

Constraint-Aware Recommendation of Complex Items

Mathias Uta^{1,2}, Alexander Felfernig² and Denis Helic²

¹Siemens Energy AG, Freyeslebenstraße 1, 91058, Erlangen, Germany

²Graz University of Technology, Rechbauerstraße 12, 8010, Graz, Austria

Abstract

In contrast to basic items such as movies, books, and songs, configurable items consist of individual subcomponents that can be combined following a predefined set of constraints. Due to the increasing size and complexity of configurable items (e.g., cars and software), a simple enumeration of all possible configurations in terms of a product catalog is not possible. Configuration systems try to identify a solution (configuration) that takes into account both, the preferences of the user and a set of constraints that defines in which way individual subcomponents are allowed to be combined. Due to time limitations, cognitive overloads, and missing domain knowledge, configurator users are in many cases not able to completely specify their preferences with regard to all relevant component properties. As a consequence, recommendation technologies need to be integrated into configurators that are able to predict the relevance of individual components for the current user. In this paper, we show how the determination of configurations can be supported by neural network based recommendation. This approach helps to predict user-relevant item properties using historical interaction data. In this context, we introduce a semantic regularization approach that helps to take into account configuration constraints within the scope of neural network learning. Furthermore, we demonstrate the applicability of our approach on the basis of an evaluation in an industrial configuration scenario (high-voltage switchgear configuration).

Keywords

Recommender systems, Knowledge representation and reasoning, Neural networks,

1. Introduction

In contrast to basic items such as books, movies, and songs, configurable items are composed of subcomponents which must be combined conform to a set of predefined constraints [1]. For reasons of combinatorial explosion, it is in many cases impossible to enumerate the individual items (configurations) in terms of a product catalog. Related example domains are *automotive* [2], *software* (e.g., configuration of operating systems) [3], and *telecommunication infrastructures* [4]. Due to the increasing size and complexity of configurable items, it becomes important to integrate recommendation algorithms into configuration processes to support users in component and/or parameter selection.

Informally, configuration can be regarded as a product design activity where the resulting item (also denoted as product or configuration) is composed of elements of a pre-defined set of basic components/parameters [1]. In this context, the chosen components must be consistent with a given set of constraints that define restrictions regarding the possible component combinations. On the

knowledge representation level, configuration problems can be defined, for example, as a constraint satisfaction problem (CSP) [5] or in terms of a rule-based representation [6]. Using CSP representations, possible combinations of individual components are defined in terms of constraints with a strict separation of domain knowledge and problem solving knowledge [1]. In contrast, in rule-based approaches product domain knowledge and problem solving knowledge are intermingled. In this paper, we use a rule-based knowledge representation which is applied in the reported application domain of *high-voltage switchgear configuration*.

Due to the increasing size and complexity of configurable items, recommendation technologies are needed, that proactively support underlying choice processes. There exist a couple of approaches that already support the recommendation of complex items. First, knowledge-based recommender systems [7] support recommendation processes on the basis of a product catalog and determine recommendations either on the basis of a set of strict selection criteria (constraints) [8] or similarity metrics [9]. The ranking of items is often implemented on the basis of a utility analysis [10] or further evaluation criteria that measure to which extent the preferences of the user are satisfied by individual decision alternatives [11, 9, 12]. Importantly, with a few exceptions [13, 14, 15], the approaches to the handling of user preferences in configuration-related scenarios do not take into account the preferences of other users but focus more on different types of decision-theoretic optimizations. An overview of existing integration approaches of recommendation

3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) & 5th Edition of Recommendation in Complex Environments (ComplexRec) Joint Workshop @ RecSys 2021, September 27–1 October 2021, Amsterdam, Netherlands

✉ mathias.uta@siemens-energy.com (M. Uta);
afflefern@ist.tugraz.at (A. Felfernig); dhelic@tugraz.at (D. Helic)

🌐 <https://felfernig.ist.tugraz.at/> (A. Felfernig)

🆔 0000-0002-1670-7508 (M. Uta); 0000-0003-0108-3146

(A. Felfernig); 0000-0003-0725-7450 (D. Helic)

© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)



technologies into configuration systems is provided a.o. in Falkner et al. [13]. Existing integrations focus on a 2-phase process where recommendations of feature settings are predetermined and then recommended to the user. In the case of inconsistent recommendations, alternative recommendations are calculated repeatedly until a consistent recommendation can be presented.

Compared to existing approaches to the integration of recommendation algorithms with configuration, we show how to take into account configuration constraints already in the learning phase and thus minimize the probability of inconsistent recommendations to be detected in the subsequent configuration phase. In this paper, we follow the idea of case-based recommendation [16] where historical configurations with similar parameter settings as those already specified by the current user are used as a basis for identifying nearest-neighbor configurations. In our work, we use such a case-based approach as a baseline version. This version is then compared with two different versions of a feed-forward neural network based configurator integration. The *first version* focuses on the prediction of configuration parameter settings relevant for the user. The *second version* follows the same goal but also takes into account the fact that recommendations should be consistent with the underlying constraint set. To support this goal, we propose a *semantic regularization* of a feed-forward (multi-class and multi-branch) neural network that is used as a configuration parameter prediction model.

The major contributions of this paper are the following. (1) we introduce a semantic regularization approach specifically useful for integrating case-based recommendation with rule-based configuration environments, (2) we compare the predictive quality of the developed approach on the basis of a real-world dataset from a complex industrial configuration task (high-voltage switchgear configuration) with regard to the evaluation criteria of prediction quality and recommendation consistency, and (3) we show how the presented results can be further generalized to be applicable for configuration scenarios beyond rule-based configuration.

The remainder of this paper is organized as follows. In Section 2, we introduce a working example in terms of a simplified configuration knowledge base from the automotive domain. In this context, we also introduce the concepts of a configuration task and a corresponding configuration. Thereafter, in Section 3, we introduce our neural network based approach to the recommendation of configuration parameter settings. In Section 4, we summarize our evaluation approach and report the results of an evaluation conducted on the basis of a real-world dataset from the domain of high-voltage switchgear configuration. The paper is concluded with an overview of future research issues (Section 5).

2. Working Example

As a basis for the following discussions on integrating neural network based predictions of user preferences, we first introduce the definition of a *configuration task* (see Definition 1).

Definition 1. A *configuration task* can be defined by a tuple (V, D, R, REQ) where $V = \{v_1, v_2, \dots, v_n\}$ is a set of finite domain variables, $D = \{dom(v_1), dom(v_2), \dots, dom(v_n)\}$ is a set of corresponding domain definitions, and $R = \{r_1, r_2, \dots, r_m\}$ is a set of rules that define how a configuration can be derived from a given set of customer requirements $REQ = \{v_\alpha = val_\alpha, \dots, v_\gamma = val_\gamma\}$ where elements of REQ are regarded as variable value assignments.

A simple example of a configuration task definition is the following (see Example 1) where *pdcc* represents a *park distance control* feature and *fuel* represents *fuel consumption* in *gallons/100miles*.

Example 1: Configuration Task.

- $V = \{type, pdc, fuel, skibag, 4-wheel, color\}$
- $D = \{dom(type) = \{city, limo, combi, xdrive\}, dom(pdc) = \{yes, no\}, dom(fuel) = \{1.7, 2.6, 4.2\}, dom(skibag) = \{yes, no\}, dom(4-wheel) = \{yes, no\}, dom(color) = \{red, blue\}\}$
- $R = \{r_1 : 4-wheel = yes \rightarrow type = xdrive, r_2 : skibag = yes \rightarrow type \neq city, r_3 : fuel = 1.7 \rightarrow type = city, r_4 : fuel = 2.6 \wedge type = xdrive \rightarrow false, r_5 : type = combi \rightarrow skibag = yes, r_6 : type = limo \rightarrow pdc = yes\}$
- $REQ = \{type = city, pdc = yes, fuel = 1.7\}$

Given the definition of a configuration task (V, D, R, REQ) , we are able to introduce the definition of a corresponding configuration (solution for a configuration task) – see Definition 2.

Definition 2. A configuration for a given configuration task definition (V, D, R, REQ) is a set of variable assignments $CONF = \{v_1 = val_1, \dots, v_n = val_n\}$ where $\forall \{v_i = val_i\} \subseteq CONF : val_i \in dom(v_i)$ and *consistent*($CONF \cup R \cup REQ$). A configuration is *complete* if each variable in V has an assignment in $CONF$.

An example configuration $CONF$ for the configuration task of Example 1 is the following (see Example 2).

Example 2. $CONF = \{type = city, pdc = yes, fuel = 1.7, skibag = no, 4-wheel = no, color = red\}$

We regard a configuration as *complete* if each of the variables in V is associated with a corresponding value assignment and these assignments are consistent with the rules in R . As already mentioned, in many configuration scenarios users are not able or do not want to specify values for all the defined variables in V but are interested in recommendations that help to more easily complete a

Table 1

A simple example of a collection of already completed configuration sessions (one hot encoding). The abbreviation *pd* denotes a *park distance control* feature. Furthermore, *fuel* denotes the *fuel consumption* in gallons/100 miles. Finally, session *current* is an ongoing session where variable settings for $\{skibag, 4-wheel, color\}$ should be recommended.

attribute	type				pd		fuel			skibag		4-wheel		color		
	session	city	limo	combi	xdrive	yes	no	1.7	2.6	4.2	yes	no	yes	no	red	blue
1	1	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0
2	0	0	0	0	1	1	0	0	0	1	0	1	1	0	0	1
3	0	1	0	0	0	1	0	0	1	0	0	1	0	1	0	1
current	0	0	1	0	0	0	1	0	1	0	?	?	?	?	?	?

configuration session [13]. We now introduce a definition of a recommendation in the context of a configuration task (see Definition 3).

Definition 3. Given the definition of a configuration task (V, D, R, REQ) , a corresponding recommendation $REC = \{v_\beta = val_\beta, \dots, v_\delta = val_\delta\}$ is a set of variable value assignments of $v_i \in V$. A recommendation REC is consistent if $REC \cup REQ \cup R$ is consistent, i.e., a solution can be found.

Example 3. $REC = \{skibag = no, 4-wheel = no, color = red\}$

Following the approach of case-based reasoning [16], it can be the case that recommended variable value assignments are *inconsistent* with the already defined user requirements and the rules (constraints) defined in the knowledge base. This is the case if recommendations are determined from already completed configuration sessions without taking into account configuration constraints (rules in R). In the following, we provide a simple example of a case-based reasoning approach and then focus on how to take into account configuration rules in terms of a *semantic regularization* when optimizing a neural network responsible for recommending variable settings.

3. Recommending Configurable Items

As already sketched in the previous section, recommendations in the context of configuration scenarios are represented by a set of attribute assignments, i.e., a recommendation could include a single attribute setting but also numerous settings recommended at the same time. In this section, we discuss different approaches to recommend variable value settings in the context of knowledge-based configuration scenarios.

Case-based Recommendation Table 1 represents a simple example of a set of already completed configurations that can be used as a basis for determining recommendations. In this example, configuration sessions

1–3 have already been completed. The *current* session is ongoing and we are interested in a recommendation for the variables *skibag*, *4-wheel*, and *color*. For the purposes of this example and also for discussing the neural network based recommendation approach, we apply a *one hot encoding* of the configuration variables, for example, in *session 1*, the configured car is of type *city*. In the scenario shown in Table 1, a simple case-based reasoning recommender would search for one or more nearest neighbors (NN) and recommend the variable settings that were chosen most often by the nearest neighbors. In our example, the nearest neighbor (session) of the *current* session is *session 3* (in terms of the number of equivalent variable values). If we assume $|NN|=1$, we would recommend $REC = \{skibag = no, 4-wheel = no, color = blue\}$ to the user in the *current* session (if we intend to predict all unspecified variable values at the same time).

Importantly, since the *current* user is interested in a car of type *combi* which requires the inclusion of a *skibag* (see Example 1), such a recommendation (REC) induces an inconsistency between the user requirements and the configuration knowledge base (the set of rules R). A traditional approach to deal with such a situation is to test the next recommendation for consistency and do this until a consistent recommendation could be found [13]. Our approach (that will be introduced in the following) to deal with such a situation is to introduce a *semantic regularization* into the neural network learning phase which helps to avoid inconsistent recommendations as far as possible.

Neural Network based Recommendation Our basic approach to recommend variable values in the context of rule-based configuration is based on the feed-forward neural network structure depicted in Figure 1. In such networks, the *input layer* consists of possible values (represented in terms of a one hot encoding) that have already been specified by a user. For example, the variable values (preferences) that have already been specified by the user in session *current* are $\{type = combi, pd = no, fuel = 2.6\}$. Networks as those depicted in Figure 1 can be trained in a domain-dependent fashion on the basis of a dataset

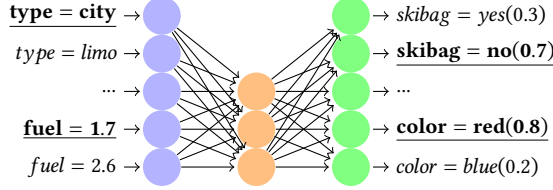


Figure 1: A simple neural network architecture with an *input layer* representing specified (e.g., *type* and *fuel*) and un-specified (e.g., *skibag* and *color*) variables, one *hidden layer*, and an *output layer* that helps to estimate variable values of relevance.

comprised of already completed configuration sessions (see Sessions 1–3 in Table 1). Furthermore, the *hidden layer* is used for learning dependencies between input values selected by the user and corresponding variable values of potential relevance for the user. The number of nodes in the hidden layer is regarded as hyper-parameter to be optimized in an item domain dependent fashion (in [17] an equal amount of neurons in the input layer and the hidden layer showed the best performance). Finally, the *output layer* supports a multi-branch approach (one branch per variable) where each branch b is splitted into bo output nodes representing the different domain values of variable v_b . In contrast to the input and hidden layer which use a *ReLU* activation function, classification is implemented using *softmax*. The choice of the training hyper-parameters has been made based on several test executions. Optimizer “Adam” [18] has shown the best performance compared to other gradient decent optimizers like “ADAGRAD”, “RMSProp” and “SGD” [19]. “Adam” uses adaptive estimation of first order and second order moments, which slows down the adjustment of neuron weights the more steps have been done. The selected parameters for the “Adam” optimizer were an initial learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Please note that this network architecture assumes categorical variables (e.g., similar to our Example 1) – other variable types require preprocessing such as binning or alternative architectures. Our neural network derived from the knowledge base introduced in Section 2 consists of 6 output nodes and also 9 input nodes (assuming the example from above).

In the basic version of our approach, neural networks are trained on the basis of training dataset (see, e.g., sessions 1–3 in Table 1). The usage scenario of this basic neural network approach is the following: if a user interacts with a configurator and has already specified a set of initial requirements (*REQ*), the neural network can be exploited for the recommendation of variable values. Since the basic version of the neural network can only learn constraints/rules from the available set of completed configurations, it can be the case that predictions induce an inconsistency with the underlying rule set.

Constraint-Aware Recommendation For reasons of potentially inconsistent recommendations, we have introduced an enhanced neural network learning phase including a *semantic regularization* where inconsistent recommendations are taken into account as regularization term. In other words, although parts of the domain-specific rules/constraints can be learned from the underlying training dataset, it can be the case that some or even many constraints are neglected and the resulting variable value recommendations induce an inconsistency. We denote this approach as *constraint-aware neural networks* which are extremely relevant in recommendation scenarios where domain-specific constraints/rules have to be taken into account by the recommender. To reduce the probability of inconsistent variable value recommendations, knowledge base rules/constraints are taken into account in the learning process. This goal is achieved by integrating the results of a consistency check of the proposed recommendation *REC* (more precisely, $REQ \cup R \cup REC$) into a corresponding loss function as shown in Formula 1.

$$L(\theta) \leftarrow L_{train}(\theta) + \Omega(\theta) + \mu \times \Pi(\theta) \quad (1)$$

In this context, $L(\theta)$ denotes a loss function on the vector θ of weights in the neural network, $L_{train}(\theta)$ denotes the prediction loss, $\Omega(\theta)$ represents a corresponding L_2 regularization term, μ represents a hyper-parameter that controls the impact of an inconsistency on the overall loss, and $\Pi(\theta)$ indicates whether the recommendation resulting from θ is consistent (0 is returned) or inconsistent (1 is returned). As $\Pi(\theta)$ is a discrete non-differentiable function, the optimization of our loss function has to resort to approximation of gradients via computation of finite differences.

User Interaction and Knowledge Representation

Our approach to neural network based variable value recommendation for knowledge-based configuration helps to reduce the probability of inconsistency-inducing recommendations (see Section 4) and thus also can help to make configuration processes less time-consuming for users. The proposed recommendation approach is flexible in the sense that recommendations for single-variable assignments as well as combined variable assignment

recommendations can be supported. In our working example, we did not take into account settings, where a configuration task is organized in phases where in each phase a specific subcomponent of the product is configured (e.g., software configuration as part of the configuration of a whole computer). On the user interface level, recommendations are mostly related to variables within a specific phase. However, recommendations can also be determined on the basis of existing variable settings from different phases.

Recommendation Consistency The achievable degree of recommendation consistency also depends on the used knowledge representation. In the case of a rule-based knowledge representation [6], it is not always feasible to correctly predict if it is possible to complete a partial configuration, i.e., given a (consistent) set of customer requirements, is it possible to find assignments for the remaining variables in such a way that a consistent and complete configuration can be achieved. If a more compact representation of all satisfiable variable assignments is available [20], our approach can be applied to recommend the most relevant option among the consistent ones. In a similar fashion, constraint-based approaches [5] can be applied to infer remaining variable assignments that are still consistent with *REQ* and *R*. In the following, we present the results of an empirical analysis of our constraint-aware recommendation approach using a real-world dataset from the domain of *high-voltage switchgear configuration*.

4. Evaluation

The configurator application for the *high-voltage switchgear domain* has been developed with the goal to reduce engineering effort during the offering stage of these highly complex systems. The underlying dataset includes $N = 720$ complete configurations developed by skilled sales employees from *Siemens Energy AG*. Each entry of the dataset consists of $M = 60$ attribute settings (assignments), i.e., each configuration is described by 60 variables (representing features and subcomponents). In this context, 10 out of the 60 variables have been defined as basic switchgear features which are assumed to be selected by the user before a recommendation can be triggered (e.g., *basic switchgear category* to be installed). The *focus of recommendation* are the remaining 50 more specific features with a sometimes lower degree of understandability where it is often an issue for users to find good or even optimal settings (e.g., *AC supply voltage*). The dataset is composed of consistent configurations that have been built on the basis of a rule-based configuration system.¹ As a baseline in our evaluation, we have de-

¹www.camos.de.

veloped a case-based reasoning approach (see Section 3) that recommends variable value settings on the basis of the preferences of the N nearest neighbors (in the given setting, $N = 1$ achieved the highest prediction quality). Furthermore, the two versions of the neural network based approach have been implemented on the basis of the Keras API [21].² The learning of the neural network model is based on 32 iterations during the learning phase of the model where 80% of the data is used for training purposes and 20% for testing. The *first version* of the neural network model has been trained without taking into account the rules in the configuration knowledge base whereas the learning of the model for the *second version* is based on the *loss function* included in Formula 1. For validation of the models they have been applied separately to 20 configurations which were not part of the training or testing data.

Prediction Quality Our first goal was to analyze the prediction quality of the three variable value recommendation approaches discussed in this paper: (1) case-based recommendation, (2) neural network based recommendation, and (3) neural network based recommendation with semantic regularization. To measure the prediction quality, precision has been chosen as the key performance indicator. The precision of the recommendation of a configuration A_{Rec} has been measured in terms of the share of predictions part of the configuration accepted by the user A in relation to the total number of predictions contained in A_{Rec} , see equation 2. In the context of our evaluation, we were specifically interested in the predictive performance depending on the number of already known attribute values. The prediction task was specified in such a way that given a chosen set of input attributes (the known settings representing *REQ*), the task was to predict all other missing attributes to complete the configuration. Since our configurator is organized in 5 configuration phases, the phase number of a *to be predicted* variable had to be equal to the number of the current configuration phase and the phase number of known variables (*REQ*) had to be lower or equal to the phase number of the *to be predicted* variable. Consequently, the missing attributes were iterative predicted by selecting in each iteration the values for those attributes part of the configuration phase with the lowest phase number. In the following iteration the previously predicted variables have been utilized as known variables for predicting the variables of the next configuration phase. This has been repeated till the configuration was complete.

$$\text{configuration recommendation precision} = \frac{|A_{Rec} \cap A|}{|A_{Rec}|} \quad (2)$$

²<https://github.com/MaUt89/ConLearn>

Figure 2 provides an overview of the outcome of our evaluation by averaging the precision achieved for each of the 20 validation configurations applied to the models. The results clearly indicate the potential of prediction quality improvements that can be achieved by the inclusion of semantic regularization concepts. Whereas, (1) case-based recommendation achieves only a precision of 60.5% when starting with ten initially specified variable values, the neural network based recommendation approaches predict the variable values with 74% (2) and 76.33% (3) precision. Noticeable is that, the neural network based recommendation with semantic regularization (3) outperforms the approach without semantic regularization (2) in every validation scenario. Finally, with 50 out of 60 variable values given as an input for the configuration both neural network based approaches reach a precision of 100%.

Consistency We were also interested in the degree of consistency of the determined recommendations REC , i.e., $\text{consistent}(REC \cup REQ \cup R)$. The consistency of the recommendation of a configuration A_{Rec} has been measured in terms of the share of knowledge base consistent predictions A^* part of the recommendation in relation to the total number of predictions contained in A_{Rec} , see equation 3. As can be seen in Figure 3, the semantic regularization helps to decrease the inconsistency degree of recommendations (compared to the CBR and the basic neural network based approach). Starting with a consistency of 95.27% (ten variables values already known) the neural network based approach with semantic regularization (3) reaches already with 30 initially known variable values a consistency of 100%. Both other approaches (1) and (2) achieve poorer results with ten initially given variable values 93.59% (1) and 94.88% (2) and reach a consistency of 100% not until 40 variables are initially specified. All in all, the higher consistency of approach (3) including the semantic regularization has been expected since this approach is penalizing non-consistent predictions during the learning phase of the model. Nevertheless, the impact could have been higher and the consistency especially with a low number of initially known variable values is improvable. To achieve this an optimization of the hyper-parameter μ is desirable and part of future work.

$$\text{configuration recommendation consistency} = \frac{|A^* \cap A_{Rec}|}{|A_{Rec}|} \quad (3)$$

5. Conclusions and Future Work

We have introduced an approach to the integration of recommendation features into knowledge-based configu-

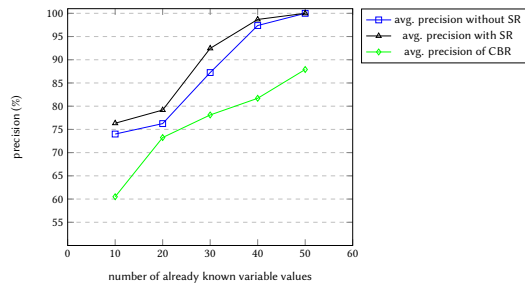


Figure 2: Precision of high voltage switchgear related predictions (SR = semantic regularization, CBR = case-based reasoning).

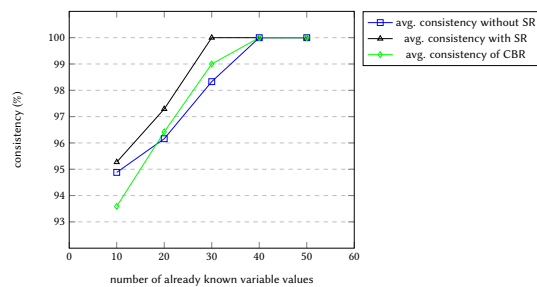


Figure 3: Consistency of high voltage switchgear related predictions (SR = semantic regularization, CBR = case-based reasoning).

rators and compared it with *case-based and basic neural network based recommendation*. To improve the prediction quality and consistency of recommendations, we have introduced a semantic regularization approach that helps to further increase the consistency degree of recommendations especially in the context of rule-based configuration scenarios. The presented approach has been integrated into a industrial rule-based configuration environment that focuses on the configuration of high-voltage switchgears. The presented approach can increase both, prediction quality and consistency of the determined recommendations. Furthermore, the approach is generalizable to other types of configuration knowledge representations such as constraint satisfaction problems (CSPs).

Future work will focus on the integration of the developed concepts into *model-based configuration knowledge representations* such as CSPs [5]. Furthermore, we will extend the scope of considered machine learning approaches a.o. with an integration of matrix factorization based variable value prediction. The dataset size used for the evaluation presented in this paper can be considered as a limitation of this work, in particular w.r.t. neural network models. A major focus of future work will be the evaluation of our approach with larger industrial configuration datasets. The neural network based prediction models will also be evaluated for their applicability in

the context of diagnosis scenarios, i.e., scenarios where users receive recommendations regarding requirements changes that help to get out from an inconsistent situation. Also, we plan to investigate alternative formulations of the optimization problem, for example, with consistency conditions being (partially) defined as optimization constraints. Finally, although already integrated into the configuration environment of Siemens Energy, the evaluation of the proposed recommendation approach will be further extended especially with regard to the quality of the user interface and the need of additional explanations for the proposed recommendations.

References

- [1] M. Stumptner, An overview of knowledge-based configuration, *AI Communications* 10 (1997) 111–125.
- [2] J. Landahl, D. Bergsjö, H. Johannesson, Future Alternatives for Automotive Configuration Management, *Procedia Computer Science* 28 (2014) 103–110.
- [3] J. Sincero, W. Schröder-Preikschat, The linux kernel configurator as a feature modeling tool, in: Workshop on Analyses of Software Product Lines, ASPL, Limerick, Ireland, 2008, pp. 257–260.
- [4] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, M. Stumptner, Configuring large systems using generative constraint satisfaction, *IEEE Intelligent Systems* 13 (1998) 59–68.
- [5] E. Tsang, *Foundations of Constraint Satisfaction, Computation in cognitive science*, Academic Press, London, UK, 1993.
- [6] D. Dhungana, C. Tang, C. Weidenbach, P. Wischniewski, Automated Verification of Interactive Rule-based Configuration Systems, in: 28th IEEE/ACM Intl. Conference on Automated Software Engineering, IEEE, Silicon Valley, CA, USA, 2013, pp. 551–561.
- [7] R. Burke, Knowledge-based recommender systems, *Encyclopedia of Library and Information Systems* 69 (2000) 180–200.
- [8] A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, Constraint-based Recommender Systems, in: *Recommender Systems Handbook*, 2 ed., Springer, Boston, MA, 2015, pp. 161–190.
- [9] L. Chen, P. Pu, Critiquing-based Recommenders: Survey and Emerging Trends, *UMUAI* 22 (2012) 125–150.
- [10] D. Winterfeldt, W. Edwards, *Decision Analysis and Behavioral Research*, Cambridge University Press, Cambridge, England, 1986.
- [11] C. Boutilier, R. Patrascu, P. Poupart, D. Schuurmans, Constraint-based optimization and utility elicitation using the minimax decision criterion, *Artificial Intelligence* 170 (2006) 686–713.
- [12] J. Goldsmith, U. Junker, Preference handling for artificial intelligence, *AI Magazine* 29 (2008) 9–12.
- [13] A. Falkner, A. Felfernig, A. Haag, Recommendation technologies for configurable products, *AI Magazine* 32 (2011) 99–108.
- [14] M. Zanker, A Collaborative Constraint-based Meta-level Recommender, in: *ACM RecSys*, ACM, Lausanne, Switzerland, 2008, pp. 139–146.
- [15] D. Jannach, L. Kalabis, Incremental prediction of configurator input values based on association rules – a case study, in: *Proceedings Workshop on Configuration*, 2011, pp. 32–35.
- [16] B. Smyth, Case-Based Recommendation, in: *The Adaptive Web*, volume 4321 of *LNCS*, Springer, Berlin, Heidelberg, 2007, pp. 342–376.
- [17] M. Uta, A. Felfernig, Towards machine learning based configuration, in: C. Forza, L. Hvam, A. Felfernig (Eds.), *22nd International Configuration Workshop*, 2020, pp. 25–28.
- [18] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [19] S. Ruder, An overview of gradient descent optimization algorithms, *CoRR* abs/1609.04747 (2016). URL: <http://arxiv.org/abs/1609.04747>.
- [20] H. Andersen, H. Hulgaard, Boolean Expression Diagrams, in: *12th IEEE Symp. on Logic in Computer Science*, IEEE, Warsaw, Poland, 1997, pp. 88–98.
- [21] F. Chollet, Keras, <https://github.com/fchollet/keras>, 2015.