Short paper

On Breaking The Curse of Dimensionality in Reverse Engineering Feature Models

Jean-Marc Davril and **Patrick Heymans**[1] and **Guillaume Bécan** and **Mathieu Acher**[2]

Abstract. Feature models have become one of the most widely used formalism for representing the variability among the products of a product line. The design of a feature model from a set of existing products can help stakeholders communicate on the commonalities and differences between the products, facilitate the adoption of mass customization strategies, or support the definition of the solution space of a product configurator (i.e. the sets of products that will be and will not be offered to the targeted customers). As the manual construction of feature models proves to be a time-consuming and error prone task, researchers have proposed various approaches for automatically deriving feature models from available product data. Existing reverse engineering techniques mostly rely on data mining algorithms that search for frequently occurring patterns between the features of the available product configurations. However, when the number of features is too large, the sparsity among the configurations can reduce the quality of the extracted model. In this paper, we discuss motivations for the development of dimensionality reduction techniques for product lines in order to support the extraction of feature models in the case of high-dimensional product spaces. We use a real world dataset to illustrate the problems arising with high dimensionality and present four research questions to address them.

1 Introduction

Feature Models (FMs) have been first introduced for representing the commonalities among the software systems of a software product line [15]. An FM specifies the features that form the potential products (also called configurations) of a product line and how these features can be combined to define specific products. In [4] Berger et al. survey the adoption of variability modeling techniques in industry and report that FMs are the most frequently observed notation.

Defining an FM over a set of existing configurations can bring valuable support in the adoption of *mass customization* strategies. Tseng and Jiao [18] define mass customization as the process of "*producing goods and services to meet individual customer's needs with near mass production efficiency*". A key phase in the development of a mass customization strategy is the definition of the *solution space* that should be offered by the provider [16] - that is, all the product configurations that should be made available in order to satisfy customer demand.

An FM is a concise representation of the solution space of a product line. It can be used to engineer mass customization systems, such as configurators [5, 9]. In this case the FM serves as the knowledge for the configuration system [10]. While deriving a configuration system solely from a set of existing products is not desired in practice, a reverse engineered FM can be used by stakeholders to collect valuable insights about the product domain and to assess the fit between the product line solution space and customer expectations. Depending on the complexity of the solution space and the richness of the product domain, the manual construction of an FM can prove to be time-consuming and prone to errors.

For these reason researchers have provided significant contributions in the development of techniques for automating the extraction of FMs from sets of existing products specifications (or *configurations*) - e.g. see [7, 19, 1, 14, 20, 2]. Existing approaches mostly rely on *logical techniques* that search for statistically significant relationships between the feature values. For instance, if two feature values never occur together in the configurations, one could infer a constraint stating that these values exclude each other. Similarly, two values that frequently occur together can imply a configuration dependency between the two features.

In this paper, we discuss the pitfalls related to the extraction of FMs from product configuration data. In particular, we highlight the limitation of applying logical approaches to *high dimensional* data (i.e. when the product space is defined by a very large number of features). When the number of features grows, logical methods require an increasing number of available configurations to maintain statistical significance. This means that while automatic support is desirable to help practitioners manage high dimensional product space, an FM extraction process that solely relies on logical techniques does not cope well with high dimensionality. We argue that, even when a large volume of configuration data is available, one cannot consistently assume that a logical pattern extracted from the data should be used to define the boundaries of the configuration space. It follows that while it is desirable to support practitioners with logical techniques, an FM extraction process should also consider available product domain knowledge. In the following sections, we highlight the need for formal methods that operate on both logical techniques and background domain knowledge.

The remainder of the paper is organised as follows. Section 2 describes feature modeling and prior work related to the synthesis of FMs. Section 3 illustrates the curse of dimensionality in the context of the FM synthesis problem with a real world example. In section 4 we elaborate a research plan with four research questions.

2 Feature Model Synthesis

An FM is an explicit representation of the underlying variability among the products of a product line. It defines the set of features that compose the products and how these features can be combined into products. An FM can be represented as a tree in which nodes are

[1] University Namur, Belgium, email: firstname.lastname@unamur.be
[2] Inria and Irisa, Université Rennes 1, France, email: firstname.lastname@inria.fr

features and edges between the features represent hierarchical relationships (see Figure 1). In the tree hierarchy, each parent-child decomposition constrains the valid combinations of features that can be found in product configurations. The FM in Figure 1 shows a *XOR-decomposition* for the feature `Screen` into the features `High definition` and `Basic` (i.e. the two child features form a *XOR-group*), which specifies that exactly one of the two child features has to be included in each configuration. Other usual decomposition types are *OR-groups* and *Mutex-groups*, which respectively define that when the parent feature is selected, all features, or at most one feature, must be included. As shown in Figure 1, filled circles and full circles represent mandatory and optional child features respectively. It is also possible to define cross-tree constraints such as the *implies* relationship in Figure 1.

An FM is *attributed* if there are typed attributes associated to its features. Figure 1 shows an attribute `size` of type *integer* under the feature `Screen`. Such FMs are referred hereafter as *Attributed Feature Models (AFM)*.

The semantics of an FM fm, noted $[\![fm]\!]$, is commonly defined as the sets of products (i.e. the sets of sets of features) that satisfy the constraints specified by fm [17]. Table 2 lists the three valid product configurations for the sample FM in Figure 1.

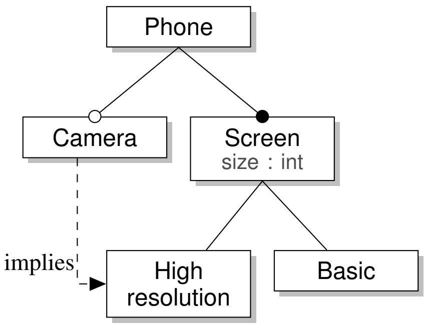

Figure 1. A sample FM for a product line of phones

Camera	High resolution	Basic
X	X	
	X	
		X

Table 1: The valid product configurations for the FM in Figure 1

The *FM synthesis problem* consists in the extraction of an FM from an existing set of products. The synthesis can be decomposed into two steps. First, a logical formula over the inclusion of features in products is mined from the set of configurations. Then, an FM is extracted from the logical formula [7]. Many different FMs can be built for the same logical formula [19]. Therefore, the FM extraction requires heuristics for guiding the selection of the edges that will form the tree hierarchy [19, 8, 20, 2].

The initial set of configurations can be represented as a *configuration matrix*, which presents the features that are included in the existing configurations, as well as the values for the feature attributes. Table 2 shows a possible initial *configuration matrix* from which the FM in Figure 1 could be synthesised.

There are multiple examples of prior work related to the synthesis of FMs from a logical formula, or from sets of formally defined configurations [7, 19, 1, 14, 20]. A major challenge in the synthesis

Camera	High resolution	Basic	size
X	X		8
X	X		7
	X		6
		X	7

Table 2: A potential product matrix that could lead to the extraction of the FM in Figure 1

of FMs is the elicitation of the FM hierarchy. She et al. [19] propose an interactive approach to recommend users with the likely parent candidates for specific features. In [8] we proposed to weight edges between features on the basis of both probabilistic dependencies between the features and similarity between their textual descriptions. We then considered the selection of the hierarchy as the computation of an optimum branching between the features [21]. The FM synthesis techniques proposed in [2] aim at producing FMs that convey a hierarchy that is conceptually sound w.r.t. ontological aspects. In [3], Bécan et al. use domain knowledge to distinguish features from attributes and propose a procedure to mine AFM.

Other works address the extraction of FMs from less structured artefacts such as textual product descriptions [23, 8, 11].

In [6] Czarnecki et al. use probabilistic logic to formalise the foundations of Probabilistic Feature Models (PFMs). The authors also propose an algorithm to build PFMs upon constraints extracted from sets of configurations. PFMs can contain *soft constraints* which express probabilistic dependencies between features.

3 The curse of dimensionality

A machine learning algorithm suffers from the effects of the so-called *curse of dimensionality* when it does not scale well with high-dimensional data. For example, performance issues can arise when the complexity of the algorithm is exponential in the number of dimensions of the dataset. High dimensionality can also impact the quality of results when some dimensions of the dataset are not relevant to the problem to be solved, and thus feed an algorithm with distracting information. Data are referred to as being high-dimensional when they are embedded into an high-dimensional space. In the context of the FM synthesis problem, the data are formed by the existing product configurations. Consequently, data dimensionality is defined by the number of features, the number of attributes, and the size of the domains for the values of these attributes.

3.1 High dimensionality in FM synthesis

We now list the variability structures that are commonly mined from configuration matrices by existing FM synthesis approaches.

- **Binary implications:** Binary implications indicate dependencies between the feature or attribute values in the configuration matrix.
- **Hierarchy:** A tree hierarchy is built from the binary implications between the features. Conceptually, the hierarchy of an FM organizes the features into different levels of increasing detail. It also defines that the selection of a child feature in a configuration always implies the selection of its parent feature.
- **Mandatory features:** Once the hierarchy has been found, for any binary implication from a parent feature to one of its child, the child has to be made mandatory.

- **Feature groups:** *OR-groups*, *XOR-groups* and *Mutex-groups* represent how sibling features can be combined together in product configurations.
- **Cross-tree constraints:** In addition to the constraints represented in the feature hierarchy, cross-tree constraints such as *requires* or *excludes* relationships are mined.

In order to illustrate the curse of dimensionality in the context of the FM synthesis problem, we have applied an AFM synthesis algorithm to a real world dataset extracted from the *Best Buy* product catalog. *Best Buy* is an American retailer that provides consumer electronics, and publishes its products data on the web through an API [3]. We built configuration matrices for the *Best Buy* data by merging extracted sets of products that have at least 75% of features and attributes in common. A description of the AFM synthesis algorithm is out of the scope of this paper and can be found in [3].

We have considered 242 extracted configuration matrices. Table 3 shows statistics about these matrices. The number of Configurations is the number of products in the matrix while the number of Variables is the number of columns (corresponding to features or attributes). Domain size is the number of distinct values in a column. The properties of the configuration matrices are quite representative of high dimensional product spaces. The number of products is low w.r.t. to the total number of distinct cell values. In our dataset, there is almost the same number of variables (columns) as configurations; and in average there are more than 5 values per variable. The application of the AFM synthesis algorithm to this dataset brings to light the need for further research efforts as summarised below.

	Min	Median	Mean	Max
Configurations	11	27.0	47.1	203
Variables (columns)	23	50.0	49.6	91
Domain size	1	2.66	5.45	47.18

Table 3: Statistics on the Best Buy dataset

Firstly, the *Best Buy* configuration matrices contain empty cells. The average proportion of empty cells in the matrices is 14.4%, and in the worst case, the proportion is 25.0% The problem with empty cells is that they do not have a clearly defined semantics in terms of variability. One might consider that an empty cells translates the absence of the corresponding feature in the configuration. However it is unsure whether the feature is really excluded, or if its value is simply unknown. This uncertainty is important because different interpretations of empty cells could lead to different synthesised FMs.

Another concern is the ability to distinguish features from attributes among the columns of the matrix. As for the empty cells, different heuristics for this task could result in very different synthesised FMs. Furthermore, each attribute should be associated to the appropriate parent feature, and automating the association resolution becomes harder as the number of features and attributes grows.

One possible direction for addressing the interpretation of empty cells and the distinction between features and attributes is to rely on the specification of domain knowledge by users. This strategy would require the design of user interactions that prevent users from being overwhelmed with huge volume of variability information, notwithstanding the large number of features and attributes in the dataset.

Another important concern is related to constraints. The number of constraints synthesised from the *Best Buy* matrices average 237 with a maximum of 8906. Such large numbers of constraints put into question the validity of the extracted constraints - that is whether these are legitimate configuration constraints w.r.t. to restrictions in the product line domain. When the data is sparse, it can be hard to evaluate whether the configurations just happened to exhibit the constraint. Moreover, when the number of constraints is high, many different FMs that fit the data equally well can be derived from them. A purely statistical synthesis approach is thus limited as it cannot be used to assess the quality of the candidate FMs. Therefore, it would be useful to automatically reduce the number of irrelevant constraints, or help users assess them. Several approaches can be considered to determine a readable subset of relevant constraints to present to users, e.g. prioritization, or minimisation [22].

Our example does not illustrate the synthesis of PFMs. While the constraints mined for crisp FMs have a confidence of 100% (i.e. they cannot be violated by any product), the constraints mined for PFMs have a confidence above a predefined threshold lower than 100%. PFMs can be useful to model *variability trends* among the products. Similar to the synthesis of FMs, a high dimensional matrix can lead to the computation of a very large number of constraints with a confidence above the predefined threshold, and thus make the elicitation of the PFM structure arduous.

3.2 Dimensionality reduction

In machine learning, the term *dimensionality reduction* denotes the process of reducing the number of attributes to be considered in the data for a particular task [12]. Dimensionality reduction techniques are commonly divided into two categories: *feature selection* and *feature extraction*.

In **feature selection**, a subset of the original data attributes is selected (see [13]). This typically involves the identification of filtering criteria on the attributes (*filter* methods) or the use of the machine learning algorithm itself for ranking the relevance of the attributes (*wrapper* methods). In an FM synthesis process, feature selection could be achieved by choosing a subset of the features to be considered during the elicitation of the FM hierarchy. Once an initial hierarchy would be computed from the core features, the filtered features could then be appended to it.

Feature extraction consists in defining a projection from the high-dimensional space of the data to a space of lower dimension. Let us consider a product line featuring the features `length`, `width` and `depth`. One could define a mathematical function over the values of these three features to replace them with a new attribute `size`, thus reducing the number of dimensions by mapping three features to a single one. The intended benefit is to reduce the cognitive effort when configuring since (1) less configuration variables are presented to users; (2) the configuration variables abstract details that are typically technical, making the promise of raising the level of abstraction for domain analysts or end-users of the engineered configurator.

4 Research Agenda

We aim at addressing dimensionality reduction in the synthesis of FMs in future research. To this end, we state four research questions:

- **RQ1: How should empty cells in configuration matrices be interpreted during the FM synthesis?** An empty cell can either represent the absence of a feature (resp. attribute) or translate a lack of knowledge for the value of a feature (resp. attribute). Acknowledging different semantics for empty cells can lead to different synthesis results. It would be interesting to investigate the

[3] http://developer.bestbuy.com/

use of complementary data, such as product descriptions or user knowledge, to set these missing values.

- **RQ2: How to use background-knowledge in the construction of FMs?** Current synthesis approaches commonly use only data for constructing FMs. However, when a configuration matrix is highly dimensional, it is possible to find many different FMs that fit the data equally well. Some researchers in the machine learning community have already considered that constructing models only from observed data is a bad practice, which has been referred to as *data fishing*. In order to select the right FM, we believe that practical synthesis procedures should be guided by existing knowledge about the application domain of the targeted FM (i.e. *background knowledge*). More specifically, background knowledge could be particularly useful for (1) differentiating the columns of a configuration matrix into features and attributes, and (2) selecting the tree hierarchy among the features of the FM.
- **RQ3: How to reduce dimensionality by modeling product qualities?** Configuration matrices represent products as sets of features, which refer to a large number of technical specifications (e.g. size, weight, battery life). However, customers usually communicate and reason about the products in terms of different abstractions we call *product qualities* (e.g. ease-of-use, portability, ergonomics). An interesting research direction is the design of formal methods for augmenting configuration matrices with product qualities. Projections of qualities onto the technical features could help define a new configuration space of lower dimensionality.
- **RQ4: How to assess the quality of extracted FMs?** One usually wants to elicit an FM with a set of specific tasks to solve in mind, which means that the quality of the FM should be assessed on the basis of the degree of support it provides w.r.t. these tasks. For instance, if an FM is extracted from a set of products with the aim of providing an overview of the underlying variability (domain analysis task), then the readability and the learnability of the FM are relevant quality criteria. In the case of the synthesis of an FM to model a solution space, the conformance of the FM to the existing products should be evaluated. A framework is thus required to identify the evaluation criteria of extracted FMs.

5 Conclusion

In this paper, we discussed the problems arising during the automatic synthesis of feature models from high-dimensional configuration matrices. We first framed concepts well-studied in the machine learning community, such as the *curse of dimensionality* and *dimensionality reduction*, in the context of the feature model synthesis problem. Using a real world dataset extracted from the *Best Buy* website, we then highlighted the steps in an attributed feature model synthesis process that require further investigations (e.g., when computing constraints). We stated research questions related to the application of FM synthesis algorithms to high dimensional configuration matrices.

The motivation for enabling domain experts to apply dimensionality reduction on configuration matrices is to synthesize variability information that is relevant w.r.t. to the intention of practitioners, and to produce more useful, readable resulting feature models.

REFERENCES

[1] Mathieu Acher, Anthony Cleve, Gilles Perrouin, Patrick Heymans, Charles Vanbeneden, Philippe Collet, and Philippe Lahire, 'On extracting feature models from product descriptions', in *Sixth International Workshop on Variability Modeling of Software-Intensive Systems*. ACM, (2012).

[2] Guillaume Bécan, Mathieu Acher, Benoit Baudry, and Sana Ben Nasr, 'Breathing ontological knowledge into feature model synthesis: An empirical study', *Empirical Software Engineering*, 51, (2015).

[3] Guillaume Bécan, Razieh Behjati, Arnaud Gotlieb, and Mathieu Acher, 'Synthesis of attributed feature models from product descriptions', in *19th International Software Product Line Conference*, Nashville, TN, USA, (2015).

[4] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wsowski, 'A survey of variability modeling in industrial practice', in *Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems*, VaMoS '13, pp. 7:1–7:8, New York, NY, USA, (2013). ACM.

[5] Quentin Boucher, Gilles Perrouin, and Patrick Heymans, 'Deriving configuration interfaces from feature models: A vision paper', in *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, pp. 37–44. ACM, (2012).

[6] Krzysztof Czarnecki, Steven She, and Andrzej Wasowski, 'Sample spaces and feature models: There and back again', in *Proceedings of the 12th International Software Product Line Conference.* IEEE, (2008).

[7] Krzysztof Czarnecki and Andrzej Wasowski, 'Feature diagrams and logics: There and back again', in *Proceedings of the 11th International Software Product Line Conference, 2007. SPLC'07.* IEEE, (2007).

[8] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans, 'Feature model extraction from large collections of informal product descriptions', in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. ACM, (2013).

[9] Deepak Dhungana, Andreas Falkner, and Alois Haselbock, 'Configuration of cardinality-based feature models using generative constraint satisfaction', in *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*. IEEE, (2011).

[10] Alexander Felfernig, 'Standardized configuration knowledge representations as technological foundation for mass customization', *IEEE Transactions on Engineering Management*, **54**(1), 41–56, (2007).

[11] Alessio Ferrari, Giorgio O Spagnolo, and Felice Dell'Orletta, 'Mining commonalities and variabilities from natural language documents', in *Proceedings of the 17th International Software Product Line Conference*, pp. 116–120. ACM, (2013).

[12] Imola K Fodor. A survey of dimension reduction techniques, 2002.

[13] Isabelle Guyon and André Elisseeff, 'An introduction to variable and feature selection', *The Journal of Machine Learning Research*, (2003).

[14] Evelyn Nicole Haslinger, Roberto Erick Lopez-Herrejon, and Alexander Egyed, 'On extracting feature models from sets of valid feature combinations', in *Fundamental Approaches to Software Engineering*, Springer, (2013).

[15] Kyo C Kang, Sholom G Cohen, James A Hess, William E Novak, and A Spencer Peterson, 'Feature-oriented domain analysis (foda) feasibility study', Technical report, DTIC Document, (1990).

[16] FT Piller and P Blazek, 'Core capabilities of sustainable mass customization', *Knowledgebased Configuration–From Research to Business Cases. Morgan Kaufmann Publishers*, 107–120, (2014).

[17] P Schobbens, Patrick Heymans, and J-C Trigaux, 'Feature diagrams: A survey and a formal semantics', in *14th IEEE international conference on Requirements Engineering*, pp. 139–148. IEEE, (2006).

[18] J. Seng, M.M.and Jiao, 'Mass customization', in *Handbook of Industrial Engineering: Technology and Operations Management, Third Edition (ed G. Salvendy)*, (2001).

[19] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki, 'Reverse engineering feature models', in *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pp. 461–470. IEEE, (2011).

[20] Steven She, Uwe Ryssel, Nele Andersen, Andrzej Wasowski, and Krzysztof Czarnecki, 'Efficient synthesis of feature models', *Information and Software Technology*, **56**(9), 1122–1143, (2014).

[21] Robert Endre Tarjan, 'Finding optimum branchings', *Networks*, **7**(1), 25–35, (1977).

[22] Alexander von Rhein, Alexander Grebhahn, Sven Apel, Norbert Siegmund, Dirk Beyer, and Thorsten Berger, 'Presence-condition simplification in highly configurable systems', in *Proceedings of the International Conference Software Engineering (ICSE)*, (2015).

[23] Nathan Weston, Ruzanna Chitchyan, and Awais Rashid, 'A framework for constructing semantically composable feature models from natural language requirements', in *13th International Software Product Line Conference*, pp. 211–220, (2009).