Simulation Modelling Practice and Theory 99 (2020) 102016

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

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

Neural networks for the metamodeling of simulation models with online decision making

Fabian Dunke*, Stefan Nickel

Karlsruhe Institute of Technology, Institute of Operations Research, Discrete Optimization and Logistics, Kaiserstr. 12, 76131 Karlsruhe, Germany

ARTICLE INFO

Keywords:
Simulation metamodeling
Artificial neural network
Simulation optimization
Online optimization
Order picking system

ABSTRACT

We present a methodology for an artificial neural network (ANN) based metamodeling of simulation models in the special case when online decision making routines are invoked repetitively by the simulation model throughout the simulation run. For a practitioner, the benefit of such a simulation metamodel lies in the possibility to compare different decision making routines (operational control strategies) without excessive computational time for running multiple simulation configurations with different control strategies. Contrasting to the conventional setting of ANN based simulation metamodeling, in this paper ANNs have to take as input not only numerical parameters, but also different control strategies. The methodology is finally put into practice in a case study of an order picking system. Results show that on average the relative error of the ANN metamodel is fairly acceptable and allows for a first assessment of system parameters and control strategies. However, also large outliers in the relative error are encountered. Hence, for the analysis of different parameters and control strategies in applications, the use of ANN-based simulation metamodels requires an accompanying statistical assessment of ANN-based performance prediction accuracies.

1. Introduction

We consider real world dynamic systems which require operational decisions on a repetitive basis. Plenty of such systems can be found, for instance in the context of supply chain management, logistics, production, or power management. In the setting of this paper, these decisions are evoked through a specific decision making routine, an operational control strategy. Hence, operational decisions are intentionally modelled, determined, and not relinquished to randomness. In fact, different candidates of such operational control strategies may be of interest for operating a dynamic system, and the question may be which one of the proposed control strategies will provide the best system performance. This approach of explicit decision making is in contrast to applying random mechanisms or simple rule-based policies (priority rules) as implemented in a large number of simulation models, and it provides a coupling of simulation with operational optimization methods. The latter are also referred to as online optimization methods since decisions have to be communicated to the calling simulation model "on-the-line". As seen in the schematic Fig. 1, decisions are determined upon calling an online optimization module repetitively as part of the simulation run (inner loop between simulation and online optimization) in order to request decision proposals throughout the course of time. The explicit consideration of different online decision making methods (online algorithms) represents a significant step towards realizing the potential of computational possibilities in real world applications. In fact, improving the efficiency of operations in practice requires first to identify options for choice of action and then to devise goal-oriented methods supporting the decision making. Hence, modeling these

* Corresponding author.
E-mail addresses: fabian.dunke@kit.edu (F. Dunke), stefan.nickel@kit.edu (S. Nickel).

https://doi.org/10.1016/j.simpat.2019.102016
Received 1 August 2019; Received in revised form 20 September 2019; Accepted 18 October 2019
Available online 23 October 2019
1569-190X/

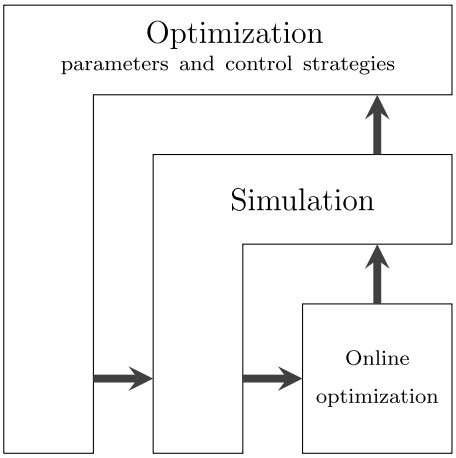

Fig. 1. Relation between optimization, simulation, and online optimization.

degrees of freedom in an optimization problem and devising related solution algorithms contributes to operating dynamic systems in accordance with a decision maker's goal system. On the other hand, the previously mentioned search for a promising online algorithm to apply in the inner loop represents an optimization problem itself. This design decision is to be determined in an interplay between an optimization for finding the most suitable algorithm and the simulation model (outer loop between optimization and simulation). Typically, also input parameters for the simulation are optimized through the outer loop between optimization and simulation. Therefore, in the outer loop the simulation is used as an evaluation function for the parameter and algorithm proposals prescribed by the outer optimization engine. This is in contrast to the inner loop where the online optimization algorithm is used as an executable element of the simulation model influencing how the system advances over time.

From the nesting between optimization and simulation (outer loop) and simulation and online optimization (inner loop), it becomes apparent that the optimization task on the upper level can be viewed as a search for an optimal point of a concatenated function. Unfortunately, online algorithms to be employed on the lower level are only executed when called by the simulation routine as part of an executable model. Therefore, it is impossible to provide an analytical model which could be used for the optimization in the outer loop. Likewise, metaheuristics will encounter the same difficulty, namely that decisions of the online algorithm cannot be encapsulated in the algorithmic outline without executing them.

Through modeling both the outer and inner feedback mechanisms by an artificial neural network (ANN) we take on another approach in this paper. This paper addresses the question whether simulation metamodeling via ANNs is a reasonable approach for the analysis of simulation models with integrated explicit decision making in the form of online optimization algorithms. While ANN-based simulation metamodeling has been shown to provide reasonable results in a number of applications (e.g., energy management [1], inventory and production [2], scheduling [3–5]), there has been neither a systematic consideration nor analysis of explicit online optimization algorithms as part of simulation models. Concerning the enormous importance of control decisions in practice, we feel that in fact one of the main tasks in planning and analyzing real world dynamic systems amounts to determining those control algorithms which are most suitable for efficient operations.

The main contributions of the paper consist of a proposed methodology for determining the suitability (in terms of accuracy) of an ANN-based simulation metamodel in the case of integrated online decision making and the subsequent assessment of this methodology in a practical setting of an order picking system. Although in this case study risks are identified concerning an ANN-based simulation metamodel as a substitute for different simulation model configurations, the analysis shows that important insights can be found concerning the quality of different operational control strategy candidates. The rest of the presentation is organized as follows: Section 2 discusses relevant literature on the relation between simulation, optimization and online (real time) optimization, as well as literature on ANN-based simulation metamodels and order picking systems. Section 3 then presents an overall methodology for using ANN-based metamodels in case of simulation models which include online decision making routines. In Section 4, we then check the applicability of the overall approach in a case study of an order picking system where we employ several candidate algorithms for online decisions with respect to batching and routing decisions. The paper closes with a conclusion and outlook for future research in Section 5.

2. Related work

We review literature on the topics of online decision making, simulation optimization, simulation metamodeling through ANNs, and order picking systems. The presentation of each topic is organized as follows: First, the most relevant literature on the topic itself is discussed. Then, at the end of each paragraph, we examine related work which addresses a combination of the topics. Online decision making is found to be a rather common feature of dynamic systems whose state trajectories are subject to choice of action. The review on simulation optimization covers the role between simulation and optimization as well as the real-time aspect which becomes crucial in operational control. ANN-based simulation metamodeling is presented as a computationally viable analysis option

for the complex task of parameter selection in simulation models. As seen in the review on order picking systems, considering different algorithms as candidates for operational control strategies represents a significant enabler of efficient logistics operations.

2.1. Online decision making

Online decision making can be employed in a dynamic system whenever the system requires a decision in order to proceed. Related computational methods are called online algorithms ([6]) and their analysis is subject to the fact that decisions have to be made without knowing all future data ([7]). A prerequisite for online decision making is the prevalence of degrees of freedom with respect to an action which needs to be carried out to prevent the system from a deadlock. Online algorithms have also been considered as so-called rolling horizon planning schemes in many applications from production and logistics ([8–10]). The goal of online decision making consists of determining decisions in a manner as beneficial as possible for the overall goal(s) associated to the dynamic system. Hence, online decision making is concerned with determining partial solutions; and the overall solution can only be composed from the partial solutions at the end of the planning horizon ([11]). As a consequence of the analogies between online decision making and general discrete event systems ([12,13]), partial solutions then have to be determined such that feasibility of the associated actions in the dynamic system is ensured. To this end, any decision making method has to take into account the constraints prescribed by the current system state and general requirements of the system under consideration. The decision making method in this paper is called the operational control strategy. Due to the lack of a specific objective function for determining partial solutions, operational control strategies can be designed in various ways ([14]). For instance, one can formulate a mathematical programming model of the snapshot problem that the decision maker selects in conviction of its benefit to the overall goal; another frequently used type of operational control strategy relies on rule-based decision making which intends to exploit known trade-offs affecting overall system performance.

As an example, consider the order picking system discussed in this paper (cf. also Sections 2.4 and 4). A general discussion on online decision making for order picking systems is surveyed in [15]. Overall goals of warehouse operators may comprise the minimization of makespan (to save personnel costs), of walking distance (to implement efficient logistics), or of waiting time (to serve customers as quickly as possible). Since items will be retrieved by pickers, decisions have to prescribe at which time which item will be picked by which picker, i.e., routing and batching decisions have to be determined such that pickers start at the consolidation point (depot), follow their route for collecting articles which do not exceed their carrying capacity, and end their route at the depot. This example shows that online decision making may also be subject to the multi-objective setting. Therefore, simulation models are a suitable means to display complex interactions between agents, their actions, and the impact of decisions on solution quality ([16]).

2.2. Simulation optimization

Due to complexity and inherent uncertainty, exact mathematical methods to determine optimal system parameter configurations (including control strategies), such as mathematical programming or dynamic programming, cannot be used for the analysis real world systems. Since “simulation is the reproduction of a system along with its dynamic processes in an executable model in order to retrieve results which are transferable to reality” ([17]), simulation may be used as an executable method participating in the parameter optimization process. Results on system performance are derived from collecting data over a sufficient number of replications which then serve as the basis to determine (point, interval or distributional) estimates for performance measures of interest ([12]). According to [18], the goal in simulation optimization is to solve with the help of simulation the optimization problem

$$\min_{\theta \in \Theta} \mathbb{E}_I(Y(i, \theta)).$$

$\mathbb{E}_I(\cdot)$ denotes the expectation over instance set I, θ is the (multi-dimensional) vector to be optimized from parameter space Θ, and Y gives the output of the simulation model using $\theta \in \Theta$ for $i \in I$. Since evaluations of simulation outputs are often carried out in an elaborate design of experiments, terms such as “factors” (synonymously to input parameters) or “responses” (synonymously to output measures) is used frequently ([19]). Also observe that the above notation suggests that a single objective is analyzed whereas in practice multi-objective settings are rather usual and still have not been addressed sufficiently in research ([20]).

Simulation optimization methodologies can be classified into ranking and selection, random search and metaheuristics, direct gradient methods, and metamodel methods ([18,20–22]). Methodologies are selected depending on the nature of the simulation model and parameter space. Ranking and selection is recommended in case of a discrete search space with limited number of solutions, random search and metaheuristics are used in case of a discrete search space with a high number of alternatives. Direct gradient methods and metamodel methods assume a detectable relation between elements in the search space and simulation outcome. While gradient methods assume a functional dependence allowing to use gradients in the search process, metamodels assume a mechanistic and causal relationship capturable through models such as response surface models or ANNs. We observe that metaheuristic-based software packages (*OptQuest, SimRunner, RiskOptimizer*) have by far gained the largest share of attention in the context of simulation optimization as these are also implemented in a vast number of simulation software packages ([19,23,24]).

The feedback between online optimization and simulation model (cf. Fig. 1) allows to integrate rational decision-making into a simulation model. To this end, online algorithms are utilized as operational control strategies providing decisions as required by the simulation logic ([16]). These methods lead to improved outcomes compared to simulation models where decision making is not modeled explicitly but implemented through random mechanisms or simple priority rules. The applicability of online algorithms within simulation models is methodologically underpinned by the relationship between online optimization and discrete event

systems ([11,16]). As a result, online algorithms can be integrated into discrete event *simulations* which provide a discrete event system with input in the form of events over time. Literature devoted to the relation between online optimization and simulation is rather sparse and only exists with respect to specific applications ([14,25–27]).

Explicit use of ANNs in simulation optimization is made by Laguna and Marti ([28]) who describe an online procedure for tuning parameters of simulation models according to the output of an ANN. The authors use (feedforward) ANNs to eliminate solutions ranked bad by the metamodel as for the related low probability of showing good behavior. The overall search for an optimal parameter setting is controlled by a scatter search algorithm. The real time training procedure is implemented for job shop scheduling. The authors acknowledge that perfect optimization of parameters in simulation models is impossible, and that (meta)heuristic approaches yield at least more satisfying solutions. In this sense, because of confidence interval requirements, combinatorial explosion of the search space (curse of dimensionality), and lack of structure for many search spaces, it may also be an option to opt for an ordinal optimization, i.e., a ranking of parameter alternatives ([24]).

2.3. ANN-based simulation metamodeling

Running simulation models can be quite expensive in terms of computational time ([1,22,29]). This holds especially true in case of integrated decision making which requires computational time itself. Simulation metamodels offer a trade-off between accuracy and efficiency needed for extensive simulative analysis with high degrees of freedom in the parameter space. Simulation metamodels try to predict the input-output-relations of a simulation model through another functional model so as to provide robust and fast decision support. As a result, metamodel-based simulation optimization represents a promising methodology: Results are deterministic, run times are short, and the metamodel can be executed easily for a vast number of input factors/design parameters ([30]).

There are several classes of simulation metamodels, each with specific advantages, disadvantages, and applicability ranges ([1,29]). Response surface models and polynomial regression analysis opts at finding (low order) polynomial functions for simulation input-output relations. Advanced regression settings such as multivariate adaptive regression splines employ a sum of basis functions. Kriging models extend these models by a Gaussian process allowing to interpolate residuals leading to so-called spatial correlation. In contrast, ANNs (for a systematic introduction see, e.g., [31]) exhibit an architecture of neurons interacting with each other in resemblance to the human brain. ANNs are designed to transform inputs from the input layer through hidden layers into outputs at the output layer. Each layer consists of a number of neurons transferring signals according to the signal strength obtained from neurons of other layers. Hence, neurons are wired through stages. From the approximability theorem ([32,33]) it follows that there is an ANN for any continuous function.

As observed by the literature review and analysis in [34], there is an increasing number of publications combining ANNs and simulation, mainly exactly for simulation metamodeling ([1]) or as part of a metaheuristic simulation optimization process as implemented, e.g., in *OptQuest, Optimiz*, or *Simul8* ([19,20]). Historically, ANNs are one of the newer methods succeeding classical regression-type metamodels and they are said to be more flexible and computing-time efficient than regression models. However, convergence analysis or other statistical properties cannot be shown due to the rather general outline. In [1], different metamodels are compared in the context of energy simulation. The authors find from their literature review that from the class of metamodeling techniques, ANNs (in case of many inputs) and Kriging (in case of a high degree of non-linearity and little noise) are recommendable in producing good approximations with respect to standard performance measures such as the root mean squared error, coefficient of determination, maximal absolute error.

ANNs were used in a number of industrial applications – mainly from production and logistics – to approximate input-output relationships of related discrete-event simulation models: [2,35] compare genetic programming and ANN-based metamodeling for inventory control, automated material handling and a production line. Kuo et al. [36] considers dispatching problems in material handling systems for semiconductor wafer fabrication[37]. considers scheduling problems in a manufacturing system. [3] uses ANNs for job shop sequencing and concludes that ANN-based simulation metamodeling – in combination with genetic algorithms for controlling the overall parameter optimization – is an efficient way for optimizing simulation model parameters. Altiparmak et al. [38,39] address strategic decisions in assembly line balancing, [40] considers the multi-product setting in manufacturing systems. From these case studies on ANN-based simulation metamodeling and through improved computing capacities over the past years, it can be concluded that ANNs are getting an edge over more traditional metamodeling approaches such as regression analysis. Therefore, [41] devises a general framework for ANN-based simulation metamodeling. Our methodology presented in Section 3 basically relies on the steps proposed in the framework and adapts them to the case where operational control strategies appear as input neurons. Moreover, ANNs also exhibit the advantage that multi-criteria evaluations are easily possible through having one neuron for each objective. The universal approximation theorem ([32,33]) gives a justification for using ANNs as simulation metamodels as it states that a feed-forward network with a single hidden layer containing a finite number of neurons is already sufficient to approximate any measurable function on compact subsets of $\mathbb{R}^n$ with any desired accuracy. However, the theorem does not give any information on how the network shall be parametrized concretely.

Related to an analysis of control strategies as considered in this paper, [4,5] utilize ANNs to assess and determine (i.e., select dynamically) dispatching rules for scheduling in manufacturing systems. The ANN is trained offline with different input scenarios where the input includes the system state and system parameters. The ANN then has one output neuron for each dispatching rule. Output neurons can be ranked by the signal strength that results, i.e., they provide a mapping of input scenarios to dispatching rule rankings. After learning the training data the ANN then gives the recommended dispatching rule for new cases. The ANN in turn is embedded into the simulation optimization search process aiming at optimizing performance depending on system parameters. Observe that this is different from the setting of this paper in that the ANN shall be employed operationally to provide a

recommendation for the most suitable dispatching rule. Thus, the ANN is used as an online algorithm rather than as a simulation metamodel.

2.4. Order picking systems

Order picking is the activity of retrieving items of customer orders from their storage locations in order to consolidate them such that they can be sent collectively to the customers ([42]). Order picking systems are typically arranged in aisles, each one having an organized form of storage space for the items in racks. Upon arrival of customer orders and their registration in the material flow controller, items of orders are collected in an organized manner either by human pickers or automated devices. Strategic warehousing problems consist of structural decisions, e.g., how the system should be integrated with other warehouse processes, ([43,44]) or designing and dimensioning the layout of the aisles and racks ([45,46]). Once a layout is fixed, the operational task amounts to item storage and retrieval. For the former task, well-known strategies are random (chaotic) storage, dedicated storage, or class-based storage ([45]); the latter task is the most cost-intensive as it is carried out on a daily basis incurring high labor costs. We will be interested most in this order picking process due to its challenging operational nature requiring online decisions repeatedly. Item retrieval involves two decisions (batching and routing) due to finite picker capacities and the need for item collection. These two interrelated decisions have to be solved repetitively throughout the day for yet unserved orders. Algorithms for both problems are surveyed in [15]. In the case study in Section 4 we will use two different batching algorithms and four different routing algorithms as batching and routing control strategies.

1. Batching: Assignment of items of orders to pickers/picking vehicles such that the total number of items of assigned orders does not exceed the picker capacity.
2. Routing: Route determination for each picker such that all items of assigned orders are visited and the route starts and ends at the depot/transfer point.

In several papers, order picking systems have been analyzed by simulative methods. The picker routing problem has been addressed with increasing frequency over the past years. Publications mainly focus on a simulation-based analysis of different routing strategies in special settings such as warehouses with narrow aisles ([47]) or special aisle configurations ([48]), pickers deviating from prescribed routes due to human errors ([49]), adaptive pick list modification during the picking process ([50]), or low-level picker-to-parts systems ([51]). Methodologically similar, also solution procedures for the item batching problem were analyzed under specific conditions such as prevalent in large-scale order picking systems ([52]), in pick-and-pass systems ([53]), or under utilization of cluster-analysis-based batching methods ([54]). There are several publications focusing on combinations of operational strategies such as batching and routing ([16,55]) or storage and routing ([56,57]). The interaction between strategic and operational decision making is under consideration in a simulation-based analysis for algorithm combinations taking into account warehouse design, storage, and routing decisions ([58]), or zoning, batching, storage, and routing decisions ([59–62]).

Similarly to the idea of simulation metamodeling, performance measures of order picking systems are estimated based on queuing theory assumptions in [63–65]. However, for these analytical models very strict assumptions have to hold such as statistical assumptions about item arrivals, order quantities, or service times in different pick zones as well as special warehouse layouts. To the best of our knowledge, there is no research paper available on the use of ANN-based simulation metamodels for order picking systems.

3. ANN-based metamodeling for simulation models with online decision making

The need for quick responses on expected simulation performance without exploiting too much computational resources is further increased in the presence of real time optimization, i.e., in case of a coupling between simulation and online optimization. This makes it even more necessary to devise simulation metamodels. However, the question remains whether such metamodels are capable of predicting simulation outcomes that are influenced by explicit decision making routines as prescribed by online algorithms. Also, optimization through simulation metamodeling has become increasingly popular due to increasing computing capacities. Questions that have to be solved concern the choice of the metamodel, the search design for finding promising parameters, the conduct of the experiments, and the assessment of the modeling adequacy ([1]). In this paper, we consider ANNs as simulation metamodels. Like every other simulation metamodel, also ANNs will only provide an estimate for

$$\min_{\theta \in \Theta} \mathbb{E}_I(Y(i, \theta))$$

resulting from an instance sample set I_{sample} used to provide an estimate for $\mathbb{E}_I(Y(i, \theta)$ upon a parameter sample set Θ_{sample}. Thus, from our sampling procedure we get for $\theta_{sample} \in \Theta_{sample}$ with N being the number of samples using θ_{sample} the value of $\frac{1}{N}\sum_{k=1}^{N} Y(i_{sample}, \theta_{sample})$ as an estimator for $\mathbb{E}_I(Y(i, \theta_{sample}))$.

3.1. General outline of ANNs

The core building block of an ANN is a neuron. As shown in Fig. 2, a neuron receives (real-valued) inputs $x_1, x_2, \ldots, x_n$ weighted with factors $w_1, w_2, \ldots, w_n$, respectively. The sum of the weighted inputs $\sum_{i=1}^{n} w_i x_i$ then produces the output $f(\sum_{i=1}^{n} w_i x_i)$ of the neuron

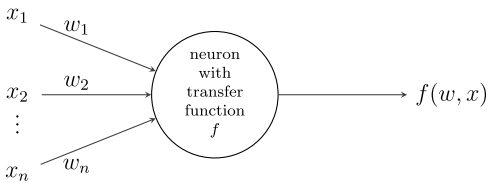

Fig. 2. Representation of a neuron (cf. [31]).

according to the chosen transfer function f. The output can be interpreted as the transmitted signal strength of the neuron. With $\Sigma := \sum_{i=1}^{n} w_i x_i$, frequently used transfer functions f are sigmoidal functions $f(\Sigma) := \frac{1}{1+e^{-\Sigma}}$, threshold functions $f(\Sigma) := \mathbf{1}_{[v,\ \infty)}(\Sigma)$, rectifier functions $f(\Sigma) := \max\{0, \Sigma\}$, or radial basis functions $f(x_1,...,x_n) = \exp(-\frac{\sum_{i=1}^{n}(x_i-\mu_i)^2}{2\sigma^2})$.

Neurons are organized in layers. Every ANN exhibits one input and one output layer with hidden layers between them. Only neurons of different layers can be connected to each other. When connections only occur between successive levels in a forward-oriented manner, we speak of a feedforward network. Backedges (recurrent networks) or arbitrary connections between arbitrary layers are also possible. The most frequently used form for function approximation is the feedforward type as shown in Fig. 3 with one or at most two hidden layers. Input neurons have no ingoing connections, but produce an output (which serves as an input for the ANN); output neurons have no outgoing connections, but transform the received input into an output (which serves as an output of the ANN).

ANNs can be cast as a network of primitive functions ([31]). Essentially, a feedforward network is a concatenation of neuron transfer functions with each transfer function receiving as input a linear combination of outputs from the previous layer; the output can then be computed easily by forward propagation. To this end, let $x_1, x_2, ...,x_n$ be the inputs to the ANN. Then, successively for each hidden layer the input values of the neurons and the edge weights are computationally processed with the transfer function of the neuron, giving the output of that hidden layer. Once, the output layer is reached, the observed input values of the previous last hidden layer and the edge weights are observed and translated into the final outputs $y_1, y_2, ...,y_m$ through the transfer functions of the output neurons.

For observed simulation outputs $\hat{y}_1, \hat{y}_2, ...,\hat{y}_m$ which are to be approximated, the question is how well the outputs of the ANN $y_1, y_2, ...,y_m$ approximate them. To this end, the most frequently used performance measure for the approximation quality of the ANN is the mean squared error (MSE):

$$MSE(y, \hat{y}) = \frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j).$$

Feedforward ANNs can be used to approximate functions. The back error propagation algorithm ([66]) is used as the algorithm of choice in order to instill learning on the network. Mechanistically, learning is achieved through minimizing (in a gradient-descent fashion) the MSE between predicted outputs y and observed outputs $\hat{y}$ in a sequential manner. In the context of ANNs, learning means that the parameters (weights) of the ANN are adapted. From now on, we denote the output of neuron j by o_j and it is given as $o_j = f(\sum_{i=1}^{n} w_{ij} o_i)$. In particular, for output neuron j we have $y_j := o_j = \sum_{i=1}^{n} w_{ij} o_i$ and for input neuron j we have $o_j := x_j$. Therefore, we can substitute y_j in $MSE(y, \hat{y})$. Since $\hat{y}$ are constant values, each y_j contributes $\sum_{i=1}^{n} w_{ij} o_i$ to $MSE(y, \hat{y})$. For each o_i, we can recursively insert $o_i = f(\sum_{k=1}^{n} w_{ki} o_k)$. From this concatenation and the bounding condition for the jth input neuron $o_j := x_j$, $MSE(y, \hat{y})$ ultimately contains products of weights and neuron input values. To decrease $MSE(y, \hat{y})$ we perform a gradient descent with respect to w, i.e.,

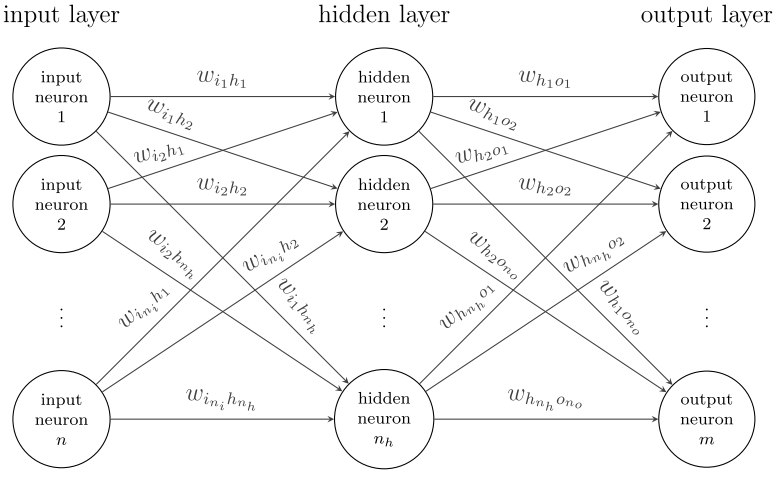

Fig. 3. Representation of a feedforward ANN with one hidden layer.

we compute $\frac{\partial MSE(y,\hat{y})}{\partial w_{ij}}$ and perform a step into the steepest descent direction which is given by updating w_{ij}: $=w_{ij} - \Delta w_{ij} = w_{ij} - \nu \frac{\partial MSE(y,\hat{y})}{\partial w_{ij}}$ with a so-called learning weight ν. In general, it can be derived that

$$\delta_j = \begin{cases} \frac{\partial MSE(y,\hat{y})}{\partial f(o_j)} \cdot \frac{df(o_j)}{do_j} & \text{if } j \text{ is an output neuron} \\ \sum_{l=1}^{n} w_{jl} \cdot \delta_l \frac{df(o_j)}{do_j} & \text{else} \end{cases} \tag{1}$$

and $\frac{\partial MSE(y,\hat{y})}{\partial w(ij)} = o_i \cdot \delta_j$. For the special case of sigmoidal transfer functions, Eq. (1) further resolves as

$$\delta_j = \begin{cases} (y_j - \hat{y}_j) \cdot o_j \cdot (1 - o_j) & \text{if } j \text{ is an output neuron} \\ \sum_{l=1}^{n} w_{jl} \cdot o_j (1 - o_j) & \text{else} \end{cases} \tag{2}$$

The procedure of sequentially updating w_{ij} in a backward manner starting with the weights of connections entering the output neurons then yields the back error propagation algorithm. From a historical point of view, it deserves mentioning that the back-propagation algorithm strongly resembles the Gauss-Newton method and also the steepest gradient method from optimization theory. Moreover, the algorithm was already used in a slightly modified form in the field of optimal control in the 1960s. Finally, we would like to remark that from practical considerations and also from utilizing MSE as a performance measure it makes sense to use normalized or standardize input values from the interval $[-1, 1]$ or $[0, 1]$, respectively, because many transfer functions typically return values in $[-1, 1]$ or $[0, 1]$ as well.

3.2. Methodology for ANN-based simulation metamodeling with online decision making

We now consider the task of deploying an ANN for the sake of using it as a simulation metamodel in the special setting of online optimization being integrated into the simulation. To this end, we look at the conceptual steps needed to establish the validity of an ANN-based simulation metamodel. We follow the guidelines in [41] and extend them by an explicit consideration of the real time decision making character that results from the integration of operational control strategies.

3.2.1. Simulation model development

As an extension to classical simulation models where often priority rules or random mechanisms are deployed to model agent behavior, we explicitly allow for the integration of online decision making routines that can be deployed by the agents whenever a decision is required. Hence, besides numerical parameters for the system configuration, control strategies (online algorithm) candidates represent nominal parameters to be varied in a simulation optimization, and a modular integration of such online algorithms into the simulation model has to be facilitated. Related to ANNs, numerical parameters and control strategy candidates will represent the inputs $x_1, \ldots, x_n$. As simulation lends itself to multicriteria analysis, simulation output quantities (responses) $\hat{y}_1, \ldots, \hat{y}_n$ will correspond to outputs $y_1, \ldots, y_n$ of the ANN. Since the goal of ANN-based simulation metamodeling consists of the estimation of expected values of simulation outputs, for each simulation configuration (selected parameter values and operational control strategies) several replications have to be executed in order to smooth out randomness of single instances. In this sense, $\hat{y}$ will hold the expectation values of the simulation outputs after N replications, i.e., $\hat{y}_j$: $=\frac{1}{N}\sum_{k=1}^{N} \hat{y}_j^k$ where $\hat{y}_j^k$ is the output of the simulation model for the jth performance measure in the kth simulation replication.

Running times of online algorithms have to be considered because running the simulation models for obtaining the training and testing data of the ANN is a necessary step in ANN-based simulation metamodeling. Since runtime is incurred each time that a decision needs to be made, online algorithms will be executed frequently. Hence, both in the simulation model and in the real world setting, the requirement has to be met that not too much computing time is elicited. For heuristic approaches to solving the snapshot problems this requirement is typically met. When one tries to determine exact solutions, care has to be taken in case of $\mathcal{NP}$-hard snapshot problems that instances do not get too large.

3.2.2. Data preparation / sampling

To generate training and testing data, we have to sample the decision space consisting of system parameters and operational control strategies. As the goal of simulation metamodeling also consists of determining suitable operational control strategies, we observe that all control strategy candidates need to be considered in the numerical experiments, otherwise it is impossible to infer any information on simulation behavior under an unknown control strategy. Hence, in contrast to numerical parameters not only a subset of all potential values can be considered, but all online algorithms actually have to be represented in the data. In a second step, we determine for the numerical parameters which sampling strategy is used, e.g., random sampling, Monte Carlo sampling, Latin hypercube sampling, or space-filling sampling. Having obtained all numerical samples, these are combined with the control strategies. In doing so, we have to take care that every control strategy is combined with every numerical sample such that the evaluation basis is the same for each control strategy.

3.2.3. ANN development

ANN representation and architecture are determined by the number of hidden layers, the number of neurons at each hidden layer, and the inputs and outputs. There are some references pointing at how to select some of these quantities. First, it is known from the

universal approximation theorem that a feedforward layer with just one hidden layer is a reasonable choice when it comes to achievable approximation qualities ([29,67,68]). Moreover, it is explicitly remarked that choosing too many neurons on the hidden layers leads to overfitting because there are too many adaptable degrees of freedom (weights) which make every piece of training data to be perfectly reconstructed by the ANN, while at the same time it will fail for new test data ([41]). Hence, the ANN may be well "trained", but it has not been "educated" which should be the overarching goal of training. Apart from these recommendations careful experimenting/testing with concrete values is suggested to obtain further insight into ANN behavior as required for a specific simulation model. Additionally, we have that operational control strategies will serve as the input for the simulation model and hence also for the ANN. However, since algorithms can only be represented as nominal variables, we have to create one input neuron for each possible realization of the operational control strategy with an input value of 1 if this control strategy is chosen and 0 otherwise. Since the number of input and output neurons is fixed by the user specification and availability of candidates for control strategies, the number of input and output neurons is fixed. For the number of hidden layers and hidden neurons, we refer to the rules of thumb given in [69–71]. These point all towards the rule that overfitting shall be heavily avoided through choosing a number of hidden neurons which is not too large, especially in comparison with the number of input neurons and output neurons, respectively.

3.2.4. Models' training and testing for system selection

Once the training and testing data is available, this data can be used in two intermittent phases: Training data is used to help the ANN model find its best configuration with respect to the connection weights and the network layout. Testing data can then be used to assess whether the approximation quality of such a specific ANN setting is also retained for data different from the training inputs, namely for the testing data. In case that to many data points lead to a high degree of adaptation, but poor performance in case of new data points, we speak of overfitting. To avoid this and to evaluate the quality of the ANN-based metamodel, validation measures such as the mean squared error, coefficient of determination, maximal absolute error have to be employed. A feedback with the data module determines whether an adequate trade off between time efficiency and approximation quality is striked. ANNs are especially appreciated in systems where no explicit causal knowledge between input and output factors is available as they allow also under these circumstances to provide surrogate models based on training data only.

The unknowns to be determined consist of the ANN layout decisions and connection weights. The latter values are obtained through executing the back error propagation algorithm for each ANN candidate and the available training data. Therefore, each ANN is optimized as much as possible bringing them in a comparable state to determine which network configuration is most advisable with respect to the achievable approximation quality on the testing data then. The goal of this step is to find the most promising network candidates including a specification of the network structure (layers, neurons, weights, transfer functions). In particular, weights at the input neurons for the operational control strategies provide a first pointer towards the influence of the control strategies on output values.

3.2.5. ANN validation / evaluation

In order to decide whether the ANN is a sufficiently good fit for the simulation model, we use simulation data which has not been utilized before in the models' training and testing for system selection. In contrast to the training and testing phase, no more adaptation of connection weights or network layout is carried out. Rather, the obtained ANN candidates from the previous step are used and it is checked whether similar accuracy performance measures are achieved on the new data. Hence, as the final validation step, we obtain performance measures for the approximation quality of the ANN candidates.

Recalling that each simulation configuration (selected parameter values and operational control strategies) was replicated several times in order to obtain expectation results, also the ANN shall return in each output neuron the average of a performance measure that would result from a large number of simulation replications. However, since training and testing data (output values for the single performance measures) are obtained replication by replication, one can translate this gradual data retrieval into the research question of how many replications N are needed for training ANNs to a sufficiently high accuracy. We do so by recurring to the index N which gives us the performance measures after N replications. Therefore, after N simulation replications with the same simulation setting, the observed expected output $\hat{\bar{y}}_j(N) = \frac{1}{N}\sum_{k=1}^{N}\hat{y}_j^k$ where $\hat{y}_j^k$ is the output of the simulation model for the jth performance measure in the kth simulation replication.

3.2.6. Selection of simulation parameters and operational control strategies

In case of a positive validation of an ANN as a simulation metamodel, we can employ this model to evaluate the response of the ANN for different parameters and operational control strategies. Hence, instead of applying a simulation optimization, the ANN metamodel allows to employ a simulation metamodel optimization. Despite the original purpose of simulation metamodels which is to facilitate an easy evaluation of different parameter settings and control strategies, there is no automated way of figuring out "optimal" settings by an overarching optimization algorithm in a multicriteria simulation environment. Rather, comparing (in the sense of ranking and selection) will enable the decision maker in a multi-criteria setting to decide according to his or her preferences which parameter settings are most desirable.

In conclusion, ANNs are not only used as (time-saving) metamodels for estimating average output values of simulation models configured with different parameter settings and operational control strategies, but also for discerning good control strategies from bad ones, and for identifying mutual effects between parameter settings and operational control strategies in the form of a response surface. As a side topic, the dependence between the number of simulation replications and attainable accuracy of ANNs can be considered.

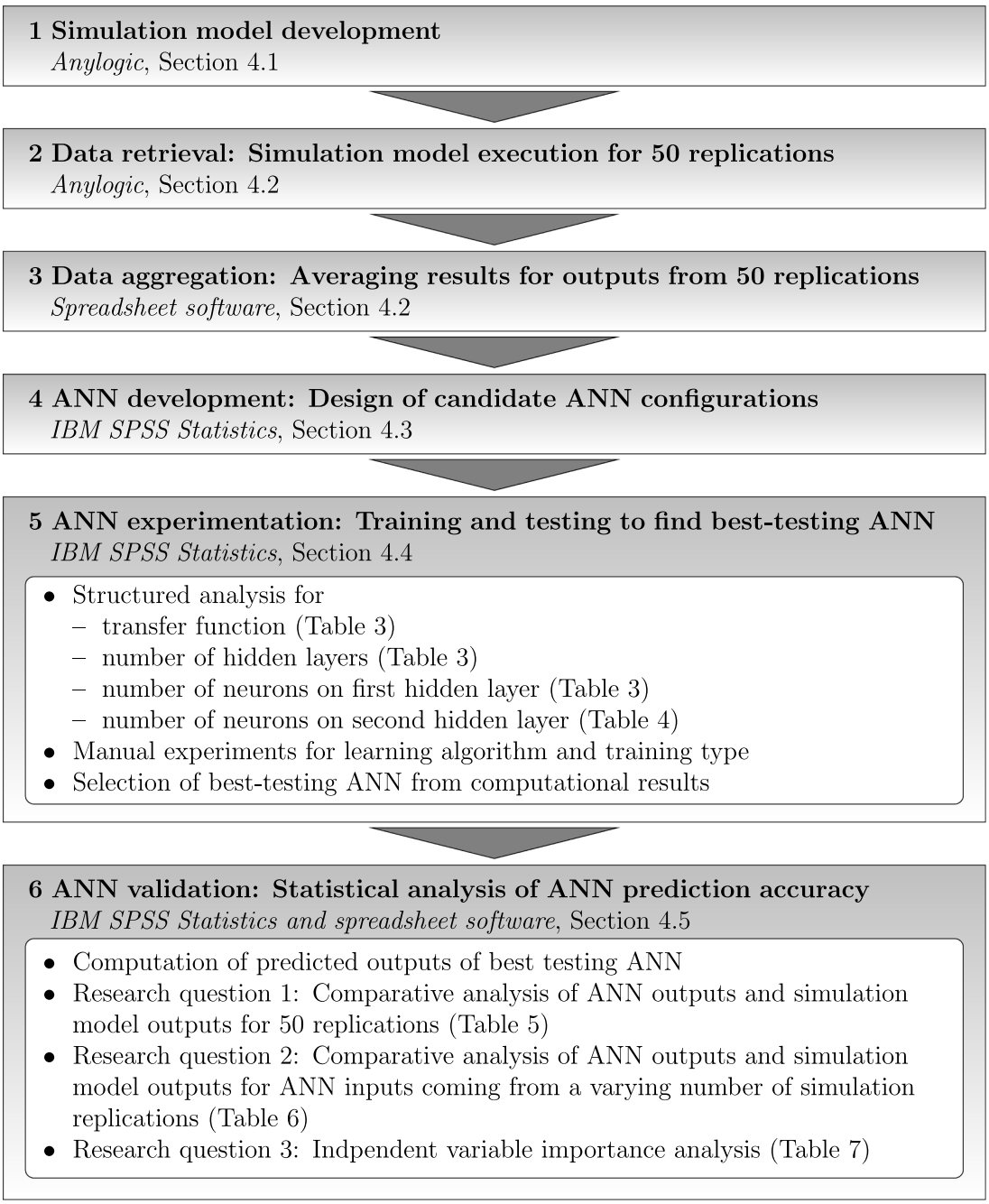

Fig. 4. Overview of methodological steps for the ANN-based simulation metamodeling of an order picking system.

4. Case study: metamodeling of an order picking system

To the best of our knowledge, no research is available on simulation metamodeling for order picking systems with online decision making. In particular, there is nothing known about ANN-based simulation metamodels in this context. Therefore, the following case study also provides practitioner's knowledge on order picking system operations. The goal of the case study encompasses the specification of an ANN as a simulation metamodel. According to the simulation optimization idea such a metamodel would support the search for recommendable parameter and control strategy settings. We facilitate an analysis by carrying out the guideline steps presented in Section 3. Fig. 4 provides an overview of the different elements of the analysis carried out over the following sections. All source files and data sets related to the case study are available online at http://dol.ior.kit.edu/english/downloads.php.

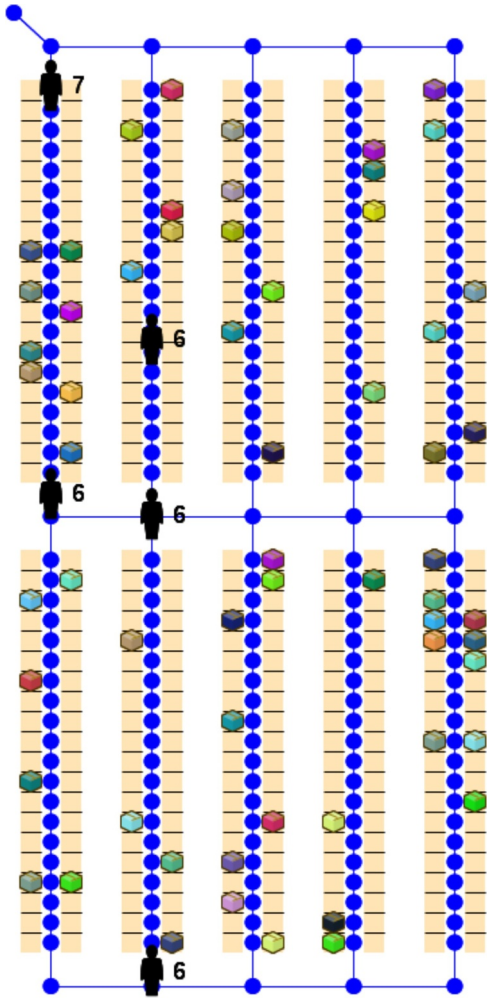

Fig. 5. Animation of the simulation model for an order picking system in *AnyLogic*.

4.1. Simulation model development

We first set up a simulation model with integrated operational control algorithms. Moreover, the simulation model shall be capable of being instantiated for different parameter settings reflecting logistics and warehouse geometry. Typical warehouse performance indicators such as overall makespan, overall walking distance of pickers, picker utilization, box throughput are obtained as outputs. Since we intend to develop a metamodel for a simulation model which will be exposed to randomness itself, the simulation metamodel shall return the expected performance over a given configuration of parameters and operational control strategies. Therefore, $N = 50$ simulation replications are executed. Each replication describes a different set of orders to be processed throughout the workday.

The simulation model is implemented as a discrete event model in the simulation software package *Anylogic* in version 8.1.0. Fig. 5 shows the order picking system as instantiated for two rows with five aisles in each row, five pickers with capacity of seven, and a number of orders available at the time of the screenshot. Table 1 summarizes parameters and control strategies that were varied in the simulation model. Implementation details for the routing algorithms (SShaped, LargestGap, Return, Optimal) and batching algorithms (Priority, Seed) are summarized in [15]. Observe that for the warehouse geometry there is only one value in the parameter specification. Since we believe that the logistical parameters related to capacities are enough to be considered in order to decide upon the suitability of ANN metamodels, we fixed the geometric parameters. However, the simulation model is capable to flexibly generate different warehouse layouts as specified by the warehouse geometry parameters (number of rows, number of aisles per row, number of cells per aisle, distance between rows, distance between aisles).

4.2. Data preparation / Sampling

The proprietary random influence on warehouse operations are the orders arriving over the day. Each order consists of several

Table 1
Parameters and outputs of the simulation model.

Parameters and specifications	
overall number of orders	100, 200, ..., 500
minimum number of boxes per order	1, 2, 3
maximum number of boxes per order	3, 4, 5
picker capacity	5, 7, 10
number of pickers	5, 7, 10
number of rows	2
number of aisles per row	5
number of cells per aisle	20
distance between rows	50
distance between aisles	75
routing algorithm	SShaped, LargestGap, Return, Optimal
batching algorithm	Priority, Seed
Outputs	
makespan	
total walking distance	
picker utilization	
box throughput	

boxes ranging between a minimum and a maximum number of boxes per order. Each order is further characterized by the box locations in the warehouse and its release time. For each order these values are generated according to a uniform distribution; e.g., for the release time a uniformly distributed value is drawn from the interval between workday beginning and (planned) workday end. (Observe that the realized workday end is always after the planned workday end as we only consider warehouses under high utilization.) With $N = 50$ seeds, input instances were created such that every possible parameter configuration is checked. Thus, we have an overall available data basis of $50 \cdot 5 \cdot 3 \cdot 3 \cdot 3 \cdot 3 \cdot 4 \cdot 2 = 162.000$ training samples. In total, this leads to averages of the output values for $\frac{162.000}{50} = 3.240$ different parameter and control strategy settings. These 3.240 data points were then partitioned into 50% training data, 25% testing data, and 25% validation data. This data will be used subsequently to establish ANN-based metamodels and to assess their approximation capabilities.

From a computational time perspective it was possible to execute the simulation model for all possible configurations. We ran a replication for each order sequence with each possible parameter and online algorithm combination. The computational time required amounted to a total of 52 hours and 17 minutes on a 64 GB RAM machine with Intel i7 CPU at 3.6 GHz. Since a metamodel needs only to be constructed once and can be re-utilized in various situations, we believe that this computing expense is worth to be invested in order to generate a rich data basis.

4.3. ANN development

For every variable system parameter and operational control strategy in Table 1, we provide an input neuron. Despite implementation of warehouse geometry with parameters for the aisle and rack layout, we decided for this analysis to consider only logistical parameters concerning the order structure and picker resources. This simplifies the presentation and already gives us an insight into the interplay between classical numerical parameters (such as a picker capacity) and categorical parameters (such as a batching algorithm). Hence, overall we have five input neurons for numerical parameters and two input neurons for the operational control strategies. However, observe that the six specifications for the algorithms are translated into six input neurons taking binary inputs. Nonetheless, most software tools for ANNs allow for a simpler representation with two neurons in the user interface. According to Table 1, the ANN is configured with four output neurons, one for each simulation output.

Further degrees of freedom comprise the transfer function type as well as the number of hidden layers and the number of neurons per hidden layer. We adhere to the rules of thumb given in [69–71] stating that one or two hidden layers are enough to produce sufficiently good approximations. Concerning the number of hidden neurons we initially experimented with 8, 10, 12, and 14 hidden neurons per layer. This is in line with the suggestion of not using too many hidden neurons. From our analysis it will be seen that good

Table 2
Configuration options of the ANN.

Input neurons	7
Output neurons	4
Type of transfer function	sigmoidal, hyperbolic tangent, radial basis
Number of hidden layers	1, 2
Number of neurons per hidden layer	8, 10, 12, 14
Type of learning algorithm	scaled conjugated gradient, gradient descent
Type of training	batch, online

results are obtained with the most popular transfer functions (sigmoidal and hyperbolic tangent transfer functions). Moreover, many ANN software packages offer slightly different learning algorithms and training procedure types. Table 2 summarizes the considered configuration options of the ANN.

4.4. Models' training and testing for system selection

ANNs are set up and optimized to approximate the response of the simulation model under different inputs concerning logistical parameters and control strategies. The overall goal consists of checking whether there is an ANN available to be used as a valid simulation metamodel. Configuration details of the ANNs examined in the order picking case study are summarized as follows and will be discussed subsequently in the search for the best testing ANN configuration:

Inputs (independent variables) number of orders, minimum number of boxes per order, maximum number of boxes per order, picker capacity, number of pickers, routing algorithm, batching algorithm.
Outputs (target variables) predictions for expected values of makespan, total walking distance, picker utilization, box throughput.
Data basis averages for each output obtained from 50 simulation replications for all combinations of input values
Loss function for ANN assessment sum-of-squares error
ANN configuration decisions type of transfer function, number of hidden layers, number of neurons on hidden layers, type of learning algorithm, type of training

Further information on the data basis are given in Section 4.2 on the simulation configurations used for data retrieval; further information on the specifications for inputs and outputs are provided in Section 4.3 on the ANN representation of the simulation model input-output structure. Computations are carried out in the neural network module of *IBM SPSS Statistics* in version 25. We used 50% of the available 3.240 data points for training. Training was succeeded by testing with another 25% of the data basis to prevent overfitting. The remaining 25% of the data basis were used as validation data.

Concerning the type of training and learning algorithm, we experimented manually with different ANN configurations. Results were rather clear such that there was no need for a systematic experimental outline on these decisions: In batch training, weights are adjusted only when all training data have been propagated; in online training, weights are adapted upon each processing of an input tuple. Both in batch and in online training, adjustments are made over several iterations over all data (epochs). In our experiments, there was no systematic difference observable between batch and online training, i.e., depending on the seed there were cases where batch training outperformed online training and vice versa. Hence, it is irrelevant which type of training is selected. In the sequel, we will consider networks trained with batch data. The same conclusion could be drawn from the manual experiments for the learning algorithm, namely that there is no dominance between classical gradient descent and scaled conjugated gradient. In the sequel, we will consider networks using the classical gradient descent. Radial basis transfer functions were eliminated due to poor approximation compared to sigmoidal and hyperbolic tangent transfer functions.

For the remaining decisions (transfer function / number of hidden layers / number of neurons per layer) a structured analysis was carried out to find the best testing network configuration. Conclusions were based on the relative total error of ANN predictions deviating from simulation averages from 50 replications. Observe that this measure is computed by *IBM SPSS Statistics* and no details are given on how the final value is composed. Therefore, in Section 4.5, we will compute relative total errors for each of the four simulation outputs when ANN results are re-transformed to actual output values to be predicted. Decisions on the transfer function type, the number of hidden layers, and the number of neurons are based on the training and testing results recorded in Table 3.

Results from ANNs with one hidden layer were outperformed by those with two hidden layers. We conclude that an optimized ANN shall exhibit two hidden layers with hyperbolic tangent transfer functions for the neurons on the hidden layers. Taking into account both the level of the relative total error and the deviations (in terms of stability over the number of hidden neurons on the second layer), the bold-typed entries show that it is recommendable to have 14 neurons on the first hidden layer. Next, to determine the number of neurons on the second hidden layer, we consider the behavior of the ANN over five different seeds as summarized in Table 4.

As can be seen from the bold-typed entries, the combination of the hyperbolic tangent transfer functions in an ANN with 14 and 10 nodes on the hidden layers, respectively, leads to the best and most stable results. Therefore, we conclude from the training and testing phase that this network configuration is most recommendable.

4.5. ANN validation / evaluation

With the remaining 25% of the available 3.240 data points, we carried out the final ANN validation and evaluation phase with the resulting best testing ANN from the previous step. The goal of this phase is to conclude whether this ANN can be used as a simulation metamodel in case of embedded online decision making and which size of a data basis would be required to do so.

4.5.1. Research question 1: Is it possible to estimate the expected output of a simulation model?

From the training and testing phase in Section 4.4 we have received the recommendable network configuration using data from 50 simulation replications. Using this best testing ANN configuration, we compute relative total errors specifically for each of the four simulation outputs (makespan, total walking distance, picker utilization, box throughput) when ANN results are re-transformed to actual output values to be predicted. Using the estimates, we can then provide output-specific relative errors as well as statistical

Table 3
Relative total error of ANNs for training, test, and validation data depending on ANN configuration.

Activation function	Number of hidden layers	Number of hidden neurons	Relative total error (training)	Relative total error (testing)	Relative total error (validation)
tanh	1	8	0.036	0.038	0.037
tanh	1	10	0.043	0.045	0.046
tanh	1	12	0.038	0.039	0.039
tanh	1	14	0.020	0.022	0.021
tanh	2	8, 8	0.032	0.035	0.035
tanh	2	8, 10	0.027	0.028	0.028
tanh	2	8, 12	0.019	0.020	0.020
tanh	2	8, 14	0.018	0.019	0.019
tanh	2	10, 8	0.015	0.016	0.016
tanh	2	10, 10	0.014	0.015	0.015
tanh	2	10, 12	0.032	0.035	0.037
tanh	2	10, 14	0.016	0.018	0.018
tanh	2	12, 8	0.013	0.015	0.013
tanh	2	12, 10	0.018	0.020	0.020
tanh	2	12, 12	0.014	0.016	0.015
tanh	2	12, 14	0.029	0.033	0.032
tanh	2	14, 8	**0.016**	**0.018**	**0.02**
tanh	2	14, 10	**0.011**	**0.013**	**0.013**
tanh	2	14, 12	**0.015**	**0.017**	**0.018**
tanh	2	14, 14	**0.013**	**0.015**	**0.014**
sig	1	8	0.063	0.065	0.067
sig	1	10	0.058	0.060	0.064
sig	1	12	0.050	0.052	0.057
sig	1	14	0.033	0.035	0.024
sig	2	8, 8	0.050	0.051	0.054
sig	2	8, 10	0.072	0.074	0.074
sig	2	8, 12	0.041	0.042	0.044
sig	2	8, 14	0.046	0.048	0.049
sig	2	10, 8	0.039	0.041	0.042
sig	2	10, 10	0.038	0.040	0.042
sig	2	10, 12	0.048	0.049	0.05
sig	2	10, 14	0.04	0.041	0.042
sig	2	12, 8	0.034	0.035	0.036
sig	2	12, 10	0.040	0.042	0.041
sig	2	12, 12	0.018	0.019	0.020
sig	2	12, 14	0.040	0.040	0.044
sig	2	14, 8	0.027	0.029	0.031
sig	2	14, 10	0.050	0.052	0.054
sig	2	14, 12	0.039	0.041	0.042
sig	2	14, 14	0.036	0.037	0.038

information. Table 5 and Fig. 6 summarize the results that were obtained from the ANN as a simulation metamodel.

From average and median performance indicators, we would first derive that to a satisfactory level of no more than 5% misestimation, ANNs represent a valid approach to predict average behavior of the simulation model. Observed standard deviations would confirm this recommendation. However, the existence of outliers shows that there is a threat for considerable misestimation. Therefore, we conclude that for an order picking system with online decision making it is not possible to provide precision guarantees when using ANN-based simulation metamodels for predicting average performance of simulation models involving operational control strategies. Since causal relations between input factors (including parameters and control strategies) are not revealed through the configuration of an ANN, we have to recognize that the interplay between the rationale of an online algorithm (operational control strategy) and logistical parameter settings cannot be anticipated with sufficient statistical security as would be required for ultimately substituting the simulation model with the ANN. Hence, combining operational control strategies and warehouse parameters incurs such a complex overall system behavior which makes it impossible to foresee the overall outcome of such a combination. In particular, outcomes of control strategies (such as determining optimal routes) strongly depend on specific instances and cannot be represented by a single number as is the case for conventional numerical parameters. Nonetheless, the ANN-based simulation metamodel can safely be used by the decision maker as a first step in order to obtain a glance at average and median quality of different control strategies. In particular, the knowledge gained can be used to eliminate weak strategies from further consideration.

4.5.2. Research question 2: To what extent do ANN predictions depend on the number of simulation replications contributing data for ANN training?

Expectation values of simulation outputs after n simulation replications are computed as $\hat{\bar{y}}_j^n := \frac{1}{n}\sum_{k=1}^{n}\hat{y}_j^k$ where $\hat{y}_j^k$ is the output of the simulation model for the jth simulation output in the kth simulation replication. Since we are already in possession of $\hat{\bar{y}}_j^{50}$, we can

Table 4
Relative total error of ANNs for training, test, and validation data depending on ANN configuration and seed values.

Activation function (hidden)	Number of hidden layers	Number of hidden neurons	seed	Relative total error (training)	Relative total error (testing)	Relative total error (validation)
tanh	2	14, 8	1	0.016	0.018	0.020
			2	0.015	0.018	0.018
			3	0.013	0.015	0.015
			4	0.012	0.016	0.015
			5	0.010	0.012	0.013
tanh	2	14, 10	1	**0.011**	**0.013**	**0.013**
			2	**0.017**	**0.018**	**0.020**
			3	**0.012**	**0.014**	**0.013**
			4	**0.010**	**0.012**	**0.014**
			5	**0.015**	**0.018**	**0.019**
tanh	2	14, 12	1	0.015	0.017	0.018
			2	0.011	0.012	0.013
			3	0.026	0.030	0.032
			4	0.013	0.014	0.014
			5	0.013	0.016	0.016
tanh	2	14, 14	1	0.013	0.015	0.014
			2	0.014	0.015	0.017
			3	0.011	0.014	0.013
			4	0.020	0.024	0.023
			5	0.012	0.015	0.015

Table 5
Information on relative errors in % for the respective performance criteria.

	Makespan	Distance	Utilization	Throughput
Average relative error	4.33	4.38	1.44	2.89
Median relative error	3.07	3.10	0.85	2.27
Lower quartile	1.40	1.44	0.35	1.06
Upper quartile	5.58	5.87	1.83	4.02
Standard deviation of relative error	4.86	4.73	1.85	2.59
Maximum relative error	51.99	108.72	24.31	28.45
Minimum relative error	0.00	0.00	0.00	0.00

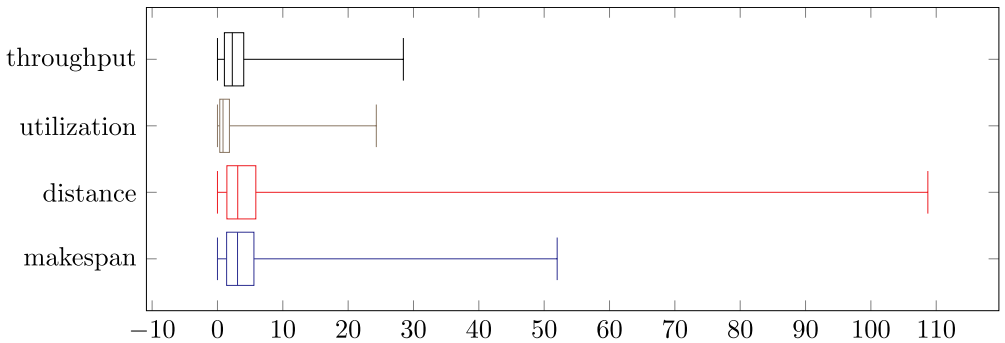

Fig. 6. Boxplots on relative errors in % for the respective performance criteria.

ask to which precision $\hat{y}_j^{50}$ can be estimated by an ANN-based simulation metamodel when only n simulation replications are used with $n < N = 50$. In other words, we ask for the degree of dispersion inherent to the simulation model and to the related ANN-based metamodel's capability of coping with it. Table 6 and Fig. 7 illustrate the approximation behavior of the ANN using simulation data from 1, 5, 10, 20, 30, 40, 50 replications.

As a result, we find that the essential characteristics of the approximation behavior of the ANN already become apparent from a rather moderate number of simulation replications such as 10–20. Therefore, no substantial additional statistical validity would be gained through further increasing the number of replications. The incapability of the ANNs to trim large misestimations is found consistently over all simulation outputs to be approximated.

Table 6
Information on relative errors in % for the respective performance criteria.

Number of replications		Makespan	Distance	Utilization	Throughput
1	average relative error	7.77	7.45	2.37	4.97
	median relative error	5.53	5.61	1.23	4.05
	lower quartile	2.51	2.58	0.53	1.89
	upper quartile	9.98	9.83	2.81	7.00
	standard deviation of relative error	8.34	7.32	3.05	4.13
	maximum relative error	80.16	67.04	21.30	32.34
	minimum relative error	0.00	0.00	0.00	0.00
5	average relative error	5.10	5.67	1.86	3.66
	median relative error	3.97	4.05	1.00	2.87
	lower quartile	1.80	1.77	0.43	1.34
	upper quartile	6.98	7.40	2.27	5.10
	standard deviation of relative error	4.75	6.20	2.45	3.19
	maximum relative error	46.11	73.29	32.07	27.15
	minimum relative error	0.00	0.00	0.00	0.00
10	average relative error	4.99	5.84	1.78	3.52
	median relative error	3.80	4.12	0.93	2.76
	lower quartile	1.67	1.94	0.38	1.30
	upper quartile	6.85	7.34	2.05	4.87
	standard deviation of relative error	4.91	6.38	2.76	3.16
	maximum relative error	61.65	63.79	44.08	36.47
	minimum relative error	0.00	0.00	0.00	0.00
20	average relative error	4.39	4.41	1.62	3.08
	median relative error	3.15	3.27	0.95	2.44
	lower quartile	1.43	1.53	0.40	1.10
	upper quartile	5.57	5.79	1.94	4.32
	standard deviation of relative error	5.27	4.27	2.38	2.74
	maximum relative error	70.71	41.10	40.08	37.91
	minimum relative error	0.00	0.00	0.00	0.00
30	average relative error	5.39	5.31	1.58	3.07
	median relative error	4.20	3.47	0.96	2.43
	lower quartile	1.81	1.60	0.43	1.11
	upper quartile	7.48	6.82	2.04	4.46
	standard deviation of relative error	5.08	6.35	1.99	2.53
	maximum relative error	54.27	78.85	28.07	18.09
	minimum relative error	0.00	0.00	0.00	0.00
40	average relative error	5.63	5.97	1.70	3.69
	median relative error	4.09	4.24	0.97	2.94
	lower quartile	1.93	1.91	0.42	1.38
	upper quartile	7.39	7.68	2.07	5.09
	standard deviation of relative error	6.09	6.24	2.36	3.19
	maximum relative error	83.23	55.22	32.07	23.88
	minimum relative error	0.00	0.00	0.00	0.00
50	average relative error	4.33	4.38	1.44	2.89
	median relative error	3.07	3.10	0.85	2.27
	lower quartile	1.40	1.44	0.35	1.06
	upper quartile	5.58	5.88	1.83	4.02
	standard deviation of relative error	4.86	4.73	1.85	2.59
	maximum relative error	51.99	108.72	24.31	28.45
	minimum relative error	0.00	0.00	0.00	0.00

4.5.3. Research question 3: According to the ANN-based simulation metamodel, what are the importances of the input factors for the simulation outputs?

Table 7 contains the information on relative input neuron importances as outputted by the neural network module of *IBM SPSS Statistics*. The input factor importance analysis reflects an ANN's sensitivity with respect to a change in the outputted prediction values upon changes in respective input factors. Input factor importances are given as relative values; normalization is obtained from dividing an input factor's importance by the largest importance among all input factors. Therefore, importance analysis allows to quantify the degree of influence of an input factor on the output and facilitates a comparison between input factors.

Using the estimations of the ANN metamodel, we can derive that – consistently over all seeds, i.e., irrespective of random influences – the most important factors for the attainable simulation output values are the number of orders, the number of pickers, and the routing algorithm. In particular, it becomes apparent that the routing algorithm plays a predominant role when compared to the batching algorithm.

4.6. Conclusions from case study

In the first research question, we wanted to know whether the approximation quality of an ANN is good enough. Since the goal of

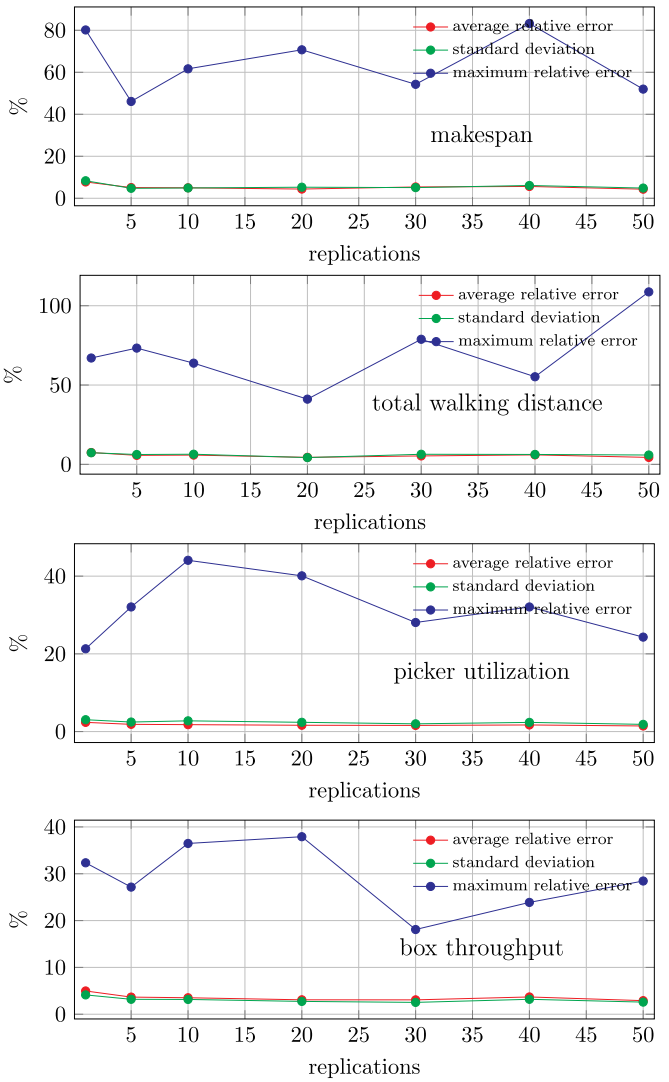

Fig. 7. Development of relative error related performance measures for makespan, total walking distance, picker utilization and box throughput depending on the number of replications.

Table 7
Information on importances of input neurons in ANN.

	1 seed	5 seeds	10 seeds	20 seeds	30 seeds	40 seeds	50 seeds
number of orders	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
min. number of boxes/order	53.8%	51.8%	51.4%	52.1%	51.9%	52.1%	52.0%
max. number of boxes/order	35.1%	36.5%	36.6%	37.5%	38.2%	37.0%	38.4%
picker capacity	68.5%	69.6%	69.2%	70.1%	70.5%	70.5%	70.9%
number of pickers	75.7%	75.6%	76.5%	77.4%	77.6%	77.5%	77.9%
routing algorithm	72.1%	73.6%	74.0%	74.3%	74.4%	74.8%	74.5%
batching algorithm	13.7%	13.9%	14.1%	14.1%	14.0%	13.7%	14.1%

a simulation metamodel encompasses a statistically secured selection of parameters and operational control strategies, we found that this cannot be guaranteed by an ANN-based simulation metamodel. Simulation results suggested that an optimal routing and a priority routing yield the most favorable results. This ordering is also reproduced by the ANN which is seen when input configuration specific results are sorted according to the respective outputs. In the vast majority of parameter settings, the ANN gives highly satisfactory estimations (in the range of +/– 5%) of the simulation outputs which would therefore allow for a rough first estimation of achievable performance indicators in specific input constellations. However, for definitively selecting a specific combination of parameters and control strategies, the risk – as can be seen by the heavy outliers – is too large. Therefore, we do not recommend ANN-

based simulation metamodels for detailed analysis in case of integrated online optimization, but suggest its use for providing a first, rough impression of expected simulation model behavior which could be used as a starting point for further analysis. With the second research question, we wanted to check whether there is hope that through a sufficiently large data basis prediction quality can be improved. However, from the computational results we can conclude that further substantial improvement is not only possible, but also not necessary already from ten replications on. Finally, the third research question has allowed us to identify which parameters or control strategies can be seen as key elements for successful logistics operations. In particular, this allows us to diagnose those input factors which should be clearly selected by some valid argumentation as these factors will influence the achievable performance sustainably.

5. Summary and outlook

Simulation metamodeling is a concept that has been around for quite some time. The approach we took in this paper is somewhat different in that we look at ANNs as metamodels for simulation models with integrated online decision making. Moreover, we do not intend to predict simulation output for a certain replication, but to predict the average output attainable over several replications. The logistics-related motivation behind this analysis is that we identify customer orders as a significant driver of random effects in many applications and that operational control strategies therefore are best evaluated over a large number of customer order realizations. To the best of our knowledge, an ANN-based metamodel for predicting average outputs over many simulation replications where each simulation run makes excessive use of an online control strategy has not been analyzed systematically before.

The paper first presents a discussion of the concepts required to create an ANN-based metamodel for a simulation model with integrated operational control strategies. To this end, the hierarchical concatenation between optimization and simulation is first discussed – showing that there must be a clear distinction between operational execution of optimization algorithms (online algorithms) and the overarching optimization searching for best possible operational control strategy and parameter configurations. Following the guidelines in [41], we then present the steps required in such an analysis when the focus is set on an analysis of the control strategies integrated in the simulation models. In a case study on an order picking system we checked whether this approach is successful also in practice. The results suggest that the overall quality of a control strategy can be predicted to a satisfactory level when considered over a large number of different parameter settings. Therefore, ANN-based simulation metamodeling allows for a first assessment of control strategy quality, guiding the search further for specific parameter settings. However, the presence of isolated outliers for the precision of the ANN-based metamodel also shows that ANN-based simulation metamodeling for simulation models with integrated online decision making comes with non-negligible risks in terms of achievable accuracy. Hence, an accompanying statistical analysis becomes indispensable in any application. We remark that when no input-output relation is demanded, but only a best possible parameter alternative from a small and fixed set is to be selected, statistical ranking and selection can be applied ([20]). As shown recently in a Bayesian setting, discrete distributions (such as simulation model output distributions) can be captured by formulating sampling and selection as a stochastic control problem ([72]).

Future research should take into account an analysis of factors which are crucial for the degree of precision that can be attained by ANN-based predictions of simulation outputs. From the case study, we know that different customer order structures lead to different ways of handling them by operational control strategies. This in turn is responsible for the unpredictability of the overall outcome. Therefore, it becomes necessary to identify under which conditions on the simulation model structure an acceptable and stable approximation quality of an ANN-based simulation metamodel can be achieved.

References

[1] L. Van Gelder, P. Das, H. Janssen, S. Roels, Comparative study of metamodelling techniques in building energy simulation: guidelines for practitioners, Simul. Modell. Pract. Theory 49 (2014) 245–257.
[2] B. Can, C. Heavey, A comparison of genetic programming and artificial neural networks in metamodeling of discrete-event simulation models, Comput. Operat. Res. 39 (2) (2012) 424–436.
[3] D. Fonseca, D. Navaresse, Artificial neural networks for job shop simulation, Adv. Eng. Inf. 16 (4) (2002) 241–246.
[4] W. Mouelhi-Chibani, H. Pierreval, Training a neural network to select dispatching rules in real time, Comput. Ind. Eng. 58 (2) (2010) 249–256.
[5] H. Pierreval, Neural network to select dynamic scheduling heuristics, J. Decis. Syst. 2 (2) (1993) 173–190.
[6] Online Algorithms: The State of the Art, in: A. Fiat, G. Woeginger (Eds.), Springer, 1998.
[7] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998.
[8] C. Bes, S. Sethi, Concepts of forecast and decision horizons: applications to dynamic stochastic optimization problems, Math. Operat. Res. 13 (2) (1988) 295–310.
[9] B. Fleischmann, H. Meyr, M. Wagner, Advanced Planning, in: H. Stadtler, C. Kilger, H. Meyr (Eds.), Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies, 5th ed., Springer, 2015, pp. 71–95.
[10] S. Sethi, G. Sorger, A theory of rolling horizon decision making, Ann. Oper. Res. 29 (1) (1991) 387–415.
[11] M. Grötschel, S. Krumke, J. Rambau, T. Winter, U. Zimmermann, Combinatorial online optimization in real time, in: M. Grötschel, S. Krumke, J. Rambau (Eds.), Online Optimization of Large Scale Systems, Springer, 2001, pp. 679–704.
[12] C. Cassandras, S. Lafortune, Introduction to discrete event systems, 2nd ed., Springer, 2010.
[13] F. Dunke, S. Nickel, A general modeling approach to online optimization with lookahead, Omega (Westport) 63 (2016) 134–153.
[14] P. Friese, J. Rambau, Online-optimization of multi-elevator transport systems with reoptimization algorithms based on set-partitioning models, Discrete Appl. Math. 154 (13) (2006) 1908–1931.
[15] S. Henn, S. Koch, G. Wäscher, Order batching in order picking warehouses: a survey of solution approaches, in: R. Manzini (Ed.), Warehousing in the Global Supply Chain, Springer, 2012, pp. 105–137.
[16] F. Dunke, S. Nickel, Evaluating the quality of online optimization algorithms by discrete event simulation, Central Eur. J. Operat. Res. 25 (4) (2017) 831–858.
[17] Verein Deutscher Ingenieure (VDI), VDI-Richtlinie 3633. Simulation Von Logistik-, Materialfluß- Und Produktionssystemen: Begriffsdefinitionen, VDI-Handbuch Materialflußund Fördertechnik, Beuth, 1996.
[18] M. Fu, Optimization via simulation: a review, Ann. Oper. Res. 53 (1) (1994) 199–247.

[19] J. April, F. Glover, J. Kelly, M. Laguna, Practical introduction to simulation optimization, Proceedings of the Winter Simulation Conference Proceedings 1 (2003) 71–78.
[20] S. Amaran, N. Sahinidis, B. Sharda, S. Bury, Simulation optimization: a review of algorithms and applications, Ann. Oper. Res. 240 (1) (2016) 351–380.
[21] S. Andradottir, An Overview of Simulation Optimization via Random Search, in: S. Henderson, B. Nelson (Eds.), Simulation, Handbooks in Operations Research and Management Science, 13 Elsevier, 2006, pp. 617–631.
[22] J. Xu, E. Huang, C.-H. Chen, L. Lee, Simulation optimization: a review and exploration in the new era of cloud computing and big data, Asia-Pacific J. Operat. Res. 32 (3) (2015).
[23] J. Boesel, J. Bowden, R.O., F. Glover, J. Kelly, E. Westwig, Future of simulation optimization, Proceedings of the Winter Simulation Conference Proceedings 2 (2001) 1466–1469.
[24] M. Fu, S. Andradottir, J. Carson, F. Glover, C. Harrell, Y. Ho, J. Kelly, S. Robinson, Integrating optimization and simulation: research and practice, Proceedings of the Winter Simulation Conference Proceedings 1 (2000) 610–616.
[25] M. Almodóvar, R. García-Ródenas, On-line reschedule optimization for passenger railways in case of emergencies, Comput. Operat. Res. 40 (3) (2013) 725–736.
[26] R. Dekker, P. Voogd, E. Van Asperen, Advanced methods for container stacking, OR Spect. 28 (4) (2006) 563–586.
[27] F. Dunke, S. Nickel, Day-ahead and online decision-making for collaborative on-site logistics, J. Simulat. (2018) 1–14.
[28] M. Laguna, R. Martí, Neural networks prediction in a system for optimizing simulations, IIE Trans. (Inst. Ind. Eng.) 34 (3) (2002) 273–282.
[29] R. Barton, M. Meckesheimer, Metamodel-based simulation optimization, Handb. Operat. Res. Manag. Sci. 13 (C) (2006) 535–574.
[30] I. Sabuncuoglu, S. Touhami, Simulation metamodelling with neural networks: an experimental investigation, Int. J. Prod. Res. 40 (11) (2002) 2483–2505.
[31] R. Rojas, Neural Networks: A Systematic Introduction, Springer, 1996.
[32] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.
[33] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[34] A. Negahban, J. Smith, Simulation for manufacturing system design and operation: literature review and analysis, J. Manuf. Syst. 33 (2) (2014) 241–261.
[35] B. Can, C. Heavey, Comparison of experimental designs for simulation-based symbolic regression of manufacturing systems, Comput. Ind. Eng. 61 (3) (2011) 447–462.
[36] Y. Kuo, T. Yang, B. Peters, I. Chang, Simulation metamodel development using uniform design and neural networks for automated material handling systems in semiconductor wafer fabrication, Simul. Modell. Pract. Theory 15 (8) (2007) 1002–1015.
[37] M. Chambers, C. Mount-Campbell, Process optimization via neural network metamodeling, Int. J. Prod. Econ. 79 (2) (2002) 93–100. Theoretical Approaches and Decision Support
[38] F. Altiparmak, D. Dengiz, A. Bulgak, Buffer allocation and performance modeling in asynchronous assembly system operations: an artificial neural network metamodeling approach, Appl. Soft Comput. 7 (3) (2007) 946–956.
[39] G. Yildiz, O. Eski, An artificial neural network based simulation metamodeling approach for dual resource constrained assembly line, Proceedings of the 16th International Conference on Artificial Neural Networks - Volume Part II, ICANN'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 1002–1011.
[40] F. Yang, Neural network metamodeling for cycle time-throughput profiles in manufacturing, Eur. J. Oper. Res. 205 (1) (2010) 172–185.
[41] D. Fonseca, D. Navaresse, G. Moynihan, Simulation metamodeling through artificial neural networks, Eng. Appl. Artif. Intell. 16 (3) (2003) 177–183.
[42] J. van den Berg, W. Zijm, Models for warehouse management: classification and examples, Int. J. Prod. Econ. 59 (1) (1999) 519–528.
[43] G. Ghiani, G. Laporte, R. Musmanno, Introduction to Logistics Systems Planning and Control, Wiley, 2004.
[44] K. Roodbergen, G. Sharp, I. Vis, Designing the layout structure of manual order picking areas in warehouses, IIE Trans. 40 (11) (2008) 1032–1045.
[45] R. de Koster, T. Le-Duc, K. Roodbergen, Design and control of warehouse order picking: a literature review, Eur. J. Oper. Res. 182 (2) (2007) 481–501.
[46] K. Roodbergen, I. Vis, A model for warehouse layout, IIE Trans. (Inst. Ind. Eng.) 38 (10) (2006) 799–811.
[47] F. Chen, G. Xu, Y. Wei, Heuristic routing methods in multiple-block warehouses with ultra-narrow aisles and access restriction, Int. J. Prod. Res. 57 (1) (2019) 228–249.
[48] K. Roodbergen, R. de Koster, Routing order pickers in a warehouse with a middle aisle, Eur. J. Oper. Res. 133 (1) (2001) 32–43.
[49] R. Elbert, T. Franzke, C. Glock, E. Grosse, The effects of human behavior on the efficiency of routing policies in order picking: the case of route deviations, Comput. Ind. Eng. 111 (2017) 537–551.
[50] W. Lu, D. McFarlane, V. Giannikas, Q. Zhang, An algorithm for dynamic order-picking in warehouse operations, Eur. J. Oper. Res. 248 (1) (2016) 107–122.
[51] H. Hwang, Y. Oh, Y. Lee, An evaluation of routing policies for order-picking operations in low-level picker-to-part system, Int. J. Prod. Res. 42 (18) (2004) 3873–3889.
[52] S. Hong, A. Johnson, B. Peters, Large-scale order batching in parallel-aisle picking systems, IIE Trans. (Inst. Ind. Eng.) 44 (2) (2012) 88–106.
[53] J. Pan, P.-H. Shih, M.-H. Wu, Order batching in a pick-and-pass warehousing system with group genetic algorithm, Omega (United Kingdom) 57 (2015) 238–248.
[54] H. Hwang, W. Baek, M.-K. Lee, Clustering algorithms for order picking in an automated storage and retrieval system, Int. J. Prod. Res. 26 (2) (1988) 189–201.
[55] C.-Y. Tsai, J. Liou, T.-M. Huang, Using a multiple-ga method to solve the batch picking problem: considering travel distance and order due time, Int. J. Prod. Res. 46 (22) (2008) 6533–6555.
[56] M. Klodawski, R. Jachimowski, I. Jacyna-Golda, M. Izdebski, Simulation analysis of order picking efficiency with congestion situations, Int. J. Simulat. Model. 17 (3) (2018) 431–443.
[57] T. Franzke, E. Grosse, C. Glock, R. Elbert, An investigation of the effects of storage assignment and picker routing on the occurrence of picker blocking in manual picker-to-parts warehouses, Int. J. Logist. Manag. 28 (3) (2017) 841–863.
[58] K. Roodbergen, I. Vis, G. Taylor, Simultaneous determination of warehouse layout and control policies, Int. J. Prod. Res. 53 (11) (2015) 3306–3326.
[59] K. Choy, N. Sheng, H. Lam, I. Lai, K. Chow, G. Ho, Assess the effects of different operations policies on warehousing reliability, Int. J. Prod. Res. 52 (3) (2014) 662–678.
[60] G. Dukic, C. Oluic, Order-picking methods: improving order-picking efficiency, Int. J. Logist. Syst. Manag. 3 (4) (2007) 451–460.
[61] C. Petersen, An evaluation of order picking policies for mail order companies, Product. Operat. Manag. 9 (4) (2000) 319–335.
[62] C. Petersen, G. Aase, A comparison of picking, storage, and routing policies in manual order picking, Int. J. Prod. Econ. 92 (1) (2004) 11–19.
[63] T. Le-Duc, R. de Koster, Travel time estimation and order batching in a 2-block warehouse, Eur. J. Oper. Res. 176 (1) (2007) 374–388.
[64] L. Tang, E.-P. Chew, Order picking systems: batching and storage assignment strategies, Comput. Ind. Eng. 33 (3–4) (1997) 817–820.
[65] M. Yu, R. de Koster, The impact of order batching and picking area zoning on order picking system performance, Eur. J. Oper. Res. 198 (2) (2009) 480–490.
[66] P. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.
[67] K.-I. Funahashi, On the approximate realization of continuous mappings by neural networks, Neural Netw. 2 (3) (1989) 183–192.
[68] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[69] M. Berry, G. Linoff, Data mining techniques: For marketing, sales, and customer relationship management, 2nd ed., Wiley, Wiley, 2004.
[70] A. Blum, Neural networks in c++: An object-oriented framework for building connectionist systems, Wiley, Wiley, 1992.
[71] K. Swingler, Applying Neural Networks : A Practical Guide, Academic Press, 1996.
[72] Y. Peng, E.K.P. Chong, C. Chen, M.C. Fu, Ranking and selection as stochastic control, IEEE Trans. Automat. Control 63 (8) (2018) 2359–2373.