Analysis Operations for Constraint-based Recommender Systems

Sebastian Lubos
Institute of Software Technology, Graz University of Technology
Graz, Austria
slubos@ist.tugraz.at

Viet-Man Le
Institute of Software Technology, Graz University of Technology
Graz, Austria
vietman.le@ist.tugraz.at

Alexander Felfernig
Institute of Software Technology, Graz University of Technology
Graz, Austria
alexander.felfernig@ist.tugraz.at

Thi Ngoc Trang Tran
Institute of Software Technology, Graz University of Technology
Graz, Austria
ttrang@ist.tugraz.at

ABSTRACT

Constraint-based recommender systems support users in the identification of complex items such as financial services and digital cameras (digicams). Such recommender systems enable users to find an appropriate item within the scope of a conversational process. In this context, relevant items are determined by matching user preferences with a corresponding product (item) assortment on the basis of a pre-defined set of constraints. The development and maintenance of constraint-based recommenders is often an error-prone activity – specifically with regard to the scoping of the offered item assortment. In this paper, we propose a set of offline analysis operations (metrics) that provide insights to assess the quality of a constraint-based recommender system before the system is deployed for productive use. The operations include a.o. automated analysis of feature restrictiveness and item (product) accessibility. We analyze usage scenarios of the proposed analysis operations on the basis of a simplified example digicam recommender.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Social and professional topics** → **Quality assurance**; • **General and reference** → **Metrics**.

KEYWORDS

Constraint-based recommender systems, evaluating recommender systems, evaluation metrics

ACM Reference Format:
Sebastian Lubos, Viet-Man Le, Alexander Felfernig, and Thi Ngoc Trang Tran. 2023. Analysis Operations for Constraint-based Recommender Systems. In *Seventeenth ACM Conference on Recommender Systems (RecSys '23), September 18–22, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3604915.3608819

RecSys '23, September 18–22, 2023, Singapore, Singapore

ACM ISBN 979-8-4007-0241-9/23/09.
https://doi.org/10.1145/3604915.3608819

1 INTRODUCTION

Constraint-based recommender systems support users in decision processes related to complex items such as financial services and digital cameras (digicams) [4, 15]. In contrast to recommendation approaches such as collaborative filtering [2] and content-based filtering [12], constraint-based recommendation is based on a predefined set of constraints that describe relationships between user preferences and the offered item assortment. User preferences are often elicited within the scope of a conversational process where users define and adapt their preferences depending on the items proposed by the system. For example, in the digicam domain, user preferences can define the main *usage* (e.g., photography of "fastmoves" in sports scenarios) and the importance of *ease of transportation*. A corresponding item catalog represents the set of offered digicams that can be recommended to the user. Selecting items for specific users is the task of the recommendation knowledge represented in terms of a set of constraints [4].

Developing constraint-based recommender systems is often an error-prone process due to cognitive overloads of knowledge engineers, missing domain knowledge, and outdated item and constraint knowledge [5]. In order to avoid the deployment of recommender systems that are based on suboptimal recommendation knowledge, we propose a set of (offline) analysis operations that help to detect issues even before a constraint-based recommender system is deployed for productive use. With this, we provide metrics that help to evaluate the quality of a recommender system with regard to a set of basic but fundamental properties, for example, for each potential user preference (represented by a variable assignment/constraint), it should be possible to recommend an item that supports this preference.

Existing approaches to evaluate the quality of a recommender system [7, 14, 16, 17] are based on (1) data-driven studies focusing on primarily analyzing the prediction quality of a recommender system (within the scope of offline experiments), (2) experimental settings with prototype systems which focus, for example, on an evaluation of different user interfaces, explanation types, and algorithmic approaches, and finally (3) field studies which often focus on the evaluation of a recommender system in a real-world setting a.o. on the basis of A/B testing. In contrast to existing evaluation approaches, we focus on analysis operations that do not require

the deployment of a recommender system in an evaluative setting or the evaluation of the system on the basis of datasets.

Providing such analysis operations is highly relevant, for example, for marketing and sales departments in charge of item set scoping (assortment optimization) [8, 10], i.e., taking strategic decisions regarding the inclusion (exclusion) of specific item features and items. Examples of such analysis operations are the following: (1) in a digicam recommender system, we are interested a.o. in the *accessibility* of individual digicams, i.e., how "easy" it is for a digicam to be part of a set of recommended items. Digicams with low accessibility could have a negative impact on sales statistics since users rarely or – in the worst case – never see the digicam. Another example is the *restrictiveness* of specific user preferences. For example, if a *watertight* feature is supported by only a minority of the offered digicams, the potential user preference of "including watertight" is too restrictive and could lead to situations where user preferences cannot be supported. This effect could be intended – if unintended, corresponding adaptations are needed, for example, in terms of extending the offered item assortment or adapting/extending item descriptions and constraints.

In this context, the major contributions of our paper are the following: (1) we propose a set of basic analysis operations (metrics) which help to evaluate the intended properties of constraint-based recommenders and thus also support item set analysis and scoping processes, (2) we extend the landscape of recommender system evaluation approaches with an *offline view* that can be used without the need of setting up user studies or providing datasets to evaluate the predictive quality of a recommender system, (3) we demonstrate the application of the proposed analysis operations on the basis of a working example from the domain of digicams.

The remainder of our paper is organized as follows. In Section 2, we introduce the concepts of a constraint-based recommendation task and a corresponding constraint-based recommendation. In Section 3, we introduce analysis operations specifically relevant to constraint-based recommendation scenarios. Thereafter, we analyze threats to validity in Section 4. Finally, we conclude the paper with a discussion of open research issues (Section 5).

2 CONSTRAINT-BASED RECOMMENDATION

The overall idea of constraint-based recommendation [4, 15] can be summarized as follows. Users specify their preferences (requirements R) with regard to a set F of pre-defined product (item) features which – combined with a corresponding set of feature constraints FC define the user preference space. In addition, an item assortment (product list I) defines the set of available items where individual items are described in terms of a set of item attributes (IA). Since a constraint-based recommendation task is defined in terms of a constraint satisfaction problem (CSP) [4, 13], the item assortment is described in terms of item constraint(s) (IC). Importantly, an additional set of constraints (so-called filter constraints $FILT$) describes relationships between potential user preferences and corresponding relevant items. Consequently, filter constraints are responsible for figuring out those items which can be regarded as recommendation candidates, i.e., items of potential relevance for the user.

We now introduce an example preference space represented by a feature model (FM) [1] from the domain of digicam recommendation (see Figure 1). This FM represents the customer preference space with regard to the offered assortment of digicams $\{i_1..i_{10}\}$ (see Table 1), i.e., it defines the variants (feature combinations) that can be selected by a customer. In this example, each customer (user) has to specify his/her main usage of the digicam (represented as a mandatory relationship between the features *digicam* and *usage*). The specified main usage has to be one (or more) out of *buildings*, *people*, and *fastmoves*. Furthermore, the user can specify his/her preference regarding the digicam weight (*easycarry* digicam preferred or not), the need for the inclusion of an exchangeable lens (feature *flexiblelens*), and the requirement that the recommended digicam is *watertight*. Finally, the FM includes an additional constraint specifying that if users want to focus on *fastmoves* photography, only digicams supporting the feature *watertight* should be recommended.

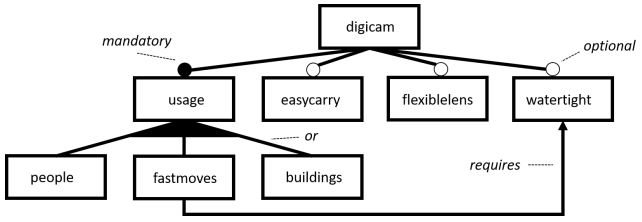

Figure 1: Example feature model (FM) which defines the user preference space regarding a digicam assortment.

The user preferences (requirements) R, the user preference space (defined by an FM as in Figure 1), and the corresponding item assortment (including recommendation constraints) can be translated into a constraint satisfaction problem (CSP) [4, 13] which allows reasoning regarding user-relevant item recommendations. A *constraint-based recommendation task* (CBR-task) can be defined as a specific type of Constraint Satisfaction Problem (CSP) (see Definition 1).

Definition 1 (CBR-task). A constraint-based recommendation task is a tuple (F, C, I, IA, IC, R) where $F = \{f_1..f_n\}$ is a set of Boolean-domain features describing user preferences and $C = FC \cup FILT$ where $FC = \{c_1..c_l\}$ is a set of relationships and cross-tree constraints in the feature model (FM) and $FILT = \{c_{l+1}..c_m\}$ defines constraints between features and product properties.[1] Furthermore, I represents the available items (the item assortment), $IA = \{ia_1..ia_r\}$ the set of finite domain type item attributes (properties), and IC a constraint in disjunctive normal form describing the item assortment. Finally, $R = \{req_1..req_q\}$ is a set of customer requirements defining intended feature inclusions.

Example 1 (Digicam CBR-task). On the basis of our working example, we can now define a CBR-task (F, C, I, IA, IC, R) where we assume $R = \{req_1 : fastmoves = true\}$ (the user is interested in fastmoves photography). We also assume that each $ia_k \in IA$ describes an item property, for example, *id* represents the identifier of an item and *resolution* describes a digicam resolution. All items of our working example, i.e., $i_1..i_{10}$, are specified in Table 1.

[1] Such constraints are also denoted as filter constraints [4].

- F ={f_1 : $digicam$, f_2 : $usage$, f_3 : $buildings$, f_4 : $fastmoves$, f_5 : $people$, f_6 : $easycarry$, f_7 : $flexiblelens$, f_8 : $watertight$}.
- FC = {c_1 : $digicam = true$, c_2 : $usage = true \leftrightarrow digicam = true$, c_3 : $usage = true \leftrightarrow buildings = true \vee fastmoves = true \vee people = true$, c_4 : $easycarry = true \rightarrow digicam = true$, c_5 : $flexiblelens = true \rightarrow digicam = true$, c_6 : $watertight = true \rightarrow digicam = true$, c_7 : $fastmoves = true \rightarrow watertight = true$}.
- $FILT$ = {c_8 : $buildings = true \rightarrow accumulator = h$, c_9 : $people = true \rightarrow resolution = h$, c_{10} : $fastmoves = true \rightarrow fps = h$, c_{11} : $easycarry = true \rightarrow weight = l$, c_{12} : $flexiblelens = true \rightarrow mount \neq no$, c_{13} : $watertight = true \rightarrow isolation = h$}.
- I = {$i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9, i_{10}$}.
- IA ={id, $accumulator$, $resolution$, fps, $weight$, $mount$, $isolation$}.
- IC = {($id = i_1 \wedge accumulator = h \wedge resolution = h \wedge fps = h \wedge weight = h \wedge mount = t1 \wedge isolation = h$) $\vee$ .. $\vee$ ($id = i_{10} \wedge accumulator = h \wedge resolution = l \wedge fps = a \wedge weight = a \wedge mount = t2 \wedge isolation = l$)}.
- R = {req_1 : $fastmoves = true$}.

Table 1: Example item table I. Items $i_j \in I$ are described in terms of their technical properties (id (*item id*), $accumulator$, $resolution$, fps (*frames per second*), $weight$, $mount$, $isolation$) using the abbreviations {$\underline{h}igh, \underline{a}verage, \underline{l}ow$}, {$t1, t2$} as specific mount types, and {no} indicates fixed lenses. Furthermore, $rf(i_j)$ is the number of features supported by item i_j (see Formula 1).

item id	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}
accumulator	h	h	h	h	h	h	h	h	h	h
resolution	h	h	a	h	a	a	l	h	l	l
fps	h	a	h	a	a	h	l	a	a	a
weight	h	l	a	l	l	a	l	l	a	a
mount	$t1$	no	$t2$	$t1$	no	$t1$	no	no	$t1$	$t2$
isolation	h	l	h	h	l	h	l	h	h	l
$rf(i_j)$	7	5	6	7	4	6	4	6	5	4

Definition 2 (Constraint-based Recommendation). A constraint-based recommendation (REC) for a CBR-task is a set of tuples $REC = \cup(rank_j, i_j)$ where $rank_j$ represents the rank assigned to item $i_j \in I$ by a recommendation function rf, and $\forall(rank_j, i_j) \in REC : consistent(C \cup IC \cup R \cup \{id = i_j\})$ which means that each recommended item i_j must support the constraints in C, IC, and R.

A constraint-based recommendation task can be solved, for example, by a constraint solver, a SAT solver, or on the basis of a conjunctive query.[2] Typically, there are different alternative items satisfying the defined user requirements (i.e., the selected features). For the purpose of item ranking, we need a recommendation function rf. For simplicity, in this paper, we choose the ranking function shown in Equation 1 (the number of features supported by item i_j).[3] For example, with the exception of not being an "easycarry" digicam (feature f_6), item (digicam) i_1 supports all other features $f_1, f_2, f_3, f_4, f_5, f_7, f_8$, i.e., $rf(i_1) = 7$.

$$rf(i_j) = |\{supported\ features(i_j)\}| \tag{1}$$

Example 2 (Digicam Recommendation). We are now able to identify three digicams supporting R = {req_1 : $fastmoves = true$}: i_1, i_3, and i_6 (these items support c_{10} : $fastmoves = true \rightarrow fps = h$). With our simplified ranking function, we can derive the following item recommendation: REC = {$(1, i_1), (2, i_3), (3, i_6)$} since $rf(i_1) = 7$, $rf(i_3) = 6$, and $rf(i_6) = 6$ (see also Table 1).

After having introduced and exemplified basic concepts of constraint-based recommendation, we now focus on analyzing the quality aspects of constraint-based recommenders. These properties are evaluated on the basis of a corresponding set of analysis operations (metrics).

3 ANALYSIS OPERATIONS FOR CONSTRAINT-BASED RECOMMENDERS

Following the idea of analyzing the potential impacts of deploying constraint-based recommenders in real-world settings, we now introduce the following analysis operations (metrics). These metrics are also exemplified on the basis of the feature model in Figure 1 and the corresponding product table (Table 1).[4]

Restrictiveness of Features. For each feature $f_i \in F$ (see also Definition 1), we are interested in the potential impact of selecting this feature, specifically, in terms of its restrictiveness, i.e., to which extent the feature contributes to a reduction of the number of recommendation candidates $i_j \in I : consistent(\{id = i_j\} \cup C \cup IC)$. For example, when including the feature *fastmoves*, this results in the recommendation candidates i_1, i_3, and i_6. Following this idea, we can define the *restrictiveness* of a feature f (see Formula 2) on the scale [0..1] with value 1 representing the extreme case that there does not exist a recommendation that includes feature f.

$$restrictiveness(f) = 1 - \frac{|\{i_j \in I : consistent(\{id = i_j\} \cup \{f = true\} \cup C \cup IC)\}|}{|\{i_j \in I\}|} \tag{2}$$

Measuring the restrictiveness of features is important to understand to which extent the item assortment (product table) I is capable of covering the preferences of a customer community. For example, the share of "watertight and fastmoves" digicams could be increased if the digicams are offered by a sports equipment company. In the extreme case, there is no item i_j supporting feature f – in such case, the feature can be regarded as a kind of *dead* feature (see Formula 3). In our working example, there is no dead feature.

$$dead(f) = \begin{cases} true, restrictiveness(f) = 1 \\ false, \text{otherwise} \end{cases} \tag{3}$$

Following Formula 2, $restrictiveness(fastmoves) = 1 - \frac{3}{10} = 0.7$. The same concept can be applied to feature sets, for example,

[2]Further implementation details can be found here: https://github.com/AIG-ist-tugraz/AO4CRS.

[3]For alternative recommendation functions we refer to [4].

[4]We have calculated the metrics on the basis of Choco (choco-solver.org).

$restrictiveness(\{fastmoves, flexiblelens\}) = 1 - \frac{3}{10} = 0.7$. The complement *excluding* both features, i.e., $restrictiveness(\{\neg fastmoves \wedge \neg flexiblelens\})$ is $1 - \frac{10}{10} = 0$. A feature can be regarded as *false optional* if it is included in every possible set of customer requirements or is always excluded (see Formula 4).

$$false\ optional(f) = \begin{cases} true, inconsistent(\{f = false\} \cup C \cup IC) \\ \quad \vee inconsistent(\{f = true\} \cup C \cup IC) \\ false, \text{otherwise} \end{cases} \tag{4}$$

Accessibility of Items (Products). We propose to measure product accessibility in terms of the number of times an item i is part of a recommendation list, i.e., an item is consistent with $C \cup IC \cup R$. In this context, accessibility can be measured (on the scale [0..1]) in terms of the share of recommendation sets including i compared to the overall number of different recommendation sets that can be generated from user requirements in R which is $|RECS|$ (see Equation 5). In the extreme case of accessibility $accessibility(i) = 1$, item i is part of each recommendation.

$$accessibility(i) = \frac{|\{REC \in RECS : (X, i) \subseteq REC\}|}{|RECS|} \tag{5}$$

Given a constraint-based recommendation task, we can calculate all possible recommendations $REC \in RECS$. The total number of recommendations in our example, i.e., $|RECS|$, is 47 since there are 47 distinct sets of customer requirements. Note that we assume that *RECS* is a bag-type set since different customer requirements could result in exactly the same recommendation *REC*. Table 2 provides an overview of the number of times an item i_j is included in a set *REC*.

Table 2: Item occurrences in recommendations.

pid	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	Σ
#occurrences	31	7	15	31	3	15	3	15	7	3	130

Following Formula 5, $accessibility(i_6) = \frac{15}{47}$ since i_6 is part of a recommendation 15 times and the total number of different consistent requirements is 47. The $accessibility(i_7)$ is $\frac{3}{47}$, i.e., 0.064 which is quite low. This could indicate different issues, for example, i_7 could be outdated or additional features are needed to make the product more accessible, for example, i_7 could have excellent usability specifically for beginners, however, this aspect is not covered (taken into account) by the currently offered set of features. Furthermore, the accessibility of i_7 could also be improved by including an upgrade of i_7 supporting additional features, for example, a "watertight version" of i_7.

If we would include a new type of low-price "fastmoves" digicam i_{11} not supporting the feature *watertight*, then $accessibility(i_{11}) = 0$ since the feature model would be too restrictive in this case. Such issues can be resolved, for example, by testing individual products with regard to their support by the corresponding feature model.

Item Set Coverage. We are also interested in the share of $i_j \in I$ that can be recommended at least once, i.e., $accessibility(i_j) > 0$. This is important due to the fact that we want to avoid situations where some products $i_j \in I$ do not have the chance of being included in at least one recommendation set – such a situation would be indicated by $coverage < 1$ (*coverage* is the product catalog coverage – see Formula 6). In our working example, all items i_j are included in at least one recommendation which results in a product catalog $coverage = 1.0$.

$$coverage = \frac{|\{i_j \in I : accessibility(i_j) > 0\}|}{|\{i_j \in I\}|} \tag{6}$$

Visibility of Items (Products). Since we are dealing with basic recommendation scenarios, we also have to take into account the item (product) ranking in recommendation lists. The more often an item is shown in a prominent position of a recommendation list[5], the higher the corresponding visibility for customers, since due to primacy effects, products at the beginning of a recommendation list are analyzed more often [11]. Taking into account a product's ranking position in different recommendation lists results in the following proposed measure of visibility (see Formula 7).

$$visibility(i) = 1 - \frac{\Sigma_{REC \in RECS:(X,i) \subseteq REC}\ X}{\Sigma_{REC \in RECS:(Y,i) \subseteq REC}\ worstrank(REC)} \tag{7}$$

For example, item i_5 is part of 3 recommendations – within these recommendations, i_5 is ranked 4th in two out of 3 recommendations and 9th in one out of three recommendations (the worst rank in those recommendations is 5, 5, and 19). When applying Formula 7, the overall visibility of i_5 (recommendation lists where i_5 occurs), is quite low ($(1 - \frac{4+4+9}{5+5+10}) = 0.15$), since i_5 is ranked in the almost worst positions. At the same time, i_5 has low accessibility (Formula 5) since it is only part of three recommendation lists (see Table 2). Having such products could be intended (a product only relevant for specific customer segments – in our case, customers interested in "easycarry" digicams). At the same time, depending on the popularity of an item, this can also be regarded as a replacement candidate in future product assortment planning. Being defined on the scale [0..1], the higher the value of $visbility(i)$, the more often item i has a high rank in the corresponding recommendation lists.

Controversy of Features. In interactive recommendation scenarios, it can be the case that a user specifies preferences that do not allow the identification of a solution (recommendation). For example, it is impossible to find an "easycarry" "fastmoves" digicam in a recommendation set of our working example. In this context, we are able to measure the controversy of a feature in terms of the number of times a feature is part of a conflicting set of user requirements R.

To measure the *controversy* of a feature f, we need to figure out all possible combinations of requirements R which include f and induce an inconsistency with $C \cup IC$. Then, $controversy(f)$ represents the share of inconsistency-inducing requirements including f (see Formula 8). For example, the feature *watertight* is part of 8 different sets of requirements which induce an inconsistency. In total, we have 16 different sets R inducing an inconsistency resulting in $controversy(\{watertight\}) = \frac{8}{16} = 0.5$. Being defined on the scale [0..1], the extreme case of $controversy(f) = 1$

[5]Best *rank*=1 and *worstrank* represents the last position in a recommendation *REC*.

indicates a situation where feature f is included in each set of potential user requirements resulting in an empty recommendation, i.e., inconsistent($R \cup C \cup IC$).

$$controversy(f) = \frac{|\{R \in REQS : \{f = true\} \subseteq R \wedge inconsistent(R \cup C \cup IC)\}|}{|\{R \in REQS : inconsistent(R \cup C \cup IC\}\}|} \quad (8)$$

In this setting, we evaluate (on the level of individual features) how often requirements (R) including feature f derived from the feature model are inconsistent with the constraints in $C \cup IC$, i.e., how often a set of requirements including f does not have a corresponding recommendation. In this context, $controversy(f)$ can also be regarded as a measure of the under-constrainedness of a feature model with regard to feature f. Omitting the context of a specific feature f, we are also able to measure the global $controversy_g$ of the feature model (i.e., all features).

$$controversy_g = \frac{|\{R \in REQS : inconsistent(R \cup C \cup IC)\}|}{|\{R \in REQS\}|} \quad (9)$$

Note that in such scenarios, if available, we can also apply conflict detection and diagnosis algorithms that help to determine minimal sets of feature selections (so-called conflict sets [9]) that need to be adapted in order to be able to find a recommendation. In our working example, for each feature f we can observe a $controversy(f) < 1$. This also holds for the global controversy ($controversy_g = \frac{16}{63} = 0.25$) where the total number of different requirement sets is 63 (thereof, 47 are consistent, i.e., lead to a corresponding recommendation).

Redundant Constraints. A constraint $c \in FC \cup FILT$ can be regarded as redundant, if its deletion does not change the user preference space and the corresponding solution space, i.e., the possible sets of recommended items (see Formula 10). In our working example, there are no redundant constraints.

$$redundant(c \in FC \cup FILT) = \begin{cases} true, solutions(FC \cup FILT - \{c\}) \\ \quad -solutions(FC \cup FILT) = \emptyset \\ false, \text{otherwise} \end{cases} \quad (10)$$

4 THREATS TO VALIDITY

We have introduced a simplified working example from the domain of digicams. In this context, we used a simplified definition of technical product properties to make the example also accessible for non-digicam experts. Depending on the application scenario, the proposed analysis operations (evaluation metrics) could also require the calculation of larger result sets – in general, the result set size increases with the number of offered features and the number of offered items [3]. In cases where preference and solution spaces become too large, we have to apply approximation methods stemming, for example, from model counting [6].

5 CONCLUSIONS

We have proposed analysis operations (evaluation metrics) that help to analyze the potential impacts of applying a constraint-based recommender system in productive use. We have introduced such analysis operations on the basis of a simplified working example from the domain of digicams where feature models are used to specify customer preference spaces, i.e., which combinations of features (requirements) can be selected by customers. These requirements are the basis for recommending relevant items. The discussed evaluation metrics can support domain experts and knowledge engineers in the process of scoping item sets and corresponding user preference spaces, i.e., deciding about different variability aspects of the envisioned item assortment (when setting up a new product assortment but also in the context of adapting already existing product assortments and corresponding preference spaces). We have introduced a basic set of such analysis operations and regard these as a basis for future work focusing on extending this set but also proposing solutions that assure applicability when dealing with complex preference spaces. Related analysis operations could also help to analyze impacts on the underlying production processes, for example, what it means to include new features in terms of needed investments and production capacities. We regard this issue as a major topic for future research.

ACKNOWLEDGMENTS

The work presented in this paper has been partially conducted within the scope of the OpenSpace project funded by the Austrian research promotion agency (project FO999891127).

REFERENCES

[1] D. Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *SPLC'2005*. LNCS, Vol. 3714. Springer, San Diego, CA, USA, 7–20.
[2] M. Ekstrand, J. Riedl, and J. Konstan. 2011. Collaborative Filtering Recommender Systems. *Found. Trends Hum.-Comput. Interact.* 4, 2 (feb 2011), 81–173. https://doi.org/10.1561/1100000009
[3] A. Falkner, A. Felfernig, and A. Haag. 2011. Recommendation Technologies for Configurable Products. *AI Magazine* 32, 3 (2011), 99–108. https://doi.org/10.1609/aimag.v32i3.2369
[4] A. Felfernig and R. Burke. 2008. Constraint-Based Recommender Systems: Technologies and Research Issues. In *Proceedings of the 10th International Conference on Electronic Commerce* (Innsbruck, Austria). ACM, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/1409540.1409544
[5] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. 2006. An Integrated Environment for the Development of Knowledge-Based Recommender Applications. *Int. J. Electron. Commerce* 11, 2 (dec 2006), 11–34. https://doi.org/10.2753/JEC1086-4415110201
[6] C. Gomes, A. Sabharwal, and B. Selman. 2009. Model Counting. In *Handbook of Satisfiability*, A. Biere, M. Heule, H. vanMaaren, and T. Walsh (Eds.). Frontiers in Artificial Intelligence and Applications, Vol. 185. IOS Press, Amsterdam, 633–654. https://doi.org/10.3233/978-1-58603-929-5-633
[7] A. Gunawardana and G. Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.* 10 (2009), 2935–2962.
[8] J. Hense and A. Hübner. 2022. Assortment optimization in omni-channel retailing. *European Journal of Operational Research* 301, 1 (2022), 124–140.
[9] U. Junker. 2004. QuickXPlain: Preferred Explanations and Relaxations for Over-constrained Problems. In *AAAI 2004*. AAAI, San Jose, CA, USA, 167–172.
[10] L. Marchezan, E. Rodrigues, W. Klewerton Guez Assunção, M. Bernardino, F. Basso, and J. Carbonell. 2022. Software Product Line Scoping: A Systematic Literature Review. *Journal of Systems and Software* 186 (2022), 111189. https://doi.org/10.1016/j.jss.2021.111189
[11] J. Murphy, C. Hofacker, and R. Mizerski. 2012. Primacy and Recency Effects on Clicking Behavior. *Computer-Mediated Comm.* 11 (2012), 522–535.
[12] M. J. Pazzani. 1999. A Framework for Collaborative, Content-Based and Demographic Filtering. *AI Review* 13, 5-6 (Dec. 1999), 393–408.
[13] F. Rossi, P. van Beek, and T. Walsh. 2006. *Handbook of Constraint Programming*. Elsevier, Amsterdam, The Netherlands.

[14] G. Shani and A. Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. Kantor (Eds.). Springer US, Boston, MA, 257–297. https://doi.org/10.1007/978-0-387-85820-3_8

[15] T. Ulz, M. Schwarz, A. Felfernig, S. Haas, A. Shehadeh, S. Reiterer, and M. Stettinger. 2017. Human Computation for Constraint-Based Recommenders. *J. Intell. Inf. Syst.* 49, 1 (2017), 37–57. https://doi.org/10.1007/s10844-016-0433-4

[16] M. Uta, A. Felfernig, V. Le, A. Popescu, T.N.T. Tran, and D. Helic. 2021. Evaluating Recommender Systems in Feature Model Configuration. In *25th ACM International Systems and Software Product Line Conference - Volume A* (Leicester, United Kingdom). ACM, New York, 58–63. https://doi.org/10.1145/3461001.3471144

[17] E. Zangerle and C. Bauer. 2022. Evaluating Recommender Systems: Survey and Framework. *ACM Comput. Surv.* 55, 8, Article 170 (2022), 38 pages. https://doi.org/10.1145/3556536