# Applying machine learning for the anticipation of complex nesting solutions in hierarchical production planning

**Christian Gahm, Aykut Uzunoglu, Stefan Wahl, Chantal Ganschinietz, Axel Tuma**

**Angaben zur Veröffentlichung / Publication details:**

Gahm, Christian, Aykut Uzunoglu, Stefan Wahl, Chantal Ganschinietz, and Axel Tuma. 2022. "Applying machine learning for the anticipation of complex nesting solutions in hierarchical production planning." *European Journal of Operational Research* 296 (3): 819–36. https://doi.org/10.1016/j.ejor.2021.04.006.

# Applying machine learning for the anticipation of complex nesting solutions in hierarchical production planning

Christian Gahm*, Aykut Uzunoglu, Stefan Wahl, Chantal Ganschinietz, Axel Tuma

*Chair of Business Administration, Production & Supply Chain Management, University of Augsburg, 86159 Augsburg, Germany*

*Keywords:*
(R) machine learning
Hierarchical planning
Anticipation
Nesting
Irregular strip packing

A B S T R A C T

In hierarchical production planning, the consideration of interdependencies between superior top-level decisions and subordinate base-level decisions is essential. In this respect, the anticipation of base-level reactions is highly recommended. In this paper, we consider an example from the metal-processing industry: a serial-batch scheduling problem constitutes the top-level problem and a complex nesting problem constitutes the base-level problem. The top-level scheduling decision includes a batching decision, i.e., the determination of a set of small items to be cut out of a large slide. Thus, to evaluate the feasibility of a batch, the base-level nesting problem must be solved. Because solving nesting problems is time consuming even when applying heuristics, it is troublesome to solve it multiple times during solving the top-level scheduling problem. Instead, we propose an approximative anticipation of base-level reactions by machine learning to approximate batch feasibility. To that, we present a prediction framework to identify the most promising machine learning method for the prediction (regression) task. For applying these methods, we propose new feature vectors describing the characteristics of complex nesting problem instances. For training, validation, and testing, we present a new instance generation procedure that uses a set of 6,000 convex, concave, and complex shapes to generate 88,200 nesting instances. The testing results show that an artificial neural network achieves the lowest expected loss (root mean squared error). Depending on further assumptions, we can report that the approximate anticipation based on machine learning predictions leads to an appropriate batch feasibility decision for 98.8% of the nesting instances.

## 1. Introduction

Almost every manufacturing company applying production planning and scheduling methods uses the concept of hierarchical production planning. This concept leads back to the works of Hax and Meal (1973) and Bitran, Haas and Hax (1981) (amongst others). Particularly the conceptual framework of Schneeweiß (1995) describes the hierarchical aspects of production planning systems. In this framework, the interdependencies between superior top-level decisions (defining the instructions or top-down influence) and the subordinate base-level decisions are explicitly emphasized. In addition, the anticipation of base-level reactions (bottom-up feedback) by top-level decisions is highly recommended. This anticipated reaction can be modelled by a so-called anticipation function "calculating" hypothetical reactions of the base-level founded on hypothetical instructions of the top-level.

In this paper, we investigate a hierarchical production planning example from the metal-processing industry: a serial-batch scheduling problem constitutes the top-level decision problem and a complex nesting problem (CNP; also known as two-dimensional, highly irregular strip packing problem; cf., e.g., Leao, Toledo, Oliveira, Carravilla & Alvarez-Valdés, 2020) constitutes the base-level decision problem. Thereby, the top-level scheduling decision comprises a batching decision (i.e., the determination of a set of items to be cut out of one metal slide) and a scheduling decision (i.e., the allocation of batches to machines and their sequencing). Important regarding the batching decision is that the base-level nesting problem must be solved to evaluate the feasibility of batches. Because solving a CNP is time consuming even when applying heuristics, it is troublesome to solve it multiple times during solving the top-level scheduling problem. Instead, we propose to use machine learning for the anticipation of base-level reactions (particularly batch feasibility) instead of iteratively solving CNPs. Without losing the ambition to propose machine

* Corresponding author.
*E-mail address:* christian.gahm@wiwi.uni-augsburg.de (C. Gahm).

learning as a general anticipation method, the application context of CNPs is motivated by several aspects:

– First, CNPs, a special kind of packing/cutting problems, occur in many industrial sectors like metal-processing, carbon fiber, or textile manufacturing and therefore, make our approach relevant in many different industrial application fields (Burke, Kendall & Whitwell, 2009).
– Second, nesting problems, or packing/cutting problems in general, very often constitute a subordinated problem of a superior decision problem (e.g., a machine scheduling problem) or are at least closely related to it (cf., e.g., Chryssolouris, Papakostas & Mourtzis, 2000 or Helo, Phuong & Hao, 2019).
– Third, solving CNPs causes high computation times, even if heuristics are used (note, that solving CNPs within an iterative solution method for the superior problem would even be troublesome if computation times are only a few seconds; cf., e.g., Mundim, Andretta, Carravilla & Oliveira, 2018).

Therefore, a most accurate and also efficient anticipation method for complex nesting solutions would be valuable to improve the solution quality and/or reduce computation times for many superior decision problems in a broad range of applications and industries.

Our main contribution to literature is the new anticipation approach, i.e., applying machine learning for the anticipation of complex nesting solutions, in hierarchical production planning. In contrast to Rohde (2004) who only uses artificial neural networks to anticipate lot-size inventories and setup times in the context of master production scheduling, we recommend to use a broad range of machine learning techniques to improve anticipation accuracy. To that, we propose a prediction framework to identify the most suitable technique. Further contributions to literature are the new instance generation procedure for CNPs (based on real-world shapes and controllable attributes) and the feature vectors to model CNP characteristics used by the machine learning techniques. The complete data set used in this paper is available at Mendeley data (Gahm, 2020).

The structure of this paper is as follows: After the detailed description of the investigated decision problem in Section 2, we discuss the related work of our anticipation approach in Section 3. The anticipation method itself is described in Section 4, followed by the prediction framework used for the anticipation (Section 5). Section 6 is dedicated to the evaluation of the applied machine learning techniques and the proposed anticipation approach. The paper closes with a conclusion of our results and an outlook on further research topics in Section 7.

## 2. Problem description

An incisive example of the interdependencies between a top-level decision and a base-level decision in hierarchical production planning can be found in the metal-processing industry. Fig. 1 illustrates the serial-batch scheduling problem of the top-level and the complex nesting problem of the base-level.

In this industry, cutting operations performed by laser or water jet cutting machines are necessary to fabricate a wide variety of metal pieces (items) out of large base metal slides. Thereby, the items should be grouped into batches to avoid unnecessary machine setups and/or to increase resource efficiency by avoiding waste. All items grouped in one batch are then cut out from the same metal slide. Because cutting operations are performed sequentially, the processing time of a batch is equal to the sum of processing (cutting) times of all the items assigned to this batch ("serial batching"). The machine setups are required to extract completed slides, to insert new slides, and to adjust machine parameters. Because setup efforts depend on the base slide characteristics material type and thickness, sequence-dependent setup times must be considered. These base slide characteristics also define incompatible item families because items with different material type and/or thickness requirements cannot be cut out of the same base metal slide and therefore cannot be grouped in one batch. The metal items have arbitrary shapes and sizes and the base slides have a limited capacity corresponding to their size. Thus, the items that can be grouped within one batch are not only restricted by their family but additionally by the slide capacity. The batching decision (item to batch assignment) combined with the scheduling decision (batch to machine allocation – in the case of multiple cutting machines – and sequencing of batches) defines a serial-batch scheduling problem representing the top-level decision (for an overview on different kinds of batch scheduling problems see for instance Potts & Kovalyov, 2000).

To implement the final schedule and also to evaluate the feasibility of a batch (the items of a batch must not exceed the slide capacity and may not overlap), the items must undergo a packing task which specifies the spatial arrangement of the items' shape. This packing or nesting task represents the base-level decision problem. Actually, this base-level decision problem could be formulated as "two-dimensional bin packing problem" (or "two-dimensional finite bin packing problem"; cf., the typology of Wäscher, Haußner & Schumann, 2007). Instead, we consider the more general "two-dimensional strip packing problem" having one open (infinite) dimension (cf., e.g., Martello, Monaci & Vigo, 2003). For this problem, the planning objective is the minimization of the required strip height whereas the second dimension (width) is fixed. Then, to evaluate batch feasibility, the calculated height can be compared to any reference height, e.g., the height of standard metal slides or the height of smaller metal slides remaining from previous cutting processes (to increase resource efficiency). Regarding the cutting items, it must be considered that the customer specific items have arbitrary shapes that are typically irregular (also called non-regular; cf., Wäscher et al., 2007; for an overview of relevant application areas see e.g., Dowsland & Dowsland, 1995). Accordingly, the problem is called "irregular strip packing problem" or "nesting problem" (cf., Oliveira, Gomes & Ferreira, 2000; in textile manufacturing, the problem is called "marker-making problem", cf., e.g., Li & Milenkovic, 1995). Burke, Hellier, Kendall and Whitwell (2007) use the term "highly irregular" shapes if the item shapes include concavities and/or holes. Complexity further increases if the items can be rotated (at fixed angles, a given number of times, or even arbitrarily). In consequence, for evaluating the feasibility of a batch in the context of serial-batch scheduling in metal processing, the "highly irregular strip packing problem with free rotations" must be solved. We call this problem the "complex nesting problem (CNP)" in the following. The CNP represents the base-level decision in the hierarchical planning context.

To summarize the problem description, we depict the relationship between the theoretical concept of hierarchical production planning and the concrete manifestations of the application case's decision problems (models) in Fig. 2. The bold terms represent the concepts and elements as used in literature (cf., e.g., Schneeweiß, 1995 and Schneeweiß, 2003), whereas the italic terms depict the concrete manifestations regarding the metal processing application case. The numbers in brackets show the basic course of action.

Due to the high complexity of the top-level serial-batch scheduling problem, heuristic solution methods will most likely be used for solving. During the execution of these solution methods, a huge number of potential batches is created and their feasibility must be evaluated. Because the iterative solving of CNPs (to evaluate batch feasibility) is very time consuming even when applying heuristic nesting methods, approximative approaches

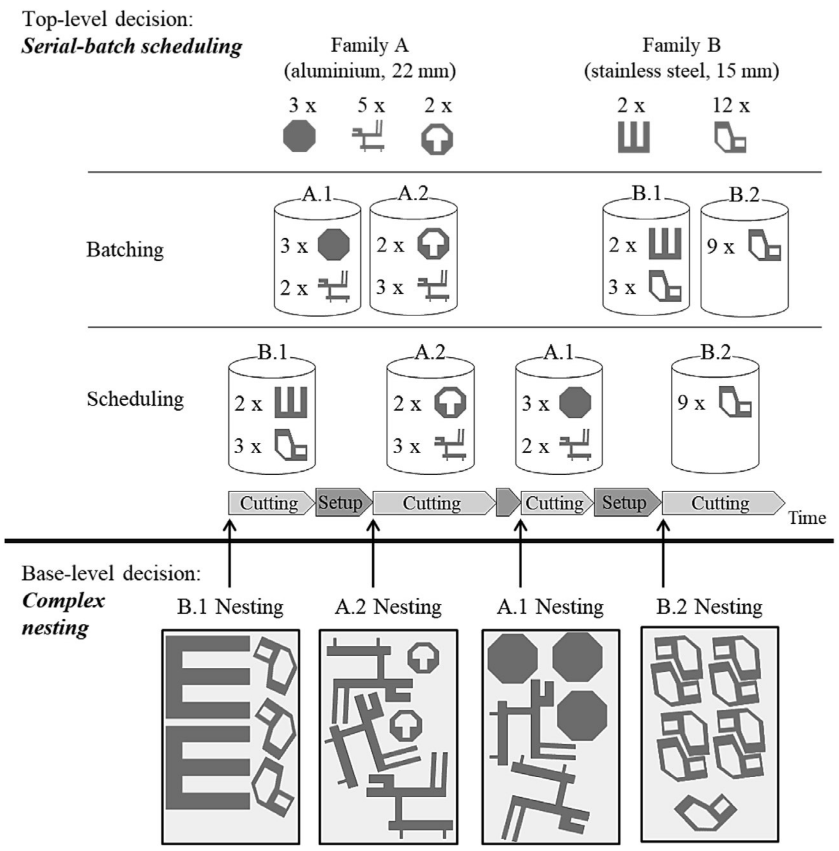

**Fig. 1.** Top-level and base-level decisions.

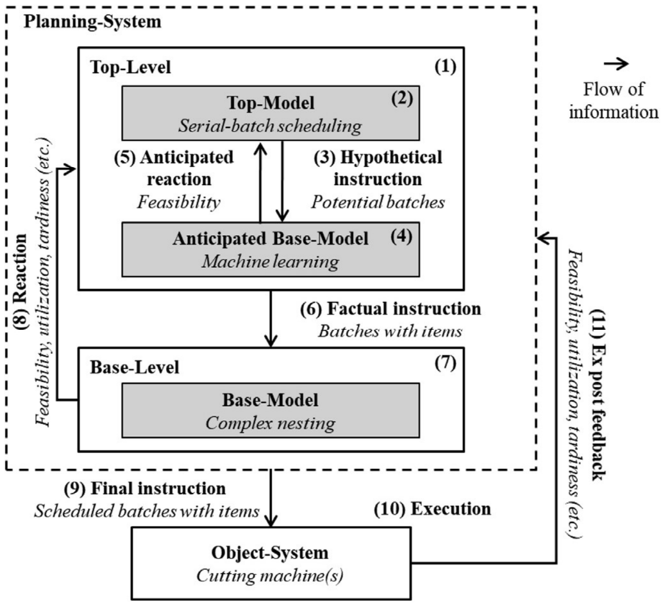

**Fig. 2.** Hierarchical integration of the approximate anticipation by machine learning.

are commonly used. To the best of our knowledge, only simple approximation approaches are currently used in literature (cf., Section 4.2 and e.g., Helo et al., 2019). Since these simple approximation approaches lack accuracy and solving CNPs by heuristics is not efficiently enough, we propose to use machine learning for the anticipation of complex nesting solutions and thus, for the decision about the feasibility of a batch. In this way, we aim at a most suitable trade-off between feasibility-decision accuracy and response time (i.e., the time required for deciding about batch feasibility).

## 3. Background and related work

The literature review to examine related work is divided into three parts. The first part is dedicated to anticipation functions in the context of hierarchical production planning. The second part contains contributions related to the application of machine learning in the area of packing and cutting, whereas the third part introduces major machine learning concepts in general and briefly describes potential machine learning techniques appropriate for the prediction task.

### *3.1. Basics on anticipation functions*

In Schneeweiß (2003), four archetypical types of anticipation functions are described: perfect anticipation (reactive), approximate anticipation (reactive), implicit approximation (reactive), and non-reactive anticipation. In the case of perfect anticipation, the base-level decision model is completely known and exactly integrated into the top-level model. Only the parameters (information) used by the top-level model might not be known deterministically and change when solving the base-level model. In contrast, approximate anticipation is performed by using some approximative anticipation function (models or methods). For an implicit anticipation, only some (most important) aspects of possible base-level reactions are considered within the anticipation function. Note that for these three anticipation types, explicit anticipation functions are used, whereas for the last type, non-reactive anticipation, no explicit anticipation function is used. Instead, some major aspects of the base-level decision are incorporated within the top-level model. Hereby, these aspects are not (reactively) depending on the top-level instructions. Because the complexity of the top-level model is often already too high, using non-reactive anticipation or even perfect anticipation is not possible. This is particularly the case for the models considered in this paper: the serial-batch scheduling problem (e.g., with minimizing total weighted tardiness as objective, it is NP-hard since it is a reduction from the corresponding NP-hard single machine sequencing problem) and the CNP (already the strip packing problem with rectangular items is NP-hard in the strong sense; cf., Martello et al., 2003). Therefore, we propose to use an approximative anticipation function that has also an implicit anticipation aspect as we only consider the height (representing the major aspect) to decide on batch feasibility and do not require the complete CNP solution.

In literature, different approaches for approximative anticipation functions can be found: for instance, exponential smoothing (Selçuk, Fransoo & Kok, 2006), clearing functions (cf., e.g., Graves, 1986 or Asmundsson, Rardin, & Uzsoy et al., 2006), or simulation (cf., e.g., Venkateswaran & Son, 2005 or Albey & Bilge, 2011). To the best of our knowledge, only Rohde (2004) uses a machine learning technique (artificial neural networks) for anticipation as we propose. Giving a good example for implicit anticipation, Kallestrup, Lynge, Akkerman and Oddsdottir (2014) emphasize that for non-perfect anticipations, the actual reaction from the object-system may be quite different from the anticipated reaction and that this problem can be reduced by increasing the quality of the anticipation. Therefore, we are not going to rely our prediction on a single machine learning technique like Rohde (2004) but propose a prediction framework to identify the most suitable technique.

## *3.2. Machine learning applied to packing and cutting problems*

In contrast to the almost non-existing application of machine learning techniques for anticipation purposes, these techniques are commonly used by hyper-heuristics and less frequently as solution method itself. Hyper-heuristics can be defined as "an automated methodology for selecting or generating heuristics to solve hard computational search problems" (Burke et al., 2010; for a similar definition cf., Pappa et al., 2014). To that, the central idea of hyper-heuristics is to learn which heuristic (or operator) will perform best by exploiting information about the current problem instance to solve and/or information from already solved problem instances. In this context, Burke et al. (2010) propose the distinction between online and offline learning. Online learning takes place while a heuristic solves a single problem instance, whereas offline learning aims to gather knowledge (e.g., in form of rules) from a set of training instances and to use this knowledge for solving unknown instances better. For recent advances and an extended classification scheme of hyper-heuristics see Drake, Kheiri, Özcan and Burke (2020). Most relevant publications concerning the development of hyper-heuristics for solving cutting and packing problems are López-Camacho, Terashima-Marín, Ochoa and Conant-Pablos (2013b), López-Camacho, Terashima-Marín, Ross and Ochoa (2014), Segredo, Segura and León (2014), Sim, Hart and Paechter (2012), Terashima-Marín, Ross, Farías-Zárate, López-Camacho and Valenzuela-Rendón (2010), and Gomez and Terashima-Marín (2018). Although these publications have a different scope, they are relevant concerning the prediction task at hand as the authors use features to characterize their problems that are related to CNPs. An explicit performance prediction to estimate minimum reference values for evaluation purposes or as termination criteria of iterative solution methods for the rectangular two-dimensional strip-packing problem is proposed by Neuenfeldt Júnior, Silva, Gomes, Soares and Oliveira (2019). Besides the differing prediction purpose, another main difference between their and our prediction task is their limitation to the problem with rectangular shaped items (in contrast to highly irregular ones). Remember, the goal of our approach is to predict the height of the strip that would result from the application of the CNP solution method that will be actually used for solving the CNPs defined by the superior scheduling (batching) decision.

Another way of applying machine learning techniques is to use them as autonomous solution method or as a part of a solution method. For instance, to calculate initial solutions for a two-dimensional cutting problem, Han and Na (1996) combine self-organizing feature maps (an unsupervised learning architecture) and fuzzy c-means. Dagli and Poshyanonda (1997) use a back-propagation neural network to generate larger patterns out of smaller input patterns for the nesting of rectangular patterns. Wong and Guo (2010) use a learning vector quantization neural network to classify items in order to select packing rules.

## *3.3. Major machine learning concepts*

Recent advances in machine learning have risen new possibilities for its application. Particularly deep learning (a synonym for deep neural networks; cf., Goodfellow, Bengio & Courville, 2017 or Kraus, Feuerriegel & Oztekin, 2020) has become a very powerful technique in the last years. Besides deep learning and "traditional" artificial neural networks, also other machine learning techniques might be appropriate for the height prediction task at hand. To be appropriate, the technique must be capable to perform a univariate multiple regression for the following reasons: First, as the strip height to be predicted is a single dependent continuous variable ("outcome variable"), classification and clustering models (methods) are not useful but univariate regression models. Second, as we do not expect that the height is related to a single CNP instance characteristic, several independent variables (also called "predictors", "covariates", or "features") must be considered by a multiple regression model. Summarizing, machine learning techniques that are capable to perform univariate multiple regressions and that belong to the class of supervised machine learning methods are basically appropriate. To select and evaluate the most appropriate machine learning method, we will briefly review models and methods in the following. For easier reading, we use the term regression model, regression method, or the abbreviation RM if we refer to corresponding machine learning models and methods in the remainder of this paper.

Because a detailed description of RMs is out of the scope of this paper, we only carve out the main aspects related to the prediction task at hand and refer to several books that are good starting points to deepen the topics. Remark that the basic idea of supervised machine learning is the use of training data (containing training instances or samples comprising independent variables and dependent/response variables) to determine the parameters of a model in such a way that a sufficient generalization is achieved. In the context of machine learning, a sufficient generalization means that a RM performs well on new, previously unseen inputs (cf., Goodfellow et al., 2017) and not only on the training data. Because we are not interested in inference (i.e., understanding the relationship between independent and dependent variables) but in most accurate predictions, the aspect of interpretability is not that important and thus, even very flexible parametric and non-parametric regression models are suitable. An important theoretical result of statistics and also machine learning is that a model's "generalization error" (i.e., its error rate on unknown data; also called "out-of-sample error", "test error", or, related to prediction tasks, "prediction error") consists of three very different errors (cf., Géron, 2019, p. 195): bias, variance, and irreducible error. Accordingly, the challenge is to find a RM for which bias and variance are low (this is referred to as the bias-variance trade-off;

cf., e.g., James, Witten, Hastie & Tibshirani, 2013 or Goodfellow et al., 2017). This is challenging because in general, more flexible RMs have a lower bias but a higher variance compared to inflexible ones. Note that the generalization error is not measured by using the training data but by a special hold-out data set called "test set". This test set may not be used in advance, neither during development, training, nor during the validation (evaluation) of different RMs. For the validation of different models (i.e., the identification of the most suitable RM), a special hold-out set, the "validation set", can be separated from the training data set. Behind this background, we briefly describe some main aspects of appropriate machine learning techniques in the following.

The group of rather inflexible, linear models includes multiple linear regression and regularized linear models like Ridge regression, Lasso regression or Elastic net. More flexible are non-linear models like Polynomial regression (based on non-linear transformations of the predictors; cf., e.g., Géron, 2019) and even more flexible is Support vector regression. Central hyper-parameter (hyper-parameters control the general behavior of a RM, e.g., how the model is trained; in contrast, a "normal" parameter is part of the model and determined during training) of Support vector regression is the used kernel: e.g., linear, polynomial with a specific degree, Gaussian RBF, or sigmoid. The kernels are used for "virtually enlarging" the feature space without having the (computational) drawback of an actually large feature space (cf., e.g., Bishop, 2006 or Murphy, 2013).

Another group of regression model bases on decision trees. Besides their advantages (e.g., interpretability or visualizability), decision trees have several disadvantages, i.e. they are non-robust (instable to even small changes in the data; cf. e.g., Géron, 2019) and tend to overfitting. To eliminate these disadvantages, for example Bagging regression trees, Random patches and Random subspaces, Random forests (particularly helpful to determine a feature's importance), or Boosting (e.g., Gradient boosted regression trees) have been developed. Because these RMs use several decision trees to construct a more accurate prediction model, they belong to the group of "ensemble models" where the final prediction is the (weighted) mean value of the individual predictions of the ensemble's members (e.g., decision trees). Generally, the result of an ensemble model has a similar bias but a lower variance than a single prediction model (cf., James et al., 2013). Instead of using the mean value to determine the final prediction, also an additional RM could be used. This type of ensemble model, where one or more layers of RMs are used and the prediction of the preceding layer is the input (feature) of the succeeding layer, is called stacking or blending (cf., e.g., Géron, 2019). An idea that is also used by the following RM.

Artificial neural networks are maybe the most flexible machine learning technique for predicting the strip height, as they do not make any assumptions about the relationship between independent variables and dependent variable (cf. Bishop, 1995). Because we are not going to discuss biological neural networks, we use the term Neural networks (or NN) for simplicity in the following. Due to their nonlinear nature, NNs perform very well for modeling complex data structures where the functional form is most likely nonlinear. Because the number of NN architectures is almost infinite (a good visualization is provided on www.asimovinstitute.org), we are not going into more detail about all available architectures. In this paper, we concentrate on a "standard" feedforward NN architecture with two hidden layers and evaluate several hyper-parameters like number of neurons per hidden layer and learning epochs. Such a NN with two hidden layers can be seen as a deep learning method.

Summarizing the related work, we conclude that there is no existing approach that uses and evaluates different machine learning techniques for the approximative anticipation of base-level reactions in the context of complex nesting problems.

## 4. Approximate anticipation by machine learning predictions

Central goal of the application of machine learning as approximative anticipation function is a most accurate anticipation of base-level reactions. Regarding the metal-processing application case, the anticipation function is used to predict batch feasibility, i.e., if the items assigned to a batch can be placed on a metal slide. Note that goal of the prediction is not a minimum height or lower bound but to hit the result of the nesting solution method that would be actually used for solving the finally created CNPs. Instead of solving the corresponding CNP, we seek to find an efficient RM (used as anticipation/prediction function $f^P$) and its parameters that predicts the height $\hat{H}_i$ of a CNP instance $I_i$ based on a set of instance features $\theta_i$ such that the prediction error is minimum. To measure the prediction error, we use the root mean squared error (RMSE) because larger errors are penalized stronger and it has the same unit (and similar scale) as the predicted value. Besides using the RMSE as prediction error measure in the training, hyper-parameter tuning, and validation phase, the RMSE is also used to estimate the overall expected loss of prediction function $f^P$ regarding unknown instances. Depending on the purpose, instance set $X$ in Eq. (1) can either be a training data set ($T$), a validation set ($V$), or a test set ($D$):

$$L(X,\ f^P) = \sqrt{\frac{1}{|X|}\sum_{i\in X}\left(\hat{H}_i^{f^P} - H_i^{f^{SM}}\right)^2} \qquad (1)$$

In (1), $\hat{H}_i^{f^P}$ depicts the height predicted by prediction function $f^P$ and $H_i^{f^{SM}}$ ($H_i$ for readability in the following) depicts the height calculated by solution method $f^{SM}$ for CNP instance $I_i$: $H_i = H_i^{f^{SM}} = f^{SM}(I_i)$. Thus, $H_i$ depicts the known response variable in the regression analysis and is called label in the context of supervised machine learning. Note that whenever the prediction of the height $\hat{H}_i^{f^P}$ is directly compared to the label $H_i$ and the RMSE definition (1) is used by a RM, we call this "absolute labeling strategy" ("AbsLab") in the remainder of the paper.

Before describing three simple approximative anticipation functions and their evaluation in terms of prediction accuracy, notations and further terms are introduced.

### 4.1. Notations

Basic task of the CNP as considered in this contribution is to position a set $P_i = \{p_{j=1}, \ldots, p_n\}$ of $n$ items on a large object (the strip) having a fixed width $W_i$ by minimizing the "used" height of the object. A feasible solution (called "nesting") is a placement of items without overlaps and with no item outside the object limits. An item $p_j$ is represented by a tuple of polygons $\Psi_j = (\rho_{j,1}, \rho_{j,2}, \ldots, \rho_{j,n_j})$ with $n_j \geq 1$. The first (enclosing) polygon $\rho_{j,1}$ (also depicted by $E$ in the following) defines the enclosing line and thus, separates the "interior" from the "exterior" of an item shape. All other polygons define holes of an item (these are also called difference polygons). Each of the polygons is defined by a tuple of vertices $\Lambda_{j,\rho} = (v_1, v_2, \ldots, v_{k_\rho})$ in $\mathbb{R}^2$. The edges defined by the vertices of a polygon are given by $\Gamma_{j,\rho}$ and their lengths by $\Omega_{j,\rho}$.

Basic properties of an item's shape are the height $h_j$ and width $w_j$ of the shape's minimum bounding rectangle (MBR; as we allow arbitrary rotations, the dimension naming is not relevant but we use the convention that without rotation, the width of an item is not smaller than its height, i.e., $w_j \geq h_j$). The area of the enclosing polygon is depicted by $a_j^E$, the area of the convex hull of the

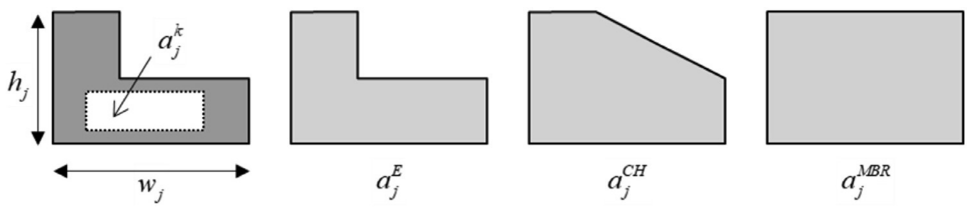

**Fig. 3.** Item shape properties.

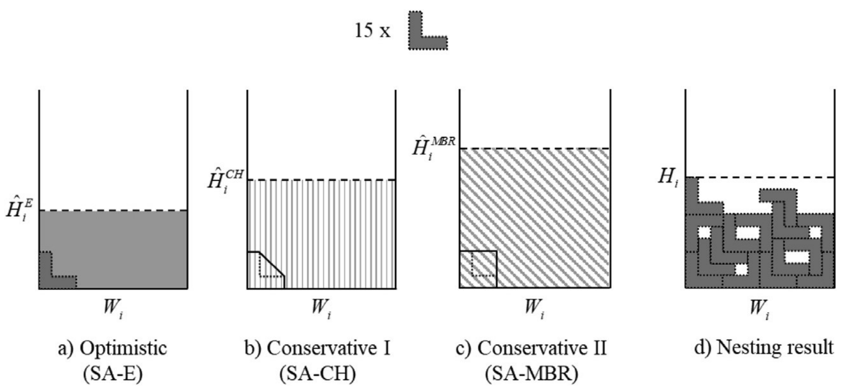

**Fig. 4.** Illustration of simple approximation approaches by a single CNP instance.

enclosing polygon by $a_j^{CH}$, and the area of the minimum bounding rectangle by $a_j^{MBR} = h_j \cdot w_j$ (cf. Fig. 3). The area of a hole defined by a polygon $\rho_{j,k\in\{2,\ldots,n_j\}}$ is depicted by $a_j^k$.

### 4.2. Simple approximation approaches

Based on the "area lower bound" idea proposed by Martello et al. (2003) for the strip-packing problem, three very simple and fast computable problem-specific approximate anticipation functions can be derived: SA-E (2), SA-CH (3), and SA-MBR (4).

The first function "fills" the strip with the total area of the enclosing polygons and represents an inordinate optimistic anticipation approach:

$$\mathrm{SA-E}: \quad \hat{H}_i^E = \max\left\{\max_{j\in P_i}\{h_j\}, \sum_{j\in P_i} a_j^E / W_i\right\} \tag{2}$$

The second, more conservative approach, fills the strip with the total area of the convex hulls:

$$\mathrm{SA-CH}: \quad \hat{H}_i^{CH} = \max\left\{\max_{j\in P_i}\{h_j\}, \sum_{j\in P_i} a_j^{CH} / W_i\right\} \tag{3}$$

The third, most conservative approach, fills the strip with the total area of the MBR of all items:

$$\mathrm{SA-MBR}: \quad \hat{H}_i^{MBR} = \max\left\{\max_{j\in P_i}\{h_j\}, \sum_{j\in P_i} a_j^{MBR} / W_i\right\} \tag{4}$$

Fig. 4 illustrates the three simple approximation approaches compared to a nested solution by an example with 15 items:

The low prediction accuracy of the simple approximation approaches is illustrated by the three diagrams in Fig. 5. In each diagram of Fig. 5, the abscissa depicts the heights obtained by the CNP solution method and the ordinate depicts the heights estimated by the simple approximation approaches (the light grey diagonal shows the perfect estimation). The illustrated approximations base on test data set D consisting of 17,640 CNP instances (cf. Section 5.4.1).

The diagrams of Fig. 5 show that the simple approximation methods are not suitable for an accurate decision on the batch feasibility: SA-E and SA-CH most likely will underestimate the height and lead to infeasible batches, whereas the accuracy of SA-MBR decreases with increasing height. Further details, substantiating the low prediction accuracy of the simple approximation methods are listed in Table 5 in Section 6.3. The lack of accuracy of these methods manifests the need for more accurate techniques e.g., from (supervised) machine learning.

However, the information provided by the simple approximation methods should or can be used by the RMs. One way for integration is to use them as descriptive features (independent variables). Another way is to use one of them for defining a new response variable $\Delta H_i$ based on the difference between $H_i$ and $\hat{H}_i^{MBR}$: $\Delta H_i = H_i - \hat{H}_i^{MBR}$ ($\hat{H}_i^{MBR}$ is used because it achieved the lowest RMSE of all simple approximation approaches). In this case, the RM is no longer predicting the height $\hat{H}_i^{f_P}$, but predicts the difference $\hat{\Delta}H_i^{f^P}$ between $H_i$ and $\hat{H}_i^{MBR}$. The "predicted" height can then be easily derived by $\hat{H}_i^{f^P} = \hat{H}_i^{MBR} + \hat{\Delta}H_i^{f^P}$. With this difference labeling strategy ("DiffLab"), we use the following RMSE definition to measure prediction errors during training and validation:

$$RMSE\,(X,\ f^P) = \sqrt{\frac{1}{|X|}\sum_{i\in X}\left(\hat{\Delta}H_i^{f^P} - \Delta H_i\right)^2} \tag{5}$$

## 5. The prediction framework

For the approximate anticipation, we propose the following framework to identify and apply the most suitable RM for the strip height prediction. Fig. 6 illustrates the framework's main components and the flow of information and data regarding the identification of the best RM (grey arrows) and the application of the RM by the scheduling method (dashed black arrows). It also highlights the relationship between the top-level and the base-level decisions.

To identify the most suitable RM ($f^{BEST}$), CNP instances are required for training, validation, and testing. These instances, provided by the "Data acquisition" component, can be acquired

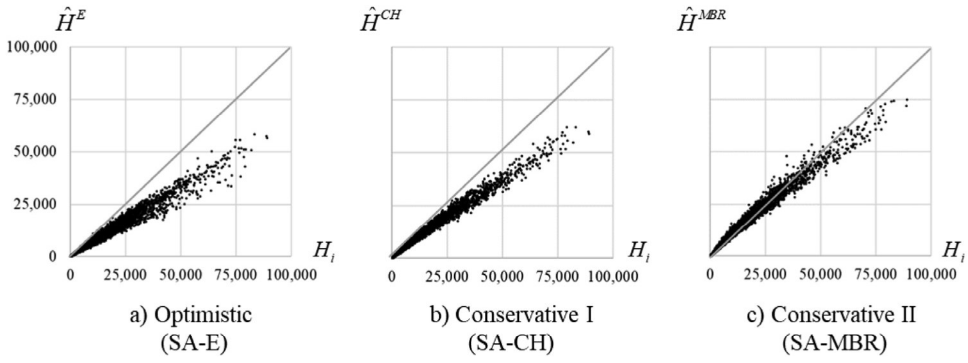

**Fig. 5.** Predictions by simple approximation approaches.

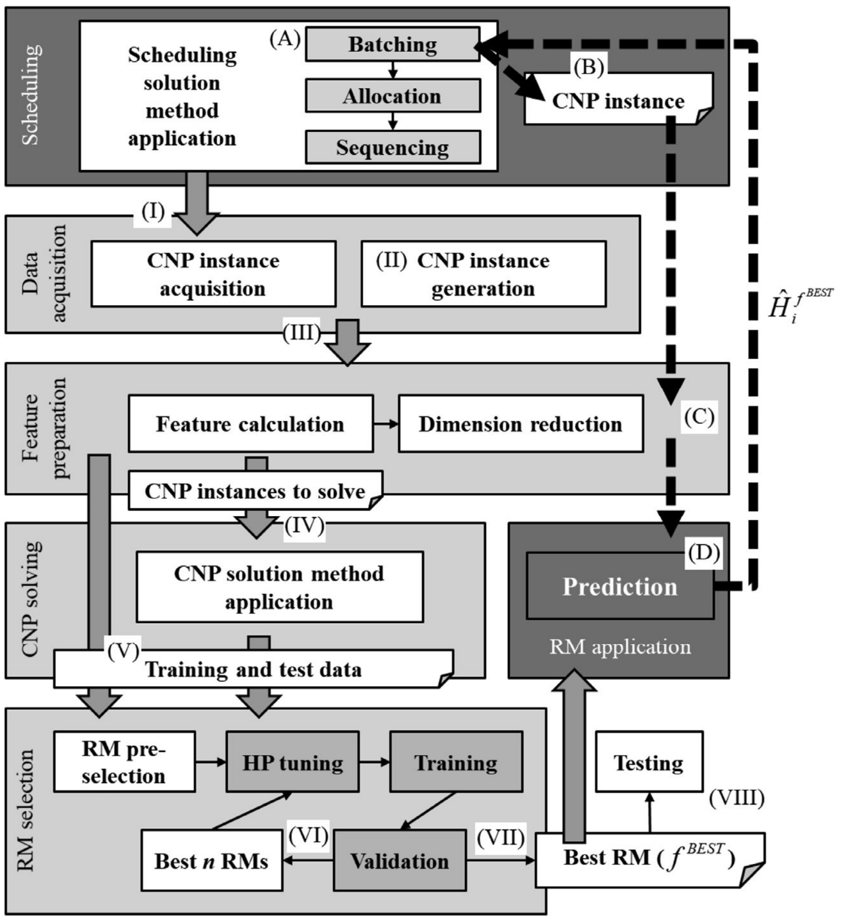

**Fig. 6.** Prediction framework.

"real-world" instances (obtained by historical scheduling method applications; (I)) and/or generated instances (II) that best possibly represent "real-world" instances (the latter aspect is discussed in detail in 5.1).

All acquired instances (III) are then used by the component "Feature preparation" that is responsible for calculating the descriptive features of the CNP instances (cf. 5.2.1) which in turn are given to the "Dimension reduction" component (cf. 5.2.2). Note that these two steps must also be performed when applying the best RM ($f^{BEST}$) for the prediction of the height. Therefore, both steps should be very efficient.

For CNP instances that are not yet solved (IV), it is important to calculate the actual heights $H_i$ (response variables, labels) with the same CNP solution method with which the "real-world" instances are solved (cf. 5.3). If the solution method changes, all available instances must be solved again by using this solution method. This is part of the component "CNP solving".

The resulting set of instances with features and labels are then split to training and test data sets (V; cf. 5.4.1).

The component "RM selection" uses these data sets for the RM (pre-) selection, hyper-parameter tuning (HP-tuning), training, validation, and testing (cf. 5.4). Thereby, we recommend to use two phases of HP-tuning, training, and validation: In the first phase (VI), all generally applicable RMs (cf. Section 3.3) are evaluated by means of their RMSE. To reduce computation times, this evaluation is based on the half of the training data and a limited set of hyper-parameter settings (HP-settings). In the second phase (VII), a selection of the $n$ most promising RMs is investigated with a greater set

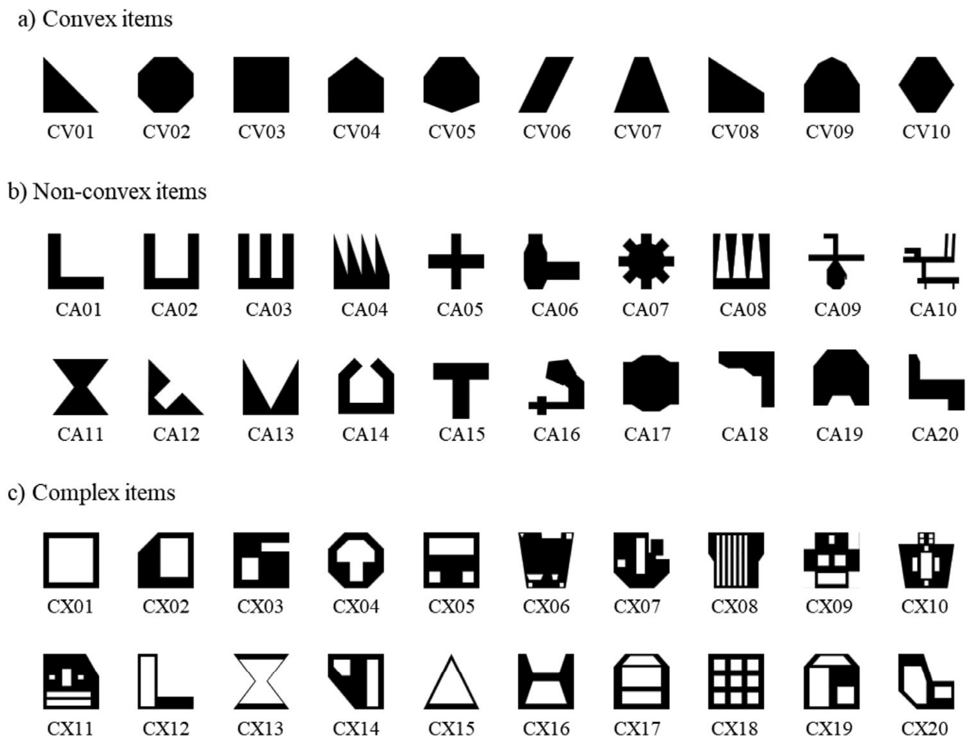

**Fig. 7.** Elementary item shapes.

of HP-settings and the complete training data set. In this way, the complete potential of each promising RM can be exploited and the best RM is identified and trained in an efficient manner.

After that, the expected loss of the best RM can be estimated based on the test data set and this trained RM is ready to be applied by the scheduling method (VIII).

The scheduling solution method uses this best RM $f^{BEST}$ to evaluate if a constructed batch (A), which in turn defines a CNP instance (B), is feasible. To enable the height estimation, features for this CNP instances must be calculated and the following dimension reduction provides the information used by $f^{BEST}$ (C). The predicted height $\hat{H}_i^{f^{BEST}}$ (D) is then used to decide if a nesting is possible, i.e., if the constructed batch is feasible.

## 5.1. Data acquisition and CNP instance generation

If there is no sufficient number of real-world CNP instances for training, validation, and testing of RMs available (not sufficient means that the number of samples is too small and thus, sampling noise is induced), instances can be generated (e.g., also proposed by Feng, Li, Cen & Huang, 2003). To avoid sampling biases, these instances must be representative for the complete, diverse instance space of real-world instances. Therefore, we basically rely on the procedures and aspects discussed by López-Camacho et al. (2014) and Neuenfeldt Júnior et al. (2019), Silva, Oliveira and Wäscher (2014), Wang and Valenzela (2001), Wäscher et al. (2007) for generating instances. In addition, we particularly follow the advice of Smith-Miles and Bowly (2015) to generate instances by controllable characteristics (attributes) to achieve a high instance dissimilarity and method discrimination (instances should elicit different behaviors; cf., Smith-Miles, Baatar, Wreford & Lewis, 2014). Note that the following instance generation procedure could be applied or easily adapted to any type of (complex) cutting and packing problems.

Because the quality of the training data is substantial for applying machine learning techniques, we put great emphasize on the instance generation procedure and its description.

### 5.1.1. Basic item shapes

In contrast to the generation of shapes for regular, rectangular, or (simple) irregular packing and cutting problems, the shapes of CNPs have a high degree of freedom and a completely automated generation of shapes would likely lead to unrealistic shapes and consequently, to unrealistic problem instances. To address this problem, we use shapes based on technical drawings from a sheet metal manufacturer operating several laser cutting machines (Fig. 7).

In doing so, we take care that the generated instances represent the diversity of real-world shapes. Accordingly, the instances base on 50 elementary item shapes with three basic types $BT \in \{CV, CA, CX\}$: 10 shapes are convex ($CV$; regular and irregular), 20 shapes are concave ($CA$), and 20 shapes are complex ($CX$). These elementary shapes are then used to derive a comprehensive set of diverse item shapes by scaling according to the attributes item widths $IW$ and item height $IH$.

Based on the item width attribute specified by $IW \in \{S, M, L, XL\}$, the width of the new item's MBR is drawn from a uniform distribution with restrictions related to the reference strip width $W^R = 1,000$ scaled by parameter $\zeta$: $w_j(IW) \sim U(0.9 \cdot \zeta \cdot W^R, 1.1 \cdot \zeta \cdot W^R)$ with $\zeta = 0.1$, $\zeta = 0.25$, $\zeta = 0.5$, and $\zeta = 0.75$ for $S$, $M$, $L$, and $XL$, respectively. To reflect the approach of quadratic and narrow items proposed by several authors (cf., e.g., Terashima-Marín et al., 2010 or Silva et al., 2014), we use quadratic, half-narrow, and narrow items. Accordingly, the item height attribute $IH \in \{Q, HN, N\}$ defines how the height of an item's MBR is drawn from a uniform distribution with restrictions related to the item's width $w_j$ scaled by parameter $\psi$: $h_j(IH) \sim U(0.9 \cdot \psi \cdot w_j, 1.1 \cdot \psi \cdot w_j)$ with $\psi = 1.0$, $\psi = 0.5$, and $\psi = 0.25$ for $Q$, $HN$, and $N$, respectively. After determining the MBR according to $IW$ and $IH$, the

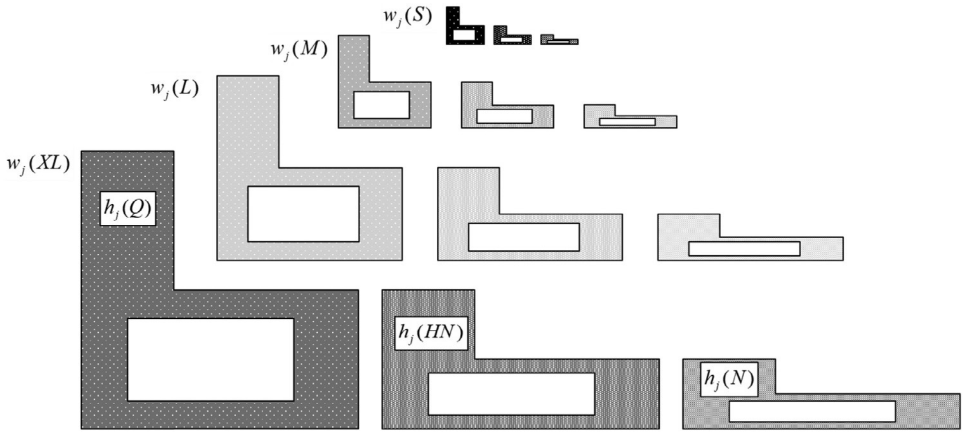

**Fig. 8.** Illustration of the shape scaling mechanism.

elementary shape is maximized (scaled) within the bounds ($w_j$ and $h_j$) to create the shape of the new item (Fig. 8 illustrates the scaling for the elementary shape depicted in Fig. 3).

For each of the 12 combinations of width and height attribute, we derive ten MBRs by drawing $w_j$ and $h_j$ ten times. Then, we scale each of the 50 elementary shapes to fit in the 120 MBRs. This leads to a shape repository of 6,000 different shapes that are separated into 36 categories according to their attributes type $BT \in \{CV, CA, CX\}$, width $IW \in \{S, M, L, XL\}$, and height $IH \in \{Q, HN, N\}$: e.g., the category ($CV$, $S$, $Q$) contains 100 basic shapes or the category ($CA$, $L$, $HN$) contains 200 basic shapes.

### *5.1.2. Instance attributes and classes*

All generated instances are categorized by classes based on several attributes (generation parameters). First attribute of a generated instance $I_i$ is the width $W_i$ of the strip. Here, we use the attributes $OW \in \{SW, MW, LW\}$ with the corresponding widths of $W_i(SW) = 1,000$, $W_i(MW) = 2,000$, and $W_i(LW) = 4,000$. Together with the number of items $n$ and the item related parameters explained in the following, the strip width determines if the strip (or nesting) aspect ratio is basically rather "quadratic" or "narrow" (on the importance of the strip aspect ratio cf., e.g., Neuenfeldt Júnior et al., 2019). Item related attributes used in our generation procedure described in the following section are item type assortment ($ITA$), item type heterogeneity ($ITH$), item width assortment ($IWA$), and item height assortment ($IHA$).

Before describing these attributes in detail, we introduce the mechanism of "random attribute selection with shuffling" that is used later on (cf. Section 5.1.3). As pointed out by Silva et al. (2014), beta distributions with probability density function $f(x, \alpha, \beta)$ should be preferred for instance generation because of their flexibility to model different probabilities depending on the parameters $\alpha$ and $\beta$. Because randomly parameterized beta distributions would result in too similar instances (contradicting the request for diverse instances), we use a set $B$ of 81 beta distributions based on all combinations of $\alpha$ and $\beta$ values from the set {1.0, 1.5, 2.0, ..., 5.0}. The resulting density functions and the attribute selection intervals are illustrated in Fig. 9. A uniformly drawn beta distribution is named $beta^R$ in the following.

As the evaluation of the resulting attribute selection by these distributions has shown undesired accumulations due to the law of large numbers (cf. Fig. 10), we enhance the attribute selection by a "shuffling" mechanism, i.e., a randomized assignment of attributes to selection intervals. The randomly determined assignment of an attribute to a selection interval is called *attPerm* in the following.

The positive effect of the shuffling mechanism is illustrated in Fig. 10. For the illustration of the shuffling mechanism, a

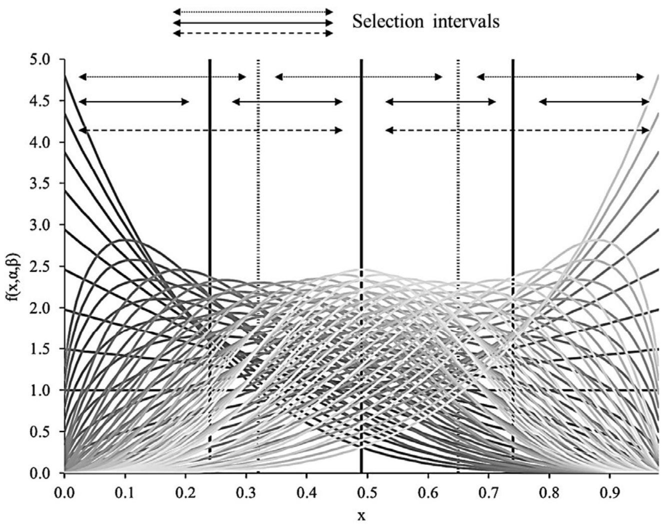

**Fig. 9.** Visualization of selectable beta distributions.

simulation with 1,000 iterations is used. In each iteration, one beta distribution (out of the set $B$ containing 81 distributions) was drawn and based on this beta distribution, 1,000 attributes have been drawn out of three or four classes, respectively. Summarizing, the "random attribute selection with shuffling" ($RBetaS$) mechanism uses two randomly defined parameters: $beta^R$ and $attPerm$.

The parameter item type assortment $ITA$ defines the composition of different items with regard to the elementary shape types convex ($CV$), concave ($CA$), and complex ($CX$): $ITA \in \{CV, CA, CX, CV+CA, CV+CX, CA+CX, CV+CA+CX\}$. If more than one type is specified, the selection is done by $RBetaS$.

To further improve instance dissimilarity, we use the parameter item type heterogeneity $ITH \in \{WH, SH\}$ and only consider randomly defined subsets of all elementary shapes (specified by the other parameters) for selection. Thereby, for $WH$ ("weakly heterogeneous") and $SH$ ("strongly heterogeneous"), the number of elementary shapes in the subsets is drawn from a discrete uniform distribution restricted by $[0.2 \cdot \upsilon \cdot n_{est}, 0.4 \cdot \upsilon \cdot n_{est}]$ with $\upsilon = 1$ and $\upsilon = 2$, respectively. Here, $n_{est}$ depicts the total number of elementary shapes per type (e.g., $n_{est} = 10$ for convex shapes).

The parameter item width assortment $IWA$ defines the composition of items with different widths according to $IW \in \{S, M, L, XL\}$: $IWA \in \{S+M, M+L, L+XL, S+M+L, M+L+XL, S+M+L+XL\}$. If more than one type is specified, the selection is done by $RBetaS$.

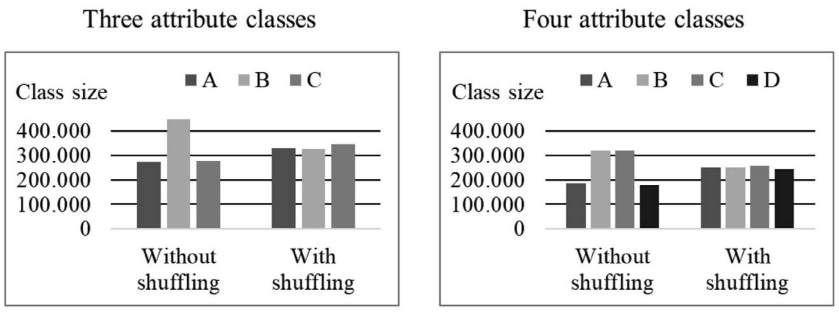

**Fig. 10.** Influence of the shuffling mechanism.

The item height assortment parameter *IHA* defines the composition of items with different heights according to $IH \in \{Q, HN, N\}$: $IHA \in \{Q, HN, N, Q{+}HN, Q+N, HN{+}N, Q{+}HN{+}N\}$. If more than one type is specified, the selection is done by *RBetaS*.

According to the attributes *OW*, *ITA*, *ITH*, *IWA*, and *IHA*, a total number of 1,764 instance classes is available. Each class can be specified by a tuple like $(SW, CV{+}CX, WH, M+L{+}XL, Q{+}HN)$. Regarding the aspect of weakly and strongly heterogenous items (cf. Wäscher et al., 2007), we classify all instances with item type heterogeneity *WH* and an item width assortment from the subset $\{S+M, M+L, L{+}XL\}$ to be weakly heterogenous and all others to be strongly heterogenous.

### 5.1.3. Generation procedure

The following pseudo-code illustrates the main course of action of the instance generation procedure:

```
createInstances($N_c$:= number of instances per class, $lb^n$, $ub^n$, $shapeRepository[]$)
For each $OW \in \{SW, MW, LW\}$
  For each $ITA \in \{CV, CA, CX, CV{+}CA, CV{+}CX, CA{+}CX, CV{+}CA{+}CX\}$
    For each $ITH \in \{WH, SH\}$
      For each $IWA \in \{S+M, M+L, L{+}XL, S+M+L, M+L{+}XL, S+M+L{+}XL\}$
        For each $IHA \in \{Q, HN, N, Q{+}HN, Q+N, HN{+}N, Q{+}HN{+}N\}$
          For $i = 1$ to $N_c$
            $beta^R_{ITA}$:= $\sim U(B)$; $beta^R_{IWA}$:= $\sim U(B)$; $beta^R_{IHA}$:= $\sim U(B)$;
            $attPerm_{ITA}$:= getPerm($ITA$);
            $attPerm_{IWA}$:= getPerm($IWA$);
            $attPerm_{IHA}$:= getPerm($IHA$);
            $S[]$:= getShapeSubsets ($shapeRepository[]$, $ITA$, $ITH$, $IWA$, $IHA$)
            $n$:= $\sim U(lb^n, ub^n)$
            For $j = 1$ to $n$
              $basicType_j$:= getTypeAttribute ($beta^R_{ITA}$, $attPerm_{ITA}$, $ITA$); // e.g., CX
              $w_j$:= getWidthAttribute ($beta^R_{IWA}$, $attPerm_{IWA}$, $IWA$); // e.g., L
              $h_j$:= getHeightAttribute ($beta^R_{IHA}$, $attPerm_{IHA}$, $IHA$); // e.g., IH
              $item$:= selectItemFromSubset ($S[]$, $basicType_j$, $w_j$, $h_j$)
              addItemToInstance ($item$);
            Next
        ...
Next
```

Based on this procedure, we generate 50 instances ($N_c = 50$) for each of the 1764 instance classes, leading to 88,200 CNP instances in total. The lower bound of the number of items per instance ($lb^n = 50$) is derived by the number of items used by instances from literature, whereas the upper bound ($ub^n = 150$) is justified by the application case.

## 5.2. Feature preparation

The descriptive features used by the RMs are based on aggregated item (shape) properties and further instance related characteristics. Thereby, we use properties and concepts known from literature and new ones, particularly developed to characterize CNPs. Feature selection and combination are part of the feature engineering that has an important influence on the prediction performance of RMs (e.g., on the one hand, too less features can lead to underfitting but, on the other hand, irrelevant features can lead to overfitting; these aspects will be discussed in more detail in Section 5.2.2).

Instead of numerically descriptive features used by our RMs, it would be generally possible to use the information how many items with a specific shape are part of an CNP instance: e.g., the instance contains eight items with shape A, 34 items with shape B, 12 items with shape C, and so on. This kind of feature preparation could be very fruitful if item shapes are not varying but constant because using this kind of features, other prediction methods, like bag-of-words models as used in Naive Bayes spam filtering, could be used. Unfortunately, the number of different shapes is anything but constant in most companies and particularly in most sheet metal manufacturing companies. In this case, whenever a new shape has to be cut or packed, new instances for training, validation, and testing would have to be available or generated. As this is not an adequate approach for real-world applications, we use descriptive numerical features as these provide a higher degree of flexibility (regarding new item shapes).

### 5.2.1. Feature calculation

Besides the basic properties described in Section 4.1, further item properties are used to define the descriptive features $\theta_i$ of a CNP instance *i*. Some of these properties are obvious, some of them are derived from literature (López-Camacho et al., 2013b, 2014; Wang & Valenzela, 2001, and Neuenfeldt, Silva, Miguel Gomes & Oliveira, 2018), and some are newly defined to particularly express highly irregular shapes. The complete set of 43 item properties is listed in Table 8 in Appendix A-1. Based on these item properties, we use the functions *SUM*, *MED* (median), *MIN*, *MAX*, *VAR* (variance), *Q1* (first quartile), *Q3* (third quartile), *P10* (10% percentile), *P90* (90% percentile), and *SKEW* (Fisher-Pearson coefficient of skewness; cf., Zwillinger & Kokoska, 2000) to determine aggregated instance related features. Note that mean values are omitted because they are directly related to *SUM*. In contrast to Neuenfeldt Júnior et al. (2019), we are not using the ratios *Q3*/*Q1* and *P90*/*P10* but use the "unrelated" values and leave the combination of both to the RM. Besides these 430 ($43 \cdot 10$) aggregated features (named $SUM(h_j)$, $MED(h_j)$ etc. in the following), additional instance related features are listed in Table 1 (features marked by an asterisk * originate from López-Camacho et al., 2013b).

The number of totally 452 features ($43 \cdot 10 + 22$) seems to be quite large and can cause difficulties for some RMs. To that, we use dimension reduction as described in the following section. In addition, we define two sets containing different instance features to evaluate the influence of features on the prediction quality and computation times. The first set contains the total number of available instance features (TIF), whereas the second set only contains a reduced number of instance features (RIF). This second set RIF does not contain features having an average-case computation complexity higher than O($n$), e.g., O($n$ log $n$): *MED*, *Q1*, *Q3*, *P10*, *P90*, and *SKEW*. Accordingly, feature set RIF only contains

Table 1
Additional instance features.

| Feature | Description |
|---|---|
| $n$ | Total number of items |
| $W$ | Width of the strip |
| $\hat{H}^{E}$ | Predicted height based on the area of the enclosing polygon |
| $\hat{H}^{CH}$ | Predicted height based on the area of the enclosing polygon's convex hull |
| $\hat{H}^{MBR}$ | Predicted height based on the area of the enclosing polygon's MBR |
| $n^{D}$ | Number of different item categories; two items have a different category if they are not completely identical regarding the combination of the attributes $BT \in \{CV, CA, CX\}$, $IW \in \{S, M, L, XL\}$, and $IH \in \{Q, HN, N\}$. |
| $MIN^{\#IpC}$ | Minimum number of items regarding all item categories |
| $MAX^{\#IpC}$ | Maximum number of items regarding all item categories |
| $MEAN^{\#IpC}$ | Mean of the number of items regarding all item categories |
| $MED^{\#IpC}$ | Median of the number of items regarding all item categories |
| $VAR^{\#IpC}$ | Variance of the number of items regarding all item categories |
| $SKEW^{\#IpC}$ | Skewness of the number of items regarding all item categories |
| $Q1^{\#IpC}$ | First quartile of the number of items regarding all item categories |
| $Q3^{\#IpC}$ | Third quartile of the number of items regarding all item categories |
| $P10^{\#IpC}$ | 10% percentile of the number of items regarding all item categories |
| $P90^{\#IpC}$ | 90% percentile of the number of items regarding all item categories |
| $n^{LI}$ | Number of large items ($w_j \geq 0.75 \cdot W$); * |
| $n^{SI}$ | Number of small items ($w_j \leq 0.25 \cdot W$); * |
| $n^{HR}$ | Number of items with a high rectangularity ($r_j^E > 0.9$); * |
| $n^{LR}$ | Number of items with a low rectangularity ($r_j^E \leq 0.5$); * |
| $n^{NCON}$ | Number of non-convex items (items with $n_j^{XIA} > 0$) |
| $n^{COMP}$ | Number of complex items (items with $n_j > 1$) |

188 features ($43 \cdot 4 + 16$). Note that because item properties only have to be computed once, their amount influences computation efficiency only minimally.

As many RMs require or perform better on scaled feature values, we use a normalization (min-max scaling) between 0.0 and 1.0 for scaling all features.

#### 5.2.2. Dimension reduction

At first sight, a large number of descriptive features seems to be favorable for characterizing an instance in the best possible way and thus, increase prediction accuracy. This is indeed the case if these features are actually relevant for the prediction. If features are not relevant, prediction accuracy can get worse and training time unnecessarily increases. This is because such "noise" features, while increasing the dimensionality, exacerbating the risk of overfitting and adding bias (cf., e.g., James et al., 2013). Even if features are relevant, the variance incurred in determining their coefficients may compensate the benefits that they bring. Furthermore, the increased dimensionality leads to the need for larger data sets due to the curse of dimensionality (cf., e.g., Géron, 2019). To handle the trade-off between benefits and drawbacks of more or less descriptive features, dimension reduction methods can be very helpful.

One of the most popular dimension reduction methods is the Principal Component Analysis (PCA) belonging to the area of unsupervised machine learning. Within the prediction framework, PCA is used to convert a set of features into a set of uncorrelated descriptive variables (called "principal components") by retaining most of the variance of the original features. Main parameter for the PCA is the number of resulting components. Instead of defining a fixed number, we use the approach to specify a minimum value for the training data set's variance to be preserved (e.g., PCA(98%); as proposed by Géron, 2019).

Of course, also other dimension reduction methods like "Kernel PCA", "Locally Linear Embedding", or "Linear and Quadratic Discriminant Analysis" could be applied.

### 5.3. CNP solving

Because the RM is used to predict strip heights calculated with one specific CNP solution method, this solution method, that is not only used for solving training instances but also applied on the real CNP instances resulting from the superior scheduling task, must be determined. The CNP solution method used in this contribution is the open source nesting software "deepnest" (https://deepnest.io) that bases on "SVGnest" (https://svgnest.com) which in turn bases on the works of Burke et al. (2007), Kendall (2000), and López-Camacho, Ochoa, Terashima-Marín and Burke (2013a). All used sources and libraries are available at github.com. For solving the CNP instances, we used the following parameters: population size = 10 (standard value), mutation rate = 10 (standard value), optimization ratio = 0 (only material utilization is optimized, not cutting length), maximum computation time = 180 seconds, and number of threads = 2. Note that the last two parameters are set in order to achieve a reasonable trade-off between solution quality and computation time.

### 5.4. RM selection

The determination of the most appropriate RM, its parameters, and hyper-parameters starts with the separation of the available data into training data and test data. Only the training data is used for the next steps of hyper-parameter tuning, model training, and model validation. Final result of these steps is the RM achieving the lowest RMSE with regard to the training data.

#### 5.4.1. Data separation: training and test data

As it is common practice to use at least 20% of the available data for testing, we follow this approach and separate our 88,200 CNP instances into a training data set (T) consisting of 70,560 instances and a test data set (D) of 17,640 instances. Because we use an integrated training and validation method (cf., Section 5.4.3), an explicit validation set is not required. To have a greatest possible diversity in the test data set, we randomly select 10 from each of the 1,764 instance classes for the test data set and put the remaining instances into the training data set. This results in a stratified sampling.

#### 5.4.2. RM pre-selection

Because we have only very limited information about previous RM applications related to the prediction task at hand (only Neuenfeldt Júnior et al., 2019 address a similar problem), we have pre-selected a relatively large number of 18 RMs to be evaluated in the first phase of the RM selection process (the pre-selection is based on the different groups of RMs described in Section 3): Multiple linear regression (MLR), Ridge regression (RR), Lasso regression (LR), Elastic net (EN), LARS Lasso (LL), Polynomial regression (PR; with the best tuned linear model), Stochastic gradient descent (SGD), K-nearest neighbors regression (KNN), Kernel ridge regression (KRR), Support vector regression (SVR), Decision tree – CART (CART), Bagging regression trees (BRT; uses one of the base estimators PR, SGD, KNN, KRR, SVR, or CART with best hyperparameters), Random forest regressor (RF), Extremely randomized trees (ERT), AdaBoost with R2 (ABR2; uses one of the base estimators PR, SGD, KNN, KRR, SVR, or CART with best hyperparameters), Gradient boosted decision trees (GBDT), Stochastic gradient boosted decision trees (SGBDT), and Neural networks (NN). Because some of the RMs (PR, BRT, and ABR2) base on other RMs, a corresponding sequence of experiments must be respected.

#### 5.4.3. Hyper-parameter tuning, training, and validation

Because hyper-parameter tuning, training, and validation are strongly related to each other, these aspects are explained together.

To determine the best hyper-parameter setting (HP-setting) of a RM, approaches like manual search, grid search, or randomized search are common. In manual search, hyper-parameter values are set manually, in grid search, given value vectors for each hyper-parameter are combined to a full factorial analysis, and in randomized search, random combinations of the values given by predefined vectors for each hyper-parameter are used. In all three cases, one RM is iteratively trained and validated to determine the best hyper-parameter setting.

The training and validation of a RM with a specific hyper-parameter setting is performed based on the cross validation technique (cf., e.g., Bishop, 1995 or Goodfellow et al., 2017). This technique is commonly used by learning methods to achieve a good generalization, to compare the performance of several RMs (cf., Wong, 2015), and to estimate prediction errors (cf., Fushiki, 2011). Therefore, for determining the most suitable RM and its parameters, we use grid search and a 5-fold cross-validation with shuffling. Shuffling means that the instances in the training data set are randomly reordered before they are split into the different folds. To reduce computation times, we restrict the number of folds to five and do not perform any repetitions as we can assume that our training data set is large enough (35,280 instances in the first phase and 70,560 in the second phase; for discussions about the usefulness of a higher number of folds and repetitions cf., e.g., Bengio & Grandvalet, 2004 or Baets, Manderick, Rademaker & Waegeman, 2012).

The result of the 5-fold cross-validation of a RM is the mean validation RMSE of all five folds. This mean validation RMSE is the major criteria when determining the most suitable RMs in the first phase and the best RM in the second phase. Further criteria are the mean computation times (regarding all folds and the best hyper-parameter setting) and the coefficient of determination ($R^2$).

### *5.5. Testing and RM application (prediction)*

To evaluate whether a RM generalizes well on new CNP problem instances, we follow the common approach and apply the best RM on the test data set (containing instances the RM has never "seen" before) to determine its expected loss.

The best (most accurate) RM is finally used for the prediction of the strip height, i.e., this trained RM provides a highly efficient anticipation function for the superior scheduling problem to anticipate batch feasibility (and quality).

## 6. Evaluation

The accuracy of a machine learning technique does not only depend on its basic capabilities (e.g., the possibility to model nonlinear relations) but also on the engineered features, the dimension reduction (parameters), and the labeling strategy. Accordingly, we investigate the influence of the two feature sets RIF and TIF combined with three parameter settings:

- "PCA(98%)-AbsLab":
  PCA with 98% variance preservation combined with the absolute height label "AbsLab" (cf., (1) in Section 4)
- "PCA(98%)-DiffLab":
  PCA with 98% variance preservation combined with the difference label "DiffLab" (cf., (5) in Section 4.2)
- "PCA(90%)-SimApp":
  PCA with 90% variance preservation combined with the absolute height label "AbsLab" and the additional information of the simple height approximations as unscaled features (to compensate the reduced number of principal components resulting from PCA with 90%)

This latter setting is used to analyze the performance (efficiency and effectiveness) of RMs with regard to the number of components (features).

The proposed prediction framework is implemented in Python 3.7, instance data, training and test results are persisted in a PostgreSQL database, and statistical analyses are performed by RStudio. For implementing the various regression methods, we use the "scikit-learn" package (Pedregosa et al., 2011) despite for SGBDT and NN. For SGBDT, we use "XGBoost" (xgboost.ai) and for NN, we use "Keras" (keras.io) and "TensorFlow" (www.tensorflow.org). To provide the possibility to reproduce our results, an executable Jupyter-Notebook and the complete set of instance samples with instance attributes, features, labels, simple approximations, and the results of our best RM method are provided within a data set hosted on Mendeley data (Gahm, 2020).

All experiments are executed on workstations with Intel Xeon 3.0 GigaHertz CPUs and 64 GigaBytes RAM.

### *6.1. Results of RM selection phase I*

In phase I, we evaluate the 18 pre-selected RMs in terms of their mean validation RMSE and with regard to the different feature parameter settings previously described. Besides the mean validation RMSEs, Table 2 additionally depicts the number of tuned hyper-parameters ("Num. HP") and the number of investigated HP-settings ("Num. HP-settings") for each RM. Note that PR(EN) names Polynomial regression combined with tuned Elastic net, BRT(PR(EN)) names Bagging regression trees with tuned PR(EN), and so on. Unfortunately, not all RMs have been able to produce reasonable results and the return codes have not been meaningful in all cases. Thus, in Table 2, abbreviation OOM indicates "out of memory" errors and UKE indicates "unknown errors".

The results in Table 2 show that the calculation of all defined instance features (TIF) justifies the additional computation times by improved predictions. Particularly when comparing the computation times: the RIF instance feature set can be calculated in 0.09 seconds on average and the TIF instance feature set can be calculated in 0.39 seconds on average. However, if computation times are very critical also the RIF set in combination with NN, PCA(98%), and "DiffLab" is an opportunity. The results also show that the linear models MLR, RR, LR, EN, and LL are underfitting and thus, are not suitable for the height prediction task. Furthermore, the results clearly indicate the benefit of using the information provided by the simple approximation approaches, especially if $\hat{H}_i^{MBR}$ is used by the "DiffLab" label (cf., Eq. (5)).

Most promising RMs after this first phase are marked bold in Table 2 and analyzed in detail in Table 3. Besides the RMSE, we report the explained variance ($R^2$) and the mean training time ("MTT"; mean with regard to the best HP-setting and the five folds including training and validation).

Because the solution quality of ABR2(PR(EN)) is similar to that of the other most promising RMs but its mean computation is remarkably higher, we decided to analyze PR(EN), BRT(PR(EN)), and NN in RM selection phase II.

The results of the RM selection phase I also show the benefit of the proposed prediction framework that emphasizes the evaluation of a broad range of RMs, different feature sets, different labeling strategies, and dimension reduction parameters in order to identify the most suitable ones.

### *6.2. Results of RM selection phase II*

In phase II, we concentrate on the three RMs PR(EN), BRT(PR(EN)), and NN and the two configurations with RIF and TIF combined with PCA(98%) and "DiffLab". Here, we use the complete training data set with 70,560 CNP instances. Table 4 depicts the

**Table 2**
RMSEs with regard to RM, feature set, PCA variance, and labeling.

| RM (Num. HP, Num. HP-settings) | RIF | | | TIF | | |
|---|---|---|---|---|---|---|
| | PCA(98%) -AbsLab | PCA(98%) -DiffLab | PCA(90%) -SimApp | PCA(98%) -AbsLab | PCA(98%) -DiffLab | PCA(90%) -SimApp |
| MLR (0, 1) | 6786.6 | 822.3 | 788.3 | 6600.8 | 750.8 | 767.5 |
| RR (2, 120) | 6786.6 | 822.3 | 788.3 | 6600.7 | 750.8 | 767.5 |
| LR (3, 240) | 6786.6 | 822.3 | 788.3 | 6600.7 | 750.8 | 767.5 |
| EN (4, 2160) | 6786.6 | 822.3 | 788.3 | 6600.7 | 750.8 | 767.5 |
| LL (2, 120) | 6786.6 | 822.3 | 788.3 | 6600.7 | 750.8 | 767.5 |
| PR(EN) (4, 1080) | 753.1 | 464.8 | 443.5 | 432.3 | **389.7** | 413.6 |
| SGD (4, 3240) | 6818.9 | 828.2 | UKE | 6641.8 | 758.7 | UKE |
| KNN (3, 48) | 3550.6 | 805.8 | 628.3 | 4159.0 | 800.1 | 628.8 |
| KRR (2–4, 1140) | OOM | OOM | OOM | OOM | OOM | OOM |
| SVR (3–5, 4650) | 2447.9 | 772.0 | UKE | OOM | OOM | OOM |
| CART (5, 256) | 2604.1 | 840.8 | 727.6 | 2505.4 | 826.8 | 745.5 |
| BRT (PR(EN)) (2, 8) | 751.9 | 462.9 | 442.8 | 430.1 | **386.4** | 410.6 |
| RF (6, 1024) | 1684.6 | 684.1 | 558.9 | 1725.4 | 642.8 | 550.1 |
| ERT (6, 1600) | 1612.8 | 664.0 | 541.3 | 1548.1 | 609.9 | 526.8 |
| ABR2 (PR(EN)) (2, 10) | 752.2 | 463.6 | 442.5 | 429.8 | **386.8** | 411.1 |
| GBDT (6, 1536) | 1399.8 | 664.0 | 553.8 | 1390.1 | 577.0 | 536.8 |
| SGBDT (7, 4608) | 1244.0 | 590.5 | 550.8 | 1280.7 | 552.6 | 528.3 |
| NN (5, 225) | 424.9 | 389.5 | 732.4 | 361.7 | **341.7** | 725.4 |
| Number of principal components | 57 | 57 | 20+3 | 120 | 120 | 30+3 |

**Table 3**
Key figures of most promising RMs in phase I (all with TIF, PCA 98%, and “DiffLab”).

| RM | RMSE | $R^2$ | MTT [seconds] |
|---|---|---|---|
| PR(EN) | 389.7 | 0.8813 | 115.96 |
| BRT(PR(EN)) | 386.4 | 0.8833 | 6256.07 |
| ABR2(PR(EN)) | 386.8 | 0.8831 | 45,182.49 |
| NN | 341.7 | 0.9088 | 962.95 |

key figures with regard to RM and feature set, and additionally the percentage changes compared to phase I.

As we can see by the percentage changes of the mean computation times, computational efforts remarkably increase (particularly for NN) whereas prediction accuracy and explained variance are only slightly improved (despite NN with RIF where we have a larger improvement). Note that the remarkably increase of the mean computation time of the NN in phase II can be traced back not only to the larger number of CNP instances in the training set but also to the different best HP-settings: in phase I, we have two hidden layers with 512 and 64 neurons and 200 epochs and in phase II we have two hidden layers with 1,024 and 1,024 neurons and 500 epochs.

Summarizing the RM selection process, artificial neural networks (with two hidden layers each containing 1,024 nodes, a dropout rate of 0.3 for each node in the hidden layers and a L2 weight regularization rate of 0.001 for 500 epochs) are identified to provide the most accurate predictions.

### 6.3. Testing results

The final testing of the best RM (NN with TIF, PCA(98%), and “DiffLab”) is used to estimate the overall expected loss of the prediction. The standard test of machine learning techniques to estimate $L(X, f^P)$ bases on the test data set separated from the complete data set before starting training and validation (cf., Section 5.4.1). In Table 5, the results of the testing are depicted with regard to the complete test set D and with regard to the different instance classes (in the last four rows, the number in brackets indicates the number of instances).

The results of Table 5 clearly show the remarkable benefit of using NN as anticipation function compared to the simple approximation methods: For any case of instance characteristics, the expected loss achieved with NN is much lower compared to SA-E, SA-CH, and, SA-MBR. However, there are some CNP instance characteristics that make predictions more difficult: e.g., instances with a small strip width ($OW=SW$), instances where items with a large or extra-large width constitute the majority of items ($IWA=L+XL$ or $IWA=M+L+XL$), or instances where items are all more or less quadratic ($IHA=Q$). Furthermore, the results also show that prediction accuracy decreases with increasing number of items.

Summarizing the testing results, the prediction accuracy of the NN is remarkably good (with a mean expected loss – RMSE – of 327.89), particularly when comparing the predictions of the NN illustrated in Fig. 11 to the simple approximation methods illustrated by the diagrams in Fig. 5 (Section 4.2).

**Table 4**
Key figures of most promising RMs in phase II (with PCA(98%) and “DiffLab”).

| RM | RIF | | | TIF | | |
|---|---|---|---|---|---|---|
| | RMSE | $R^2$ | MTT [seconds] | RMSE | $R^2$ | MTT [seconds] |
| PR(EN) | 471.58 | 0.8383 | 42.19 | 383.96 | 0.8926 | 240.40 |
| | (1.46%) | (0.90%) | (72.13%) | (−1.47%) | (1.28%) | (107.31%) |
| BRT(PR(EN)) | 470.84 | 0.8389 | 3284.89 | 383.22 | 0.8930 | 17,082.95 |
| | (1.72%) | (0.81%) | (149.90%) | (−0.82%) | (1.10%) | (173.06%) |
| NN | 374.96 | 0.8978 | 6135.63 | 339.17 | 0.9162 | 56,569.90 |
| | (−3.73%) | (1.88%) | (191.26%) | (−0.74%) | (0.81%) | (5774.65%) |

**Table 5**
Testing results.

| | $f^P$ | $L(X,\ f^P)$ | | | | $R^2$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | SA-E | SA-CH | SA-MBR | NN | SA-E | SA-CH | SA-MBR | NN |
| **Testing (D)** | | **3688.03** | **2474.07** | **1230.48** | **327.89** | **0.8286** | **0.9229** | **0.9809** | **0.9986** |
| *OW* | *SW* | 5918.05 | 4062.95 | 1931.17 | 509.31 | 0.7797 | 0.8962 | 0.9765 | 0.9984 |
| | *MW* | 2148.95 | 1218.10 | 812.13 | 217.55 | 0.8435 | 0.9497 | 0.9777 | 0.9984 |
| | *LW* | 1078.64 | 609.66 | 391.47 | 125.73 | 0.8430 | 0.9498 | 0.9793 | 0.9979 |
| *ITA* | *CV* | 3129.45 | 3129.45 | 1249.70 | 275.33 | 0.8760 | 0.8760 | 0.9802 | 0.9990 |
| | *CA* | 4601.00 | 1611.84 | 1146.52 | 350.19 | 0.6833 | 0.9611 | 0.9803 | 0.9982 |
| | *CX* | 3149.15 | 2502.35 | 1586.27 | 357.52 | 0.8932 | 0.9326 | 0.9729 | 0.9986 |
| | *CV+CA* | 3977.66 | 2424.98 | 1051.82 | 303.41 | 0.7877 | 0.9211 | 0.9852 | 0.9988 |
| | *CV+CX* | 3119.24 | 2792.69 | 1251.61 | 334.86 | 0.8843 | 0.9072 | 0.9814 | 0.9987 |
| | *CA+CX* | 3949.07 | 2090.48 | 1147.13 | 335.36 | 0.8030 | 0.9448 | 0.9834 | 0.9986 |
| | *CV+CA+CX* | 3631.07 | 2478.72 | 1103.30 | 331.17 | 0.8319 | 0.9217 | 0.9845 | 0.9986 |
| *ITH* | *WH* | 3615.56 | 2419.76 | 1292.59 | 356.02 | 0.8312 | 0.9244 | 0.9784 | 0.9984 |
| | *SH* | 3759.10 | 2527.21 | 1165.05 | 297.11 | 0.8260 | 0.9214 | 0.9833 | 0.9989 |
| *IWA* | *S+M* | 451.00 | 270.59 | 169.20 | 77.85 | 0.8356 | 0.9408 | 0.9769 | 0.9951 |
| | *M+L* | 2021.32 | 1208.99 | 724.60 | 225.19 | 0.8231 | 0.9367 | 0.9773 | 0.9978 |
| | *L+XL* | 6644.81 | 4589.43 | 2186.82 | 506.77 | 0.7525 | 0.8819 | 0.9732 | 0.9986 |
| | *S+M+L* | 1303.67 | 752.53 | 507.64 | 170.35 | 0.8440 | 0.9480 | 0.9763 | 0.9973 |
| | *M+L+XL* | 4310.76 | 2773.85 | 1415.28 | 402.00 | 0.7944 | 0.9149 | 0.9778 | 0.9982 |
| | *S+M+L+XL* | 3589.52 | 2422.33 | 1219.77 | 375.32 | 0.8140 | 0.9153 | 0.9785 | 0.9980 |
| *IHA* | *Q* | 6352.49 | 4394.10 | 2125.28 | 545.00 | 0.7792 | 0.8943 | 0.9753 | 0.9984 |
| | *HN* | 2813.26 | 1737.32 | 865.29 | 249.37 | 0.8066 | 0.9263 | 0.9817 | 0.9985 |
| | *N* | 1108.51 | 727.32 | 258.17 | 133.20 | 0.7966 | 0.9124 | 0.9890 | 0.9971 |
| | *Q+HN* | 4473.39 | 2957.50 | 1504.45 | 360.63 | 0.8178 | 0.9204 | 0.9794 | 0.9988 |
| | *Q+N* | 3608.03 | 2460.09 | 1232.61 | 346.02 | 0.8350 | 0.9233 | 0.9807 | 0.9985 |
| | *HN+N* | 1920.72 | 1150.45 | 530.59 | 182.41 | 0.8270 | 0.9379 | 0.9868 | 0.9984 |
| | *Q+HN+N* | 2999.24 | 1967.05 | 1096.40 | 304.28 | 0.8451 | 0.9334 | 0.9793 | 0.9984 |
| $50 \leq n \leq 75$ (4547) | | 2215.57 | 1429.96 | 725.22 | 221.77 | 0.8221 | 0.9259 | 0.9809 | 0.9982 |
| $75 < n \leq 100$ (4396) | | 3145.23 | 2032.87 | 1000.57 | 289.51 | 0.8137 | 0.9222 | 0.9811 | 0.9984 |
| $100 < n \leq 125$ (4312) | | 3967.05 | 2695.10 | 1331.19 | 353.93 | 0.8238 | 0.9187 | 0.9802 | 0.9986 |
| $125 < n \leq 150$ (4385) | | 4922.76 | 3349.28 | 1673.09 | 417.48 | 0.8208 | 0.9170 | 0.9793 | 0.9987 |

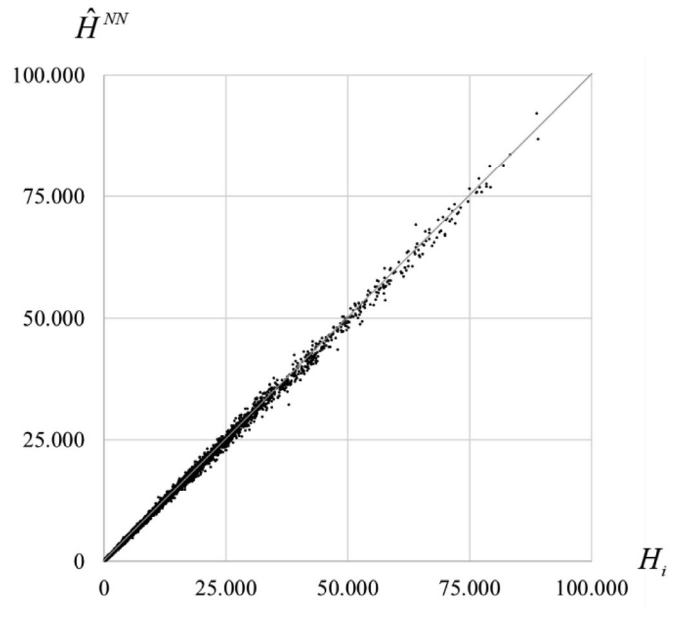

**Fig. 11.** Height predictions by NN with TIF, PCA(98%), and "DiffLab" (test data set).

Against the background of the hierarchical planning model (cf., Fig. 2), not only prediction accuracy but also response time can be important (depending on the solution method of the superior decision problem). The mean total response time to predict the height of a CNP instance with NN based on TIF, PCA(98%), and "DiffLab" is 440.81 microseconds (instance feature calculation: 397.56 microseconds; feature scaling: 0.10 microseconds; PCA computation: 0.09 microseconds; NN prediction: 43.06 microseconds). If this response time is not suitable, prediction accuracy could be traded for a lower response time by using the RIF feature set. With the RIF feature set, the NN (with PCA(98%), and "DiffLab") achieves a mean expected loss of 368.46 (+11.01%) and a mean total response time of 140.13 microseconds (−214.57%; instance feature calculation: 99.91 microseconds; feature scaling: 0.09 microseconds; PCA computation: 0.09 microseconds; NN prediction: 40.04 microseconds). For a better assessment of the response times, we compare them to the computation times (CT) of the nesting solution method for calculating first solutions and best solutions (with settings as described in 5.3) in Table 6. The comparison bases on 50 randomly selected instances having a corresponding number of items. Note that Mundim et al. (2018) report similar computation times for their general heuristic when solving two-dimensional nesting problems.

**Table 6**
Nesting computation times and prediction response times.

| Number of items ($n$) | First solutions | | Best solutions | Response times | |
|---|---|---|---|---|---|
| | Min. CT [s] | Mean CT [s] | Mean CT [s] | RIF [s] | TIF [s] |
| 50 | 5.16 | 8.75 | 119.93 | 0.11 | 0.40 |
| 75 | 5.28 | 11.38 | 102.72 | 0.12 | 0.42 |
| 100 | 6.12 | 14.83 | 104.54 | 0.13 | 0.43 |
| 125 | 8.36 | 19.44 | 112.63 | 0.15 | 0.45 |
| 150 | 7.74 | 21.00 | 106.14 | 0.16 | 0.46 |

The much higher computation times of the nesting solution method show the superiority of the approximative anticipation by machine learning in terms of time. However, the time benefit comes at the costs of a lower prediction accuracy compared to the perfect "prediction" (best solution) achieved by the nesting solution method. The actual influence of the prediction accuracy on the feasibility-decision is discussed in the following section.

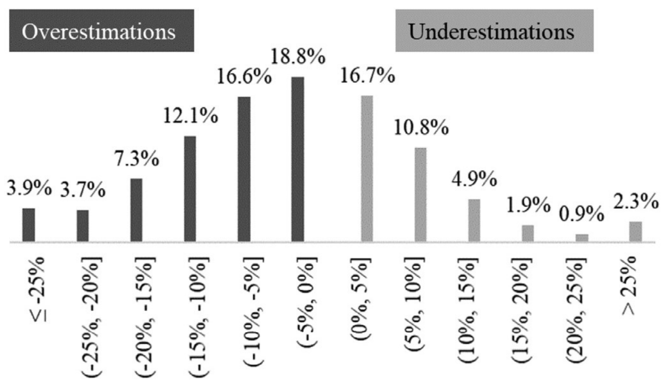

**Fig. 12.** Histogram of relative percentage errors by SA-MBR (test data set).

**Table 7**
Comparison of anticipation functions regarding batch feasibility decisions.

| | | Over-estimations | Underestimations below or equal to | |
|---|---|---|---|---|
| | | | 5% | 10% |
| SA-MBR | | 62.4% | 79.2% | 89.9% |
| NN with PCA(98%), | RIF | 60.8% | 92.9% | 98.6% |
| “DiffLab”, and | TIF | 58.5% | 93.4% | 98.8% |

## 6.4. Discussion

In this paper, we propose using machine learning for the approximate anticipation of base-level reactions instead of iteratively solving CNPs to achieve a most suitable trade-off between feasibility-decision accuracy and response time. As discussed so far, on the one hand, the application of a nesting solution method leads to perfect batch feasibility-decision accuracy at the expense of high computation times and, on the other hand, the simple approximation approaches have computation times of almost zero but lack batch feasibility-decision accuracy. The question yet to be answered is the investigation of the feasibility-decision accuracy achieved with our machine learning based predictions.

Remember, during solving the superior serial-batch scheduling problem, the predicted height is used to estimate whether or not a created batch is feasible, i.e., a batch is feasible if the predicted height is smaller than the maximum height of any currently available metal slide. In this context, we must be aware of two cases: false negative feasibility decisions and false positive feasibility decisions (the two other cases, i.e., true positive and true negative, will not cause any problems). False negative feasibility decisions mean that the height has been overestimated and thus, a batch has been declared to be infeasible despite being feasible. In this case, we might skip a good, or even the optimum, scheduling solution (i.e., scheduling solution quality is affected). False positive feasibility decisions mean that the height has been underestimated and thus, a batch has been declared to be feasible despite being infeasible. In this case, the most critical one, not only solution quality but solution feasibility is affected and the scheduling solution might be infeasible.

Because we do not have any information about actually available metal slides, we show the benefits of our approach by the following line of arguments. Each of the histograms in Fig. 12–14 depicts 12 classes of relative percentage errors. Thereby, negative errors indicate overestimations and positive errors underestimations.

Assuming that false negative decisions are bearable, all overestimations are acceptable (cf. column two in Table 7). Furthermore, if we can assume that underestimations below or equal to 5% only lead to a negligible share of false positive conclusions, up to 93.4% of the overall decisions would be implementable (NN with TIF).

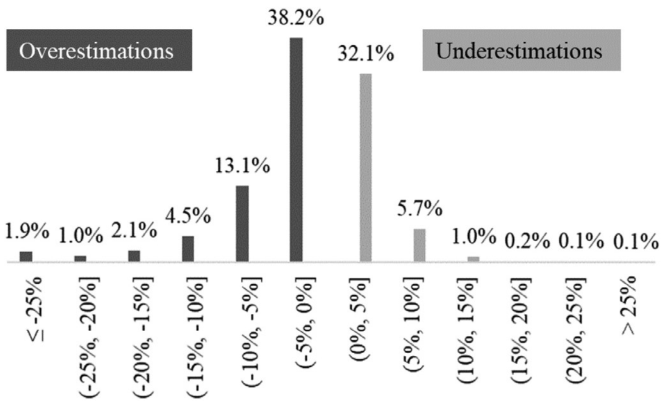

**Fig. 13.** Histogram of relative percentage errors by NN with RIF (test data set).

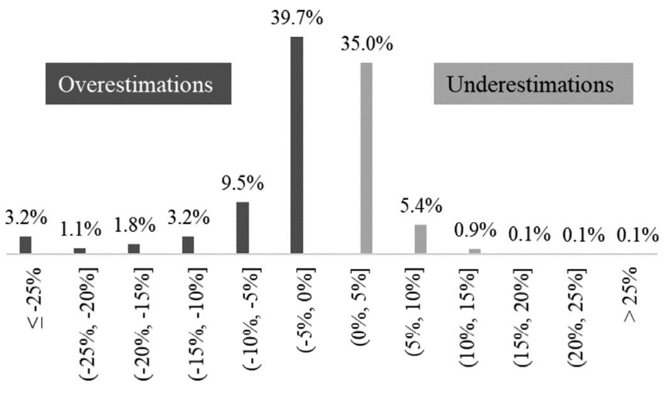

**Fig. 14.** Histogram of relative percentage errors by NN with TIF (test data set).

Relaxing this assumption to underestimations below or equal to 10%, this value even increases to 98.8% (cf. Table 7, the underlying values can be derived from Fig. 12, Fig. 13, and Fig. 14,).

The superiority of the approximate anticipation of base-level reactions by NN compared to the best simple approximation SA-MBR in terms of feasibility-decision accuracy becomes obvious by comparing the key figures depicted in Table 7.

## 7. Final remarks

Goal of our investigation has been to analyze the potential benefits of applying machine learning techniques for the approximate anticipation of base-level reactions instead of using simple approximate anticipation functions or solving the base-model. Although our investigation is related to a specific application case constituting the two related decision models serial-batch scheduling (top-level) and complex nesting (base-level), the underlying concept of using machine learning techniques as anticipation function in a hierarchical planning system is general and universal. To support the identification and application of the most suitable machine learning technique and its (hyper-) parameters, we propose a prediction framework comprising several components like data acquisition, instance generation, feature engineering, dimension reduction, and RM selection. The testing results demonstrate that the proposed prediction framework is capable to identify, train, and validate the best machine learning technique for the univariate multiple regression task at hand. The concluding discussion underscores the importance of a most accurate height prediction and shows the suitability of the proposed anticipation approach and its superiority compared to simple approximate anticipation functions.

Potential to improve prediction accuracy further is endowed by developing new, and enhancing or combining existing machine learning techniques. For instance, the application of deeper neural networks or stacking models. Additionally, providing prediction or confidence intervals quantifying the uncertainty of a prediction would be helpful for the batch feasibility decision. Also, research into machine learning stack models directly deciding on batch feasibility is of interest.

Besides the application case of sheet metal manufacturing, other areas of application can be found. For instance, the emerging area of additive manufacturing constitutes a very similar decision environment: top-level batching and scheduling decisions combined with base-level three-dimensional nesting decisions. Here, depending on the additive manufacturing technology, not only the feasibility of a batch is of interest but also the processing (build) time of a batch depends on the nesting decision (e.g., on the build orientation or on the maximum height of the batch; cf., Griffiths, Scanlan, Eres, Martinez-Sykora & Chinchapatnam, 2019 and Zhang, Yao & Li, 2020, respectively).

Beyond the application of machine learning techniques as approximate anticipation function in the area of production scheduling, the concept is applicable to other hierarchical decision environments and its potential should be investigated.

## Appendix

### A-1: List of item properties

## References

Albey, E., & Bilge, Ü. (2011). A hierarchical approach to FMS planning and control with simulation-based capacity anticipation. *International Journal of Production Research, 49*(11), 3319–3342. https://doi.org/10.1080/00207543.2010.482570.

**Table 8**
Item properties.

| Property | Description |
|---|---|
| $h_j$ | Height of the MBR |
| $w_j$ | Width of the MBR |
| $r_j^{h,w} = h_j/w_j$ | Aspect ratio of the dimensions of the MBR |
| $r_j^h = h_j/d_j$ | Ratio between the height and the diagonal of the MBR |
| $r_j^w = w_j/d_j$ | Ratio between the width and the diagonal of the MBR |
| $a_j^E$ | Area of the enclosing polygon (including areas of the difference polygons) |
| $a_j^{CH}$ | Area of the convex hull |
| $a_j^{MBR}$ | Area of the MBR |
| $r_j^{D-E} = 1 - (a_j^D/a_j^E)$ | Filling degree of the shape with $a_j^D = \sum_{k=2}^{n_j} a_j^k$ depicting the total area of all difference polygons ($a_j^k$ depicting the area of a difference polygon $k$; cf., white area in Fig. 3) |
| $r_j^{O-MBR} = a_j^O/a_j^{MBR}$ | Filling degree of the MBR with $a_j^O = a_j^E - a_j^D$ depicting the total occupied area |
| $r_j^E = a_j^E/a_j^{MBR}$ | Elementary rectangularity based on MBR (cf., López-Camacho et al., 2013b; for alternative measures of rectangularity cf., e.g., Rosin, 1999) |
| $r_j^{CH} = a_j^{CH}/a_j^{MBR}$ | Rectangularity of the convex hull |
| $n_j^E$ | Total number of vertices (edges) of the enclosing polygon |
| $l_j^E$ | Total length of the enclosing polygon's edges |
| $l_j^{MEAN-A,E} = (1/k_E) \cdot \sum_{z=1}^{k_E} l_z$ | Mean absolute length of the enclosing polygon's edges |
| $l_j^{MEAN-R,E} = l_j^{MEAN-A,E}/d_j$ | Mean relative length of the enclosing polygon's edges |
| $l_j^{MED-A,E}$ | Median of the absolute lengths of the enclosing polygon's edges |
| $l_j^{MED-R,E} = l_j^{MED-A,E}/d_j$ | Median of the relative lengths of the enclosing polygon's edges |
| $l_j^{VAR-A,E}$ | Variance of the absolute lengths of the enclosing polygon's edges |
| $l_j^{VAR-R,E} = l_j^{VAR-A,E}/d_j^2$ | Variance of the relative lengths of the enclosing polygon's edges |
| $l_j^{MIN-A,E}$ | Minimum absolute length of the enclosing polygon's edges |
| $l_j^{MIN-R,E} = l_j^{MIN-A,E}/d_j$ | Minimum relative length of the enclosing polygon's edges |
| $l_j^{MAX-A,E}$ | Maximum absolute length of the enclosing polygon's edges |
| $l_j^{MAX,E} = l_j^{MAX-A,E}/d_j$ | Maximum relative length of the enclosing polygon's edges |
| $l_j^C = l_j^E + \sum_{\rho=2}^{n_j} l_j^\rho$ | Total cutting length, with $l_j^\rho$ depicting the length of the perimeter of a difference polygon $\rho$. |
| $n_j^{RIA}$ | Number of "right" interior angles (between 85° and 95°) |
| $r_j^{RIA-IA} = n_l^{RIA}/n_j^E$ | Relative number of "right" interior angles |
| $n_j^{XIA}$ | Number of reflex interior angles (angles between 180° and 360°); if $n_j^{XIA} > 0$, the item has an irregular shape |
| $n_j^{XIA-IA} = n_j^{XIA}/n_j^E$ | Relative number of reflex interior angles |
| $n_j$ | Total number of polygons per item; if $n_j > 1$, the item has a complex shape |
| $c_j^{MAX}$ | Maximum degree of concavity (cf., López-Camacho et al., 2013b and Wang, 1998) |
| $c_j^{SUM}$ | Total degree of concavity measured by the sum of concavity of all reflex interior angles (e.g., to reflect star-shaped items) |
| $d_j^{CE-CMBR}$ | Euclidean distance between the centroid of the enclosing polygon $c_l^E$(defined by the means of the x and y coordinates of all vertices in$\Lambda_{j,E}$) and the centroid of the MBR $c_j^{MBR}$ (cf., Fig. 3) |
| $c_j^{MEAN-A,E} = MEAN(\delta)$ | Mean absolute distance between the enclosing polygon's vertices and centroid $c_l^E$ (with $\delta$ depicting the set of all distances between the enclosing polygon's edges and $c_l^E$) |
| $c_j^{MEAN-R,E} = c_j^{MEAN-A,E}/d_j$ | Mean relative distance between the enclosing polygon's vertices and the centroid $c_l^E$ |
| $c_j^{MED-A,E} = MED(\delta)$ | Median of absolute centroid distances |
| $c_j^{MED-R,E} = c_j^{MED-A,E}/d_j$ | Median of relative centroid distances |
| $d_j^{VAR-A,E} = VAR(\delta)$ | Variance of absolute centroid distances |
| $d_j^{VAR,E} = d_j^{VAR-A,E}/d_j^2$ | Variance of relative centroid distances |
| $c_j^{MIN-A,E} = MIN(\delta)$ | Minimum of absolute centroid distances |
| $c_j^{MIN-R,E} = c_j^{MIN-A,E}/d_j$ | Minimum of relative centroid distances |
| $c_j^{MAX-A,E} = MAX(\delta)$ | Maximum of absolute centroid distances |
| $c_j^{MAX-R,E} = c_j^{MAX-A,E}/d_j$ | Maximum of relative centroid distances |

Asmundsson, J., Rardin, R. L., & Uzsoy, R. (2006). Tractable nonlinear production planning models for semiconductor wafer fabrication facilities. *IEEE Transactions on Semiconductor Manufacturing, 19*(1), 95–111. https://doi.org/10.1109/TSM.2005.863214.

Baets, B. de, Manderick, B., Rademaker, M., & Waegeman, W. (2012). On estimating model accuracy with repeated cross-validation. In *Proceedings of the 21st Belgian-Dutch conference on machine learning*.

Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of K-fold cross-validation. *Journal of Machine Learning Research, 5*, 1089–1105.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Bishop, C. M. (2006). *Pattern recognition and machine learning. information science and statistics*: 01. New York, USA: Springer Science+Business Media.

Bitran, G. R., Haas, E. A., & Hax, A. C. (1981). Hierarchical production planning: A single stage system. *Operations Research, 29*(4), 717–743.

Burke, E. K., Hellier, R., Kendall, G., & Whitwell, G. (2007). Complete and robust no-fit polygon generation for the irregular stock cutting problem. *European Journal of Operational Research, 179*(1), 27–49. https://doi.org/10.1016/j.ejor.2006.03.011.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In M. Gendreau, & J.-Y. Potvin (Eds.), *International series in operations research & management science: vol. 146. handbook of metaheuristics* (2nd ed.) (pp. 449–468). Springer Science+Business Media.

Burke, E. K., Kendall, G., & Whitwell, G. (2009). A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem. *INFORMS Journal on Computing, 21*(3), 505–516. https://doi.org/10.1287/ijoc.1080.0306.

Chryssolouris, G., Papakostas, N., & Mourtzis, D. (2000). A decision-making approach for nesting scheduling: A textile case. *International Journal of Production Research, 38*(17), 4555–4564. https://doi.org/10.1080/00207540050205299.

Dagli, C. H., & Poshyanonda, P. (1997). New approaches to nesting rectangular patterns. *Journal of Intelligent Manufacturing, 8*(3), 177–190. https://doi.org/10.1023/A:1018517106992.

Dowsland, K. A., & Dowsland, W. B. (1995). Solution approaches to irregular nesting problems. *European Journal of Operational Research, 84*(3), 506–521. https://doi.org/10.1016/0377-2217(95)00019-M.

Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research, 285*(2), 405–428. https://doi.org/10.1016/j.ejor.2019.07.073.

Feng, S., Li, L., Cen, L., & Huang, J. (2003). Using MLP networks to design a production scheduling system. *Computers & Operations Research, 30*(6), 821–832. https://doi.org/10.1016/S0305-0548(02)00044-8.

Fushiki, T. (2011). Estimation of prediction error by using K-fold cross-validation. *Statistics and Computing, 21*(2), 137–146. https://doi.org/10.1007/s11222-009-9153-8.

Gahm, C. (2020). *Mendeley data, V1*. https://doi.org/10.17632/scr79cbxxd.

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems* (2nd edition). Beijing, Boston, Farnham, Sebastopol, Tokyo: O'Reilly.

Gomez, J. C., & Terashima-Marín, H. (2018). Evolutionary hyper-heuristics for tackling bi-objective 2d bin packing problems. *Genetic Programming and Evolvable Machines, 19*(1–2), 151–181. https://doi.org/10.1007/s10710-017-9301-4.

Goodfellow, I., Bengio, Y., & Courville, A. (2017). *Deep learning. adaptive computation and machine learning series*. Cambridge, Mass: MIT Press.

Graves, S. C. (1986). A tactical planning model for a job shop. *Operations Research, 34*(4), 522–533. https://doi.org/10.1287/opre.34.4.522.

Griffiths, V., Scanlan, J. P., Eres, M. H., Martinez-Sykora, A., & Chinchapatnam, P. (2019). Cost-driven build orientation and bin packing of parts in Selective Laser Melting (SLM). *European Journal of Operational Research, 273*(1), 334–352. https://doi.org/10.1016/j.ejor.2018.07.053.

Han, G. C., & Na, S.-. J. (1996). Two-stage approach for nesting in two-dimensional cutting problems using neural network and simulated annealing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 210*(6), 509–519. https://doi.org/10.1243/PIME_PROC_1996_210_150_02.

Hax, A. C., & Meal, H. C. (1973). Hierarchical integration of production planning and scheduling (Sloan working papers no. 656-73*)*. MA, USA: Massachusetts Institute of Technology (MIT). Cambridge.

Helo, P., Phuong, D., & Hao, Y. (2019). Cloud manufacturing – scheduling as a service for sheet metal manufacturing. *Computers & Operations Research, 110*, 208–219. https://doi.org/10.1016/j.cor.2018.06.002.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in R* (Corrected at 8th printing 2017). *Springer texts in statistics*. New York: Springer Science+Business Media.

Kallestrup, K. B., Lynge, L. H., Akkerman, R., & Oddsdottir, T. A. (2014). Decision support in hierarchical planning systems: The case of procurement planning in oil refining industries. *Decision Support Systems, 68*, 49–63. https://doi.org/10.1016/j.dss.2014.09.003.

Kendall, G. (2000). *Applying meta-heuristic algorithms to the nesting problem utilising the no fit polygon* PhD Thesis. Nottingham: University of Nottingham.

Kraus, M., Feuerriegel, S., & Oztekin, A. (2020). Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research, 281*(3), 628–641. https://doi.org/10.1016/j.ejor.2019.09.018.

Leao, A. A., Toledo, F. M., Oliveira, J. F., Carravilla, M. A., & Alvarez-Valdés, R. (2020). Irregular packing problems: A review of mathematical models. *European Journal of Operational Research, 282*(3), 803–822. https://doi.org/10.1016/j.ejor.2019.04.045.

Li, Z., & Milenkovic, V. (1995). Compaction and separation algorithms for non-convex polygons and their applications. *European Journal of Operational Research, 84*(3), 539–561. https://doi.org/10.1016/0377-2217(95)00021-H.

López-Camacho, E., Ochoa, G., Terashima-Marín, H., & Burke, E. K. (2013a). An effective heuristic for the two-dimensional irregular bin packing problem. *Annals of Operations Research, 206*(1), 241–264. https://doi.org/10.1007/s10479-013-1341-4.

López-Camacho, E., Terashima-Marín, H., Ochoa, G., & Conant-Pablos, S. E. (2013b). Understanding the structure of bin packing problems through principal component analysis. *International Journal of Production Economics, 145*(2), 488–499. https://doi.org/10.1016/j.ijpe.2013.04.041.

López-Camacho, E., Terashima-Marín, H., Ross, P., & Ochoa, G. (2014). A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications, 41*(15), 6876–6889. https://doi.org/10.1016/j.eswa.2014.04.043.

Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip-packing problem. *INFORMS Journal on Computing, 15*(3), 310–319. https://doi.org/10.1287/ijoc.15.3.310.16082.

Mundim, L. R., Andretta, M., Carravilla, M. A., & Oliveira, J. F. (2018). A general heuristic for two-dimensional nesting problems with limited-size containers. *International Journal of Production Research, 56*(1–2), 709–732. https://doi.org/10.1080/00207543.2017.1394598.

Murphy, K. P. (2013). *Machine learning: A probabilistic perspective (4. print. (fixed many typos)). adaptive computation and machine learning series*. Cambridge, Mass: MIT Press.

Neuenfeldt Júnior, A., Silva, E., Gomes, A. M., Soares, C., & Oliveira, J. F. (2019). Data mining based framework to assess solution quality for the rectangular 2D strip-packing problem. *Expert Systems with Applications, 118*, 365–380. https://doi.org/10.1016/j.eswa.2018.10.006.

Neuenfeldt, A., Júnior, Silva, E., Miguel Gomes, A., & Oliveira, J. F. (2018). The two-dimensional strip packing problem: what matters? In A. I. F. Vaz, J. P. Almeida, J. F. Oliveira, & A. A. Pinto (Eds.), *Springer proceedings in mathematics & statistics, operational research* (pp. 151–164). Aveiro, Portugal: Springer Cham.

Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2000). TOPOS – a new constructive algorithm for nesting problems. *OR Spektrum, 22*(2), 263–284. https://doi.org/10.1007/s002910050105.

Pappa, G. L., Ochoa, G., Hyde, M. R., Freitas, A. A., Woodward, J. R., & Swan, J. (2014). Contrasting meta-learning and hyper-heuristic research: The role of evolutionary algorithms. *Genetic Programming and Evolvable Machines, 15*(1), 3–35. https://doi.org/10.1007/s10710-013-9186-9.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Potts, C. N., & Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research, 120*(2), 228–249. https://doi.org/10.1016/S0377-2217(99)00153-8.

Rohde, J. (2004). Hierarchical supply chain planning using artificial neural networks to anticipate base-level outcomes. *OR Spectrum, 26*(4), 471–492. https://doi.org/10.1007/s00291-004-0170-x.

Rosin, P. L. (1999). Measuring rectangularity. *Machine Vision and Applications, 11*(4), 191–196. https://doi.org/10.1007/s001380050101.

Schneeweiß, C. (1995). Hierarchical structures in organisations: A conceptual framework. *European Journal of Operational Research, 86*(1), 4–31. https://doi.org/10.1016/0377-2217(95)00058-X.

Schneeweiß, C. (2003). Distributed decision making––a unified approach. *European Journal of Operational Research, 150*(2), 237–252. https://doi.org/10.1016/S0377-2217(02)00501-5.

Segredo, E., Segura, C., & León, C. (2014). Memetic algorithms and hyperheuristics applied to a multiobjectivised two-dimensional packing problem. *Journal of Global Optimization, 58*(4), 769–794. https://doi.org/10.1007/s10898-013-0088-4.

Selçuk, B., Fransoo, J. C., & Kok, A. G. de (2006). The effect of updating lead times on the performance of hierarchical planning systems. *International Journal of Production Economics, 104*(2), 427–440. https://doi.org/10.1016/j.ijpe.2005.04.005.

Silva, E., Oliveira, J. F., & Wäscher, G. (2014). 2DCPackGen: A problem generator for two-dimensional rectangular cutting and packing problems. *European Journal of Operational Research, 237*(3), 846–856. https://doi.org/10.1016/j.ejor.2014.02.059.

Sim, K., Hart, E., & Paechter, B. (2012). A hyper-heuristic classifier for one dimensional bin packing problems: improving classification accuracy by attribute evolution. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, … J. C. Mitchell (Eds.), *Lecture notes in computer science. parallel problem solving from nature - PPSN Xii* (pp. 348–357). Berlin, Heidelberg: Springer.

Smith-Miles, K., Baatar, D., Wreford, B., & Lewis, R. (2014). Towards objective measures of algorithm performance across instance space. *Computers & Operations Research, 45*, 12–24. https://doi.org/10.1016/j.cor.2013.11.015.

Smith-Miles, K., & Bowly, S. (2015). Generating new test instances by evolving in instance space. *Computers & Operations Research, 63*, 102–113. https://doi.org/10.1016/j.cor.2015.04.022.

Terashima-Marín, H., Ross, P., Farías-Zárate, C. J., López-Camacho, E., & Valenzuela-Rendón, M. (2010). Generalized hyper-heuristics for solving 2D regular and irregular packing problems. *Annals of Operations Research, 179*(1), 369–392. https://doi.org/10.1007/s10479-008-0475-2.

Venkateswaran, J., & Son, Y.-. J. (2005). Hybrid system dynamic—discrete event simulation-based architecture for hierarchical production planning. *International Journal of Production Research, 43*(20), 4397–4429. https://doi.org/10.1080/00207540500142472.

Wang, P. Y., & Valenzela, C. L. (2001). Data set generation for rectangular placement problems. *European Journal of Operational Research, 134*(2), 378–391. https://doi.org/10.1016/S0377-2217(00)00263-0.

Wang, W. X. (1998). Binary image segmentation of aggregates based on polygonal approximation and classification of concavities. *Pattern Recognition, 31*(10), 1503–1524. https://doi.org/10.1016/S0031-3203(97)00145-3.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research, 183*(3), 1109–1130. https://doi.org/10.1016/j.ejor.2005.12.047.

Wong, T.-. T. (2015). Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition, 48*(9), 2839–2846. https://doi.org/10.1016/j.patcog.2015.03.009.

Wong, W. K., & Guo, Z. X. (2010). A hybrid approach for packing irregular patterns using evolutionary strategies and neural network. *International Journal of Production Research, 48*(20), 6061–6084. https://doi.org/10.1080/00207540903246631.

Zhang, J., Yao, X., & Li, Y. (2020). Improved evolutionary algorithm for parallel batch processing machine scheduling in additive manufacturing. *International Journal of Production Research, 58*(8), 2263–2282. https://doi.org/10.1080/00207543.2019.1617447.

Zwillinger, D., & Kokoska, S. (2000). *CRC standard probability and statistics tables and formulae*. Boca Raton, Flo: Chapman & Hall/CRC.