Proceedings of the 2015 Winter Simulation Conference
L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, eds.

APPROXIMATION OF DISPATCHING RULES FOR MANUFACTURING SIMULATION USING DATA MINING METHODS

Soeren Bergmann
Niclas Feldkamp
Steffen Strassburger

Department for Industrial Information Systems
Ilmenau University of Technology
P.O. Box 100 565
98684 Ilmenau, GERMANY

ABSTRACT

Discrete-event simulation is a well-accepted method for planning, evaluating, and monitoring processes in production and logistics contexts. In order to reduce time and effort spent on creating the simulation model, automatic simulation model generation is an important area in modeling methodology research. When automatically generating a simulation model from existing data sources, the correct reproduction of the dynamic behavior of the modelled system is a common challenge. One example is the representation of dispatching and scheduling strategies of production jobs. When generating a model automatically, the underlying rules for these strategies are typically unknown but yet have to be adequately emulated. In previous work, we presented an approach to approximate the behavior through artificial neural networks. In this paper, we investigate the suitability of various other data mining and supervised machine learning methods for emulating job scheduling decisions with data obtained from production data acquisition.

1 INTRODUCTION

Discrete-event simulation is used in many different disciplines and application areas. In the area of production and logistics, simulation is a well-accepted tool for the planning, evaluation, and monitoring of all relevant processes. It is further applied as an operational decision support tool in manufacturing control. In all cases, it is essential to model the reality in a simulation model adequately, especially the dynamic behavior, for example buffer scheduling strategies. Because the quality of all simulation based predictions directly depends on the quality of the model, building adequate models is very important (VDI 2014).

The process of modeling for achieving high quality models which are verified and validated is typically time consuming and usually requires a simulation expert. Given these challenges, approaches to automatically generate simulation models, based on existing data, seem to be very appealing. Such approaches can reduce the amount of time needed to create a simulation model as well as the expertise needed for creating and conducting simulations. Additionally the automatic model generation can also help to improve the quality of the model (Bergmann 2013; Bergmann and Strassburger 2010).

In the last decade, different approaches for the automatic model generation have been published. Most of them have gaps in the generation of dynamic behavior. The main problem is mostly that the information about the behavior is often not explicit stored in the information systems that will be typically used in the model generation approaches, or even that the behavior is not explicit known at all (Bergmann and Strassburger 2010; Reinhart and Gyger 2008).

First approaches for determining control strategies, such as dispatching rules, were examined in recent years. One idea is to use historical production data (production data acquisition - PDA) for the generation
of the dynamic behavior. For instance Selke, Gyger and Reinhart (Reinhart and Gyger 2008; Selke 2005) investigated pattern recognition techniques to identify control strategies from a set of predefined rules. Li and Olafsson (Li and Olafsson 2005) used a similar approach but with decision trees as output. In previous work, we have demonstrated an approach to approximate behavior, especially the scheduling strategies through artificial neural networks (Bergmann, Stelzer, and Strassburger 2014). This work had some limitations, e.g., that the approximated learning model is restricted to a fixed and predefined number of jobs that can be compared at once and therefore is restricted to a predefined capacity of the sorter/buffer on which scheduling rules are applied to. In the presented approach, we overcome this limitation by introducing binary decisions which break down the scheduling decision to a comparison between two jobs and we also applied additional data transformation steps in order to improve the approximated rule's accuracy. Furthermore, the presented approach extends the existing approximation approach to the use of four other data mining methods.

The remainder of the paper is organized as follows. Section 2 discusses the representation of dynamic behavior from real systems through dispatching rules. In Section 3 we briefly review the basics of data mining and the compared classification methods (k-Nearest Neighbors algorithm – k-NN , Naive Bayes Classifier – NBC, Support Vector Machines – SVM, Classification And Regression Tree – CART) and how they can be used in manufacturing control. In Section 4 we introduce the process for approximation of behavior and we present a small test scenario for the method comparison. In Section 5 we show the results of the comparison of the methods with different parameterization under various conditions, like good or bad data quality, with or without data transformation etc. Finally, Section 6 contains some concluding remarks and future research directions.

2 DISPATCHING RULES IN MANUFACTURING CONTROL

All decisions that have to be made in production planning and control (PPC) fall into four main strategy clusters: strategies of job release, sequencing, lot size or resource scheduling (Milberg and Selke 2003). In practice, dispatching rules are very popular and can be used in all of the 4 clusters. The rules are used to determine the priorities for fulfilling orders (Pinedo 2012).

We further restrict the discussions in this paper to dispatching rules for scheduling jobs, but this does not limit the direct transferability of the presented approach to other clusters. A large number of dispatching rules for scheduling jobs can be found in the literature.

Dispatching rules can be distinguished by the input information they need. The information can be classified in different manners, e.g., in terms of the scope of the information (local vs. global) or in terms of the type/characteristic of the information (entry-time, process-time, setup-time or date oriented, value/cost based etc.). According to the focused characteristic of a dispatching rule, a certain set of job parameters is needed as input data, e.g., the arrival time (in a buffer, in the whole system) or the buffer-queue length (of this buffer, all buffers, next buffer). Additional parameters can be calculated, e.g., the slack time (due date - processing time on all remaining machines). The simple atomic rules which are based on such a parameter are in practice often combined (Bergmann, Stelzer, and Strassburger 2014).

In this paper we use a subset of typical rules as illustrating examples: first in first out (FIFO), shortest processing time (SPT), and work in queue (winQ).

3 USING DATA MINING METHODS FOR JOB SEQUENCE PLANNING

The term data mining can be summarized as discovering patterns in large data sets. This does not only include the application of algorithms from a variety of disciplines like statistics, artificial intelligence, and machine learning, but also covers data managing aspects. Especially a data preprocessing step is necessary when handling large and heterogeneous data sets. The overall goal is to extract information from data and transform it into an understandable structure for further use (Fayyad, Piatetsky-Shapiro, and Smyth 1996; Han, Kamber, and Pei 2012).

A subfield of data mining is the supervised learning. This subfield explores algorithms that can learn from data. To be specific, they try to emulate certain regularities from training data in order to make predictions and decisions (Bishop 2006).

A training example is composed of an input vector and an output class which serves as a label. The supervised learning algorithms therefore aim to infer a function from the given input vector to the desired class label by learning the training data sets. After the training is finished, the learned function is used to assign one of the prior determined labels to unlabeled datasets.

Looking at data sets collected by production data acquisition systems, a whole range of collected attributes per job like entry time, due date, and processing time can serve as an input vector for the learning algorithm as well as overall system attributes. In our approach, the learning algorithm looks at the binary decision between two jobs and which of them is given the preference in the sequence (Li and Olafsson 2005), so the input vector dataset is composed of the PDA data for both of the compared jobs. Furthermore, the class label for each is either A or B. A fictional example of labeled datasets in this context is shown in Table 1. Each row in the table represents a dataset with which the decision model is trained. The first five columns represent the input vector, whereas the decision column represents the according class label.

Table 1: Example training datasets for pairwise sequence decision.

Job A		**Job B**		**System**	**Decision**
P1	**P2**	**P1**	**P2**	**S1**	
12	80	20	10	5	Prefer Job A
20	90	20	15	15	Prefer Job A
20	100	30	80	100	Prefer Job B

Breaking down even complex scheduling rules to the binary decision between two jobs increases the total number of decisions, but also brings in more flexibility, which makes the learning model's implementation independent from the surrounding model and is easily transferable to other datasets since the queue length of jobs waiting to be compared and scheduled is possibly unlimited.

As said before, the data mining process usually requires a data preprocessing step. The goal here is to pass the data to the mining algorithm in a format that is optimized for its functionality. In our case, a data transformation can increase the performance of the learning algorithm drastically, so transformation of data was also part of our investigation. We covered three methods of data transformation: attribute selection, normalization, and attribute construction.

Attributes selection means to find attributes that are irrelevant for the class assignment and to eliminate those attributes before even applying the learning algorithm. This increases the effect that irrelevant attributes falsify the decision model. This is especially useful in a PDA database, where the number of captured attributes is potentially high, but the number of attributes that are relevant for the job sequence planning is possibly just a fraction of this number. This process is also known as feature subset selection, since only a subset of relevant features is selected for model construction. As a side effect, a smaller feature subset also reduces the computation times of the learning algorithms (Guyon and Elisseeff 2003).

Another useful transformation method is normalization. Even after eliminating unnecessary and misleading attributes, there may occur problems regarding the comparability between the remaining attributes. When using algorithms based on distance measures, results may be biased if the value ranges differ dramatically (e.g. [-1;1] for attribute 1 and [-10000:10000] for attribute 2). On the other hand, normalization is susceptible to outliers, because outliers cause a biased shift of the edges of the value ranges. To avoid those effects, we also applied a special case of normalization to the data called z-score (or standardization), whereby afterwards the dataset has a 0 mean and 1 standard deviation (El Attar 2005).

Construction of attributes is concerned with finding the optimal depiction space. In relation to our approach, the decision which job to prefer in the job sequence planning may not be dependent on distinct input attributes but on totaling, difference, or multiplication of two or more attributes (Wnek and Michalski 1994). We therefore also tested if attribute creation may increase the decision accuracy. Taking for example the FIFO strategy when comparing Job A and Job B, the dataset extracted from PDA would tell us the entry times of A and B. Building the difference between those values (thus constructing a new attribute) would make the decision easy considering whether this value would be smaller or greater than 0.

Construction of attributes, in a naive and multilevel approach, would let the number of constructed attributes grow rapidly. For example, having 20 starting attributes would after the third stage of construction already result in over 160 million attributes. In a more sophisticated approach, performing attribute selection after each stage of construction can improve this process drastically. Therefore it became clear that attribute construction and selection methods benefit from one another. In order to fully exploit these methods, we therefore did not apply these steps individually but combined them in a data transformation framework that in our result analysis was either applied fully or not at all. Still, we applied the standardization method separately, because computational application is easy and independent from attribute construction and selection.

There is a variety of data mining algorithms that can be used for learning decisions. In this work we focused on four of the most popular ones, which are k-Nearest Neighbors, decision trees, Naive Bayes Classifier and Support Vector Machines. The basic concept of functionality for those algorithms will be explained below.

3.1 K-Nearest Neighbors (k-NN)

This algorithm tries to classify an unlabeled dataset according to the label of the majority of its k nearest neighbors in the feature space. For example, with k = 1, the algorithm searches for the most similar dataset in the training data and assigns the same label to the unlabeled dataset. The choice of the k parameter is important for the effectiveness of the algorithm and depends on the data to be analyzed. Typically, larger values of k make the procedure more robust against effects of noise, whereas smaller values of k produce a more distinct separation between classes. Similarity between two datasets p, q is called distance d, and $d(p, q)$ is calculated through a distance metric. (1) shows the Euclidian distance, which is commonly used when calculating distance similarity between vector pairs.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(d_i - p_i)^2} \qquad (1)$$

In fact, the k-NN algorithm does not build a decision model or function, but the whole set of training data composes the model, whereas the algorithm compares the pairwise distances between the unlabeled datasets and the training data and searches for the most similar ones (Altman 1992).

3.2 Naive Bayes Classifier (NBC)

This algorithm is a probabilistic classifier. This means, the decision model labels a dataset according to the highest probability. During the training, the algorithm creates probability distributions of the input vector parameters for each class. When an unlabeled dataset comes in, the algorithm combines probabilities for each parameter and builds an overall probability value for each class which describes the likeliness that the unlabeled dataset belongs to that respective class. Finally, the class with the highest likeliness is labeled to the dataset (Becker, Kohavi, and Sommerfield 2002). NBC's biggest disadvantage is that the distribution function of continuous attributes has to be set in advance. A false distribution function may lead to false likelihoods and therefore increases the probability of an incorrect decision. A solution of this problem is the application of more complex distribution functions like the multivariate normal distribution (Rennie et al. 2003).

3.3 Support Vector Machines (SVM)

In contrast to the other methods, this algorithm always separates training data into two categories. Since we compare pairwise sequencing decisions between jobs as described above, our training examples have two distinct class labels anyway. To conduct the separation, the input parameters are distributed in an n-dimensional space according to length of the input vector. Then the algorithm tries to span a plane (which is called hyper plane if n > 3) through the space in order to separate the classes, whereas the hyper plane serves as a border. Through shifting and rotating of the plane, the number of false categorized examples should be minimized. Unlabeled datasets can then be labeled according on which side of the plane they are located (Han, Kamber, and Pei 2012). Problems occur when the data is not linearly separable. The solution is the application of the *kernel trick*, which translates the dataset into a feature space of higher dimension in which it is then separable. The kernel trick depends on an adequate kernel function, which is an additional parameter that needs to be configured before conducting the SVM algorithm (Boser, Guyon, and Vapnik 1992).

3.4 Decision Trees

Decision tree models are the most commonly known method when building decision models. A decision tree is composed of nodes and leafs, whereas decisions about parameter values are made in the nodes that ultimately leads to a specific class in a leaf at the bottom of the tree. To classify according to an existing tree is trivial, the induction of a new tree from a set of training examples is challenging, and a variety of algorithms has been developed to cover requirements like noise robustness, accuracy, and fast computation. A commonly known algorithm for decision tree induction is the CART algorithm, which we used in this work. CART induces binary trees, which means each node has exactly two children. The parameter at each node is chosen according to which so called split creates the highest information entropy. This value is calculated with a split criterion, for example the Gini index that measures the impurity of the two classes after the split (Breiman 1984).

4 SCENARIO DESCRIPTION

Our test case consist of three phases. For the training data, we used a simulation model of a production system to artificially create PDA data. This allowed us to quickly develop an adequately sized collection of datasets and flexibly replace the sequence decision rule. For simulation software we used *Tecnomatix Plant Simulation.*

Figure 1 shows a screenshot of the simulation model. In this model, we have five different job types that differ in a variety of attributes like due date and processing time. The actual sequence scheduling of jobs takes places in a sorter queue that is located before the first processing station. Therefore, the corresponding sequence decision rule is applied to this sorter.

For comprehensive analysis we created data sets for each decision rule with 10.000 decisions each. So in each training data set the decision rule is fixed so that we are able to investigate if certain decision rules are more difficult/less accurate to approximate than the others. The attributes of each job that decisions are built on were completely randomized on creation. Thus we created large, heterogeneous data that is conform to real world historical production data. To make our artificial data even more realistic, we added a proportion of noise and outliers to the datasets. Therefore we are able to investigate on how the data mining algorithms and the accuracy of the corresponding decision models are responding to noisy data.

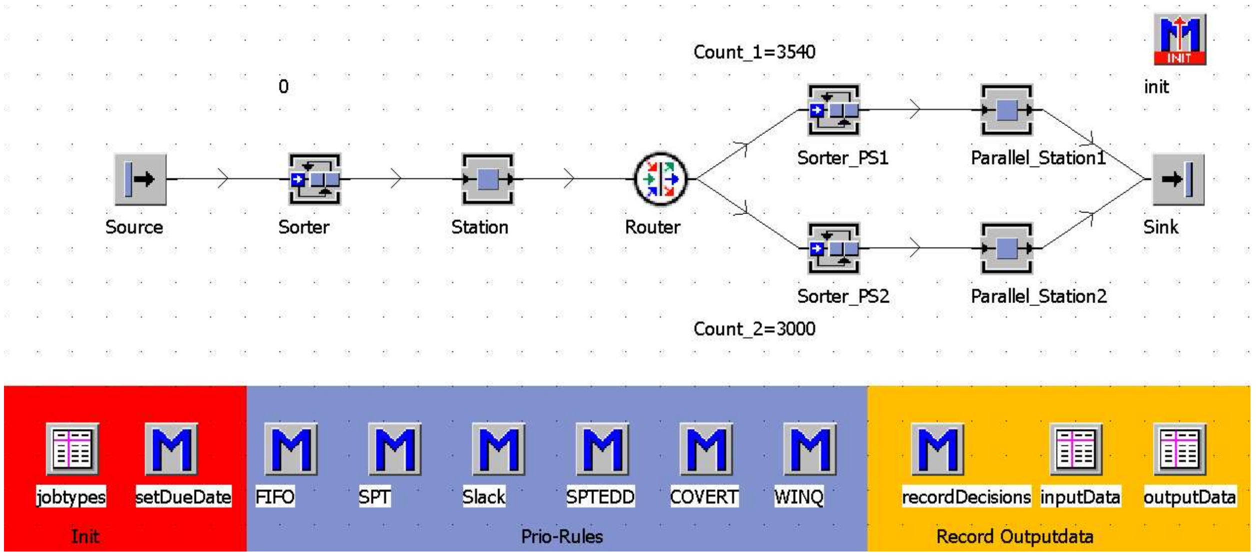

Figure 1: Screenshot of the Plant Simulation model for data generation.

We built four different validation scenarios (shown in Table 2) with different qualities of the data. Scenario A, the base scenario with the best data quality, contains no noisy or additional attributes. In Scenario B we added 10 attributes to the base scenario which are initialized with random values. Note that these additional attributes are completely irrelevant for the actual decision. Scenario C extends the base scenario by 1% outliers, which means that the value of 1 % of the attributes will be randomly changed. In scenario D, 0,5% of all attributes are modified by noise. In the context of data quality we define noise as a misclassified training data set, i.e., for 0,5 % of data sets the preferred job will be switched from A to B or accordingly changed from B to A.

Furthermore, we split the created data into a set of training data which is passed to the data mining algorithms and a set of validation data.

Table 2: Validation scenarios with different data quality settings.

Scenario	Irrelevant Attributes	Outliers	Noise
A	0	0 %	0 %
B	10	0 %	0 %
C	0	1 %	0 %
D	0	0 %	0.5 %

The next phase consist of applying data transformation methods as described above and then building the actual decision models using the presented data mining algorithms. In phase 3, we evaluate the accuracy of the decision models by passing the validation data set. The performance measure for the accuracy is called c-value, which is the percentage of correct decisions. Figure 2 summarizes the three phases of covering the test scenario.

5 RESULTS

The evaluation of results is based on the c-value (correctness), which is the percentage of correct decisions made by the decision model.

Before even comparing the different algorithms, it became clear that varying the settings and parameters of the algorithms causes huge variations in the accuracy. So the challenge was to find a setting configuration for each algorithm that produces the best accuracy in our scenario. Furthermore, c-value also

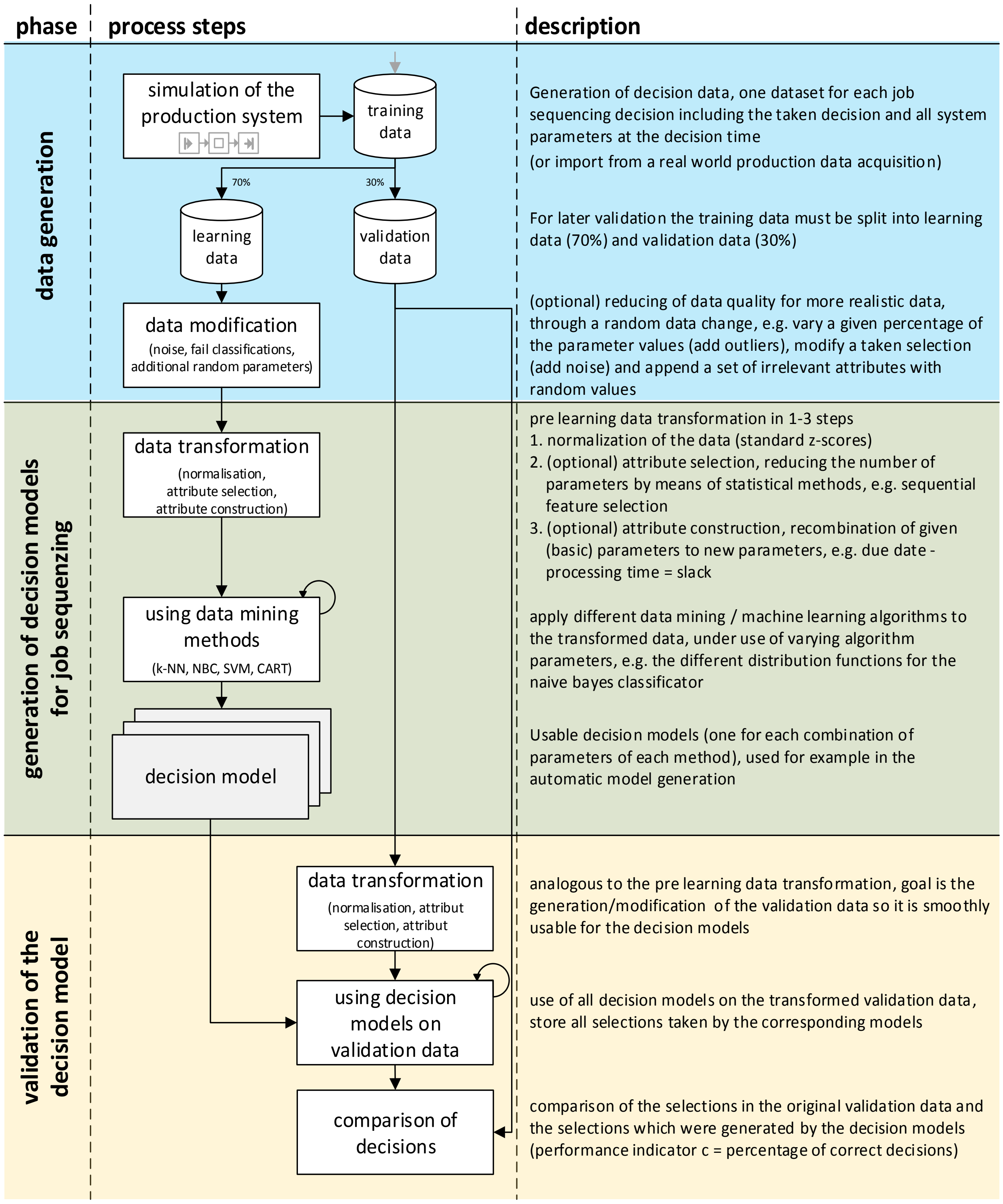
phase
process steps
description
data generation
simulation of the production system
training data
70%
30%
learning data
validation data
data modification
(noise, fail classifications, additional random parameters)
Generation of decision data, one dataset for each job sequencing decision including the taken decision and all system parameters at the decision time
(or import from a real world production data acquisition)
For later validation the training data must be split into learning data (70%) and validation data (30%)
(optional) reducing of data quality for more realistic data, through a random data change, e.g. vary a given percentage of the parameter values (add outliers), modify a taken selection (add noise) and append a set of irrelevant attributes with random values
generation of decision models for job sequenzing
data transformation
(normalisation, attribute selection, attribute construction)
using data mining methods
(k-NN, NBC, SVM, CART)
decision model
pre learning data transformation in 1-3 steps
1. normalization of the data (standard z-scores)
2. (optional) attribute selection, reducing the number of parameters by means of statistical methods, e.g. sequential feature selection
3. (optional) attribute construction, recombination of given (basic) parameters to new parameters, e.g. due date - processing time = slack
apply different data mining / machine learning algorithms to the transformed data, under use of varying algorithm parameters, e.g. the different distribution functions for the naive bayes classificator
Usable decision models (one for each combination of parameters of each method), used for example in the automatic model generation
validation of the decision model
data transformation
(normalisation, attribut selection, attribut construction)
using decision models on validation data
comparison of decisions
analogous to the pre learning data transformation, goal is the generation/modification of the validation data so it is smoothly usable for the decision models
use of all decision models on the transformed validation data, store all selections taken by the corresponding models
comparison of the selections in the original validation data and the selections which were generated by the decision models (performance indicator c = percentage of correct decisions)

Figure 2: Process of generation and validation of decision models for job sequencing using data mining methods.

varies in dependency of data quality. Exemplarily for the NBC and SVM algorithm, Figure 3 shows the huge impact that different algorithm parametrization and data quality have on the decision model's correctness. Note that c-values around 50% are worthless, especially in our two class scenario. Since the decision would always either be "prefer A" or "prefer B", a random decision would also have a 50% correctness. When having the intention to utilize an algorithm for decision approximation in an auto-generated simulation model, a c-value around 95% or higher would be adequate.

Furthermore, Figure 3 shows that regardless of parametrization, for example SVM is very sensitive to data quality. This is due to the mechanics of the algorithm, since outliers may land on the wrong side of the hyper plane, causing a very influential impact of false classified training datasets. The correctness of the NBC on the other hand is strongly depended on parametrization, to be specific on the chosen distribution function. Interestingly, the multivariate normal distribution is very sensitive to the addition of irrelevant attributes. Since NBC applies an obligatory distribution to all of the parameters, a normal distribution function fits best because in our test scenario, irrelevant attributes have been added with a normally distributed probability. NBC with an underlying normal distribution (mvmn) on the other hand performs bad when we add outliers. The 1% of outliers seem to shift the center of the normal distribution in a way that falsifies the classifications strongly, whereas the mvmn distribution is resistant to this effect.

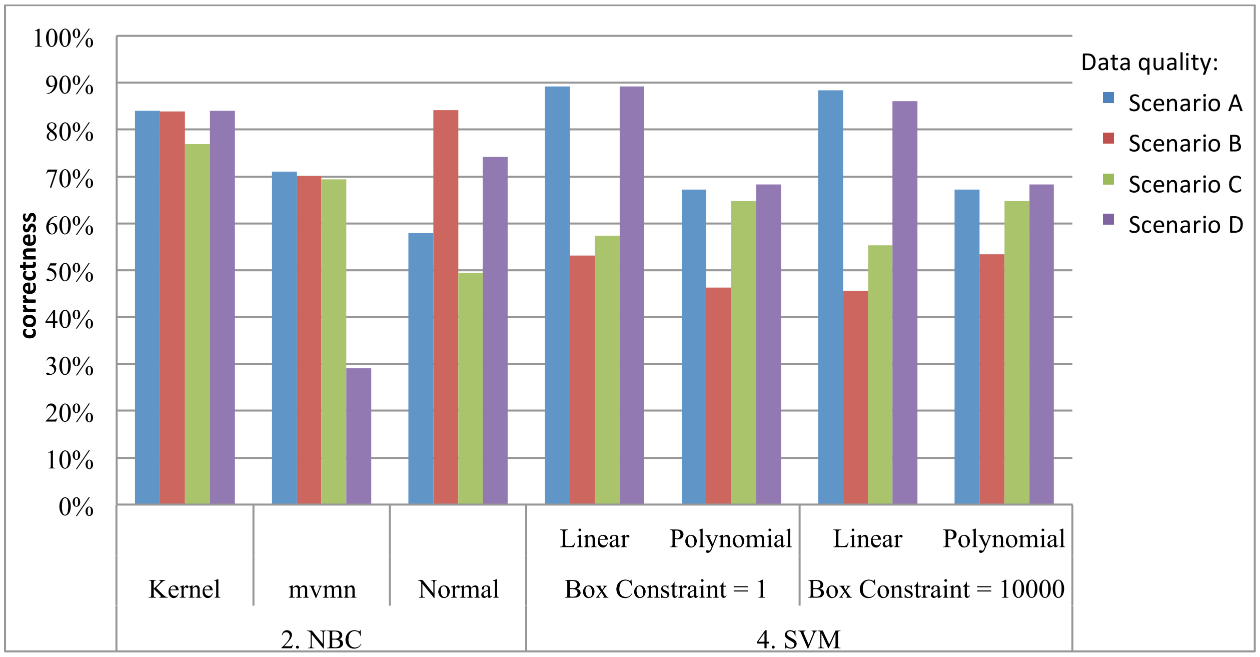

Figure 3: Comparison of the methods NBC and SVM with different parametrizations under various data quality scenarios.

For comparable benchmarks, we therefore determined the best parameter setting for each algorithm and determined the average c-value over different data quality scenarios. The results of this comparison are shown in Figure 4. The results in this figure are broken down to their actual utilized decision rule. Although the algorithms should approximate the sorters behavior regardless of the actual decision rule, some algorithms (especially NBC) show variation in their c-value depending on the decision rule of the test case.

In addition, the application of data transformation methods has an impact on the effectiveness of the investigated algorithms.

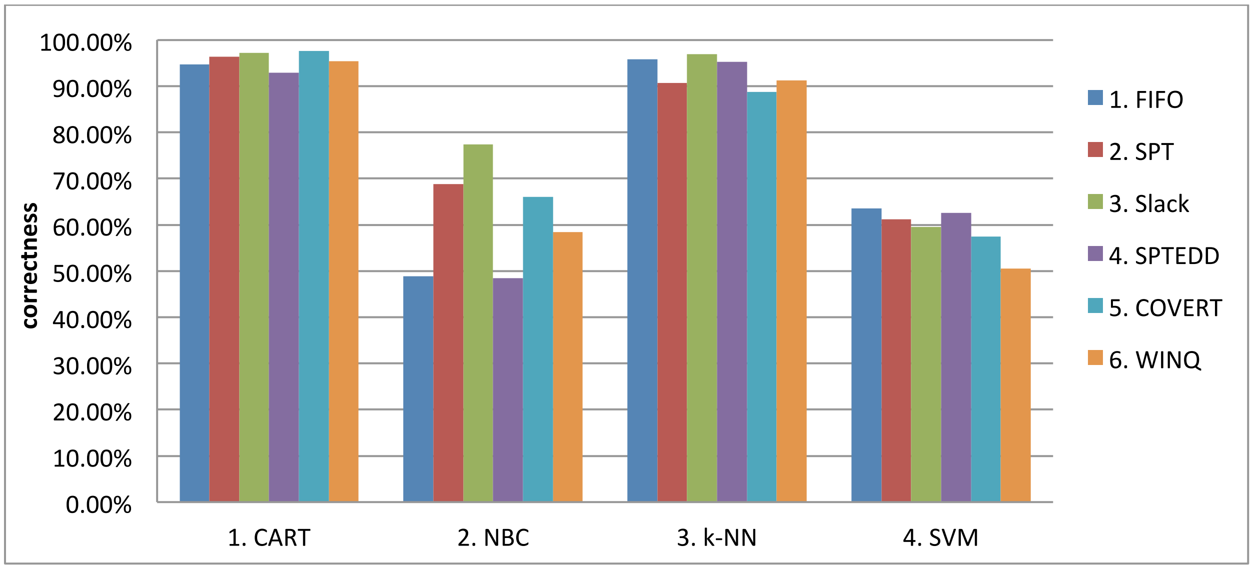

Figure 4: Comparison of the methods for different decision rules, all with best parametrization, normalization, irrelevant attributes and noise, without data transformation.

Figure 5 shows that data standardization surprisingly results in lower c-values. This is due to the fact that time dependent attributes like entry time, due date, and system clock time are continuously increasing. Recognized patterns in the training data have to be applied to increasing value ranges in the validation data, and the use of standardization seems to have a negative impact to this effect.

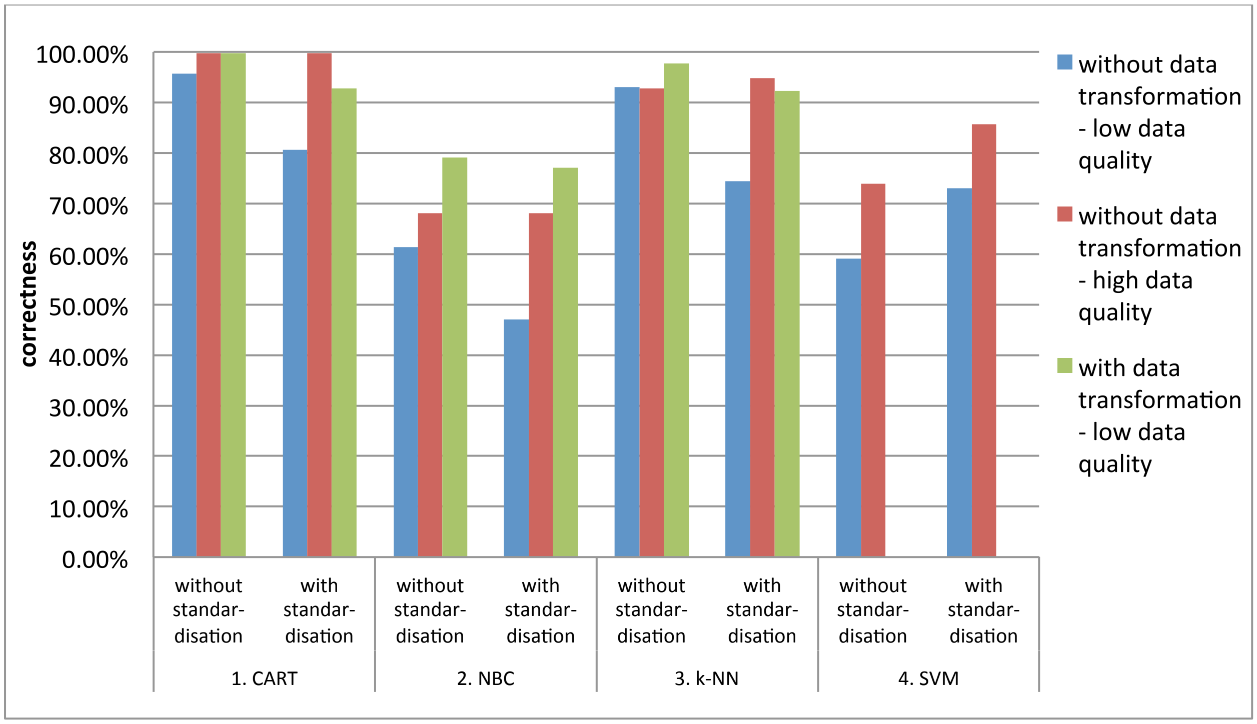

Figure 5: Comparison of the methods (best parametrization) for different data qualities, with/without data transformation and standardization.

Optimization of the depiction space through attribute selection and construction resulted in the expected positive impact. The correctness of all of the investigated algorithms could be increased. Therefore, a data transformation process with attribute selection and construction of attributes is advisable, but on the other hand computation time needed for these additional preparation steps increases also. In fact, for attribute selection, several hundreds of decision models had to be created in order to determine the optimal set of relevant attributes, which took up to 6 hours of computation time on our test systems (3.4 GHz dual core).

An exception here is the SVM algorithm in combination with data transformation. Since SVM already performed poorly in terms of runtime, reiterating this algorithm multiple times in combination with attribute creation and selection resulted in unacceptable runtime although using multiple instances of Matlab on parallel machines. So after hundred hours of runtime we canceled the computation, which is why corresponding result data (green bar) is missing in Figure 5.

In general, the results reveal that decision trees build using the CART algorithm perform best, with up to 99% correctness. The k-NN algorithms reaches similar c-values, although CART performs slightly better except for test cases utilizing the FIFO rule. Note that the K-NN algorithm is very simple and easy to implement and therefore was also quite fast both in the training and validation phases. On our test system, building a decision model with the k-NN algorithm took nearly a second, although the data transformation process may take additional computation time as described above. Therefore, we recommend both CART and k-NN for utilization in a simulation model generation task.

On the other hand NBC and SVM perform dramatically worse reaching c-values with an accuracy similar to a random decision and are therefore not usable for the task of approximating decision rules in a job sequence planning scenario.

6 SUMMARY AND DISCUSSION

We have introduced an advanced methodology for emulating dynamic behavior, based on data mining methods, which is usable directly in the simulation. We demonstrated, in a manageable scenario, that certain data mining methods in combination with an effective data transformation are usable for approximation of decision rules. Depending on the chosen data mining method and the selected settings we were able to reach quite high decision qualities (up to near 100%) under all restrictions (low data quality, additional random parameter etc.).

With that, this paper represents a step to effectively use data mining methods in the context of automatic simulation model generation. Our approaches represent a solution to the common problem of existing model generation approaches which often fail to reproduce the dynamic behavior of the real system.

It must be further examined how well this approach performs for more complex and real world scenarios. On the other hand it will be interesting to see, how the results change when the parameters of data mining methods are changed, other methods are used, or the data transformation is improved. In detail especially more methods for attribute selection and construction could be examined.

Finally, it is interesting to investigate how far and how fast individual data mining methods are capable to adapt a changing system behavior.

REFERENCES

Altman, N. S. 1992. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression." *The American Statistician* 46:175–185.

Becker, B., R. Kohavi, and D. Sommerfield. 2002. "Visualizing the Simple Baysian Classifier: Information Visualization in Data Mining and Knowledge Discovery." edited by U. Fayyad, G. G. Grinstein, and A. Wierse, 237–249, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Bergmann, S., and S. Strassburger. 2010. "Challenges for the Automatic Generation of Simulation Models for Production Systems." In *International Summer Simulation Multiconference,* 545–549, [S.l.]: Omni Press.

Bergmann, S. 2013. *Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen*, Ilmenau: Univ.-Verl.

Bergmann, S., S. Stelzer, and S. Strassburger. 2014. "On the Use of Artificial Neural Networks in Simulation-Based Manufacturing Control." *Journal of Simulation (JOS)* 8:76–90.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*, New York: Springer.

Boser, B. E., I. M. Guyon, and V. N. Vapnik. 1992. "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory,* edited by D. Haussler, 144–152.

Breiman, L. 1984. *Classification and Regression Trees*, New York: Chapman & Hall/CRC; Chapman & Hall.

El Attar, R. 2005. *Lecture Notes on Z-Transform*, [S.l.]: Lulu Com.

Fayyad, U. M., G. Piatetsky-Shapiro, and P. Smyth. 1996. "From Data Mining to Knowledge Discovery in Databases." *AI Magazine* 17:37–54.

Guyon, I., and A. Elisseeff. 2003. "An Introduction to Variable and Feature Selection." *Journal of Machine Learning Research* 3:1157–1182.

Han, J., M. Kamber, and J. Pei. 2012. *Data Mining Concepts and Techniques,* 3rd ed., Amsterdam: Elsevier/Morgan Kaufmann.

Li, X., and S. Olafsson. 2005. "Discovering Dispatching Rules Using Data Mining." *Journal of Scheduling* 8:515–527.

Milberg, J., and C. Selke. 2003. "Automatic Generation of Simulation Models for Production Planning and Control." *Annals of the German Academic Society for Production Engineering* 10:115–118.

Pinedo, M. 2012. *Scheduling. Theory, Algorithms, and Systems,* 4th ed., New York: Springer.

Reinhart, G., and T. Gyger. 2008. "Identification of Implicit Strategies in Production Control." In *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM 2008),* 302–306, Piscataway, N.J.: IEEE Xplore.

Rennie, J. D., L. Shih, J. Teevan, and D. R. Karger. 2003. "Tackling the Poor Assumptions of Naive Bayes Text Classifiers." In *Proceedings of the Twentieth International Conference on Machine Learning,* 616–623.

Selke, C. 2005. *Entwicklung von Methoden zur automatischen Simulationsmodellgenerierung*, München: Utz.

VDI. 2014. *Simulation of Systems in Materials Handling, Logistics and Production - Fundamentals,* VDI 3633-1, Last modfied 2014, Berlin: Beuth.

Wnek, J., and R. S. Michalski. 1994. "Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments." *Machine Learning* 14:139–168.

AUTHOR BIOGRAPHIES

SÖREN BERGMANN holds a Doctoral and Diploma degree in business information systems from the Ilmenau University of Technology. He is a member of the scientific staff at the Department for Industrial Information Systems. Previously he worked as corporate consultant in various projects. His research interests include generation of simulation models and automated validation of simulation models within the digital factory context. His email is soeren.bergmann@tu-ilmenau.de.

NICLAS FELDKAMP holds bachelor and master degrees in business information systems from the University of Cologne and the Ilmenau University of Technology, respectively. He is currently working as a doctoral student at the Department of Industrial Information Systems of the Ilmenau University of

Technology. His research interests include data science, business analytics, and industrial simulations. His email address is niclas.feldkamp@tu-ilmenau.de.

STEFFEN STRASSBURGER is a professor at the Ilmenau University of Technology and head of the Department for Industrial Information Systems. Previously he was head of the "Virtual Development" department at the Fraunhofer Institute in Magdeburg, Germany and a researcher at the DaimlerChrysler Research Center in Ulm, Germany. He holds a Doctoral and a Diploma degree in Computer Science from the University of Magdeburg, Germany. He is further an associate editor of the Journal of Simulation. His research interests include distributed simulation, automatic simulation model generation, and general interoperability topics within the digital factory context. He is also the Vice Chair of SISO's COTS Simulation Package Interoperability Product Support Group. His web page can be found via www.tu-ilmenau.de/wi1. His email is steffen.strassburger@tu-ilmenau.de.