Order sequencing in automated storage and retrieval systems: A neural network-assisted simulation-optimisation approach

Computers & Industrial Engineering 210 (2025) 111537

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Order sequencing in automated storage and retrieval systems: A neural network-assisted simulation-optimisation approach

Andrea Ferrari [a,*], Canan Gunes Corlu [b], Giulio Mangano [a]

[a] Department of Management and Production Engineering, Politecnico di Torino, Torino, Italy
[b] Metropolitan College, Boston University, Boston, MA, USA

ARTICLE INFO

Dataset link: https://github.com/log-lab-polito/MetaSimOpt

Keywords:
Automated storage and retrieval systems
Order sequencing
Simulation optimisation
Metamodelling
Neural networks
Particle swarm optimisation

ABSTRACT

Automated warehouses are critical to modern supply chains. One type of automation is the automated storage and retrieval system (AS/RS), which improves efficiency, reduces operational costs, and enhances inventory control. Among several optimisation problems in AS/RS, this paper focuses on the order sequencing problem (OSP) in a multi-level shuttle system, which involves finding the optimal sequence of a given number of picking orders to minimise the makespan. Because the objective function of the OSP is difficult to formulate mathematically, stochastic simulation is often used for evaluation, despite its potentially high computational cost. To address this challenge, a neural network-based metamodel (or surrogate model) is proposed to approximate the objective function. The predictive capabilities of recurrent neural networks (RNN), long short-term memory networks (LSTM), and gated recurrent units (GRU) were evaluated, with GRU identified as the most effective in predicting the makespan required to complete the order set. The GRU model was then embedded into an improved particle swarm optimisation (PSO) algorithm designed to solve the OSP. Experimental results demonstrated the effectiveness of the GRU-based PSO approach, showing competitive performance against established methods such as first-in-first-out, random search, multi-start simulated annealing, genetic algorithm, and standard PSO.

1. Introduction

In modern supply chains, automated warehouses have been used to increase the operational efficiency and to enhance the responsiveness of logistics services (Marolt et al., 2022; Zhen et al., 2025). A first example of this automation can be found in the automated storage and retrieval systems (AS/RS), which handle the storage and retrieval of unit loads (ULs) without human intervention (Azadeh et al., 2019). Their adoption spans various sectors, such as manufacturing, large-scale distribution, e-commerce, and healthcare. AS/RS are known to reduce labour costs, increase space utilisation, and improve the speed of retrieval and inventory control (Finco et al., 2023). According to a recent market analysis conducted by PrecedenceResearch (2024), the benefits of AS/RS are boosting their employment–the global warehouse automation market size accounted for USD 21.81 billion in 2024 and is expected to be worth around USD 95.45 billion by 2034, at a compound annual growth rate of 15.9% from 2024 to 2034.

Numerous approaches have been explored to investigate and optimise AS/RS. Several studies have focused on using analytical models (Lehmann & Hußmann, 2021), or simulation approaches (Epp et al., 2017) to provide accurate performance measures for specific system configurations. Other studies have concentrated on optimising specific processes, such as task scheduling, through heuristic (Li et al., 2020) or metaheuristic (Fandi et al., 2022) techniques. To overcome the limitations of each single methodology, researchers developed approaches that combine two or more techniques in an integrated framework. Within this line, simulation-optimisation (SO) represents a promising way to solve complex, uncertain, and dynamic problems (Boareto et al., 2025). In particular, the integration of simulation and optimisation offers a robust framework for addressing complex operations research problems. By combining the descriptive power of simulation with the prescriptive capability of optimisation, this approach enables more informed decision-making, improves solution accuracy, and increases adaptability, ultimately leading to more efficient and dependable outcomes (Nili et al., 2021). SO, also called simulation-based optimisation or optimisation-via-simulation, can be defined as a group of methods used to optimise stochastic simulations (Abid & Mhada, 2021). Compared to algebraic model-based mathematical programming, SO does not assume that an algebraic description of the simulation is available (Amaran et al., 2016).

* Corresponding author.
E-mail address: andrea.ferrari@polito.it (A. Ferrari).

https://doi.org/10.1016/j.cie.2025.111537
Received 29 November 2024; Received in revised form 11 September 2025; Accepted 15 September 2025
Available online 24 September 2025

List of Acronyms

AS/RS	Automated storage and retrieval system	MTTR	Mean time to repair
CPSO	Canonical particle swarm optimisation	MSSA	Multi-start simulated annealing
DES	Discrete event simulation	NN	Neural network
FIFO	First-in-first-out	OSP	Order sequencing problem
GA	Genetic algorithm	PS	Picking station
GRU	Gated recurrent unit	PSO	Particle swarm optimisation
HM	Handling machine	RNN	Recurrent neural network
IPSO	Improved particle swarm optimisation	RQ	Research question
LSTM	Long short term memory	RS	Random search
MAE	Mean absolute error	SA	Simulated annealing
MAPE	Mean absolute percentage error	SKU	Stock keeping unit
MLS	Multi-level shuttle	SO	Simulation-optimisation
MTBF	Mean time between failures	UL	Unit load

This paper presents a comprehensive approach to address the order sequencing problem (OSP) in AS/RS by leveraging SO. The OSP, which aims to determine the optimal sequence of multiple picking orders while minimising the makespan, poses a significant challenge in automated warehousing processes. In fact, order picking, which is the process of retrieving the required items from their storage locations (Jamili et al., 2025), is one of the most time-consuming and resource-intensive operations in warehouses and distribution centres (Ma et al., 2025). Thus, efficient order picking sequencing is crucial for minimising operational costs, enhancing productivity, and improving customer service (Ko & Han, 2022). Simulation becomes essential in solving the OSP because its objective function is hard to formulate mathematically (Suemitsu et al., 2022). Furthermore, the uncertainty and variability inherent in any industrial system, and thus in automated warehouse processes, adds an additional layer of difficulty to optimisation problems, making the associated models particularly difficult to solve (Juan et al., 2023). Therefore, SO methods have been empirically validated in the literature as valuable tools for exploring diverse system configurations in complex logistics and manufacturing systems characterised by inherent uncertainty (Li et al., 2009). However, the traditional simulation in an SO framework tends to be computationally expensive, which limits the use of the approach in practice. To speed up the solution process, metamodel-based SO arises as an alternative approach, which is also the approach used in this paper. The main idea is to approximate the objective function with a metamodel, or surrogate model, that is computationally less expensive than a traditional simulation model. Metamodels are often built on simulation output to provide quick predictions without the need to run time-consuming simulations (Xu et al., 2015). Simulation metamodelling plays a crucial role in supporting tactical decision-making under tight time constraints and in accelerating SO by enabling efficient evaluation of performance metrics. Among the various approaches available for metamodelling and metamodel-based optimisation, neural networks (NNs) have demonstrated particular effectiveness, due to their capacity to capture complex, non-linear relationships between inputs and outputs based on simulation data (Haas, 2024). Recently, Ferrari and Corlu (2024) illustrated the power of replacing a discrete event simulation (DES) model with a NN-based metamodel for order picking in automated warehouses.

Thus, this paper has the objective to answer the following research questions (RQs):

1. Is it possible to develop a metamodel to accurately predict the makespan for a given set of orders in an AS/RS?
2. What is the value of using a metamodel versus a traditional simulation model?
3. Can the OSP be effectively addressed using NN-assisted SO techniques within the context of AS/RS?

This paper presents a NN-based SO approach to solve the OSP in a multi-level shuttle (MLS) system, which is a particular configuration of a mini-load AS/RS used to handle ULs in space-constrained environments. As part of the NN-based metamodelling, recurrent neural network (RNNs), long short-term memory network (LSTM), and gated recurrent unit (GRU) are employed due to their ability to handle sequential data (i.e., the sequence of picking orders considered in this study). In addition, this work demonstrates the time-saving benefits of replacing a traditional DES model with an NN-based metamodel in an SO framework. Moreover, it proposes and evaluates a novel NN-assisted improved version of particle swarm optimisation (PSO) to address the OSP. The improved particle swarm optimisation (IPSO) is used as the optimisation algorithm due to its proven ability to solve discrete combinatorial optimisation problems (Emambocus et al., 2021). Ultimately, this work illustrates the effectiveness of the NN-assisted IPSO against state-of-the-art methods such as first-in-first-out (FIFO), random search (RS), multi-start simulated annealing (MSSA), genetic algorithm (GA), and canonical PSO (CPSO).

The remainder of this paper is structured as follows. Section 2 reviews the related literature and identifies the research gap and contributions. Section 3 presents the problem setting and the solution methodology. Section 4 discusses the DES model, the metamodels developed, and their training procedure, while Section 5 presents the IPSO algorithm. Section 6 presents the numerical experiments and results. Finally, Section 7 concludes the paper with a discussion of findings and future research directions.

2. Literature review

This section reviews the SO techniques used to improve warehouse operations (Section 2.1), followed by the OSP in automated warehouses (Section 2.2). The research gap and the contributions of this study are discussed in Section 2.3.

2.1. Simulation-optimisation to improve warehousing processes

Recent research on optimising automated warehouse operations has introduced a variety of methodologies and objectives. Hsu and Wang (2023) addressed the dual-command crane scheduling problem in a UL double-deep AS/RS, considering energy consumption as a crucial factor by hybridising the whale optimisation algorithm with PSO and DES. The primary objective was to minimise the total operational time, a critical AS/RS efficiency metric. It was demonstrated that the approach offers a simple structure, fewer operators, fast convergence speed, and balanced exploration and exploitation capabilities. However, it was noted that the whale optimisation algorithm might lack performance if not supported by complementary techniques. Another paper that studies energy consumption in AS/RS is Rizqi and Chou (2024). Authors proposed a surrogate model based simulation-optimisation approach to balance responsiveness and energy efficiency. Urnauer et al. (2019) shifted the focus to sequencing buffer allocation in AS/RS for the automotive industry. This study employed the greedy hill climbing heuristic and Monte Carlo simulation to optimise buffer capacity allocation,

with the heuristics being noted for its solution space convexity and speed. The primary objective, in this case, was to maximise sequence adherence, highlighting the importance of maintaining production line efficiency in the automotive sector.

The topic of order sequencing was also addressed by Suemitsu et al. (2022), who introduced a fast simulation-based approach for order sequence optimisation in a complex warehouse pick-place-carry-pick-place system comprising automated guided vehicles, picking robotic arms, and conveyor modules. Their methodology combined discrete-event simulation (DES), Bayesian recurrent neural networks (BRNN), and simulated annealing (SA) to minimise makespan under the constraint of avoiding deadlocks. Key advantages of the approach include rapid computation enabled by BRNN and the explicit consideration of order sequencing. A potential drawback, however, is the need for BRNN pre-training.

Turning to more traditional warehousing systems, another example of SO leveraging metamodelling can be found in Zhao and Wang (2021), where DES and response surface methodology were employed to minimise both makespan and the occurrence of deadlocks in a distribution system. Maniezzo et al. (2021) addressed the stochastic pre-marshalling problem in block stacking warehouses using a combination of metamodelling and a heuristic method. This study highlighted the advantages of fast computation enabled by metamodels over traditional simulation. The layout design problem was also addressed by Chen et al. (2021), who integrated DES, GA, and local search techniques to mitigate the risk of getting trapped in local optima and to improve solution quality. More recently, Chen et al. (2023) focused on the parcel hub scheduling problem in a closed-loop sortation system with shortcuts, using DES, GA, and PSO. Amorim-Lopes et al. (2021) aimed to enhance picking performance in a large retail warehouse by combining probabilistic simulation and DES with mixed integer linear programming to minimise the total distance travelled during the picking process.

Malde et al. (2022) focused on optimising team formation and job assignment for put-away processes in the automotive industry. Their approach involved the integration of DES as a simulation method and the metaheuristic based optimiser OptQuest. Derkinderen et al. (2023) focused on the resource assignment problem under uncertain activity duration. The authors proposed a method based on DES and iterated local search to minimise execution delay time.

A common feature among the reviewed studies is the difficulty in mathematically formulating the objective function, which arises from several factors. Warehouse operations often involve complex interactions composed of discrete steps (e.g., pick, place), conditional logic (e.g., if a rack is idle, perform an action), and sequencing rules. These characteristics make it challenging to represent the objective function or constraints in closed-form mathematical expressions. Furthermore, many warehousing processes are inherently stochastic. Random order arrivals, fluctuating travel times, and equipment breakdowns introduce uncertainty that complicates the development of deterministic optimisation models. Due to these challenges, simulation becomes a necessary tool for both system evaluation and optimisation. This paper focuses on order sequencing in an AS/RS environment under stochastic conditions and proposes a simulation-optimisation approach. The following section reviews the literature on order sequencing problems in automated warehouses.

2.2. Order sequencing problem in automated warehouses

Han et al. (1987) is among the earliest papers studying order sequencing in a conventional AS/RS. The authors compared the performance of the first-come-first-served and the nearest-neighbour sequencing rule using Monte Carlo simulation. Ko and Han (2022) focused on optimising order processing sequences in a robotic compact storage and retrieval system to minimise the total number of ULs processed. Due to the computational intractability of finding an optimal solution using a dynamic programming framework, a rollout heuristic algorithm was proposed. A time window-based routing algorithm was applied to shuttle-based storage and retrieval systems (Lienert & Fottner, 2018). This approach allowed storage items to be retrieved directly from the system in a desired sequence, eliminating the need for additional conveyor technology to re-sequence the ULs containing the items to be picked. Using DES, it was found that the retrieval-in-sequence method resulted in a loss of throughput, similar to observations made in stacker crane-based AS/RS.

Several studies in the literature take into account the joint optimisation of OSP with another related problem. For example, Yang et al. (2021) considered the joint optimisation of order sequencing and rack scheduling. The authors developed a mixed integer linear programming model and solved it using a two-stage solution procedure. Wang et al. (2022) explored the optimisation of order assignment, order sequencing, rack selection, and rack sequencing in a robotic mobile fulfilment system with multiple picking stations. A two-stage hybrid heuristic algorithm was proposed to address these challenges. The study found that allowing inter-station operations can significantly improve order throughput time, and it presented a mathematical model to minimise the number of rack moves as a surrogate objective for optimisation. A column generation-driven heuristic was developed for the order-scheduling and rack-sequencing problem in a robotic mobile fulfilment system with the objective of minimising rack visits in a picking station (Justkowiak & Pesch, 2023). The heuristic combined optimisation techniques for order batching and the travelling salesman problem to generate an initial solution, which was further refined using a mixed-integer linear programming approach, provided that a feasible order-processing schedule was available.

2.3. Research gap and contribution

The review of the literature reveals that SO techniques remain underutilised in AS/RS, particularly in the context of metamodel-based approaches. These methods are specially rare in solving the OSP. To the best of the authors' knowledge, Suemitsu et al. (2022) represents one of the most comprehensive attempts to address the OSP using metamodel-based SO in a warehousing context. This paper sets itself apart by exploring a different class of metamodels and integrating them with a distinct optimisation algorithm. Additionally, the operational setting examined in Suemitsu et al. (2022) differs significantly from the setting considered in this paper. While their study focuses on a robotic picking system, the current work addresses the OSP in an MLS system.

Although several forms of PSO algorithms have been investigated in the literature, the discrete combinatorial variant has been primarily applied to problems such as the travelling salesman problem, as seen in Emambocus et al. (2021). To the best of the authors' knowledge, the current work is the first to implement PSO for solving the OSP. Moreover, this paper proposes an improved version of PSO, including dynamic behaviour of coefficients, forgetting ability, and local search techniques, and it demonstrates its superiority over existing methods including FIFO, RS, MSSA, GA, and CPSO.

The primary contribution of this paper is an NN-based SO approach – specifically, a GRU-assisted IPSO method – to solve the OSP in an MLS system. Other contributions can be summarised as follows:

1. Consideration of the OSP in an MLS system, which integrates multiple aisles, handling machines, and picking stations.
2. Development and evaluation of RNN, LSTM and GRU-based metamodels to predict the makespan in OSP.
3. Demonstration of the NN-based metamodels' computational benefits compared to a traditional DES model.
4. Integration of the best NN-based metamodel (i.e., a GRU) with an IPSO algorithm to solve the OSP.
5. Comparison of the proposed IPSO algorithm with other sequencing algorithms such as FIFO, RS, MSSA, GA, and CPSO.

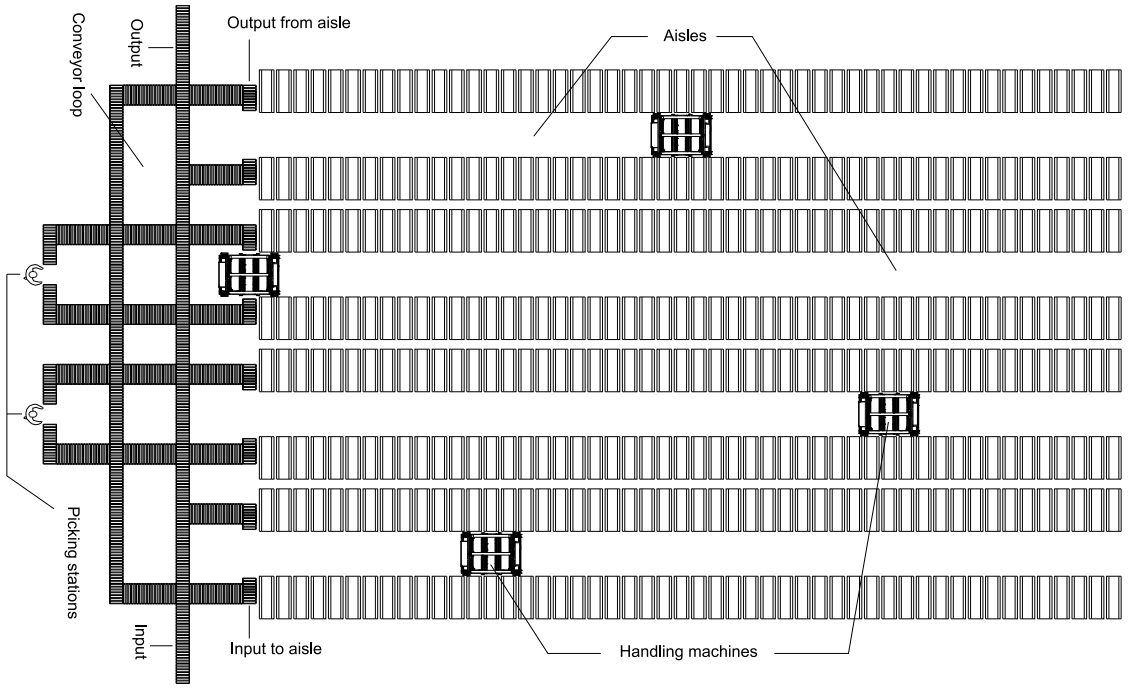


Fig. 1. The multi-level shuttle system.

3. Problem description and research methodology

3.1. The multi-level shuttle system

This paper expands the perspective of previous studies that have dealt with MLS systems (Ferrari et al., 2023; Ferrari & Corlu, 2024; Ferrari et al., 2024b, 2022) by integrating multiple aisles, multiple handling machines (HMs), and multiple picking stations. The overall system configuration was inspired by Wang et al. (2016), who studied a modified mini-load AS/RS designed to meet the needs of e-commerce logistics (Fig. 1).

From a design perspective, the HM can move mini-load ULs, such as plastic totes, along three axes, with horizontal and vertical movements occurring simultaneously. The system is designed to handle two distinct types of ULs, referred to as type A and type B. Type A ULs have dimensions of 600 × 300 × 220 mm, while type B ULs measure 400 × 300 × 220 mm. The HM is capable of simultaneously moving, storing, and retrieving two ULs of the same type. The MLS is an aisle-captive system, meaning the HM cannot change working aisles. The system under consideration consists of four aisles, each featuring a double-front rack (50 columns, 10 tiers) with multi-deep storage locations. Each location can store either two ULs of type A or four ULs of type B simultaneously. Additionally, the MLS system includes an input/output roller conveyor at one end of the aisle. In front of the storage area, a closed-loop conveyor system supplies two picking stations with the necessary ULs to complete an order and returns the ULs to the racks. If a UL is empty after picking, it is removed from the system; otherwise, it is returned to the automated warehouse for replenishment.

From an operational perspective, the HM can operate using single command, dual command, or multi command modes. The dwell policy employed is the point-of-service-completion, which means that the HM stays at the position of the last operation once it completes the backlog of storage/retrieval tasks. Additionally, the MLS system utilises class-based storage and nearest-neighbour relocation policies (Ferrari et al., 2023).

3.2. Problem setting

Order processing in a multi-picking station parts-to-picker system involves several crucial decisions, such as order sequencing, order allocation, UL selection, and UL sequencing (Wang et al., 2022). In this study, the workflow follows a fixed sequence. First, order sequencing is performed, followed by the assignment of orders to picking stations. Once the orders are allocated to the stations, the corresponding ULs are selected to fulfil the order requirements. Finally, the selected ULs are sequenced to minimise travel time during their transfer to the assigned destinations.

Each decision constitutes a sub-problem, many of which have been extensively studied in the literature. Most of these sub-problems are recognised as NP-hard, necessitating sophisticated methods for effective solutions and optimal results. This study adopted a holistic approach, emphasising the overall functionality of the system rather than the optimisation of the individual sub-problems. Consequently, simple and efficient heuristics, shown to work well in the literature, were used to approximate real operations while maintaining generalisability and reliability.

Incoming orders are assigned to picking stations to evenly distribute workloads among the available stations (Tadumadze et al., 2023). This is achieved by assigning the next order to the least busy station with the fewest pre-assigned orders. Once an order is assigned to a picking station, the ULs containing the stock keeping units (SKUs) specified in the order lines must be selected. Each product UL can accommodate only one SKU at a time, and the total number of SKUs available in the system is 500. Since the same SKU may be found in multiple ULs, it is practical to choose ULs from the aisle corresponding to the picking station to minimise UL travel distance (Wang et al., 2016). Therefore, for each order line j, a list of potential ULs containing the SKU is compiled. These ULs are first sorted by their corresponding aisles, prioritising those closest to the PS assigned to the order. Next, for each UL t, the maximal hit rate is calculated as $min\{o_j, a_t\}$, where o_j is the quantity required in the order line j and a_t in quantity stored in the UL t. The ULs are then sorted by descending hit rate values to select the UL that best matches the order line and minimises the number of ULs moved (Tadumadze et al., 2023). Additionally, within

each hit rate group, the ULs are further sorted by descending fill rate values, calculated as o_j/a_t. This prioritises ULs containing quantities closest to the order requirement and aims to empty the UL as much as possible. If no single UL contains all the items required in the order line, multiple ULs are selected, adhering to the hit rate sorting, until the order quantity is satisfied.

In order to prevent long waiting times at the picking stations, orders are triggered when 70% of the previous picking order is completed. The sequence of missions for the HMs to retrieve the ULs selected for an order is generated based on a dynamic programming algorithm that aims to minimise the total completion time of a retrieval set (Ferrari et al., 2024a). The shorter conveyor path to transport the ULs to the pre-assigned picking station is then calculated taking into account the number of ULs currently being conveyed and the number of ULs that have been preassigned to pass through the conveyor.

After completing the picking process, any ULs with leftover items are assigned to the least busy aisle for return to the AS/RS. This is done by evaluating the queue with the fewest tasks, the least number of incoming ULs from previously completed orders, and the lowest fill rate.

The demand distribution of SKUs is approximated with the typical ABC curve, where 20% of SKUs account for 80% of the total demand (Hausman et al., 1976; Silva et al., 2022; Xu et al., 2019). Therefore, the warehouse racks are divided into three zones each (A, B, C), based on the SKU turnover rate, and the number of storage locations assigned to each zone is proportioned based on the number of SKUs falling in the class (A: 20%, B: 30%, C: 50%) (Dijkstra & Roodbergen, 2017). The assignment of a storage location to a particular class is determined by the travel time distance from the input/output point (Zhou et al., 2022). Additionally, an across-aisle storage strategy is employed, meaning that SKUs from all classes are distributed across all aisles. Within each zone, SKUs are randomly assigned to storage locations (Gagliardi et al., 2015).

HMs and conveyor stops were also taken into account. In this study, machine failures were not considered to be major disruptive breakdowns, but rather minor errors that are manageable during daily warehouse operations. Examples of system downtime causes include electrical and sensor errors, mechanical errors, human errors, or UL defects (Giner et al., 2023), all of which can be quickly resolved without causing long-term system disruptions.

3.3. Solution methodology

This work investigates the OSP in an MLS system. The objective of the OSP is to find the optimal sequence S of the given N picking orders O that minimises the makespan. The OSP can be mathematically formulated as follows (Suemitsu et al., 2022):

$$\begin{aligned} \min_{S} \quad & F(X_S, Y) \\ s.t. \quad & S = \{o_1, o_2, \dots, o_n\} \\ & o_i \in \{o_1, o_2, \dots, o_n\} \qquad \forall\, i \in \{1, 2, \dots, n\} \\ & o_i \neq o_j \qquad \forall\, i \neq j \in \{1, 2, \dots, n\} \end{aligned}$$

The makespan depends on a set of variables X describing the order sequence S and a given set of system variables Y. The constraints ensure that the orders in set S are not replicated. The problem of optimally sequencing a given number of requests is NP-hard (Han et al., 1987).

Furthermore, since the objective function in the OSP is difficult to formulate mathematically, simulation becomes essential to evaluate and solve the problem. While it is true that establishing mathematical models for AS/RS with a predefined sequence is relatively straightforward, the real challenge lies in using these models to quickly and accurately derive system performance metrics, such as makespan. The complexity of AS/RS operations—coupled with the variability in order sequences and dynamic system conditions—renders these metrics difficult to compute directly from mathematical models, often necessitating the reliance on simulations for accurate assessment (Rizqi et al., 2024). However, using simulation incurs significant computational costs. Simulations require multiple iterative runs to account for stochastic variations, which can lead to prohibitive time and resource consumption to solve the OSP, particularly for large-scale AS/RS. To mitigate these challenges, a promising alternative is to employ surrogate models or metamodels during the optimisation process. These metamodels approximate the underlying behaviour of the simulation model by leveraging sampled solutions to construct statistical representations of the system. By doing so, they offer a means to predict system performance metrics – such as makespan – quickly and efficiently without the need for repeated simulation runs (Barton & Meckesheimer, 2006). The adoption of metamodels not only addresses the computational expense associated with simulations but also highlights the trade-off between accuracy and efficiency in solving the OSP. By capturing the structural information of the original problem, metamodels enable a more resource-effective approach to optimisation, particularly in contexts where iterative evaluations of the objective function are required (Xu et al., 2015).

The metamodel-based SO framework proposed in this paper is shown in Fig. 2.

A detailed DES model, capable of replicating the operations of an MLS system, was developed using the commercial software AnyLogic™. Random order sets were generated and simulated using this DES model. Then, based on the obtained dataset, a metamodel based on a NN was constructed. Specifically, a set of RNN-based metamodels were developed and compared to identify the one with the highest accuracy and prediction speed. Finally, to solve the OSP, an IPSO was employed in the search phase to iteratively explore and optimise the order sequences. In general, PSO is well-suited for finding optimal or near-optimal solutions to complex optimisation problems. The best-performing metamodel was then used to quickly evaluate the objective function of solutions proposed by the IPSO algorithm, eliminating the need for numerous resource-intensive stochastic simulations. Section 4 describes the details of the DES and the metamodels while Section 5 discusses the PSO algorithm.

4. Simulation and metamodelling

4.1. Discrete-event simulation model

To generate the data required for training the metamodel, a DES model of the MLS system was developed. DES allows for the examination of complex systems in detail and the development of virtual experiments to be performed. It is particularly useful when systems are too complex to be analysed directly through observation or mathematical analysis (Law et al., 2007). The simulation was implemented in AnyLogic™, a platform well-suited for modelling diverse logistics and supply chain operations.

The developed model incorporated key components of the warehousing system, including a conveyor network and multiple picking stations. It captured the dynamic nature of the system accounting for uncertainties in the MLS handling times, blocking events, and picking process durations. A set of assumptions was made in the development of the DES model:

- Inbound and outbound processes were not considered; hence, ULs with SKU were pre-stored at the beginning of the simulation, and products picked were removed from the simulation.
- No replenishment was required, thus, the number of items in stock was sufficient to satisfy the order set, which is a group of picking orders.
- Both for HM and conveyors, mean time between failures (MTBF) and mean time to repair (MTTR) were assumed to follow a Weibull distribution.

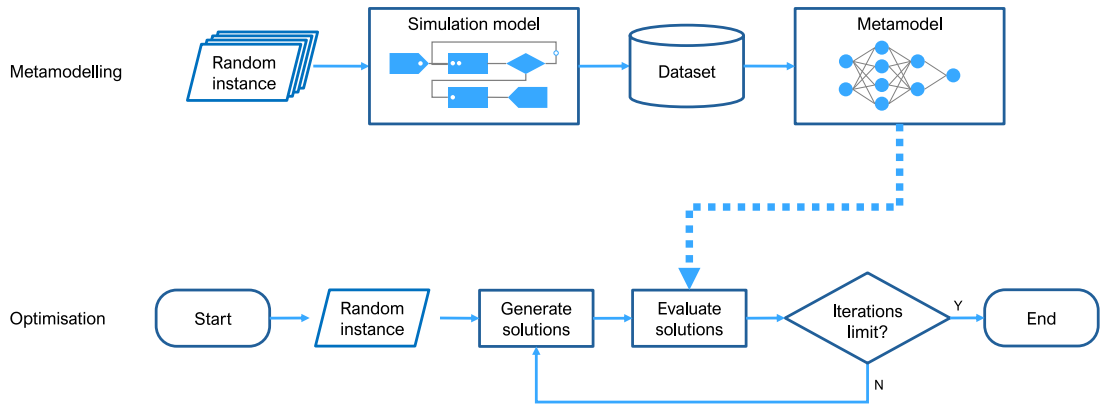


Fig. 2. Flowchart of the research methodology steps.

The decision-making logic in the order fulfilment process, including order assignment, UL selection, and routing, was implemented as custom functions within the AnyLogic environment. When predefined libraries or integrated functions in AnyLogic were insufficient to meet the specific needs of the simulation, Java programming was used to develop the necessary custom logic. This allowed for greater flexibility and precision in modelling the complex behaviours and interactions within the MLS system.

Table 1 displays the main parameters of the DES model. Most parameters were derived from a laboratory MLS installation (Ferrari et al., 2024a, 2023), with some being extracted from the system specifications and others directly measured. Examples of parameters taken from the system specifications are the velocities, accelerations of both HM and conveyors, as well as the characteristics of the ULs. On the other hand, all parameters regarding the handling times were measured. Additionally, parameters from comparable studies in the literature were incorporated. For instance, the Weibull shape parameters for conveyors and HMs failures were set to 1, since with a shape $\beta = 1$, the failure rate of the system remains constant over time, meaning the likelihood of failure is independent of the elapsed time (Das, 2008). The Weibull shape parameters for conveyors repair, HMs repair and picking time were set to 3.678, because the Weibull distribution is like a truncated normal distribution when the shape parameter is set to $\beta = 3.678$ (Lamont-Smith, 2019). Finally, the remaining parameters were attributed reasonable arbitrary values either because the current literature did not provide relevant examples or they were related to the unique aspects of the DES model developed in this work.

The DES model was verified through the debugging and tracing tools available in AnyLogic. Specifically, a step-by-step visual inspection and behavioural tracing were employed to ensure that the system components interacted as intended under various configurations. Moreover, the model was built upon previously developed and validated simulation models, which were progressively scaled up and adapted to the complexity of the current study (Ferrari et al., 2024a, 2023, 2024b). These earlier models were validated through comparisons with operational data, providing further confidence in the accuracy and reliability of the current implementation.

4.2. Metamodels

The metamodels tested were all based on a NN architecture designed for a regression task and implemented using the PyTorch framework. In machine learning, a regression task involves predicting a continuous numeric value, e.g., the makespan, based on input data, e.g., the picking orders set. The selection of a NN as a metamodel was driven by the recognition that specific architectures, including RNNs and their derivatives, demonstrate remarkable proficiency in the management of sequential data, such as the sequence of picking orders.

RNNs are designed to capture and model dependencies in sequential data by maintaining a hidden state that evolves as each element in the sequence is processed. Consequently, RNNs offer a promising approach for addressing problems involving the sequencing of picking orders, as demonstrated in Suemitsu et al. (2022). However, standard RNNs often suffer from issues such as vanishing gradients, which can limit their effectiveness in capturing long-term dependencies. LSTM networks, a specialised variant of RNNs, address these limitations through the use of memory cells and gating mechanisms. These components allow LSTMs to selectively retain or discard information across time steps, enabling the modelling of long-range temporal dependencies (Song et al., 2019). Nevertheless, due to their high model capacity and structural complexity, LSTMs may be prone to overfitting, particularly when training data is limited or when attempting to generalise to previously unseen dynamics (Ahn et al., 2025). An alternative architecture capable of handling sequential data is the GRU, which captures long-term dependencies using a more streamlined structure with fewer parameters compared to the LSTMs. GRUs rely on two primary gating mechanisms: the update gate, which controls how much of the past information is carried forward, and the reset gate, which determines the extent to which past information is forgotten. The current hidden state is then computed by combining the new input with the selectively filtered past information (John et al., 2025).

Therefore, in this paper, RNN, LSTM, GRU were adopted since they are well-established and widely recognised as standard baselines for sequence modelling (Mienye et al., 2024). The three types of NNs were comparatively evaluated, with the expectation that the GRU would outperform the standard RNN and LSTM architectures due to its balance between complexity and effectiveness. The general structure of the metamodels is illustrated in Fig. 3. All architectures shared the same overall structure, differing only in the type of recurrent cell employed.

The recurrent layers were implemented as a stack of recurrent cells to capture sequential dependencies in the input data. The recurrent layers receive the sequential data of the order set as input. At the same level as the recurrent layers, a set of linear layers (fully-connected) was used to process features describing the system state at the beginning of the simulation. The output of the recurrent layers and the linear layers were concatenated and passed to other linear layers. The model had an output layer to produce the final output. Dropout was applied after each layer to prevent overfitting.

During initialisation, the number of recurrent and linear layers, their hidden sizes, and the dropout rates were defined. For the recurrent layers, weight initialisation was performed using Xavier initialisation for input weights and orthogonal initialisation for hidden weights. For the linear and output layers, weight initialisation was again performed using Xavier initialisation for hidden weights and constant initialisation for biases (Glorot & Bengio, 2010; Saxe et al., 2013). Rectified linear unit activation functions were employed after each linear layer, contributing to the non-linearity and expressive power of the metamodel (Goodfellow et al., 2016).

Table 1
DES model parameters.

Parameter	Value	Source
Acceleration conveyor	0 m/s^2	(Ferrari et al., 2024a, 2023)
Horizontal acceleration HM	1.5 m/s^2	(Ferrari et al., 2024a, 2023)
Vertical acceleration HM	1.6 m/s^2	(Ferrari et al., 2024a, 2023)
Deceleration conveyor	0 m/s^2	(Ferrari et al., 2024a, 2023)
Horizontal deceleration HM	1.5 m/s^2	(Ferrari et al., 2024a, 2023)
Vertical deceleration HM	1.6 m/s^2	(Ferrari et al., 2024a, 2023)
Weibull shape parameter (HM failure)	1	(Das, 2008)
Weibull shape parameter (conveyor repair)	3.678	(Lamont-Smith, 2019)
Weibull shape parameter (HM repair)	3.678	(Lamont-Smith, 2019)
Weibull shape parameter (picking time)	3.678	(Lamont-Smith, 2019)
Load/unload time from/to conveyor	~ Empirical distribution	(Ferrari et al., 2024a, 2023)
Handling times	~ Empirical distribution	(Ferrari et al., 2024a, 2023)
Idle times	~ Empirical distribution	(Ferrari et al., 2024a, 2023)
Transfer time conveyor	~ Triangular(1.5, 2, 2.5) s	Arbitrary value
Weibull scale parameter (conveyor failure)	180	Arbitrary value
Weibull scale parameter (HM failure)	120	Arbitrary value
Weibull scale parameter (conveyor repair)	5.5	Arbitrary value
Weibull scale parameter (HM repair)	5.5	Arbitrary value
Weibull scale parameter (picking time)	3.3	Arbitrary value
Maximum horizontal velocity	4 m/s	(Ferrari et al., 2024a, 2023)
Maximum vertical velocity	0.8 m/s	(Ferrari et al., 2024a, 2023)
Velocity conveyor	0.5 m/s	(Ferrari et al., 2024a, 2023)
Velocity for on board transfer	0.12 m/s	(Ferrari et al., 2024a, 2023)

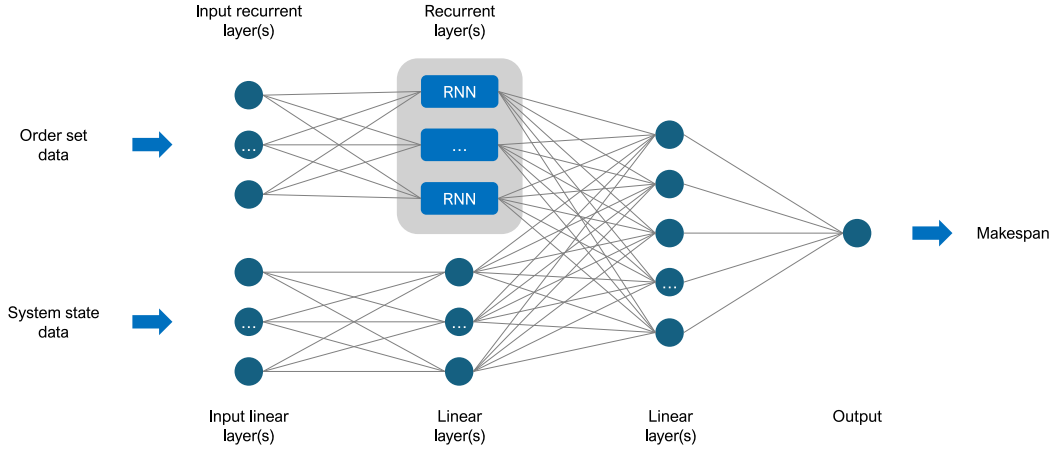


Fig. 3. The structure of the metamodels.

4.3. Feature generation

The DES model was utilised to simulate random order sets and monitor the makespan. The simulation process began by initialising a random scenario, where the initial warehouse fill rate ranged from 50% to 80%. Each UL was assigned an SKU in a predetermined quantity, based on the available fill rate and the demand associated with each SKU. Additionally, the initial positions of the HMs, indicated by their last visited storage location, were randomly selected from all available storage locations within the HM aisle. Subsequently, an order set containing a random number of orders (ranging from 10 to 50) was generated. Each order included between 1 and 10 order lines, with the quantity of each SKU requested in each order line randomly sampled from a normal distribution. The mean of this distribution was equivalent to the SKU demand rate, and the standard deviation was set to 20% of the demand rate.

Two main classes of features were generated. The first class, order set features, consisted of six features describing each order in the order set. Specifically, for each of the three product classes (A, B, and C), the number of SKUs and the quantity requested in each order were considered. The second class, system state features, comprised 40 features capturing key aspects of the system state. This included details such as the last failure or repair time of conveyors and handling HMs, as well as the total number of ULs stored in the warehouse. The order set features were used as input for the recurrent layers, while the system state features were used as input for the linear layers. The selection of features from these two classes aimed to minimise the correlation between input variables and maximise the information captured by each variable, avoiding redundancy and irrelevant information. Additionally, features were also iteratively included based on the metamodel improvement.

Each simulation was run until the last order line of the last order was completely picked. The makespan was traced to be later used as the target variable for the metamodel. Because of the stochasticity of the DES model, every order set was replicated 100 times. The initial state of the system, such as the composition of the order set, the position of the HMs, the position of the ULs within the racks, and the content of each UL, was kept constant between the 100 runs. The results of the replicates were then averaged and an average makespan was computed. A total of 3,500 independent order sets were simulated, therefore the initial dataset was composed of 3,500 simulations · 100 replicates = 350,000 data.

Given the stochastic nature of the DES model and the multitude of possible parameter combinations, this sample size was chosen to ensure a comprehensive representation of the parameter space and it was deemed adequate to capture the inherent variability in MLS system operations. Furthermore, the random generation of input states facilitated the evaluation of a wide range of conditions. Therefore, selecting 3,500 independent order sets struck a balance between acquiring statistically robust data and managing computational resources, thereby facilitating a practical and feasible approach to model development and analysis. The entire dataset was subsequently divided into training, validation, and testing subsets following the widely adopted 80–20 rule. Specifically, 80% of the data were allocated to model development – including both training and validation – while the remaining 20% were held out as an independent test set (Géron, 2022). This partitioning strategy is commonly employed to ensure that model evaluation is performed on previously unseen data, thereby providing a more reliable estimate of its generalisation performance.

4.4. Training and metamodels comparison

Prior to training, the input data were normalised using min–max scaling to ensure stable and consistent learning across features with different value ranges. A 10-fold cross-validation strategy was adopted to provide a robust assessment of the model's generalisation ability and to mitigate the risk of overfitting. The dataset was divided into 10 equally sized folds and in each iteration, 1 fold was used for validation while the remaining 9 were used for training. This process was repeated 10 times, ensuring that each data point was used for both training and validation. As a result, the approach maximised data utilisation and reduced estimation bias (Malakouti et al., 2023).

The core of the training process was an iterative training loop executed for each fold. Batches of data were fed into the model, and the training loop involved calculating the loss, performing backpropagation, and updating the model parameters using the Adam optimiser. Mean absolute error (MAE) was chosen as the loss function because it provides a simple and interpretable measure of prediction accuracy by capturing the average magnitude of errors, making it a robust metric for evaluating regression models. To mitigate overfitting, L2 regularisation was incorporated into the training loop, thereby enhancing the model's ability to generalise (Sun et al., 2025). L2 regularisation introduces a penalty term proportional to the sum of the squared weight coefficients, effectively constraining the magnitude of these weights. This encourages the model to adopt a simpler structure that is less susceptible to overfitting, ultimately resulting in a more robust and interpretable solution (Feliciani Merizio et al., 2025). Throughout the training process, the mean absolute percentage error (MAPE) was also tracked.

The training process commenced by initialising crucial hyperparameters that define the architectural and learning characteristics of the metamodel. The hyperparameters encompassed the configuration of recurrent layers (number and size), their bidirectionality, the configuration of linear layers (number and size), the dropout rates, learning rate, batch size, and weight decay for regularisation. Additional hyperparameter values that could have been investigated during the hyperparameter search were defaulted to the values specified by the PyTorch library. The best combination of hyperparameters was explored via random search, followed by manual fine-tuning by leveraging prior knowledge from tested combinations. Table 2 shows the hyperparameters search space and the values selected for each metamodel which yields the best results in terms of MAE and MAPE.

Fig. 4 presents the training and validation values of MAE and MAPE, averaged across the 10-folds. The slightly higher training errors, in terms of both MAE and MAPE, compared to the validation errors, can be attributed to the absence of regularisation during the validation inference phase. Nevertheless, it can be stated that the metamodels performed well as the training time increased, with no clear evidence of overfitting or underfitting. To further assess the predictive performance of the metamodels, MAE and MAPE were also computed on the test dataset, which was not involved in the training or validation processes. Table 3 reports the MAE and MAPE values for the training, validation, and test sets. LSTM and GRU exhibited similar performance, both clearly outperforming the RNN, which showed notably weaker results, particularly on the testing dataset.

During the testing phase, a residual analysis was also conducted to evaluate the presence of unexpected trends in the metamodel behaviour and deviations between the metamodel predictions and the actual target values. Comprehensive insights into the metamodels' performance are provided by the graphical representations shown in Fig. 5. The histograms display the frequency distribution of the residuals, while the scatter plots show the residuals plotted against the predicted values. Across all metamodels, the residuals approximated a normal distribution, which supports the reliability of the prediction intervals and underscores the robustness of the models. The scatter plots also revealed a consistent pattern of homoscedasticity across all predictions and metamodels. However, the histogram and scatter plot of the RNN model suggested a mild bias towards overestimating the true values. Overall, the residual analysis provided strong evidence that the metamodels generalised well, with LSTM and GRU demonstrating superior predictive performance on the test data.

To evaluate the effectiveness of replacing the computationally intensive DES model with metamodels, the simulation time for an order set using the DES model was compared to the prediction time of the makespan using the pre-trained NNs. Various scenarios were considered, including order sets of varying sizes (10, 20, 30, 40, 50) and different warehouse fill rates (50%, 65%, 80%). For each scenario, the simulation with the DES model and the predictions with the metamodels were replicated 100 times. The results of this comparison are presented in Table 4. The simulations and predictions were performed on a computer with an Intel Core i9-13900K 3.00 GHz 32-core CPU and 64 GB RAM. The findings reveal that all metamodels outperformed the DES model in terms of prediction speed. Notably, unlike DES, the predictions generated by the NNs were scenario-independent, meaning that the time required to infer the makespan did not vary with scenario characteristics such as the number of orders or the warehouse fill rate. This efficiency highlights the metamodels' potential for enabling faster decision-making while maintaining a balance between predictive accuracy and computational efficiency.

To identify the best-performing metamodel, the results from the previous evaluation tests were analysed. Notably, the RNN achieved its best performance with a significantly more complex architecture (more layers and neurons) compared to the LSTM and GRU. Despite this increased complexity, the RNN consistently underperformed relative to the other two metamodels. LSTM and GRU produced similar outcomes in terms of training loss, testing loss, and prediction speed, with GRU slightly outperforming LSTM in across all metrics. Given its superior performance and simpler internal-cell architecture, GRU was selected as the best metamodel.

5. Optimisation

The GRU-based metamodel was employed to solve the OSP by approximating the objective function F, which represents the makespan. Given that OSP is a combinatorial NP-hard problem, exact methods are not feasible for solving the OSP to optimality. Instead, metaheuristics – powerful algorithmic frameworks designed to solve NP-hard combinatorial problems – serve as alternatives to exact methods, enabling solutions within a reasonable time frame. Metaheuristics provide approximate solutions by intelligently exploring the solution space, balancing exploration and exploitation through techniques such as population-based search, probabilistic decision-making, and memory-based strategies. These mechanisms help escape local optima and guide the search towards high-quality solutions. This section describes a metaheuristic algorithm known as the CPSO (Section 5.1), which was used as the basis to develop the IPSO algorithm to solve the OSP (Section 5.2).

Table 2
Hyperparameters of the metamodels.

Hyperparameter	Range of values	RNN	LSTM	GRU
Recurrent layers	1, 2, 3	3	1	1
Recurrent layers size	8, 16	(8, 8, 8)	8	8
Recurrent layers bidirectionality	True, False	True	True	True
Recurrent layers dropouts	0%, 10%, 20%, 30%	(0%, 0%, 0%)	0%	0%
Linear layers (pre-concat)	1, 2	1	1	1
Linear layers sizes (pre-concat)	32, 64, 128	128	32	32
Linear layers dropouts (pre-concat)	0%, 10%, 20%, 30%	30%	30%	30%
Linear layers	1, 2	1	1	1
Linear layers sizes	32, 64, 128	128	64	64
Linear layers dropouts	0%, 10%, 20%, 30%	30%	10%	10%
Learning rate	1e−4, 5e−4, 1e−3, 5e−3, 1e−2	5e−4	5e−4	5e−4
Batch size	16, 32, 64	32	32	32
Epochs	200, 300, 400, 500	400	400	400
Weight decay (L2)	1e−4, 1e−3, 1e−2	1e−3	1e−3	1e-3

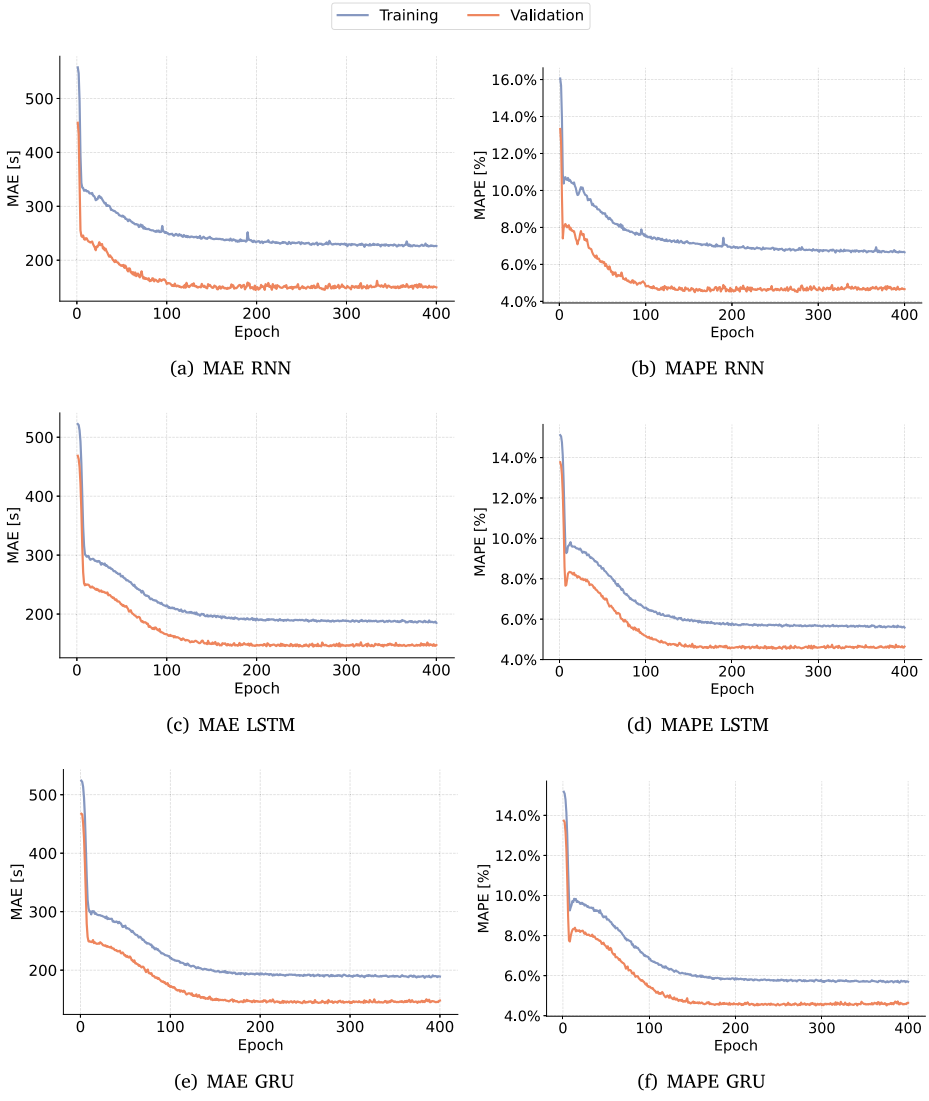
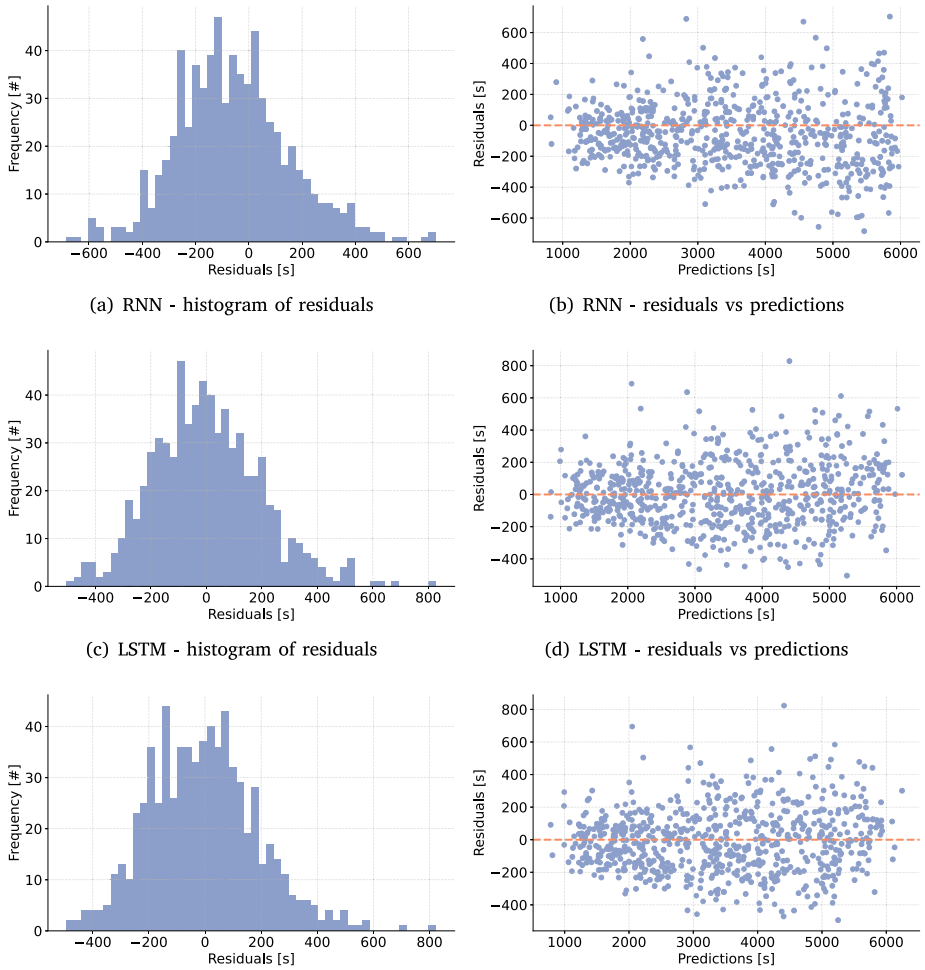


Fig. 4. Metamodels training and validation curves.

5.1. Canonical particle swarm optimisation

PSO is a well-known swarm intelligence algorithm inspired by the flocking behaviour of birds. In PSO, a population of particles searches for optimal solutions by updating their positions and velocities based on their own experiences and those of their neighbours (Xia et al., 2020). Although PSO is particularly suited for continuous optimisation problems, it has demonstrated good performance in discrete combinatorial problems such as the travelling salesman problem (Wang et al., 2003), and is therefore applicable to the OSP as well. The combinatorial variant of PSO retains the same steps as the continuous version, including initialisation of particles, updating velocities and positions, and updating personal and global best solutions (Emambocus et al., 2021). The key difference is that, in the discrete combinatorial



(a) RNN - histogram of residuals (b) RNN - residuals vs predictions

(c) LSTM - histogram of residuals (d) LSTM - residuals vs predictions

(e) GRU - histogram of residuals (f) GRU - residuals vs predictions

Fig. 5. Analysis of residuals.

Table 3
Metamodels training, validation, and testing results.

Metamodel	Training		Validation		Testing	
	MAE [s]	MAPE [%]	MAE [s]	MAPE [%]	MAE [s]	MAPE [%]
RNN	224.71	6.63%	144.66	4.49%	174.47	5.64%
LSTM	184.95	5.58%	144.68	4.53%	154.51	5.01%
GRU	186.77	5.65%	143.31	4.51%	152.67	5.01%

version, a particle position represents a complete sequence of elements (the sequence of orders in the order set), and a particle velocity is a sequence of swap operators. Each swap operator is a pair of indices indicating which two elements of the particle (two orders of an order sequence) should be swapped. For instance, considering the solution $S = \{o_1, o_2, \dots, o_n\}$ and the swap operator $\phi = \{i, j\}$, where ϕ swaps the elements at indexes i and j of sequence S, the new solution is defined as:

$$S' = S + \phi = \{o'_1, o'_2, \dots, o'_n\} \tag{1}$$

where each element o'_k of the sequence S' is given by:

$$o'_k = \begin{cases} o_j & \text{if } k = i \\ o_i & \text{if } k = j \\ o_k & \text{otherwise} \end{cases} \tag{2}$$

For a specific example, if $S = \{o_1, o_2, o_3, o_4\}$ and swap operator $\phi = \{1, 2\}$, $S' = S + \phi = \{o_2, o_1, o_3, o_4\}$.

At each iteration i, the velocity v of each particle is then updated using the following equation:

$$\begin{aligned} v_i = {} & \omega \cdot v_{i-1} \oplus \\ & c_1 \cdot r_1 \cdot (P \ominus S_{i-1}) \oplus \\ & c_2 \cdot r_2 \cdot (G \ominus S_{i-1}) \end{aligned} \tag{3}$$

In Eq. (3), ω is the inertia weight, determining how much of the velocity of the particle at the previous iteration is preserved. c_1 and c_2 are two acceleration coefficient representing the influence of the personal best solution P and global best solution G found up to iteration i, respectively. r_1 and r_2 are two random numbers uniformly distributed in the interval $[0, 1]$. S_{i-1} and v_{i-1} are the position of the particle, which is an order sequence, and the velocity of the particle, which is a sequence of swap operators, in the previous iteration $i - 1$. Finally, the operator $\oplus$ means that the two swap sequences are merged to form a new swap sequence. For instance, let $\phi_1 = \{i_1, j_1\}$, $\phi_2 = \{i_2, j_2\}$, $\phi_3 = \{i_3, j_3\}$ be three swap operators and $\Phi_1 = \{\phi_1, \phi_2\}$, $\Phi_2 = \{\phi_3\}$ two swap sequences, i.e. a set of swap operators. The new swap sequence is defined as:

$$\Phi' = \Phi_1 \oplus \Phi_2 = \{\{i_1, j_1\}, \{i_2, j_2\}\} \oplus \{\{i_3, j_3\}\} = \{\{i_1, j_1\}, \{i_2, j_2\}, \{i_3, j_3\}\}$$

On the other hand, the operator $\ominus$ indicates the subtraction of two particle positions, i.e. a solution, which generates a swap sequence. Let $S_1 = \{o^1_1, o^1_2, \dots, o^1_m\}$ and $S_2 = \{o^2_1, o^2_2, \dots, o^2_m\}$ be two solutions. The

Table 4
Runtime comparison between the simulation model and the metamodels.

		DES		RNN		LSTM		GRU	
Orders [#]	Fill rate [%]	Mean [s]	Std [s]	Mean [s]	Std [s]	Mean [s]	Std [s]	Mean [s]	Std [s]
10	50%	0.9393	0.2376	0.0019	0.0008	0.0019	0.0007	0.0019	0.0008
10	65%	1.1388	0.3018	0.0021	0.0007	0.0019	0.0007	0.0019	0.0006
10	80%	1.2062	0.3147	0.002	0.0007	0.0018	0.0007	0.0019	0.0007
20	50%	1.6682	0.4086	0.0025	0.0011	0.0021	0.0009	0.0022	0.0009
20	65%	2.1863	0.4933	0.0024	0.001	0.0022	0.0009	0.0023	0.001
20	80%	2.3908	0.5439	0.0024	0.0011	0.0022	0.0009	0.0022	0.001
30	50%	3.0149	0.6904	0.0028	0.0014	0.0024	0.001	0.0026	0.0012
30	65%	3.3619	0.7287	0.0029	0.0014	0.0026	0.0012	0.0026	0.0011
30	80%	3.389	0.7583	0.0029	0.0013	0.0025	0.0012	0.0027	0.0013
40	50%	4.4683	0.9571	0.0029	0.0013	0.0029	0.0014	0.0026	0.0011
40	65%	4.5705	1.0101	0.0033	0.0015	0.0029	0.0014	0.0028	0.0013
40	80%	4.8359	1.0849	0.0033	0.0015	0.0029	0.0014	0.003	0.0015
50	50%	6.1713	1.2729	0.0031	0.0014	0.003	0.0014	0.0029	0.0014
50	65%	6.3712	1.3478	0.003	0.0014	0.0029	0.0014	0.0033	0.0016
50	80%	6.8095	1.3665	0.0034	0.0014	0.003	0.0014	0.0032	0.0016
Average values		3.5015	0.7678	0.0027	0.0012	0.0025	0.0011	0.0025	0.0011

swap sequence Φ' generated by $S_1 \ominus S_2$ is defined as the sequence of swaps needed to transform S_1 in S_2. Thus, $\forall\ i \in \{1, 2, \dots, m\}$, if $o_i^1 \neq o_i^2$, find the index $j = min\{k \mid k > 1 \wedge o_k^1 = o_i^2\}$, generate the swap operator $\phi = \{i, j\}$, calculate the new sequence S_1', where each element o_k' is given by Eq. (2), and update the swap sequence as $\Phi' = \Phi' \cup \phi = \Phi' \cup (\{i, j\})$.

The position of the particles is then updated using the following equation.

$$S_i = S_{i-1} \oplus v_i \tag{4}$$

In Eq. (4), the $\oplus$ operator indicates that each swap operator in the swap sequence v_i is applied sequentially to S_{i-1} to generate the new solution S_i. Therefore, let $v_i = \{\phi_1, \phi_2, \dots, \phi_z\}$ be the particle velocity at iteration i. To obtain the new solution S_i, the swap operators are applied sequentially, updating the solution at each step. This can be described as:

$$S_i^j = S_i^{j-1} + \phi_j \qquad \forall\ j \in \{1, 2, \dots, z\} \tag{5}$$

where $S_i^0 = S_{i-1}$, that is the initial solution before any swaps applied, and $S_i^z = S_i$ is final solution after all swaps applied.

5.2. Improved particle swarm optimisation

5.2.1. Algorithm structure

This section introduces several enhancements to the CPSO approach described in Section 5.1 and presents an improved variant referred to as IPSO. Building on the approach proposed by Emambocus et al. (2021), IPSO incorporates several modifications aimed at improving performance. In IPSO, at each iteration i, the velocity of each particle is updated using the following equation:

$$\begin{aligned} v_i = {} & \omega \cdot v_{i-1} \oplus \\ & c_1 \cdot r_1 \cdot (P \ominus S_{i-1}) \oplus \\ & c_2 \cdot r_2 \cdot (|1-\gamma| \cdot G \ominus S_{i-1}) \oplus \\ & c_3 \cdot r_3 \cdot (|1-\gamma| \cdot L \ominus S_{i-1}) \oplus \\ & \lambda \cdot m \end{aligned} \tag{6}$$

From Eq. (6), it is evident that IPSO introduces additional terms compared to CPSO. Specifically, IPSO incorporates the influence of the local best solution L, associated with the acceleration coefficients c_3. r_1, r_2 and r_2 are random numbers in the interval $[0, 1]$, which are unique for each particle in the population of solutions. Furthermore, IPSO integrates a forgetting mechanism, denoted by the forgetting ability γ, which regulates the influence of both the local and global best solutions during the velocity update process. Finally, to enhance exploration capabilities, IPSO includes a local search strategy inspired by the random deviations observed in the flight paths of birds. This is implemented by introducing small stochastic mutations into the velocity swap sequence. Specifically, λ denotes the mutation rate, while m represents a single swap operator added to the velocity of the particle.

5.2.2. Algorithm coefficients

IPSO uses five global coefficients ω, c_1, c_2, c_3, and λ, representing the inertia weight, the acceleration coefficients associated with the personal best, global best, local best, and the mutation rate. These coefficients are the same for all the particles in the population and are static along the optimisation process. In addition, in IPSO, r_1, r_3, and r_2 are specific for each particle and they are dynamic during the optimisation process, meaning that they are updated when the global best solution G does not improve after I_s iterations. The values of these coefficients are randomly generated using Gaussian distribution with mean μ_1, μ_2, μ_3 respectively, and standard deviation of 0.1. The initial values of μ_1, μ_2, μ_3 are set to a random value in the interval $[0, 1]$. Then, during the optimisation process, the values of μ_1, μ_2, μ_3 are updated based on the population of particles. Specifically, let $\Gamma = \{S_1, S_2, \dots, S_n\}$ be the set of all the particles of the swarm of dimension n. The particles are sorted in ascending order by values of the fitness value f such that $f_1 \leq f_2 \leq \cdots \leq f_n$. The set of elite particles E is defined as the first k particles in the sorted list Γ'. The number k is calculated as

$$k = \lfloor n \cdot \alpha \rfloor \tag{7}$$

where n is the total number of particles and α is a real number in the interval $[0, 1]$. Thus, $E = \{S_1', S_2', \dots, S_k'\}$. Given E, the values of μ_1, μ_2, μ_3 are calculated according to the following expression:

$$\mu_i = (1-\beta) \cdot \mu_i + \beta \cdot \frac{\sum_{j \in E} \mu_{i,j}}{k} \qquad i \in \{1, 2, 3\} \tag{8}$$

where β represents the learning weight for knowledge from elite particles and $\mu_{i,j}$ is the value of μ_i of the elite particle j in the set E.

The sampled values of r_1, r_2, and r_3 are normalised to be in the interval $[0, 1]$ and then multiplied by the corresponding acceleration coefficient c.

In Eq. (6), the coefficients of the five terms, namely ω, $c_1 \cdot r_1$, $c_2 \cdot r_2$, and $c_3 \cdot r_3$ determine the probability of including a swap operator in the new swap sequence. Considering a generic swap sequence Φ, for each swap operator ϕ in the sequence, a random value r is generated from a uniform distribution in the interval $[0, 1]$, and compared to coefficient c associated with the sequence. If the random value r is less than the value of the coefficient, the swap operator is included in the new velocity of the particle v', otherwise it is not used. Thus:

$$v' = \{\phi \mid r \sim Uniform(0, 1) \leq c\} \qquad \forall \phi \in \Phi \tag{9}$$

Regarding the local search strategy, a random value r is drawn from a uniform distribution over the interval $[0,1]$, and compared to the mutation rate λ. If r is less than or equal to λ, a mutation sequence consisting of a single random swap operator ϕ is generated. Formally:

$$m = \begin{cases} \{\phi\} & \text{if } r \sim \text{Uniform}(0,1) \leq \lambda \\ \emptyset & \text{otherwise} \end{cases} \tag{10}$$

5.2.3. Forgetting ability

In IPSO, the concept of forgetting ability γ is incorporated into the velocity update formula to reduce the influence of outdated information from social learning exemplars, accounting for the unique characteristics of each particle. γ is specific for each particle and it is dynamic during the optimisation process, meaning that it is updated when the global best solution G does not improve after I_s iterations. The first step in calculating the forgetting ability involves determining the distance d between a particle and the global best G. Let $\Gamma = \{S_1, S_2, \dots, S_n\}$ be the set of all the particles of the population. The distance d of ith particle can be calculated by the following equation:

$$d_i = f_i - f_G \tag{11}$$

where f_i corresponds to the fitness value of the ith particle, while f_G is the fitness value of the global best solution. Both f_i and f_G are evaluated using the GRU-based metamodel as fitness function F. The particles of all the population are then sorted in ascending order by values of d. Let Γ' be the sorted sequence of particles, such that $d'_1 \leq d'_2 \leq \cdots \leq d'_n$.

After that, the forgetting ability of each ith particle in the sorted sequence Γ' is then calculated with

$$\gamma_i = \begin{cases} e^{\frac{i}{n}} \cdot (f_W - f_G) \cdot 0.01 & i > k \\ 0 & otherwise \end{cases} \tag{12}$$

where n is the number of particles in the swarm, f_W and f_G are the maximum and the minimum values of the makespan of the entire population, also in this case calculated using the GRU metamodel.

In Eq. (6), $|1-\gamma|$ determines how much information is extracted from a swap sequence. Let $\Phi = \{\phi_1, \phi_2, \dots, \phi_z\}$ be the original swap sequence. The length z of the swap sequence is increased if the constant $|1-\gamma|$ is greater than one, otherwise it is truncated. Firstly, the new length z' of the new swap sequence Φ' is calculated by multiplying the constant by its length z and rounding it down to the nearest integer:

$$z' = \lfloor |1-\gamma| \cdot z \rfloor \tag{13}$$

If $z' > z$ holds, additional swap operators are appended by repeating the existing operators in sequence until the desired length is reached. In this case, the new swap sequence is defined as: $\Phi' = (\phi_1, \phi_2, \dots, \phi_z, \phi_1, \phi_2, \dots, \phi_{z'-z})$. Conversely, if $z' < z$, the new sequence is obtained by truncating the original swap sequence to match the new length. This is done by selecting the first z' operators and discarding the remainder: $\Phi' = (\phi_1, \phi_2, \dots, \phi_{z'})$

5.2.4. Dynamic behaviour

To prevent stagnation of the global best solution, IPSO introduces a dynamic behaviour after n_s consecutive iterations without improvement by recalculating specific parameters. This mechanism promotes exploration and helps the population escape deep local minima. In particular, the personal acceleration coefficients r_1, r_3, r_2, as well as the forgetting ability γ, are re-initialised. In addition, the local neighbourhood of each particle is redefined. Let $\Gamma = \{S_1, S_2, \dots, S_n\}$ denote the set of all particles in the population. A new population ordering $\Gamma' = \{S'_1, S'_2, \dots, S'_n\}$ is obtained by randomly permuting the elements of Γ. For each particle S'_i, the neighbourhood $\mathcal{N}$ is defined as the particle itself along with its immediate left and right neighbours in Γ', such that:

$$\mathcal{N}(S'_i) = \begin{cases} \{S'_n, S'_1, S'_2\} & \text{if } i = 1 \\ \{S'_{n-1}, S'_m, S'_1\} & \text{if } i = n \\ \{S'_{i-1}, S'_i, S'_{i+1}\} & \text{if } 1 < i < n \end{cases} \tag{14}$$

Among the three particles in the neighbourhood, the one with the highest fitness value is selected as the local best particle L.

5.3. Pseudocode

Algorithm 1 shows the pseudocode of the IPSO.

6. Numerical experiments and results

6.1. Experimental setting

To evaluate the performance of the proposed NN-assisted IPSO, a series of numerical experiments were conducted. Several experimental factors were varied in order to cover a broad range of realistic scenarios. Specifically, problem instances were generated with varying order set sizes (10, 20, 30, 40, and 50 orders) and warehouse fill rates (50%, 65%, and 80%).

In addition, a comparative analysis was carried out against a selection of benchmark algorithms, encompassing both simple heuristics and well-established metaheuristic approaches. The methods included in this comparison are FIFO, RS, MSSA, GA, and CPSO.

FIFO represents a basic heuristic baseline, in which the solution to the OSP corresponds to the original sequence of arrival of the picking orders, with no adaptive or optimisation mechanism applied. While elementary, it provides a useful lower-bound reference for assessing the effectiveness of more advanced methods. RS explores the solution space by sampling candidate solutions uniformly at random and serves as a naive stochastic baseline. MSSA augments the traditional SA algorithm by incorporating a multi-start mechanism, thereby enhancing the algorithm's ability to escape local optima and improving global search performance. The mutation strategy adopted in MSSA involves random swaps between elements in the solution sequence. The GA, inspired by the principles of natural selection, evolves a population of candidate solutions through repeated application of selection, crossover, and mutation operations. In this study, roulette wheel selection was used to guide the selection of parents, order crossover was employed to preserve the feasibility of offspring solutions, and swap-based mutation was applied to introduce diversity. Additionally, elitism was incorporated to retain the best solutions. CPSO represents the standard PSO algorithm adapted to discrete combinatorial problems, as described in Section 5.1. For both CPSO and IPSO, the initial velocity of each particle was defined as a number of swap operations equal to the length of the sequence, ensuring a sufficiently diverse exploration at the start of the optimisation.

The selection of these benchmark algorithms was motivated by the need to cover a representative spectrum of optimisation strategies relevant to discrete combinatorial problems like the OSP in AS/RS. FIFO and RS provide simple and naive baselines, allowing the assessment of improvements offered by adaptive methods. MSSA and GA represent established metaheuristics capable of exploring complex solution spaces and escaping local optima. CPSO, as the canonical discrete PSO, serves as a direct reference to evaluate the enhancements introduced in the proposed IPSO. Together, these algorithms offer a comprehensive comparison across diverse optimisation paradigms, ranging from simple heuristics and metaheuristics to evolutionary and swarm-based strategies (Odeyemi & Zhang, 2025; Panda, 2018), highlighting the effectiveness and reliability of the proposed approach.

The hyperparameter settings used for each optimisation algorithm are summarised in Table 5. These values reflect configurations that consistently yielded good performance during algorithmic testing.

To reduce the impact of stochastic variability and ensure stable performance estimates, each optimisation scenario was replicated 50 times. All experiments were executed on the same computing environment used for comparing the execution time of the DES model and the metamodels.

Algorithm 1: Improved Particle Swarm Optimisation

Require: $F, X, Y, I, I_s, n, c_1, c_2, c_3, \lambda, \alpha, \beta$
Ensure : Global best solution G

Step 1: Initialise population and parameters
$\Gamma \leftarrow \emptyset$; // Population
$\mathcal{F} \leftarrow \emptyset$; // Fitness values
$G \leftarrow \{\}$; // Global best solution
$W \leftarrow \{\}$; // Global worst solution
$f_G \leftarrow +\infty$; // Best fitness value
$f_W \leftarrow -\infty$; // Worst fitness value
for $i = 1$ **to** n **do**
  $v_i \leftarrow$ RandomVelocity() ; // Initial velocity
  $S_i \leftarrow$ RandomSequence() ; // Initial sequence
  $(r_1, r_2, r_3)_i \leftarrow$ SetRandomNumbers() ; // Random numbers
  $f_i \leftarrow F(X(S_i), Y)$; // Fitness value
  $P_i \leftarrow S_i$; // Set personal best
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_i\}$;
  if $f_i > f_W$ **then**
    $W \leftarrow S_i$;
    $f_W \leftarrow f_i$;
  if $f_i < f_G$ **then**
    $G \leftarrow S_i$;
    $f_G \leftarrow f_i$;
  $\Gamma \leftarrow \Gamma \cup \{S_i\}$;

Step 2: Initialise neighbourhoods
$k \leftarrow \lfloor n \cdot \alpha \rfloor$; // Number of elites
$E \leftarrow$ UpdateElites($\mathcal{F}, \Gamma, k, \beta$) ; // Set of elites
$\Gamma \leftarrow$ Shuffle(Γ) ;
foreach $i \in \Gamma$ **do**
  $\gamma_i \leftarrow$ SetForgettingAbility(f_i, G, W, n, k) ; // Forgetting ability
  $\mathcal{N}_i \leftarrow$ DefineNeighbourhood(Γ) ; // Neighbourhood set
  $L_i \leftarrow$ UpdateLocalBest($\mathcal{N}_i$) ; // Local best

Step 3: Iterate for a fixed number of iterations
$impr \leftarrow$ False;
for $j = 1$ **to** I **do**
  $f_{\text{worst}} \leftarrow -\infty$;
  $\mathcal{F} \leftarrow \emptyset$;
  Step 4: Evaluate particles, update best/worst solutions
  foreach $i \in \Gamma$ **do**
    $v_i \leftarrow$ UpdateVelocity($\omega, c_1, c_2, c_3, (r_1, r_2, r_3)_i, \gamma_i, \lambda$);
    $S_i \leftarrow$ UpdatePosition(v_i);
    $f_i \leftarrow F(X_{S_i}, Y)$;
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_i\}$;
    if $f_i > f_W$ **then**
      $W \leftarrow S_i$;
      $f_W \leftarrow f_i$;
    if $f_i < P_i$ **then**
      $P_i \leftarrow f_i$;
    if $f_i < f_{best}$ **then**
      $G \leftarrow S_i$;
      $f_G \leftarrow f_i$;
      $impr \leftarrow$ True ;
  if $impr =$ False **then**
    $n_s \leftarrow n_s + 1$; // Stagnating iterations
  else
    $n_g \leftarrow 0$;
  Step 5: Update parameters and neighbourhoods
  if $n_s \geq I_s$ **then**
    $E \leftarrow$ UpdateElites($\mathcal{F}, \Gamma, k, \beta$) ;
    $\Gamma \leftarrow$ Shuffle(Γ) ;
    foreach $i \in \Gamma$ **do**
      $(r_1, r_2, r_3)_i \leftarrow$ UpdateCoefficients(E, β, k) ;
      $\gamma_i \leftarrow$ SetForgettingAbility(f_i, G, W, n, k) ;
      $\mathcal{N}_i \leftarrow$ DefineNeighbourhood(Γ) ;
      $L_i \leftarrow$ UpdateLocalBest($\mathcal{N}_i$) ;

6.2. Comparative evaluation of optimisation runtime

The execution times for the various optimisation strategies compared are presented in Table 6. Notably, the FIFO algorithm was excluded from the table, as it does not optimise the sequence of picking orders and, therefore, its execution time is effectively negligible.

The data revealed a consistent trend across all algorithms: as the order quantity increased, execution times tended to increase accordingly. This was expected, given that larger problem instances inherently require greater computational resources. On the other hand, the fill rate did not have an impact on the execution time. Nonetheless, despite this general pattern, substantial differences in execution times were observed among the algorithms, particularly when comparing RS, GA, and MSSA with CPSO and IPSO. It is evident that CPSO and IPSO consistently required considerably less time than the other methods. The execution times of CPSO and IPSO were comparable and remained significantly lower than those of RS, GA, and MSSA across all tested configurations. This suggests that CPSO and IPSO are highly efficient from a computational standpoint, regardless of problem size or fill rate.

Among the optimisation strategies examined, CPSO and IPSO emerged as the most time-efficient. However, CPSO generally remained the fastest algorithm. Its execution times were consistently lower than those of IPSO, although the difference was marginal. IPSO, despite exhibiting slightly higher execution times compared to CPSO, maintained strong performance in terms of runtime. Specifically, execution times for IPSO ranged from 2.81 s to 7.28 s, compared to 1.52 s to 6.15 s for CPSO. Although IPSO was not as fast as CPSO, it remained a competitive approach.

Overall, the results indicated that PSO-based algorithms were the most efficient optimisation strategies across all tested scenarios, offering reduced execution times and more stable performance. While other algorithms such as RS, GA, and MSSA performed adequately, CPSO and IPSO stood out as the preferred choices for order sequencing optimisation, particularly as problem complexity increases. These findings underscored the potential of IPSO as a high-performance solution for warehouse optimisation systems, where computational efficiency and scalability are of critical importance.

6.3. Comparative evaluation of optimisation improvements

Fig. 6 illustrates the evolution of fitness values across the various algorithms for all evaluated scenarios. The curves depicted in the graphs represent the average fitness function value at each iteration, computed over the 50 replications of the optimisation process. To improve the readability of the results, the figure only includes instances with 10, 30 and 50 orders.

As in the previous analysis, FIFO was employed as a benchmark to assess the improvement achieved by the other optimisation strategies. The general trend observed in the figure indicates that all algorithms progressively enhanced their fitness values over time, albeit to varying extents. In particular, GA, CPSO, and IPSO demonstrated superior performance compared to the remaining strategies.

A notable observation across all tested instances was the consistently superior performance of the IPSO algorithm. IPSO achieved lower fitness values than the other algorithms throughout, indicating greater efficiency and effectiveness in addressing the optimisation problem. This suggests that IPSO is especially well-suited to managing the complexity and variability introduced by differing order quantities and warehouse fill rates.

Another critical insight pertains to the influence of problem size on algorithmic performance. For smaller problem instances, the performance differences among the algorithms were relatively modest. However, as the problem scale increased, these differences became more pronounced. This pattern highlights the varying scalability and robustness of the algorithms. Notably, IPSO sustains its superior performance even in larger and more complex scenarios, further highlighting its efficacy.

Table 5
Hyperparameters of the optimisation algorithms.

Hyperparameter		Optimisation strategy				
Name	Symbol	RS	MSSA	GA	CPSO	IPSO
Number of iterations	I	2000	2000	2000	2000	2000
Population size	n	100	100	100	25	25
Elite size	k	–	–	5	–	–
Crossover rate	–	–	–	0.8	–	–
Mutation rate	λ	–	0.1	0.1	–	0.1
Temperature	–	–	50	–	–	–
Cooling rate	–	–	0.9	–	–	–
Inertia weight	ω	–	–	–	0.25	0.25
Acceleration coefficient (personal)	c_1	–	–	–	0.75	0.75
Acceleration coefficient (local)	c_2	–	–	–	0.1	0.1
Acceleration coefficient (global)	c_3	–	–	–	–	0.5
Stagnation duration	I_s	–	–	–	–	20
Fraction of elite	α	–	–	–	–	0.5
Learning weight from elite	β	–	–	–	–	0.8

Table 6
Optimisation runtime.

		RS		MSSA		GA		CPSO		IPSO	
Orders [#]	Fill rate [%]	Mean [s]	Std [s]	Mean [s]	Std [s]	Mean [s]	Std [s]	Mean [s]	Std [s]	Mean [s]	Std [s]
10	50%	3.31	0.02	3.01	0.02	4.31	0.02	1.49	0.03	2.81	1.07
10	65%	3.28	0.02	2.98	0.02	4.33	0.02	1.52	0.01	2.41	0.01
10	80%	3.3	0.02	2.99	0.02	4.32	0.02	1.51	0.01	2.43	0.01
20	50%	10.28	0.07	9.11	0.1	12.92	0.09	2.39	0.02	3.3	0.02
20	65%	10.05	0.17	9.13	0.08	12.92	0.1	2.34	0.03	3.3	0.02
20	80%	10.18	0.13	9.15	0.1	12.94	0.13	2.39	0.03	3.32	0.02
30	50%	13.83	0.08	12.38	0.04	18.5	0.05	3.26	0.05	4.17	0.05
30	65%	13.81	0.07	12.31	0.08	18.34	0.06	3.4	0.67	4.19	0.04
30	80%	13.78	0.05	12.44	0.05	18.42	0.06	3.25	0.03	4.17	0.04
40	50%	16.86	0.12	14.9	0.11	23.72	0.14	5.21	0.05	6.3	0.1
40	65%	16.91	0.12	14.93	0.09	23.69	0.13	5.18	0.07	6.28	0.08
40	80%	17.16	0.15	14.91	0.08	23.7	0.08	5.2	0.06	6.31	0.1
50	50%	20.12	0.15	17.76	0.04	30.6	0.06	6.09	0.06	7.26	0.08
50	65%	20.02	0.24	17.64	0.07	30.43	0.1	6.09	0.06	7.16	0.14
50	80%	20.23	0.17	17.43	0.09	30.17	0.16	6.15	0.06	7.28	0.15
Average values		12.87	0.11	11.4	0.07	17.95	0.08	3.7	0.08	4.71	0.13

Table 7 presents the relative improvement in the objective function value compared to the FIFO baseline, across the algorithms evaluated, and under different order quantities and fill rates. IPSO consistently demonstrated the highest mean improvement across all scenarios, with an average increase of 7.19%. This confirmed IPSO as the most effective algorithm in enhancing performance relative to FIFO. GA and CPSO also achieved strong results, with average improvements of approximately 7.06% and 6.97%, respectively. In contrast, MSSA and RS exhibited lower average improvements, around 6.61% and 5.05%, respectively. The standard deviations across all algorithms remained relatively low, indicating stable performance. IPSO recorded the lowest standard deviation (0.06%), reinforcing its status as both the best-performing and most consistent method. CPSO and RS exhibited the highest variability, with standard deviations of 0.16% and 0.15%, respectively. A further significant trend concerns the impact of problem scale. As the number of orders increased, the relative improvements in performance became more substantial. This was particularly evident in the larger instances (i.e., 40 and 50 orders), where the differences in mean improvements were more marked. As in the previous analyses, the fill rate did not appear to exert a discernible influence on the improvement rate. Overall, IPSO emerged as the most effective algorithm, particularly in scenarios involving a higher number of orders. GA and CPSO also performed well, often closely trailing IPSO. Conversely, MSSA and RS exhibited the least improvement.

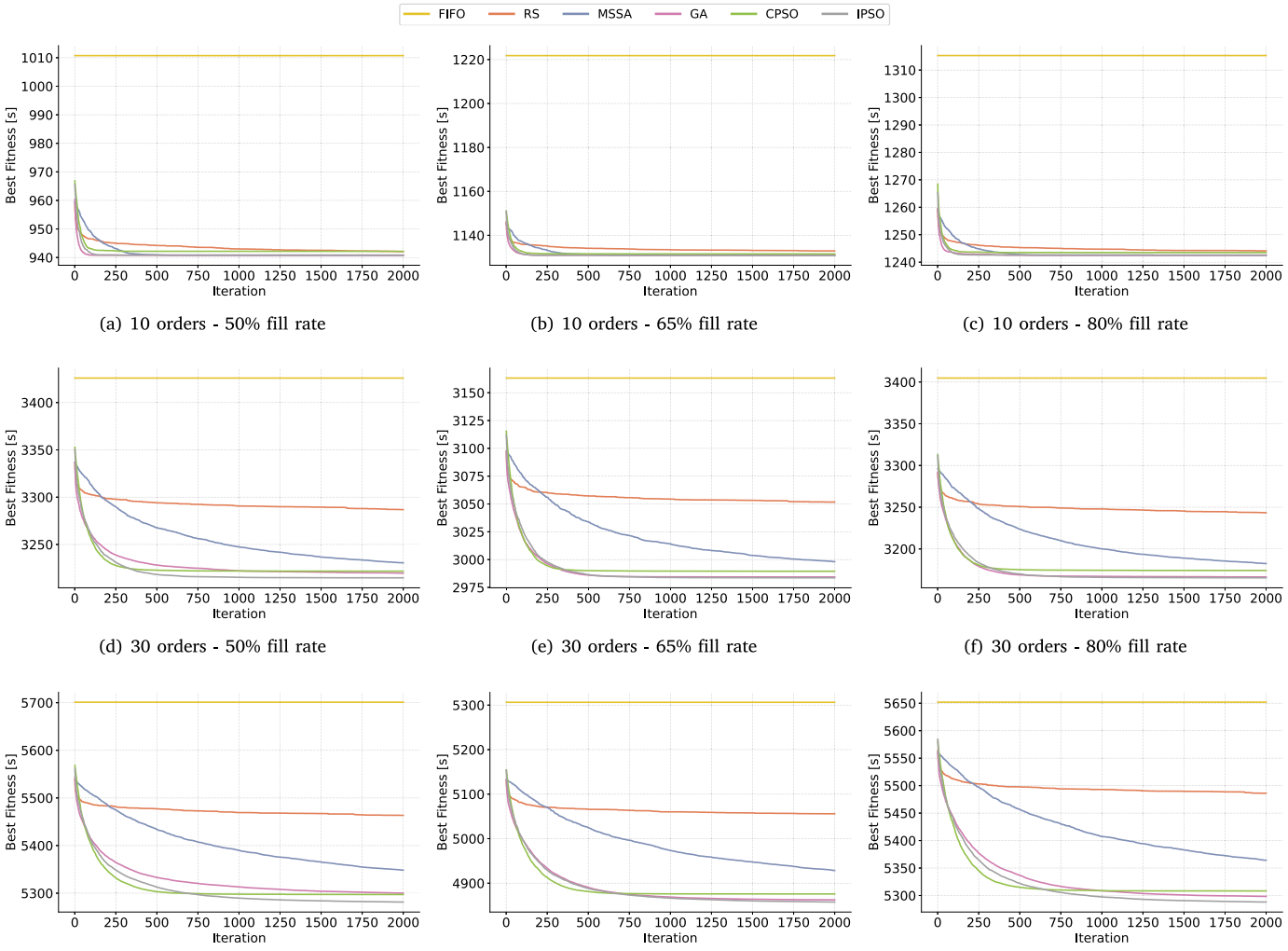
This analysis highlighted the superior and consistent performance of the IPSO algorithm across a range of scenarios. It also emphasised the importance of accounting for problem size when assessing optimisation strategies. In larger and more complex instances, the differences between algorithms became increasingly significant, with IPSO maintaining a clear advantage. When also considering execution times, IPSO offered a compelling balance between computational efficiency and solution quality, making it especially effective for large-scale optimisation tasks. This capacity to combine speed and effectiveness underscored IPSO as the most well-rounded optimisation approach evaluated in this study.

7. Discussions and conclusions

7.1. Main contributions

This study investigates the application of metamodel-assisted SO to address the OSP in an MLS system. A suite of NN-based surrogate models was developed and trained, leveraging the sequential learning capabilities of RNNs. The computational performance of these metamodels was then evaluated in comparison to a traditional DES model, with the aim of assessing potential improvements in execution time. Thus, as an answer to RQ1 and RQ2, the results showed that adopting an NN–based metamodelling approach enables a favourable trade-off between predictive accuracy – specifically in estimating the total completion time of an order sequence – and computational efficiency.

Among the architectures evaluated, the GRU emerged as the best-performing metamodel. This GRU-based surrogate model was then integrated into an improved version of PSO to solve the OSP. The IPSO algorithm incorporates several dynamic and adaptive mechanisms designed to enhance both exploration and exploitation throughout the search process, thereby improving its ability to identify high-quality solutions efficiently. Its computational and optimisation performance



(a) 10 orders - 50% fill rate (b) 10 orders - 65% fill rate (c) 10 orders - 80% fill rate

(d) 30 orders - 50% fill rate (e) 30 orders - 65% fill rate (f) 30 orders - 80% fill rate

(g) 50 orders - 50% fill rate (h) 50 orders - 65% fill rate (i) 50 orders - 80% fill rate

Fig. 6. Optimisation progress.

Table 7
Optimisation improvement.

		RS		MSSA		GA		CPSO		IPSO	
Orders [#]	Fill rate [%]	Mean [%]	Std [%]	Mean [%]	Std [%]	Mean [%]	Std [%]	Mean [%]	Std [%]	Mean [%]	Std [%]
10	50%	6.79%	0.11%	6.92%	0.00%	6.92%	0.00%	6.84%	0.16%	6.92%	0.01%
10	65%	7.28%	0.08%	7.44%	0.00%	7.44%	0.00%	7.38%	0.08%	7.44%	0.00%
10	80%	5.41%	0.07%	5.53%	0.00%	5.53%	0.00%	5.48%	0.08%	5.53%	0.00%
20	50%	5.58%	0.14%	6.87%	0.05%	6.90%	0.07%	6.86%	0.13%	6.96%	0.02%
20	65%	6.92%	0.16%	8.13%	0.05%	8.01%	0.23%	8.14%	0.12%	8.25%	0.05%
20	80%	5.48%	0.11%	6.35%	0.04%	6.39%	0.06%	6.37%	0.08%	6.43%	0.04%
30	50%	4.06%	0.14%	5.70%	0.10%	6.02%	0.17%	6.02%	0.11%	6.14%	0.09%
30	65%	3.52%	0.16%	5.22%	0.08%	5.65%	0.05%	5.50%	0.14%	5.67%	0.03%
30	80%	4.74%	0.16%	6.52%	0.08%	7.00%	0.08%	6.87%	0.11%	7.04%	0.05%
40	50%	4.61%	0.22%	7.32%	0.16%	8.06%	0.25%	8.18%	0.18%	8.37%	0.12%
40	65%	4.60%	0.24%	7.26%	0.13%	8.11%	0.23%	8.08%	0.16%	8.39%	0.08%
40	80%	4.92%	0.19%	7.42%	0.16%	8.23%	0.28%	8.27%	0.16%	8.49%	0.14%
50	50%	4.17%	0.20%	6.19%	0.10%	7.03%	0.15%	7.16%	0.12%	7.37%	0.08%
50	65%	4.72%	0.17%	7.12%	0.14%	8.36%	0.12%	8.18%	0.15%	8.49%	0.08%
50	80%	2.93%	0.16%	5.09%	0.14%	6.26%	0.22%	6.17%	0.16%	6.45%	0.08%
Average values		5.05%	0.15%	6.61%	0.08%	7.06%	0.13%	6.97%	0.16%	7.19%	0.06%

was benchmarked against a set of well-known optimisation strategies, namely FIFO, RS, GA, MSSA, and CPSO. The proposed SO approach was tested on a range of problem instances characterised by varying order set sizes and warehouse capacities. The integration of the GRU with IPSO proved to be an effective strategy, as the results demonstrated a significant reduction in makespan within reasonable computational times. Accordingly, RQ3 was addressed, and it was shown that an NN-assisted SO approach can substantially enhance AS/RS operations.

7.2. Implications

This study yields several key implications. From an academic standpoint, it contributes to the field of SO in automated warehousing by demonstrating the effectiveness of NN-based metamodelling, specifically through the integration of GRU with the IPSO algorithm. The findings show that this hybrid approach improves both prediction accuracy and computational efficiency, offering a robust solution for optimising AS/RS operations. The growing complexity and flexibility demands of warehouse systems expose the limitations of conventional tools. This study highlights the importance of hybrid methods that combine multiple techniques to enable more accurate system modelling and performance optimisation in dynamic environments.

Moreover, although the proposed approach was applied to a specific implementation of an AS/RS – namely, the MLS – the promising results obtained demonstrate its strong potential for broader applicability. In particular, the methodological framework adopted in this study is sufficiently flexible to be extended to a wide range of warehousing systems and configurations, including those with different storage technologies, layout constraints, and operational policies.

From a practical perspective, the successful application of SO to the OSP demonstrates that advanced optimisation techniques can be effectively employed in real-world settings, yielding substantial improvements in makespan. This highlights not only the practical value of the proposed approach, but also its potential for broader adoption within AS/RS industry. The methodology provides a reliable tool for predicting system behaviour and supporting daily operational decision-making in automated warehouse environments. AS/RS are increasingly required to handle greater complexity, driven by the high uncertainty and volatility of recent market demands. Unpredictable events in recent years have compelled many companies to partially or entirely redesign their supply chains and inventory management strategies. In this context, the ability of automated warehouse systems to rapidly adapt to change and efficiently re-sequence incoming orders can represent a critical source of competitive advantage.

7.3. Limitations and future research

Despite the promising results, this work presents several limitations. In particular, the analysis did not consider potential stock-outs and their impact on the order sequence, nor was order priority accounted for, despite its potential to significantly influence the optimisation process. Long system downtimes due to maintenance, breakdowns, or repairs were also not included in the analysis. Furthermore, the metamodel was assumed to be deterministic, which neglects the stochastic nature inherent in real-world operations. Addressing these limitations would improve both the comprehensiveness and the practical applicability of the findings.

Future research can build on this study along both contextual and methodological dimensions. From a contextual perspective, the current work was conceived to demonstrate the applicability of the NN-assisted simulation-optimisation approach within a specific operational setting, namely a fixed warehouse configuration. Despite this focus, the promising results suggest that the methodology has potential for extension to larger-scale scenarios, including instances involving hundreds of orders, and for adaptation to alternative system configurations with increased complexity. In addition, developing metamodels capable of predicting the occurrence of deadlocks or other system variables, such as energy consumption, could provide deeper insights and support more comprehensive optimisation solutions.

From a methodological standpoint, integrating stochastic elements into the predictions could enhance the robustness and reliability of the metamodel. Approaches such as Bayesian NNs could be employed to capture uncertainty and improve decision-making. Furthermore, exploring more advanced neural network architectures, including transformer-based models or temporal convolutional networks, may expand the scope of the findings and potentially yield additional performance gains. Future studies could also investigate the integration of the NN-based metamodel with additional metaheuristic and hybrid algorithms, such as Tabu Search, Ant Colony Optimisation, or memetic approaches, to further assess the scalability and adaptability of NN-assisted SO in solving complex and larger-scale problems in automated warehouse systems.

Collectively, these avenues highlight ongoing opportunities for research and improvement in the optimisation of picking processes, providing a foundation for developing more robust, efficient, and adaptable solutions to the challenges faced in modern automated warehousing.

CRediT authorship contribution statement

Andrea Ferrari: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Software, Writing – Original Draft . **Canan Gunes Corlu:** Conceptualization, Methodology, Supervision, Writing – Review & Editing . **Giulio Mangano:** Supervision, Validation, Writing – Review & Editing.

Data availability

The code and data that support the findings of this study are available on the following GitHub page (https://github.com/log-lab-polito/MetaSimOpt).

References

Abid, S., & Mhada, F. Z. (2021). Simulation optimisation methods applied in reverse logistics: a systematic review. *International Journal of Sustainable Engineering*, *14*(6), 1463–1483.

Ahn, J., Lee, Y., Han, B., Lee, S., Kim, Y., Chung, D., & Jeon, J. (2025). A highly effective and robust structure-based LSTM with feature-vector tuning framework for high-accuracy SOC estimation in EV. *Energy*, Article 136134.

Amaran, S., Sahinidis, N. V., Sharda, B., & Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, *240*, 351–380.

Amorim-Lopes, M., Guimarães, L., Alves, J., & Almada-Lobo, B. (2021). Improving picking performance at a large retailer warehouse by combining probabilistic simulation, optimization, and discrete-event simulation. *International Transactions in Operational Research*, *28*(2), 687–715.

Azadeh, K., De Koster, R., & Roy, D. (2019). Robotized and automated warehouse systems: Review and recent developments. *Transportation Science*, *53*(4), 917–945.

Barton, R. R., & Meckesheimer, M. (2006). Metamodel-based simulation optimization. *Handbooks in Operations Research and Management Science*, *13*, 535–574.

Boareto, P. A., Moretti, L. N., Safanelli, J., Liberato, R. B., Moro, C. H., Junior, J. E. P., Moro, C. M., dos S Coelho, L., Loures, E. F., Deschamps, F., et al. (2025). Simulation optimization-based model for decision-making in the stroke clinical pathway. *Computers & Industrial Engineering*, Article 111164.

Chen, T.-L., Chen, J. C., Huang, C.-F., & Chang, P.-C. (2021). Solving the layout design problem by simulation-optimization approach–a case study on a sortation conveyor system. *Simulation Modelling Practice and Theory*, *106*, Article 102192.

Chen, J. C., Chen, T.-L., & Lee, Y.-H. (2023). Simulation optimization for parcel hub scheduling problem in closed-loop sortation system with shortcuts. *Simulation Modelling Practice and Theory*, *124*, Article 102728.

Das, K. (2008). A comparative study of exponential distribution vs Weibull distribution in machine reliability analysis in a CMS design. *Computers & Industrial Engineering*, *54*(1), 12–33.

Derkinderen, V., Bekker, J., & Smet, P. (2023). Optimizing workforce allocation under uncertain activity duration. *Computers & Industrial Engineering*, *179*, Article 109228.

Dijkstra, A. S., & Roodbergen, K. J. (2017). 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.

Emambocus, B. A. S., Jasser, M. B., Hamzah, M., Mustapha, A., & Amphawan, A. (2021). An enhanced swap sequence-based particle swarm optimization algorithm to solve TSP. *IEEE Access*, *9*, 164820–164836.

Epp, M., Wiedemann, S., & Furmans, K. (2017). A discrete-time queueing network approach to performance evaluation of autonomous vehicle storage and retrieval systems. *International Journal of Production Research*, *55*(4), 960–978.

Fandi, W., Kouloughli, S., & Ghomri, L. (2022). Multi-shuttle AS/RS dimensions optimization using a genetic algorithm—case of the multi-aisle configuration. *International Journal of Advanced Manufacturing Technology*, *120*(1–2), 1219–1236.

Feliciani Merizio, I., Chavarette, F. R., & Fuzaro de Almeida, E. (2025). Machine learning techniques for fault detection in rotating mechanical systems. *Engineering Computations*, *42*(3), 1316–1334.

Ferrari, A., Campanale, C., Carlin, A., & Mangano, G. (2024a). Performance assessment of a multi-level shuttle system: an optimisation-embedded simulation study on time and energy consumption. *Journal of Simulation*, 1–16.

Ferrari, A., Carlin, A., Rafele, C., & Zenezini, G. (2023). A method for developing and validating simulation models for automated storage and retrieval system digital twins. *International Journal of Advanced Manufacturing Technology*, 1–14.

Ferrari, A., & Corlu, C. G. (2024). Metamodel-based order picking for automated storage and retrieval systems. In *2024 winter simulation conference* (pp. 1457–1468). IEEE.

Ferrari, A., Verso, A., Carlin, A., & Rafele, C. (2024b). Simulation study of a multi-level shuttle system with in-rack picking stations. *IFAC-PapersOnLine*, *58*(19), 848–853.

Ferrari, A., Zenezini, G., Rafele, C., & Carlin, A. (2022). A roadmap towards an automated warehouse digital twin: Current implementations and future developments. *IFAC-Papersonline*, *55*(10), 1899–1905.

Finco, S., Ashta, G., Persona, A., & Zennaro, I. (2023). Investigating different manual picking workstations for robotized and automated warehouse systems: trade-offs between ergonomics and productivity aspects. *Computers & Industrial Engineering*, *185*, Article 109668.

Gagliardi, J.-P., Renaud, J., & Ruiz, A. (2015). Sequencing approaches for multiple-aisle automated storage and retrieval systems. *International Journal of Production Research*, *53*(19), 5873–5883.

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc.

Giner, J., Katic, D., Kovacs, K., Glawar, R., & Sihn, W. (2023). A computer vision based approach to reduce system downtimes in an automated high-rack logistics warehouse. *Procedia CIRP*, *118*, 1078–1083.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249–256). JMLR Workshop and Conference Proceedings.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press, http://www.deeplearningbook.org.

Haas, P. J. (2024). Tutorial: Artificial neural networks for discrete-event simulation. In *2024 winter simulation conference* (pp. 116–130). IEEE.

Han, M.-H., McGinnis, L. F., Shieh, J. S., & White, J. A. (1987). On sequencing retrievals in an automated storage/retrieval system. *IIE TRansactions*, *19*(1), 56–66.

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

Hsu, H.-P., & Wang, C.-N. (2023). Hybridizing whale optimization algorithm with particle swarm optimization for scheduling a dual-command storage/retrieval machine. *IEEE Access*, *11*, 21264–21282.

Jamili, N., van den Berg, P. L., & de Koster, R. (2025). The impact of resource sharing on the design of multi-client warehouses. *IISE Transactions*, *57*(3), 336–350.

John, L. S., Yoon, S., Li, J., & Wang, P. (2025). Anomaly detection using convolutional autoencoder with residual gated recurrent unit and weak supervision for photovoltaic thermal heat pump system. *Journal of Building Engineering*, *100*, Article 111694.

Juan, A. A., Keenan, P., Martí, R., McGarraghy, S., Panadero, J., Carroll, P., & Oliva, D. (2023). A review of the role of heuristics in stochastic optimisation: From metaheuristics to learnheuristics. *Annals of Operations Research*, *320*(2), 831–861.

Justkowiak, J.-E., & Pesch, E. (2023). A column generation driven heuristic for order-scheduling and rack-sequencing in robotic mobile fulfillment systems. *Omega*, *120*, Article 102897.

Ko, D., & Han, J. (2022). A rollout heuristic algorithm for order sequencing in robotic compact storage and retrieval systems. *Expert Systems with Applications*, *203*, Article 117396.

Lamont-Smith, T. (2019). Power transforms for the Weibull and the K-distribution. *IET Radar, Sonar & Navigation*, *13*(4), 522–529.

Law, A. M., Kelton, W. D., & Kelton, W. D. (2007). *Simulation modeling and analysis*: *Vol. 3*, Mcgraw-hill New York.

Lehmann, T., & Hußmann, J. (2021). Travel time model for multi-deep automated storage and retrieval system with a homogeneous allocation structure. *Logistic Research*, *14*(1), 5.

Li, L., Chen, Z., & Zhang, T. (2020). Hybrid-grouping-matching heuristic for the storage assignment in gravity AS/RSs with duration-of-stay storage policy. In *Proceedings of the 4th international conference on computer science and application engineering* (pp. 1–6).

Li, J., González, M., & Zhu, Y. (2009). A hybrid simulation optimization method for production planning of dedicated remanufacturing. *International Journal of Production Economics*, *117*(2), 286–301.

Lienert, T., & Fottner, J. (2018). Routing-based sequencing applied to shuttle systems. In *2018 21st international conference on intelligent transportation systems* (pp. 2949–2954). IEEE.

Ma, B. J., Pan, S., Zou, B., Kuo, Y.-H., & Huang, G. Q. (2025). Operating policies for robotic cellular warehousing systems. *Transportation Research Part E: Logistics and Transportation Review*, *194*, Article 103875.

Malakouti, S. M., Menhaj, M. B., & Suratgar, A. A. (2023). The usage of 10-fold cross-validation and grid search to enhance ML methods performance in solar farm power generation prediction. *Cleaner Engineering and Technology*, *15*, Article 100664.

Malde, A. K., Işik, T., Lockaby, R., Gantt, A., & Thumser, G. (2022). Optimal team formation and job assignment to optimize warehouse operations. In *2022 winter simulation conference* (pp. 1772–1782). IEEE.

Maniezzo, V., Boschetti, M. A., & Gutjahr, W. J. (2021). Stochastic premarshalling of block stacking warehouses. *Omega*, *102*, Article 102336.

Marolt, J., Kosanić, N., & Lerher, T. (2022). Relocation and storage assignment strategy evaluation in a multiple-deep tier captive automated vehicle storage and retrieval system with undetermined retrieval sequence. *International Journal of Advanced Manufacturing Technology*, *118*(9), 3403–3420.

Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information*, *15*(9), 517.

Nili, M. H., Taghaddos, H., & Zahraie, B. (2021). Integrating discrete event simulation and genetic algorithm optimization for bridge maintenance planning. *Automation in Construction*, *122*, Article 103513.

Odeyemi, J., & Zhang, W. (2025). Benchmarking randomized optimization algorithms on binary, permutation, and combinatorial problem landscapes. arXiv preprint arXiv:2501.17170.

Panda, M. (2018). Performance comparison of genetic algorithm, particle swarm optimization and simulated annealing applied to TSP. *International Journal of Applied Engineering Research*, *13*(9), 6808–6816.

PrecedenceResearch (2024). Warehouse automation market size, share, and trends 2024 to 2034. https://www.precedenceresearch.com/warehouse-automation-market.

Rizqi, Z. U., & Chou, S.-Y. (2024). Multi-speed configuration of AS/RS amidst responsiveness and energy efficiency trade-off: metamodel-based simulation–optimization. *International Journal of Advanced Manufacturing Technology*, *134*(3), 1711–1728.

Rizqi, Z. U., Chou, S.-Y., & Khairunisa, A. (2024). Multi-objective simulation-optimization for integrated automated storage and retrieval systems planning considering energy consumption. *Computers & Industrial Engineering*, *189*, Article 109979.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv preprint arXiv:1312.6120.

Silva, A., Roodbergen, K. J., Coelho, L. C., & Darvish, M. (2022). Estimating optimal ABC zone sizes in manual warehouses. *International Journal of Production Economics*, *252*, Article 108579.

Song, X., Yang, F., Wang, D., & Tsui, K.-L. (2019). Combined CNN-LSTM network for state-of-charge estimation of lithium-ion batteries. *IEEE Access*, *7*, 88894–88902.

Suemitsu, I., Bhamgara, H. K., Utsugi, K., Hashizume, J., & Ito, K. (2022). Fast simulation-based order sequence optimization assisted by pre-trained Bayesian recurrent neural network. *IEEE Robotics and Automation Letters*, *7*(3), 7818–7825.

Sun, Y., Lin, K., Wang, J., Zhu, F., Wang, L., & Lu, L. (2025). Risk assessment of mountain tunnel entrance collapse based on PSO-LSTM surface settlement prediction. *Engineering, Construction and Architectural Management*.

Tadumadze, G., Wenzel, J., Emde, S., Weidinger, F., & Elbert, R. (2023). Assigning orders and pods to picking stations in a multi-level robotic mobile fulfillment system. *Flexible Services and Manufacturing Journal*, 1–38.

Urnauer, C., Bosch, E., & Metternich, J. (2019). Simulation-based optimization of sequencing buffer allocation in automated storage and retrieval systems for automobile production. In *2019 winter simulation conference* (pp. 1602–1611). IEEE.

Wang, K.-P., Huang, L., Zhou, C.-G., & Pang, W. (2003). Particle swarm optimization for traveling salesman problem. In *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE cat. no. 03ex693)*: *Vol. 3*, (pp. 1583–1585). IEEE.

Wang, W., Wu, Y., & Wu, Y. (2016). A multi-stage heuristic algorithm for matching problem in the modified miniload automated storage and retrieval system of e-commerce. *Chinese Journal of Mechanical Engineering*, *29*, 641–648.

Wang, B., Yang, X., & Qi, M. (2022). Order and rack sequencing in a robotic mobile fulfillment system with multiple picking stations. *Flexible Services and Manufacturing Journal*, 1–39.

Xia, X., Gui, L., He, G., Wei, B., Zhang, Y., Yu, F., Wu, H., & Zhan, Z.-H. (2020). An expanded particle swarm optimization based on multi-exemplar and forgetting ability. *Information Sciences*, *508*, 105–120.

Xu, J., Huang, E., Chen, C.-H., & Lee, L. H. (2015). Simulation optimization: A review and exploration in the new era of cloud computing and big data. *Asia-Pacific Journal of Operational Research*, *32*(03), Article 1550019.

Xu, X., Zhao, X., Zou, B., & Li, M. (2019). Optimal dimensions for multi-deep storage systems under class-based storage policies. *Cluster Computing*, *22*, 861–875.

Yang, X., Hua, G., Hu, L., Cheng, T., & Huang, A. (2021). Joint optimization of order sequencing and rack scheduling in the robotic mobile fulfilment system. *Computers & Operations Research*, *135*, Article 105467.

Zhao, N., & Wang, M. (2021). Research on parameter optimization of the express warehousing and distribution system based on the box–behnken response surface methodology. *Advances in Civil Engineering*, *2021*, 1–9.

Zhen, L., Tan, Z., de Koster, R., He, X., Wang, S., & Wang, H. (2025). Optimizing warehouse operations with autonomous mobile robots. *Transportation Science*.

Zhou, L., Zhao, J., Liu, H., Wang, F., Yang, J., & Wang, S. (2022). Stochastic models of routing strategies under the class-based storage policy in fishbone layout warehouses. *Scientific Reports*, *12*(1), 12876.