A Survey on Scalability and Performance Concerns in Extended Product Lines Configuration

Lina Ochoa[1], Juliana Alves Pereira[2], Oscar González-Rojas[1],
Harold Castro[1], Gunter Saake[2]
[1]Universidad de los Andes, Bogotá, Colombia
{lm.ochoa750, o-gonza1, hcastro}@uniandes.edu.co
[2]University of Magdeburg, Magdeburg, Germany
{juliana.alves-pereira, gunter.saake}@ovgu.de

ABSTRACT

Product lines have been employed as a mass customisation method that reduces production costs and time-to-market. Multiple product variants are represented in a product line, however the selection of a particular configuration depends on stakeholders' functional and non-functional requirements. Methods like *constraint programming* and *evolutionary algorithms* have been used to support the configuration process. They consider a set of product requirements like *resource constraints*, *stakeholders' preferences*, and *optimization objectives*. Nevertheless, scalability and performance concerns start to be an issue when facing large-scale product lines and runtime environments. Thus, this paper presents a survey that analyses strengths and drawbacks of 21 approaches that support product line configuration. This survey aims to: *i*) evidence which product requirements are currently supported by studied methods; *ii*) how scalability and performance is considered in existing approaches; and *iii*) point out some challenges to be addressed in future research.

CCS Concepts

•General and reference → Surveys and overviews; •Software and its engineering → Software product lines; *Genetic programming;* **•Theory of computation** *→ Constraint and logic programming;*

Keywords

Product line, configuration, survey, literature review, scalability, performance, product requirements

1. INTRODUCTION

In order to achieve mass customisation, *Product Line* (PL) has been introduced as a method for representing variability and commonality of a product family. With this capability, industries have achieved remarkable results such as product costs save and reduction in time-to-market. However, the configuration of large PL that optimizes stakeholders' requirements is still a challenge [1, 31]. PLs can represent products functional and non-functional properties and their relationships. Thus, the amount and complexity of options presented to the users exceed their capability of manually identifying an appropriate configuration. Hence, when the variant space grows, the manual configuration process becomes an error-prone and time-consuming task. Therefore, automatic and accurate algorithms are needed.

Given the organization demands to reach good configuration performance for large-scale PLs, many automatic configuration approaches have been proposed in the literature. The vast variety of work in this domain motivated us to carry out this survey investigating how and to what degree existing literature addresses scalability and performance aspects. The aim of this paper is thus to identify, describe, and classify current existing approaches supporting the PL configuration process and to increase the understanding of the fundamental research issues in this field. In this context, we derive answers to the following three research questions:

RQ1. Which product requirements are considered by PL configuration methods?
RQ2. How do existing PL configuration approaches address scalability and performance?
RQ3. What strengths and drawbacks are related to the identified PL configuration approaches?

To address *RQ1*, we give an in-depth view on how different requirements are supported in the PL configuration literature, such as *resource constraints*, *stakeholders' preferences*, and *optimization objectives* (*cf.* Sec. 2). Next, to address *RQ2*, we group the related approaches, collecting evidences about how scalability and performance concerns are managed. We aim to characterize the PL magnitude supported by each approach, as well as the time taken to configure suitable products. Finally, identification of approaches strengths and drawbacks are addressed by *RQ3*.

To answer these questions, we classified 21 selected studies in three different groups according to the method used to support the PL configuration process. Sec. 4 presents the main configuration approaches related to constraint programming and evolutionary methods. In addition, we provide an overview on unclassified methods. We present a clear summary of the existing methods to understand the different works and identify new research contributions. Next, in Sec. 5, we answer the research questions based on the previous section. Finally, conclusions are presented in Sec. 6.

VaMoS '17, February 01-03, 2017, Eindhoven, Netherlands

DOI: http://dx.doi.org/10.1145/3023956.3023959

2. BACKGROUND

In this section, we introduce the main concepts addressed throughout this paper.

Extended feature model. PLs can be formally specified using *Extended Feature Models* (EFMs), also referred in the literature as advanced or attributed feature models [4]. Feature models can be extended by adding *Non-Functional Properties* (NFPs) as feature attributes. NFPs are categorized in two main groups, quantitative and qualitative [3]. Quantitative NFPs are represented as numeric values, while qualitative NFPs are usually represented using an ordinal scale (*e.g.* low, medium, and high).

A common representation model for an EFM is the tree-based structure shown in Fig. 1, where nodes denote *features*, edges illustrate the relations between features, and dashed boxes represent NFPs, such as *cost* [5]. There are common features found in all products of the PL, known as *mandatory* features, and variable features referred to as *optional* and *alternative* features (*cf.* [4] for more details). Additionally, EFMs often contain *Cross-Tree Constraints* (CTC) which define relations among not directly connected features. Furthermore, they can also include complex CTC among features and NFPs. It is important to note that EFMs are required to support the product requirements during the automatic PL configuration process.

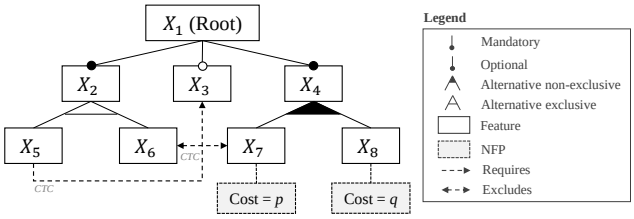

Figure 1: EFM representation.

Product configuration. PL configuration refers to the process of making decisions to (de)select a set of optimal variable features from an EFM. These decisions should comply with the EFM constraints and fulfill the product requirements [1]. We identify three main types of product requirements: *resource constraints*, *stakeholders' preferences*, and *optimization objectives*.

Resource constraints are logical relations over NFP values that admit the following operators: less than ($<$), less than and equal to ($\leq$), equal to ($=$), greater than ($>$), and greater than and equal to ($\geq$).

Stakeholders' preferences allow the specification of NFPs relative importance. Stakeholders' preferences are important specially when there are competing requirements and a trade-off is needed [13].

Optimization objective defines a maximization or minimization criteria over the aggregated values of the same NFP associated to different configured features (*cf.* Equation 2). The automatic PL configuration process can be modeled as *Single-Objective Optimization* (SOO) and *Multi-Objective Optimization* (MOO). A SOO problem arises when the stakeholders aim to optimize a single objective function over NFPs (*e.g.* minimization of cost). A MOO problem arises when the stakeholders aim to optimize multiple objective functions over NFPs (*e.g.* minimization of cost and maximization of security). Formally, it involves the optimization of a vector $F(x)$ whose elements are k objective functions $f_1(x), ..., f_k(x)$, such that $k \in \mathbb{N}$ (*cf.* Equation 1) [6].

$$F(x) = \big[f_1(x) \ ... \ f_k(x)\big]^T \tag{1}$$

Exact and approximation configuration approaches. Although the configuration of a valid product arising from EFMs may reduce the configuration space through product requirement specifications, this is still a challenge due to the diversity of application scenarios and requirements. Thus, automatic and accurate algorithms are needed to configure large-scale EFMs, where a manual configuration is an error-prone and time-consuming task [4]. There are *exact* and *approximation* PL configuration approaches. *Exact approaches* guarantee the optimality of the generated PL configuration. However, due to the large variability space and the NP-hard computational complexity of finding an optimal variant, exact approaches have inefficient exponential time (*cf.* Sec. 4.1). Therefore, *approximation approaches* are needed to generate near optimal PL configurations in an efficient polynomial time (*cf.* Sec. 4.2).

3. SURVEY METHODOLOGY

This survey follows the guidelines defined by Kitchenham et al. [14]. First, we defined three research questions to be addressed by this work (*cf.* Sec. 1). Then, for the *search process*, we built the following *search string*: *"(product line OR product family OR system family) AND (attribute OR non-functional OR quality OR preference OR requirement) AND (configuration OR product selection OR feature selection) AND (performance OR scalability)"*. The *search string* highlights four concepts: PL, NFPs, configuration, and scalability and performance. Candidate papers were obtained from five scientific databases: *ACM Digital Library*, *IEEE Xplore*, *Science Direct*, *Scopus*, and *Springer Link*.

Full papers and primary studies published in English, which appeared in journals, conferences, workshops, or symposiums after 2000[1], were considered as candidate papers in our survey revealing an initial list of 66 papers. Then, we performed a screening process over retrieved studies and excluded 45 papers that were not identified to be relevant based on the selection criteria. At the end of the screening process, we identified 19 relevant papers. We included two additional papers out of the search process given its relevance to the addressed problem, having a total of 21 primary studies selected in the scope of this survey for further analyses. From this set, 8 articles were published in journals, 9 in conference proceedings, 1 in a workshop, and 1 in a symposium. Each selected publication was then read in full detail, and its data was tabulated in a second form.

4. CONFIGURATION APPROACHES

Table 1 shows how the current literature supports the PL configuration process. The first column identifies the reference from relevant studies in this field; second column presents the employed method; third column identifies the approach related tool when mentioned in the paper; fourth column shows which product requirements are supported by these studies; fifth and sixth columns describe if the corresponding study supports runtime configuration environments and multiple PLs, respectively; the seventh column

[1] In 2000 the *Software Product Line Conference* was officially established, given the trending research on the field.

evidences the type of PL employed to perform the experiments; and, the eighth and ninth columns point out the size of the employed PL and its configuration time, respectively. Next, we provide detailed information about each study.

4.1 CP-Based Configuration

Constraint Programming (CP) is a programming paradigm that deals with *Constraint Satisfaction Problems* (CSPs). The problem of selecting a suitable set of features that satisfy the EFMs constraints and product requirements can be translated into a CSP [4]. A CSP is defined as a tuple $\langle X, P, C \rangle$, where $X = \{X_1, \ldots, X_n\}$ represents a set of n features, $P_{n \times m} = \{P_{[1,1]}, \ldots, P_{[n,m]}\}$ represents a matrix with a set of $(n * m)$ NFPs values related to each feature in X, and $C = \{C_1, \ldots, C_k\}$ represents a set of k constraints.

To support the automatic PL configuration process with CSP solvers, three main steps are performed: *i*) map functional features in X and NFPs in P from an EFM into a set of $n + (n * m)$ variables; *ii*) map EFM constraints and product requirements into a set of k constraints C restricting the variables X values; and *iii*) define a set of n binary values (*i.e.* [0,1] to indicate whether the features $\{X_1, \ldots, X_n\}$ are selected or deselected) for the variables in which all constraints C are satisfied. CP-based approaches have gained popularity in the PL field, given its ability to easily translate the PL configuration problem into a CSP.

White et al. [31] were the first authors proposing the use of CP in the context of mobile devices configuration, while considering resource constraints related to device capabilities. They use Prolog's inferencing engine and the Choco CP solver to configure in runtime a new application variant from existing software components. The complete approach, has two main steps: *i*) prune the solution space by eliminating software components (*i.e.* features) that cannot run on the target device infrastructure; and *ii*) optimize the cost function $f(x)$ in Equation 2, such that $1 \leq c \leq m$ and $c \in \mathbb{N}$. Valid solutions that satisfy the resource consumption constraints R_j in Equation 3 are searched. $R = \{R_1, \ldots, R_m\}$ represents a set of m resource constraints related to each property $\{P_{[1,1]}, \ldots, P_{[1,m]}\}$, such that $R \subset C$.

$$f(x) = min \sum_{i=1}^{n} X_i * P_{[i,c]} \tag{2}$$

$$\sum_{j=1}^{m} \left(\sum_{i=1}^{n} X_i * P_{[i,j]} \right) < R_j \tag{3}$$

Gamez et al. [10] and Leite et al. [15] also employed the Choco CP solver to deal with resource optimization on web services and inter-cloud environments. Both approaches consider runtime configuration of services from multiple EFMs. On the one hand, Gamez et al. [10] relies on the score optimization of a *Simple Additive Weighting* (SAW) function $f(x)$ in Equation 4, which ranks each valid service configuration according to its importance to the stakeholders. The weight of the NFP j is represented as w_j, where $w_j \in [0, 1]$ and $\sum_{j=1}^{m} w_j = 1$. $R_{l \times m} = \{R_{1,1}, \ldots, R_{l,m}\}$ represents a matrix with a set of $(l * m)$ NFP values related to each valid resulting configuration (*i.e.* $R_{[l,j]}$ is the corresponding value of the NFP j from the resulting valid configuration l), and the functions $max(R, j)$ and $min(R, j)$ return the maximum and minimum values of the NFP j respectively, when considering all valid resulting PL configurations.

$$f(x) = max \sum_{j=1}^{m} \sum_{i=1}^{l} w_j * \frac{max(R, j) - R_{[i,j]}}{max(R, j) - min(R, j)} \tag{4}$$

On the other hand, Leite et al. [15] relies on the MOO of three different functions, mainly number of CPU cores, RAM memory size, and financial cost per hour.

On a multi-stakeholders scenario, Ochoa et al. [20] also employ the Choco CP solver. The solver is used to detect and solve conflicts among multiple independent stakeholders' configurations performed over an EFM. As White et al. [31], their approach considers SOO over a single NFP. The authors extend a state-of-the-art tool FeatureIDE[2] with their approach. Preliminary experiments show a good performance for an EFM with up to 140 features, 13 CTCs, and 2 different stakeholders' configurations. However, for some configuration scenarios with three or more configurations, no solution was found.

White et al. [31] also provide a tool support, called Scatter[3] and present a set of performance experiments for five EFMs. Their experiments showed that Scatter can solve models with up to 50 features in about 100ms.

Gamez et al. [10] extended a state-of-the-art tool Hydra[4] and Leite et al. [15] also provide a tool support called Dohko[5]. Experimental results show that both approaches enable inexperienced users to have access to advanced configurations, without being concerned with all the characteristics of each environment. While Gamez et al. [10] use an academic PL as case study and show a computation time of 2,5s to solve models with 1.024 features, Leite et al. [15] use two industrial PLs and show a computation time of up to 10min to solve models with 46 features.

As the previous authors, Mazo et al. [17] developed a tool called VaRiaMos[6] that relies on MOO. In addition, VaRiaMos supports the filter of a set of configurations that satisfy EFMs constraints and product requirements from a partial (de)selection of features. An experiment evidenced that the approach is scalable and has a high performance (*i.e.* 515ms to find 100.440 valid configurations) by using a benchmark of 50 PLs in a broad range of application domains, such as insurance, entertainment, web applications, home automation, search engines, and databases.

Siegmund et al. [27] focus on the measurement of NFP values and on the search for an optimal variant from given product requirements. As Gamez et al. [10], in Siegmund et al. [27] a SOO can also be defined over multiple NFPs. For example, Equation 5 computes the best trade-off between two NFPs $R_{[c,i]}$ and $R_{[c,j]}$, $\forall c : 1 \leq c \leq l$, $\forall i : 1 \leq i \leq m$, and $\forall j : 1 \leq j \leq m$, such as $i \neq j$. As introduced before, l is the number of valid resulting configurations and m is the number of NFPs. Their approach consists of four main steps: *i*) select desired features; *ii*) exclude features, as well their dependencies, from further consideration if they have a negative effect on a NFP that is of interest to stakeholders; *iii*) apply CP techniques to find an optimal variant from a stakeholder-defined objective function based on feature-wise and variant-wise measurements; and *iv*) optimize a derived

[2] wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

[3] www.sf.net/projects/gems

[4] caosd.lcc.uma.es/spl/hydra

[5] dohko.io

[6] sites.google.com/site/raulmazo/products-tools/variamos

Table 1: Literature supporting the PL configuration process.

Ref.	Method	Tool Support	Requirements	Runtime	Multiple PLs	Employed PLs	PL Size	Performance
[31]	CP	Scatter	RC, SOO	Yes	Yes	A, I	50 features	100ms
[27]	CP	SPL Conqueror	RC, SOO	No	No	A, I	No information	No information
[17]	CP	VariaMos	RC, MOO	No	Yes	A, I	100.440 valid configurations	515ms
[30]	CP	FaMa	RC, SOO	No	No	G	2.000 features	12s
[20]	CP	FeatureIDE ext.	RC, SOO	No	No	A, G	140 features	296ms - 18min
[15]	CP	Dohko	RC, MOO	Yes	Yes	I	46 features	10min
[10]	CP	Hydra ext.	RC, SP, SOO	Yes	Yes	A	1.024 features	2,5s
[3]	GA	-	RC, SOO	Yes	No	G	10.000 features	0,2 - 0,5s
[11]	GA	-	RC, SOO	Yes	No	G	200 features	101,53ms
[32]	GA	-	SOO	Yes	No	I	133 features	0,25s
[13]	MOEA	PISA ext.	MOO	No	No	G, I	10.000 features	4min
[23]	MOEA	-	MOO	No	No	I	43 features	9,2min
[12]	MOEA	-	MOO	No	No	A, I	6.888 features	15min
[28]	MOEA	-	MOO	No	No	A, I	290 features	6,4 - 7s
[16]	MOEA	-	MOO	No	No	A, I	290 features	2,95 - 29,43min
[21]	MOEA	-	MOO	No	No	A, I	290 features	45min - 1.000h
[25]	MOEA	-	MOO	No	No	A, I	290 features	8min
[1]	AHP	fmp ext.	RC, SP, SOO	No	No	G	100 features	20,1 - 86,05s
[22]	Relat. modeling	ClaferMoo	MOO	No	No	A, I	85 features	11 - 18min
[2]	S-AHP	fmp ext.	SP	No	No	A	31 features	No information
[26]	Knapsack	-	RC, SOO	Yes	No	G	100 features	0,7s

We classify requirements into five types: Resource Constraints (RC), Stakeholders' Preferences (SP), Single-Objective Optimization (SOO), and Multi-Objective Optimization (MOO). Additionally, we classify the employed PLs in Academic (A), Industrial (I), and Generated Randomly (G) models.

variant by means of refactoring to improve NFPs. On step *iii*), the authors approach takes into consideration feature interactions through the usage of specific sampling heuristics to meet different feature-coverage criteria.

$$f(x) = max \sum_{c=1}^{l} \frac{R_{[c,i]}}{1.000 * R_{[c,j]}} \quad (5)$$

The general approach is implemented in a tool called SPL Conqueror[7] and its applicability is demonstrated by nine industrial and academic PLs. Once the feature measurement requires the generation of many variants, experiments show a high computation time (*e.g.* in the case of large code base, 4 days to solve models with 25 features). However, no performance information is given to the optimization process.

On the multi-stage configuration scenario, White et al. [30] propose an approach called MUSCLES (*MUlti-step Software Configuration probLEm Solver*). The authors extend a state-of-the-art tool FaMa[8] with their approach. The goal of this approach is to enable CP solvers to derive a series of intermediate optimal configurations from a starting PL configuration to a final configuration that meets the desired set of product requirements. MUSCLES supports users to decide which evolution path best fits the project's goals. Moreover, it supports the reasoning about how changes may or will impact future configuration decisions. The current experiments show that the approach can scale to EFMs with hundreds of features and multiple configuration steps (*e.g.* models with 500 features over 10 steps can be solved in about 16s, and feature models with 2.000 features over 3 steps can be solved in about 12s). However, MUSCLES assumes that the models at each step are valid and error-free.

4.2 EA-Based Configuration

In *Evolutionary Algorithms* (EA) and *Genetic Algorithms* (GA)[9], an encoded solution is known as an *individual*. Each individual is related to one or more *chromosomes* which are composed by multiple *genes* with their corresponding values or *alleles*. A group of individuals is known as a *population*. In a set of iterations the population evolves by considering different operators like *mutation*, *crossover*, and *selection*. The best individuals are selected in each generation to improve the characteristics of the off-springs based on a *fitness* function (*i.e.* it defines the degree of satisfaction of a solution). A starting population is defined and conformed by l solutions represented as chromosomes, where each chromosome is defined as a one-dimensional vector with n elements (*cf.* Fig. 2) [6]. A configuration of an EFM is represented as a chromosome, and features are represented as genes where 1 means that the feature is selected and 0 deselected.

	X_1	X_2	X_3		...		X_n
Configuration 1	1	0	1	0	0	1	1
Configuration 2	1	1	1	0	0	1	0
...							
Configuration l	1	0	0	1	0	0	1

Figure 2: EA feature model encoding.

Previous concepts are applied to EA, defining six main tasks: *i*) initialize population; *ii*) evaluate k fitness functions; *iii*) transform fitness vector into a scalar; *iv*) perform crossover; *v*) perform mutation; and *vi*) perform selection and return to task *iii* until the termination condition is satisfied (*e.g.* the number of expected generations is reached) [6]. EA-based approaches have gained popularity in the solution of the PL configuration problem, given its ability to consider stakeholders' needs as optimization objectives, and its good performance in large search spaces.

[7] www.infosun.fim.uni-passau.de/spl/projects/splconqueror/

[8] famats.googlecode.com/svn/branches/multistep/

[9] GA is considered a type of EA that uses binary strings as encoding, however the bound with other approaches is fuzzy.

4.2.1 Genetic Algorithms

Yeh and Wu [32] proposed a GA-based PL configuration approach motivated by the poor efficiency presented by other solutions [10, 15, 31] when trying to find configurations in runtime contexts. In their approach, they defined a SOO cost function $f(x)$, and the PL model as a minimum-cost flow problem, where the product configuration network is represented as a flow network. The objective is to find the cheapest path for sending an amount of flow (*cf.* Equation 6). The flow network is defined as a directed graph $G = (X, A)$, where X is the set of nodes (*i.e.* features) and A the set of arcs in the network; $c_{i,j}$ is the cost of flow between node i and node j, and X_j is the selection state of node j (1 if it is selected, 0 otherwise).

$$f(x) = min \sum_{(i,j)\in A} c_{i,j} X_j \tag{6}$$

Similarly, Bagheri et al. [3] proposed a reliability-aware configuration that aims to satisfy reliability bounds (lower and upper) when searching a configuration in the PL. Equation 7 is known as the lower reliability bound $B^L_{S_*}$, and it is the result of sequentially composing all the selected features in a configuration S_*. Two approaches were proposed to configure a PL with this optimization objective: *i*) find all configurations that satisfy functional requirements while ignoring reliability constraints. Then, the configurations that respect the reliability bounds are kept; and *ii*) use GA to respond to the SOO and functional constraints.

$$B^L_{S_*} = \prod_{i=1}^{n} B_{X_i} \tag{7}$$

Yeh and Wu [32], Guo et al. [11], and Bagheri et al. [3] approaches allow the specification of a SOO in order to configure PL optimal solutions. However, in [32] configuration filter cannot be expressed and CTC are not even managed. Nevertheless, these approaches were profiled as scalable solutions. The approach of Yeh and Wu [32] was tested against a PL with 479 billion products. The search was executed in around 260ms, which outperformed the results obtained with other searching methods such as *Uniform-cost Search* (UCS) and *Breadth-first Search* (BFS) (*i.e.* executed between 4,22s and more than 20min).

In the case of Bagheri et al. [3], their approach was tested with randomly generated EFMs between 1.000 and 10.000 features, and between 5% and 20% of CTC. The linear approach is good for configuring small models (*i.e.* less than 3.000 features), while the GA-based approach is useful for larger models. Lastly, in Guo et al. [11] 100 EFMs were generated with different feature sizes (*i.e.* 10, 50, 100, and 200). For EFMs with 200 features, their approach, took an average of 101,53ms to compute the solution.

4.2.2 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEA) are used for solving MOO problem in stochastic environments [6]. Different PL configuration approaches based on MOEA were presented with the need to improve performance and scalability of PL configuration.

Sayyad et al. [25] compared seven MOEAs, i.e. IBEA [33], NSGA-II [7], ssNSGA-II [8], SPEA2[34], FastPGA [9], MOCell [19], and MOCHC [18]. Five optimization objectives were considered, including the minimization of violated constraints and the maximization of selected features. Two academic EFMs were employed, *Web Portal* and *E-Shop*, and extended with three NFPs (*i.e. costs*, *defects*, and *used before*). IBEA outperformed the remaining algorithms in terms of quality; it favors product requirements (*i.e.* optimization objectives) over other qualities, such as absolute dominance and solutions spread. However, when performing 50.000 evaluations over the *E-Shop* EFM with 290 features, no correct solution was provided. Two million evaluations were needed in order to obtain 3% of correct solutions in around 8min. Similarly, Olaechea et al. [21] employed IBEA and five EFMs including the two models referenced by Sayyad. As results, for the *E-Shop* case, configurations were obtained after 250.000 evaluations and less than 1.000h, an extensive time for applications that demand low latency.

Lian and Zhang [16] proposed a new MOEA derived from IBEA, the *Indicator optimization and rules Violation controlling Evolutionary Algorithm* (IVEA). IVEA is designed as a polynomial time algorithm. A two-dimensional fitness function was proposed to preserve the advantage on considering product requirements presented by IBEA. The first dimension is known as *infeasibility* and it measures constraint violations of a given configuration. The second dimension is related to *product requirements* specified as a MOO.

In a similar way, Tan et al. [28] presented the feedback-directed IBEA. They use a SAT solver to remove prunable features (*i.e. common* and *dead* features) before the EA is applied. When a chromosome is created and there are constraint violations, the positions of the features involved in the violation are known as *error positions*, and mutation has a larger probability of being executed in these genes. Thus, crossover operation considers values in non-error positions in order to preserve good genes in the off-springs.

For these two last approaches, experiments were done against the same models and optimization objectives as the ones presented by Sayyad et al. [25], plus some additional scenarios. IVEA [16] outperformed other MOEAs, including IBEA. For a testing scenario with 50.000 evaluations, the algorithm took almost 3min to generate 357 correct solutions for the *E-Shop* model. On the other hand, Tan et al. [28] showed that IBEA also outperformed other MOEAs, but it was even better when combined with the feedback-directed operations and the PL pruning. For the *E-Shop* model 100% of the configurations were correct and obtained in around 6s.

Henard et al. [12] presented SATIBEA. Their approach aims to select near optimal features, while proposing a technique to handle PL constraints. *Diversity promotion* is encouraged and two smart search operators are introduced. Diversity promotion is measured with a *dissimilarity metric*. The dissimilarity between two consecutive configurations P_i and P_j is defined in Equation 8. The two proposed smart operators correspond to *smart mutation* and *smart replacement*. The first operator removes features involved in a constraint violation from an invalid configuration, and with the help of a SAT solver the partial configuration is completed. The smart replacement operator randomly picks a configuration and replaces it with a new solution. For the *Linux* model (6.888 features) SATIBEA found converging results after 15min of execution. The five considered optimization objectives are the same presented by Sayyad et al. [25].

$$d(S_i, S_j) = \frac{|S_i \cup S_j| - |S_i \cap S_j|}{|S_i \cup S_j|} \tag{8}$$

Furthermore, Hierons et al. [13] introduced *ShrInk Prioritize* (SIP), which enhances the MOO search by providing a new representation (core and parents features with a *mandatory* or *alternative* relationship are removed from the model) and organizing the objectives optimization with the *(1 + n)* approach. First, the number of non-violated constraints are considered, and then other objectives. This approach was compared to other encodings, including the ones presented by Sayyad et al. [25] and Henard et al. [12]. All results for an EFM with 10.000 features were obtained in less than 4min. In all cases SIP outperformed the other encodings. Multiple MOEAs were compared and there was not a clear winner.

Finally, Parejo et al. [23] used NSGA-II to derive configurations based on a MOO search in order to execute test suites to detect faults in a PL. Different objectives combinations are considered, including the selection of features that have a higher *Coefficient of Connectivity* (CoC), a metric that shows the amount of constraints in which a feature is involved; and the maximization of *dissimilarity* among configurations (*i.e.* differences among products in order to have a higher coverage). The results are returned in around 9,2min for an EFM with 48 features.

As main gaps of the EA-based approaches, we found that product requirements expressiveness is limited to optimization functions. Results vary depending on the selected parameters, including the number of evaluations. MOEAs tunning should be considered in future work. Moreover, the impact of CTC and the number of product requirements should also be measured, in order to characterize the scalability and performance of the approaches.

4.3 Other Configuration Methods

Other methods have been used to solve the PL configuration problem. This is the case of the knapsack-based approach presented by Shi et al. [26], which was a modification of the *Filtered Cartesian Flattening* (FCF) algorithm [29]. In this approach, the MMKP algorithm is replaced with a simple knapsack, and a greed search based in a CSP solver is used. Then, k items are selected based in their values and costs. Finally, an optimized selection is performed. The approach was tested with EFMs of different sizes. When the models have less than 100 features the approach provides an optimization level (*i.e.* approximation answer/optimal answer) greater than 80% in about 4s, where $k = 800$ in a model with 50 features. However, in this approach stakeholders cannot define their own "values" and "costs", and CTCs are not considered.

Olaechea et al. [22] presented ClaferMoo, a language and tool for managing and configuring optimal products over EFMs. ClaferMoo considers MOO over integer values. Moreover, partial configurations could be defined to filter the search space. This tool uses Molloy, an extension of Alloy which is a relational modeling language. The experiments showed that for models with around a dozen of features, ClaferMoo has a good performance (results are obtained in less than 4min). Nonetheless, for models with more than 85 features, the solution takes more than 11min to be obtained, and in the presence of multiple CTCs (*e.g.* model with 100 features and 89 CTCs) there is a time out and no optimal solution is obtained.

Finally, Bagheri et al. [2] presented the *Stratified Analytic Hierarchy Process* (S-AHP) method, and Asadi et al. [1] presented a PL configuration framework based in AHP, *Fuzzy Cognitive Maps* (FCM), and *Hierarchical Task Network* (HTN). Both approaches are AHP approximations that perform a pairwise comparison among NFPs in order to define priorities and select features according to stakeholders' preferences. In the case of Asadi et al. [1], positive and negative impacts of features over NFPs, as well as interdependency relationships over NFPs are considered. Local weights Z of NFPs are captured with AHP, while NFPs inter-dependencies T are captured with FCM. The overall weight of the NFPs is computed as $W = Z + T$. An utility function that considers relevant NFPs is defined, then products are generated based on the selection of features that maximizes the given function. Asadi et al. concluded that their approach returns results in around 16s for models with less than 200 features and with a distribution of 20% CTCs. However, when the considered NFPs increase, the execution time also increases.

5. DISCUSSION

Based on our insights from the last section, we evaluate how the studies from the literature address the research questions presented in Sec. 1.

RQ1: Which product requirements are considered by PL configuration methods?

The use of product requirements differ depending on the chosen PL configuration method. While all studied approaches based on CP and EA introduce support to optimization objectives, most of CP-based approaches provide support to SOO (only two of them [15, 17] allow also the specification of MOO). On the context of EA-based approaches, while GAs are employed in SOO scenarios, MOEAs are used in MOO. Additionally, resource constraints are fully supported by CP-based approaches. Nevertheless, in EA-based approaches just Bagheri et al. [3] and Guo et al. [11] offer such support. In particular, just Asadi et al. [1] and Gamez et al. [10] approaches handle the three recognized product requirements: *resource constraints*, *stakeholders' preferences*, and *optimization objectives*. However, none of those approaches present support to MOO.

When satisfying the product requirements, the PL constraints should be satisfied. In this context, Asadi et al. [1] and Olaechea et al. [22] approaches also present support to filters specification. Their approach aims to search for a valid configuration that satisfies a set of previously (de)selected features and maximizes an utility function that involves weights and preferences over multiple NFPs. Additionally, although most works present support to CTC, *complex CTC* introduced by Benavides et al. [5] are just partially managed in MOEAs by Sayyad et al. [25]. However, in their approach, complex CTCs are static and application engineers cannot add new expressions or modify the existing ones.

RQ2: How do existing PL configuration approaches address scalability and performance?

EA-based solutions show a better scalability and performance when configuring PLs. EAs can manage large EFMs, deriving solutions in reasonable time (few minutes or seconds). For SOO scenarios, EFMs with 1.000 to 10.000 features can be solved in less than 600ms [3, 32]. For MOO scenarios, results are obtained in 8min [25], 3min [16], and 6s [28] when considering the *E-Shop* EFM (290 features).

These experiments consider five optimization objectives [25]. Lastly, 15min were taken by Henard et al. [12] approach to find solutions with the same five objectives in the *Linux* EFM (6.888 features), and less than 4min to find solutions with the Hierons et al. [13] approach in a randomly generated EFM (10.000 features).

On the CP-based context, due to the exponential time taken by exact algorithms, lower performance is evident in the experiment results. While Hydra [10] presents the best results (1.024 features in about 2,5s) for SOO scenarios, VariaMos [17] presents the best results (100.440 valid configurations in about 515ms) for MOO scenarios. Finally, for other configuration methods, the knapsack approach presented by Shi et al. [26] shows the best results. It can manage EFMs with 850 features in around 4s. However, for all these techniques, depending on the number of considered NFPs, time could increase. To accurately measure the performance of those approaches, a set of experiments varying the number of NFPs associated to each feature need to be considered in future work.

In this context, execution time varies depending on the employed method and its considered parameters, as well as the number and type of considered product requirements. However, most of the current approaches do not present experimental results related to changes of product requirements (*e.g.* different optimization objectives) and parameters (*e.g.* type of NFPs), which can affect the scalability and performance of each solution.

RQ3: What strengths and drawbacks are related to the identified PL configuration approaches?

All methods have been proved to be capable of solving the PL configuration problem over EFMs. Nevertheless, each method presents different strengths and drawbacks that should be considered by researchers in future work.

CP-based studied approaches allow the specification of resource constraints and SOO requirements. Few solutions also allow the specification of MOO requirements [15, 17], and just one study [10] supports stakeholders' preferences. Furthermore, multiple CP-based approaches support runtime configuration [10, 15, 30], and the interaction with multiple PLs [10, 15, 17, 30]. However, for large-scale EFMs and in the presence of multiple constraints, scalability and performance can be affected [15, 20]. The employment of search heuristics can improve the performance of these solutions.

In contrast to CP-based approaches, EA-based approaches show good performance and scalability, deriving solutions for large PLs in few minutes or seconds. In the case of GA-based approaches, SOO requirements are supported and two studies also support resource constraints [3, 11]. Conversely, MOEA approaches support only MOO requirements. However, for all EA approaches, there is no PL constraints satisfaction guarantee, then the minimization of constraints violations should be treated as an additional optimization objective. To improve the quality of the obtained configurations as well as stakeholders' satisfaction, approaches that support the three product requirements pointed out in this paper (*cf.* Sec. 2) are required. In addition, tuning MOEAs parameters could enhance the quality and performance of solutions. Finally, multiple PLs and runtime configuration are not supported by most of the studies.

Other approaches have considered stakeholders' preferences as product requirements, particularly the AHP-based approaches [1, 2]. The knapsack approach [26] considers resource constraints and SOO, while the relational modeling study [22] considers MOO. Nonetheless, current experiments for these approaches only manage EFMs with less than 100 features. Further tests should be performed to prove the scalability and performance of these solutions.

For all methods, stakeholders can have very different requirements when configuring a PL. This diversity leads to heterogeneous solutions. In these cases, no support is presented to guide the user on choosing a suitable solution. Approaches like the one presented by Pereira et al. [24] have shown preliminary thoughts in this direction with the use of customized recommender systems. However, the authors approach is still not integrated with EFMs. Moreover, complex CTC [5] should also be addressed in future research.

Finally, most approaches [3, 10, 12, 16, 17, 20, 25, 30, 31, 32] conduct few experiments on academic or randomly generated PL instances and may not reflect realistic EFMs seen in industry. Additionally, the evaluation material (including algorithms and results) are not publicly available for the purpose of experiments reproducibility. The use of real industrial PLs and the availability of the evaluation material is indispensable to prove reliability of the obtained results.

6. CONCLUSION

This paper presents a survey on scalability and performance concerns related to configuration of EFMs. We analyzed 21 papers. Two methods employed during product configuration were studied, CP and EA as well as other methods found in the literature. We identified that EA favors scalability and performance given its capability of managing EFMs with up to 10.000 features and five optimization objectives. However, there is no guarantee of PL constraints satisfaction and stakeholders preferences are not considered. CP-based approaches support multiple product requirements and guarantee constraint satisfaction. Nonetheless, scalability and performance are affected when facing large-scale EFMs and a high amount of constraints. Finally, other methods like AHP, relational modeling, and knapsack are introduced to support further product requirements. For all cases, additional research is required for managing multiple PLs and runtime configuration. Further tests including industrial EFMs are needed to obtain more reliable results.

As future work, we plan to compare the different approaches by considering a same case study. Moreover, we aim to combine the presented methods to support different product requirements in order to increase stakeholders' satisfaction when deriving suitable configurations. These hybrid solutions should respond to scalability and performance requirements, which are related to large-scale PLs that demand runtime solutions. Finally, a visual guidance during the specification of requirements and selection of a configuration from a set of solutions should also be supported.

7. ACKNOWLEDGEMENTS

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) grant 202368/2014-9.

8. REFERENCES

[1] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, and E. Bagheri. Toward automated feature model

configuration with optimizing non-functional requirements. *Information and Software Technology*, 56(9):1144–1165, 2014.

[2] E. Bagheri, M. Asadi, D. Gasevic, and S. Soltani. Stratified analytic hierarchy process: prioritization and selection of software features. In *SPLC*, pages 300–315, Berlin, Heidelberg, 2010. Springer.

[3] E. Bagheri and F. Ensan. Reliability estimation for component-based software product lines. *Canadian Journal of Electrical and Computer Engineering*, 37(2):94–112, 2014.

[4] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: a literature review. *Information Systems*, 35(6):615–636, 2010.

[5] D. Benavides, P. Trinidad, and A. Ruiz-Cortés. Automated reasoning on feature models. In *CAiSE*, pages 491–503, Berlin, Heidelberg, 2005. Springer.

[6] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, Secaucus, 2006.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[8] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba. On the effect of the steady-state selection scheme in multi-objective genetic algorithms. In *EMO*, pages 183–197, Berlin, Heidelberg, 2009. Springer.

[9] H. Eskandari, C. D. Geiger, and G. B. Lamont. FastPGA: A dynamic population sizing approach for solving expensive multiobjective optimization problems. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, editors, *EMO*, pages 141–155, Berlin, Heidelberg, 2007. Springer.

[10] N. Gamez, J. El Haddad, and L. Fuentes. SPL-TQSSS: a software product line approach for stateful service selection. In *ICWS*, pages 73–80, Piscataway, 2015. IEEE.

[11] J. Guo, J. White, G. Wang, J. Li, and Y. Wang. A genetic algorithm for optimized feature selection with resource constraints in software product lines. *Journal of Systems and Software*, 84(12):2208–2221, 2011.

[12] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon. Combining multi-objective search and constraint solving for configuring large software product lines. In *ICSE*, pages 517–528, Piscataway, 2015. IEEE.

[13] R. M. Hierons, M. Li, X. Liu, S. Segura, and W. Zheng. SIP: Optimal product selection from feature models using many-objective evolutionary optimization. *Transactions on Software Engineering and Methodology*, 25(2):17:1–17:39, 2016.

[14] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering − a systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009.

[15] A. F. Leite, V. Alves, G. N. Rodrigues, C. Tadonki, C. Eisenbeis, and A. C. M. A. de Melo. Automating resource selection and configuration in inter-clouds through a software product line method. In *CLOUD*, pages 726–733, Washington, 2015. IEEE.

[16] X. Lian and L. Zhang. Optimized feature selection towards functional and non-functional requirements in software product lines. In *SANER*, pages 191–200, Piscataway, 2015. IEEE.

[17] R. Mazo, C. Salinesi, D. Diaz, O. Djebbi, and A. Lora-Michiels. Constraints: the heart of domain and application engineering in the product lines engineering strategy. *International Journal of Information System Modeling and Design*, 3(2):33–68, 2012.

[18] A. J. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, and J. J. Durillo. Optimal antenna placement using a new multi-objective CHC algorithm. In *GECCO*, pages 876–883, New York, 2007. ACM.

[19] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba. MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009.

[20] L. Ochoa, O. González-Rojas, and T. Thüm. Using decision rules for solving conflicts in extended feature models. In *SLE*, pages 149–160, New York, 2015. ACM.

[21] R. Olaechea, D. Rayside, J. Guo, and K. Czarnecki. Comparison of exact and approximate multi-objective optimization for software product lines. In *SPLC*, pages 92–101, New York, 2014. ACM.

[22] R. Olaechea, S. Stewart, K. Czarnecki, and D. Rayside. Modelling and multi-objective optimization of quality attributes in variability-rich software. In *NFPinDSML*, pages 2:1–2:6, New York, 2012. ACM.

[23] J. A. Parejo, A. B. Sánchez, S. Segura, A. Ruiz-Cortés, R. E. Lopez-Herrejon, and A. Egyed. Multi-objective test case prioritization in highly configurable systems: A case study. *Journal of Systems and Software*, 122:287–310, 2016.

[24] J. A. Pereira, P. Matuszyk, S. Krieter, M. Spiliopoulou, and G. Saake. A feature-based personalized recommender system for product-line configuration. In *GPCE*, pages 120–131, New York, 2016. ACM.

[25] A. S. Sayyad, T. Menzies, and H. Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In *ICSE*, pages 492–501, Piscataway, 2013. IEEE.

[26] R. Shi, J. Guo, and Y. Wang. A preliminary experimental study on optimal feature selection for product derivation using knapsack approximation. In *PIC*, pages 665–669, Piscataway, 2010. IEEE.

[27] N. Siegmund, M. Rosenmüller, M. Kuhlemann, C. Kästner, S. Apel, and G. Saake. SPL Conqueror: toward optimization of non-functional properties in software product lines. *Software Quality Journal*, 20(3-4):487–517, 2012.

[28] T. H. Tan, Y. Xue, M. Chen, J. Sun, Y. Liu, and J. S. Dong. Optimizing selection of competing features via feedback-directed evolutionary algorithms. In *ISSTA*, pages 246–256, New York, 2015. ACM.

[29] J. White, B. Dougherty, and D. C. Schmidt. Selecting highly optimal architectural feature sets with Filtered Cartesian Flattening. *Journal of Systems and Software*, 82(8):1268–1284, 2009.

[30] J. White, J. A. Galindo, T. Saxena, B. Dougherty, D. Benavides, and D. C. Schmidt. Evolving feature model configurations in software product lines. *Journal of Systems and Software*, 87:119–136, 2014.

[31] J. White, D. C. Schmidt, E. Wuchner, and A. Nechypurenko. Automatically composing reusable software components for mobile devices. *Journal of the Brazilian Computer Society*, 14(1):25–44, 2008.

[32] J.-Y. Yeh and T.-H. Wu. Solutions for product configuration management: an empirical study. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 19(1):39–47, 2005.

[33] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *PPSN*, pages 832–842, Berlin, Heidelberg, 2004. Springer.

[34] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *2th EUROGEN*, pages 95–100, Barcelona, 2001. CIMNE.