblems, such as order or pod sequencing. However, it may also grow in complexity, allowing for multiple decisions when solving an instance. This approach, commonly referred to as hyper-heuristics [12], allows more complex solving processes that also allow for better solutions. Alas, they are also more complex to develop. Nonetheless, we deem this next step mandatory for furthering research on RMFS.

REFERENCES

[1] Kaveh Azadeh, René De Koster, and Debjit Roy. Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, 53(4):917–945, 2019.

[2] Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

[3] Yinon Douchan and Gal A. Kaminka. *The Effectiveness Index Intrinsic Reward for Coordinating Service Robots*, volume 6. 2018.

[4] Lijuan Feng, Xinglu Liu, Mingyao Qi, Shijia Hua, and Qingte Zhou. Picking Station Location in Traditional and Flying-V Aisle Warehouses for Robotic Mobile Fulfillment System. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 1436–1440. IEEE, dec 2018.

[5] Christopher J. Hazard, Peter R Wurman, and Raffaello D'Andrea. Alphabet Soup: A Testbed for Studying Resource Allocation in Multi-vehicle Systems. *Proceedings of the AAAI Workshop on Auction-Based Robot Coordination*, 2006.

[6] Thomas Lienert, Tobias Staab, Christopher Ludwig, and Johannes Fottner. Simulation-based performance analysis in robotic mobile fulfilment systems analyzing the throughput of different layout configurations. *SIMULTECH 2018 - Proceedings of 8th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, (Simultech):383–390, 2018.

[7] Zhongqiang Ma, Guohua Wu, Bin Ji, Ling Wang, Qizhang Luo, and Xinjiang Chen. A Novel Scattered Storage Policy Considering Commodity Classification and Correlation in Robotic Mobile Fulfillment Systems. *IEEE Transactions on Automation Science and Engineering*, pages 1–14, 2022.

[8] Marius Merschformann, Lin Xie, and Hanyi Li. RAWSim-O: A simulation framework for robotic mobile fulfillment systems. *Logistics Research*, 11(1):1–11, 2018.

[9] Yaxu Niu and Frederik Schulte. Human Aspects in Collaborative Order Picking – What if Robots Learned How to Give Humans a Break? pages 541–550. Springer International Publishing, 2021.

[10] José C. Ortiz-Bayliss, Ivan Amaya, Jorge M. Cruz-Duarte, Andres E. Gutierrez-Rodriguez, Santiago E. Conant-Pablos, and Hugo Terashima-Marín. A general framework based on machine learning for algorithm selection in constraint satisfaction problems. *Applied Sciences*, 11(6), 2021.

[11] John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[12] Melissa Sanchez, Jorge M. Cruz-Duarte, Jose Carlos Ortiz-Bayliss, Hector Ceballos, Hugo Terashima-Marin, and Ivan Amaya. A systematic review of hyper-heuristics on combinatorial optimization problems. *IEEE Access*, 8:128068–128095, 2020.

[13] Yangjun Sun, Ning Zhao, and Gabriel Lodewijks. An autonomous vehicle interference-free scheduling approach on bidirectional paths in a robotic mobile fulfillment system. *Expert Systems with Applications*, 178:114932, sep 2021.

[14] Sander Teck and Reginald Dewil. Optimization models for scheduling operations in robotic mobile fulfillment systems. *Applied Mathematical Modelling*, 111:270–287, nov 2022.

[15] Cristiano Arbex Valle and John E Beasley. Order allocation, rack allocation and rack sequencing for pickers in a mobile rack environment. *Computers & Operations Research*, 125:105090, jan 2021.

[16] Shasha Wu, Cheng Chi, Wei Wang, and Yaohua Wu. Research of the layout optimization in robotic mobile fulfillment systems. *International Journal of Advanced Robotic Systems*, 17(6):172988142097854, nov 2020.

[17] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *AI Magazine*, volume 29, pages 9–19, 2008.

[18] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for sat. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.

[19] Neha Yadav, Anupam Yadav, Manoj Kumar, Neha Yadav, Anupam Yadav, and Manoj Kumar. History of neural networks. *An introduction to neural network methods for differential equations*, pages 13–15, 2015.

[20] Ning Yang. Evaluation of the Joint Impact of the Storage Assignment and Order Batching in Mobile-Pod Warehouse Systems. *Mathematical Problems in Engineering*, 2022:1–13, apr 2022.

[21] Ruiping Yuan, Juntao Li, Wei Wang, Jiangtao Dou, and Luke Pan. Storage Assignment Optimization in Robotic Mobile Fulfillment Systems. *Complexity*, 2021:1–11, nov 2021.

[22] Bipan Zou, Yeming (Yale) Gong, Xianhao Xu, and Zhe Yuan. Assignment rules in robotic mobile fulfilment systems for online retailers. *International Journal of Production Research*, 55(20):6175–6192, 2017.