Discovering configuration workflows from existing logs using process mining

Belén Ramos-Gutiérrez[1] · Ángel Jesús Varela-Vaca[1] · José A. Galindo[1] · María Teresa Gómez-López[1] · David Benavides[1]

Abstract

Variability models are used to build configurators, for guiding users through the configuration process to reach the desired setting that fulfils user requirements. The same variability model can be used to design different configurators employing different techniques. One of the design options that can change in a configurator is the configuration workflow, i.e., the order and sequence in which the different configuration elements are presented to the configuration stakeholders. When developing a configurator, a challenge is to decide the configuration workflow that better suits stakeholders according to previous configurations. For example, when configuring a Linux distribution the configuration process starts by choosing the network or the graphic card and then, other packages concerning a given sequence. In this paper, we present COnfiguration workfLOw proceSS mIning (COLOSSI), a framework that can automatically assist determining the configuration workflow that better fits the configuration logs generated by user activities given a set of logs of previous configurations and a variability model. COLOSSI is based on process discovery, commonly used in the process mining area, with an adaptation to configuration contexts. Derived from the possible complexity of both logs and the discovered processes, often, it is necessary to divide the traces into small ones. This provides an easier configuration workflow to be understood and followed by the user during the configuration process. In this paper, we apply and compare four different techniques for the traces clustering: greedy, backtracking, genetic and hierarchical algorithms. Our proposal is validated in three different scenarios, to show its feasibility, an ERP configuration, a Smart Farming, and a Computer Configuration. Furthermore, we open the door to new applications of process mining techniques in different areas of software product line engineering along with the necessity to apply clustering techniques for the trace preparation in the context of configuration workflows.

Keywords Variability · Configuration workflow · Process mining · Process discovery · Clustering

1 Introduction

Variability models, such as Feature Models (FMs) (Galindo et al. 2018), describe commonalities and variabilities in Software Product Lines (SPLs) and are used along all the SPL development process. After an FM is defined, products can be configured and derived. We can find FM depicting a diverse set of domains such as Wordpress (Rodas-Silva et al. 2019), surveillance videos (Galindo et al. 2014b; Alférez et al. 2019) or Android systems (Galindo et al. 2014a) among others. In the configuration and derivation process, users select and deselect features using a *configurator*. A configurator (Galindo et al. 2015) is a software tool that presents configuration options to the users in different stages. An example of a configurator tool is KConfig (She et al. 2010) where developers can configure the Linux kernel with more than 12.000 configuration options.

An important aspect of a configurator is to determine the *configuration workflow* (Hubaux et al. 2013), i.e., the order in which features and options are presented to configuration stakeholders. For instance, when configuring the Linux kernel using KConfig (She et al. 2010), there can be different user configuration profiles depending on interests or skills. The configuration workflow used by a configurator can impact the user experience in the configuration process. Therefore, selecting a well-suited configuration workflow is a challenge. Up to now –to the best of our knowledge– the selection of a configuration workflow is made either intuitively or following the structure and properties of a variability model (Galindo et al. 2015; Varela-Vaca and Gasca 2013; Varela-Vaca et al. 2019b).

In this paper, we present COLOSSI, a framework that takes a feature model and a set of existing configuration logs and automatically retrieves configuration workflows. A configuration log is a set of configurations performed in the past in a given domain taking into account a configuration order. Our solution relies on *process mining* (Augusto et al. 2019) techniques. Process mining is a well-established area of business process management that uses different techniques to extract business processes from traces of execution. In our approach, we conceptually map a business process model to a configuration workflow and traces to configuration logs making it possible to reuse process mining techniques to infer configuration workflows.

Although using process mining can automatically retrieve configuration workflows, the results can be difficult to interpret to domain engineers to build a configurator. This is mainly because, very often, mined processes are "spaghetti-like" models in which the same activity needs to be duplicated (van der Aalst 2011). To illustrate the difficulty, Fig. 1 shows the result of directly applying process mining techniques to the ERP system presented in Pereira et al. (2018b) and detailed in Section 3.

The simplification of spaghetti processes is an open challenge and active research area in process mining (Augusto et al. 2019). The application of clustering techniques is used to retrieve a set of simple workflows instead of a single–complicated one. These techniques are used to divide the configuration log according to different aspects. There are different techniques to create clusters that facilitate the understanding of the discovered processes. The clustering can be expressed as an optimisation problem. Thus, exhaustive techniques have been proposed for finding the best possible clustering (Hompes et al. 2015), although with potential high time- and resource-consuming. This is the reason why other algorithms have been applied, such as hierarchical algorithm (Ferreira and Alves 2011; Makanju et al. 2008 2009), k-means algorithm (Song et al. 2008) or greedy algorithm (Greco et al. 20

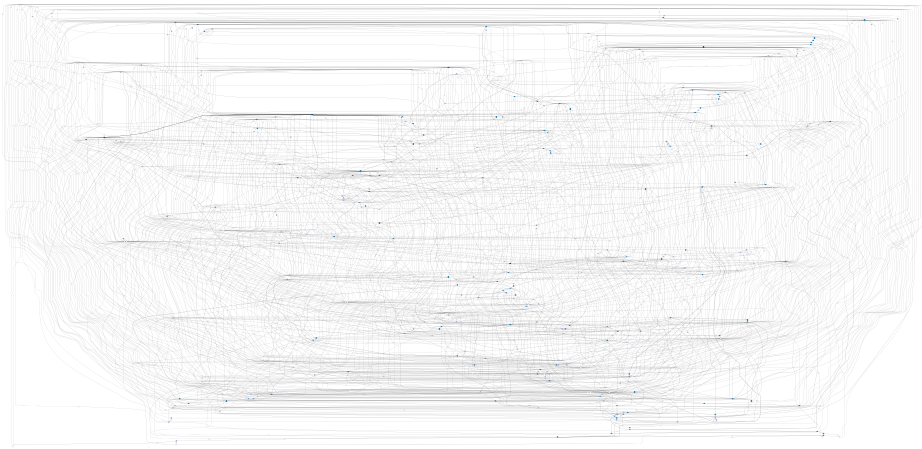


Fig. 1 Spaghetti process of the ERP presented in Pereira et al. (2018b)

De Weerdt et al. 2013). The wide types of possible algorithms to apply are caused by the most proper in each case will depend on the data log. To overcome this difficulty, COLOSSI provides the infrastructure to integrate various algorithms for clustering the traces involved in the configuration workflow, being possible the selection of the used algorithm according to the case and the necessities. Our solution takes information from the variability model as input and retrieves less complex configuration workflows that can assist the development of better configurators. Due to the size and complexity of the logs, the creation of clusters that facilitate the understanding of the discovered processes is not an easy task. Thereby, we propose four approximations based on different techniques to facilitate the creation of clusters trying to improve the distribution of the logs later used as input of the process discovery.

COLOSSI is validated using three different case studies: an ERP system, a smart farm and computer configurations taken from Pereira et al. (2016a, 2018a, b). Moreover, different clustering algorithms are applied to determine when each is more appropriate (greedy, backtracking, genetic and hierarchical algorithms). Results show that the metrics of the retrieved configuration workflows are improved depending on the clustering and the distribution algorithm of traces applied. Besides, the metrics help to support the hypotheses of reducing the complexity and enhancing the understandability of the retrieved configuration workflows. Moreover, an statistical analysis of metrics related to the distribution of traces is done to attempt to prove that the hypotheses that the selected techniques do not influence the obtained results (i.e., the algorithm has no impact on the distribution of traces).

This paper is an evolution of Varela-Vaca et al. (2019a) where the main contribution was the definition of COLOSSI as a framework to support the clustering of configuration logs to discover configuration workflows. In this paper, the previous proposal is enhanced by:

– The extension of COLOSSI to support different sort of clustering algorithms to apply.
– We extended the validation of COLOSSI with three additional case studies.

– The measurement of a set of metrics to compare the suitability of each algorithm of clustering to discover the configuration workflows that better fits the configuration traces generated by the users.

The remainder of this paper is organised as follows: Section 2 details the solution and concepts that grounds our proposal; Section 3 presents empirical results from analysing COLOSSI in different scenarios; Section 4 presents the related work and Section 5 presents concluding remarks and lessons learned.

2 COLOSSI: configration workflow process mining solution

COLOSSI combines a feature model and a configuration log to create a configuration workflow. Figure 2 shows an overview of the framework, in which, using a configuration log, it is possible to apply process mining techniques to derive a valid configuration workflow representing all the possible paths defined in the configuration logs. Very often, the resulting workflow follows the so-called spaghetti-style (van der Aalst 2011), being difficult to understand and manipulate. Nevertheless, it is important to remark that by applying some simplification techniques and extracting some metrics, these workflows could be exploited using automated process mining tools, to carry out many more additional analyses. Also, any generated configuration workflow can be already used to automatically build a configurator.

In an ideal process mining approach, as shown in path ① of Fig. 2, an event log is used to directly execute a process discovery task. However, in general, if no data preparation task is carried out in between, the results are usually unmanageable, and even more, when we are dealing with a context of high variability. That is why the data preparation phase in our proposal must necessarily include simplification mechanisms. To this end, also, to the usage of process mining techniques, we propose diverse handling and clustering methods to reduce and group similar configuration traces according to some properties. Those clusters can then be used again as an input of process mining techniques to obtaining a set of configuration workflows depending on the observed behaviour of the configuration logs. Those workflows could obtain better metrics concerning the original complex workflows of step

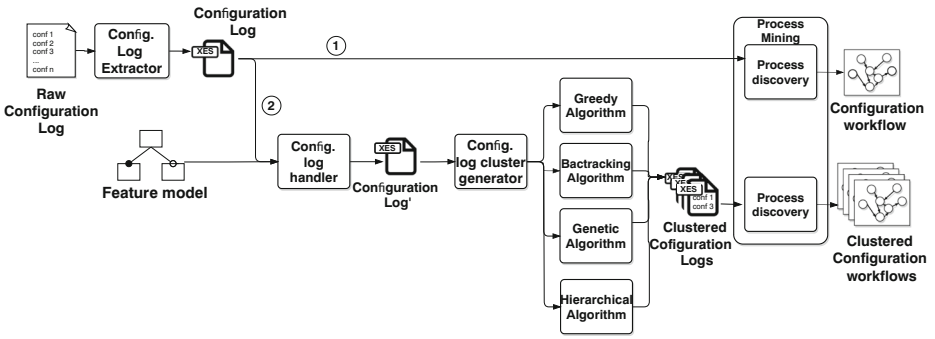


Fig. 2 COLOSSI solution overview

①. We conjecture that the resulting configuration workflows of step ② will better guide the domain engineers in the construction of a configurator as well as the analysis mentioned previously.

In the following, we describe the overall process of COLOSSI, detail its different phases and explain its implementation in general.

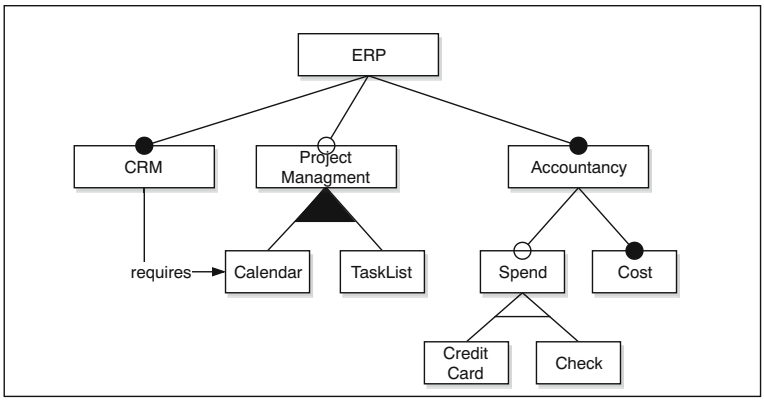
2.1 COLOSSI process

As previously mentioned COLOSSI combines a feature model and a configuration log to create configuration workflows. Consequently, it needs both as input.

A feature model is an arranged set of features that describes variability and commonality using features and relationships among them (Durán et al. 2017; Schobbens et al. 2007). FMs describe all the potential combinations of features. Figure 3 shows an excerpt of a feature model from the ERP domain where features are arranged in a tree–like structure and different relationships are established among them. FMs can be used to build configurators that are pieces of software that guide the configuration process while selecting and deselecting features. An example of a configurator is KConfig, a tool that helps to configure the Linux kernel. As an FM can define a configuration space defined by all the possible feature combinations, it can also define different possible configuration workflows that can be derived using the same FM.

To define a configuration log, we use some concepts that are used in the process mining area to describe events and traces, and we map those concepts to define a configuration log.

An event log is a multiset of traces:



Feature model

1st configuration: {*crm, tasklist, cost, accountancy*}
2nd configuration: {*project management, accountancy, credit card, calendar*}
3rd configuration: {*project management, check, crm, task list*}
4th configuration: {*crm, task list, calendar, credit card*}

Configuration log

Fig. 3 ERP domain based example

Definition 1 (Event Log). Let L be an event log $L = \{\tau_1, \cdots, \tau_m\}$ as a multiset of traces τ_i.

A trace is a tuple with an identifier and a sequence of events that occurred at some point in time:

Definition 2 (Trace). Let τ be a trace $\tau = \langle case\_id, \mathcal{E} \rangle$ which consists of a $case\_id$ which identifies the case, and a sequence of events $\mathcal{E} = \{\varepsilon_1, \cdots, \varepsilon_n\}$, ε_i occurring at a time index i relative to the other events in $\mathcal{E}$.

An event occurrence is a 3–tuple with an identifier of an activity that occurred at some timestamp and that can have additional information:

Definition 3 (Event occurrence). Let ε be an event occurrence $\varepsilon = \langle activity\_id, timestamps, others \rangle$ which is specified by the identity of an activity which produces it and the timestamps. It can store more information (i.e., states, labels, resources, etc.) which fall into the category of $others$ and which are not used in this approach.

In COLOSSI, we conceptually map elements from the feature modelling domain to the process mining domain as shown in Table 1. Concretely, an event log is conceptually a configuration log. A trace is an ordered configuration, i.e., a *configuration trace*, thus, it is a set of selected features that follow a given order. Finally, an event occurrence is a feature. Additionally, a feature can have more information like attributes associated with this feature, such as preferences, metrics or the like, which are not used in our proposal.

Thanks to this mapping, COLOSSI can be used in different scenarios to leverage process mining in variability management, like the one presented in this paper, which is the building of configurators, if the order of previous configurations are known and can be extracted. However, we envision other areas where process mining can be used to automate different tasks. Next, we describe those scenarios, also related to software product lines, from our experience and perspective:

- **Configurator building**. Up to now, configurators building is performed using manual mechanisms or, at most, using the information present in the variability model (e.g., tree traversal in feature models) (Lettner et al. 2019). With COLOSSI, we open the door to use existing configuration logs to build configurators. This novel approach can open the door to new ways of assisting configurators builders by using the generated configuration workflow to optimise configurators.
- **Data analysis**. From the generated configuration workflow it is possible to perform such analysis in terms of graph metrics. Deadlocks identifications, misalignment analysis, metrics extraction –to just mention a few– are areas where process mining techniques can be useful.

Table 1 Mapping concepts

Process mining	Product Line
Event log	Configuration log
Trace	Configuration trace
Event occurrence	Feature

- **Testing**. From the data extracted in the former item, it could be possible to define new sampling techniques (Thüm et al. 2014) that can improve the identification of bugs or feature interactions in existing product lines.
- **Variability reduction**. One of the challenges for companies that develop software product lines is variability reduction (Bosch 2018). While variability is a must in a software product line approach, it is always difficult to find a trade-off between a high degree of variability and systematic management of such variability. In this context, experts claim for techniques and tools to reduce variability while preserving configurability. Process mining techniques presented in this paper can be a first step towards defining tools to assist in the decision of variability reduction.
- **Reverse engineering**. One of the inputs used when reverse engineering feature models are configurations (a.k.a. product matrix). We envision that the techniques described in this paper can be used in reverse engineering of variability models. For instance, the generated configuration workflow can be analysed to better guide reverse-engineering algorithms.

2.2 Detailing COLOSSI steps

Once the overall process of COLOSSI has been described. We proceed to deepen the operation of each of its parts, following the structure described in Fig. 2.

2.2.1 Configuration logs extractor

A configuration log is composed of a set of configuration traces where each configuration trace encodes not only the features of a configuration but the timestamps indicating when each feature was selected. In a raw configuration log, we can find a diversity of meta-information among the selected or deselected features. Moreover, this meta-information can be presented in an unstructured or structured fashion that must be properly extracted and transformed (Valencia-Parra et al. 2019a, b) to obtain the traces in the correct format.

In this first step, we take as input a raw configuration log and output a set of configuration traces. Therefore, we need to i) search for the meta-information encoding the timestamps for each feature. Note that this might not be explicit and can be provided using other mechanisms (e.g., line numbers in a plain text format); ii) use this meta-information to represent the feature selection order, and; iii) store the set of configuration traces in a format that can be used throughout the configuration workflow retrieval process (e.g., XES serialization 2016). After this, we end up with a set of configuration traces that represent the selection order used by the users to configure the systems. However, there might be non-valid configurations and other erroneous configurations with respect to domain information. Practitioners have to make decisions to this regard depending on the kind of input logs that want to be considered in the next steps of the discovering process.

2.2.2 Configuration logs handler

At this step, the configuration log might contain non-valid configurations, erroneous partial selection of features among other domain-related errors such as those depicted in Felfernig et al. (2018). To remove clutter and noise out of the workflows, users might prefer to remove such information from the configuration log. This cleaning step consists of removing the

wrong selection of features (a.k.a non-valid partial configurations) as well as generate metrics that can be later exploited to optimise the workflow retrieval process. For example, the use of atomic-sets to complete partial configurations.

Depending on the expected workflow usage, domain engineers have to define the meaning of a valid configuration and the metrics to rely on. For example, an SPL engineer might consider only configurations with complete assignments of features to develop a configuration while other might find interesting to consider the partial assignments (i.e., to configure only the variability part of the product line, keeping aside the common parts).

COLOSSI (Fig. 2), *Process mining - process discovery* module enables to read an event log and generates a process model that fits these traces as shown in Fig. 4. In the case of the variability context, a *configuration log* is read and a *configuration workflow* is obtained using the same techniques used for classical process mining. As depicted in Fig. 2, the process discovery can be applied to a single configuration log or a set of them.

2.2.3 Configuration logs cluster generator

Configuration processes can have a high degree of variability, especially when the configuration order is defined by human decisions. The application of process discovery in this type of scenarios tends to produce spaghetti-like processes, making it necessary to apply pre-processing techniques. Configurability contexts are especially variable in relation to the executed activities derived from high human intervention. Thereby, we propose to divide the traces into subsets, to model different profiles of users, thus avoiding the discovery of non-user understandable processes. In these contexts where process discovery is used to infer spaghetti-like processes, clustering techniques such as a pre-processing step are frequently applied (Hompes et al. 2017). We propose the definition of the suitable number of clusters and the division of the configuration traces into multiple clusters before the application of a process discovery method to adapt the solution to configuration tasks. This division leads to

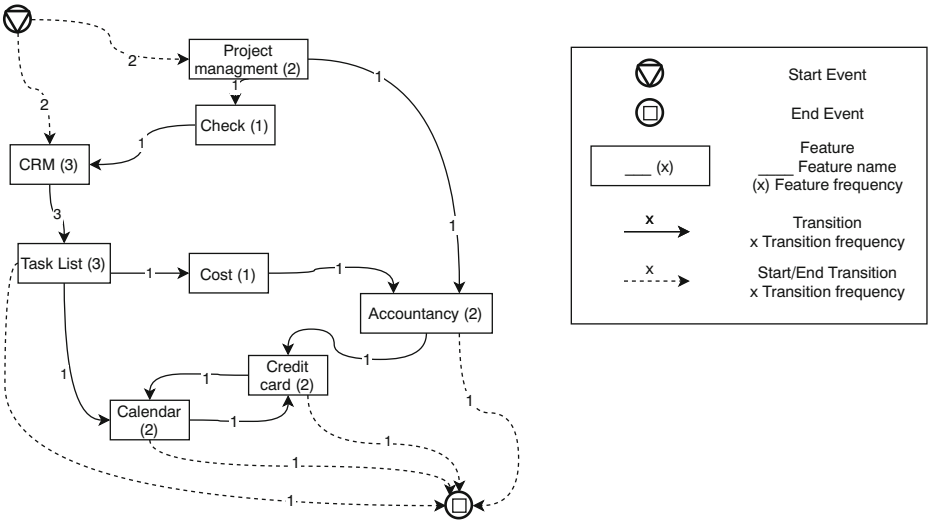


Fig. 4 Process discovered for configuration log of ERP domain based example

discover configuration workflows with more quality. This section describes what a cluster is and the metrics (e.g., entropy) used to divide the traces among a number of clusters.

Definition 4 (Number of Clusters). Let $k\text{-}cluster$ be the optimal number of clusters in which the traces must be grouped.

Like in any other clustering problem, the selection of the most optimal value of $k\text{-}cluster$ is one of the first issues to deal with, since, before being able to execute any grouping algorithm, it is necessary to previously know in how many groups the data must be distributed, in such a way that the distribution is optimal based on an established criterion.

This is a widely researched topic in which many solutions have been proposed and applied to other scenarios. In our proposal, 17 different indicators are used as reference to choose the best number of clusters: kl (Krzanowski and Lai 1988), ch (Caliński and Harabasz 1974), hartigan (Hartigan 1975), cindex (Hubert and Levin 1976), db (Davies and Bouldin 1979), duda and pseudot2 (Duda et al. 1973), ratkowsky (Ratkowsky and Lance 1978), ball (Ball and Hall 1965), ptbiserial (Milligan 1980, 1981), frey (Frey and Van Groenewoud 1972), mcclain (McClain and Rao 1975), gamma (Baker and Hubert 1975), tau (Rohlf 1974), dunn (Dunn 1974), sdindex (Halkidi et al. 2000), sdbw (Lebart et al. 2000).

Being L a configuration log composed of a set of configuration traces (i.e., $[\tau_1, \cdots, \tau_m]$), a cluster is a subset of configuration traces from L that complies certain properties.

Definition 5 (Cluster of Configuration Traces). A partition of a set of configuration traces is a set of non empty and disjoint subsets C=$\{c_1, \ldots, c_n\}$ of configuration traces, where $\bigcup_{c \in C} c = L$ and $\forall c_i, c_j \rightarrow c_i \cap c_j = \emptyset$.

The distribution of configuration traces between various clusters depends on the purpose of the practitioners. In our case, the goal is to group the more similar configuration traces. COLOSSI understanding of 'similar' is related to both features and transitions involved in the logs. Understanding as transition any edge present in the workflow that comes out of an activity.

For this reason, we adapted the classical information entropy metric (MacKay 2002) by introducing two different custom entropy metrics for clustering in the configuration context:

- *Entropy-features ($S_{features}$)* of a cluster: a metric which measures the similarity between traces according to the features that belong to the same cluster. Thus, it is the ratio between the number of features that do not appear in all configuration traces ($features_{nat}$) and the number of different features in all the configuration traces ($features_{diff}$):

$$S_{features} = \frac{|features_{nat}|}{|features_{diff}|} \tag{1}$$

- *Entropy-transitions ($S_{transitions}$)* of a cluster: a metric which measures the similarity between traces according to the transitions that belong to the same cluster. Thus, it is the ratio between the transitions that do not appear in all configuration traces ($transitions_{nat}$) and the number of different transitions in all the configuration traces ($transitions_{diff}$):

$$S_{transitions} = \frac{|transitions_{nat}|}{|transitions_{diff}|} \tag{2}$$

In order to illustrate the calculation of entropies per cluster, the $S_{features}$ and $S_{transitions}$ for the *Cluster 1* and *Cluster 2* of Fig. 5 are determined in Table 2. Reminding that we

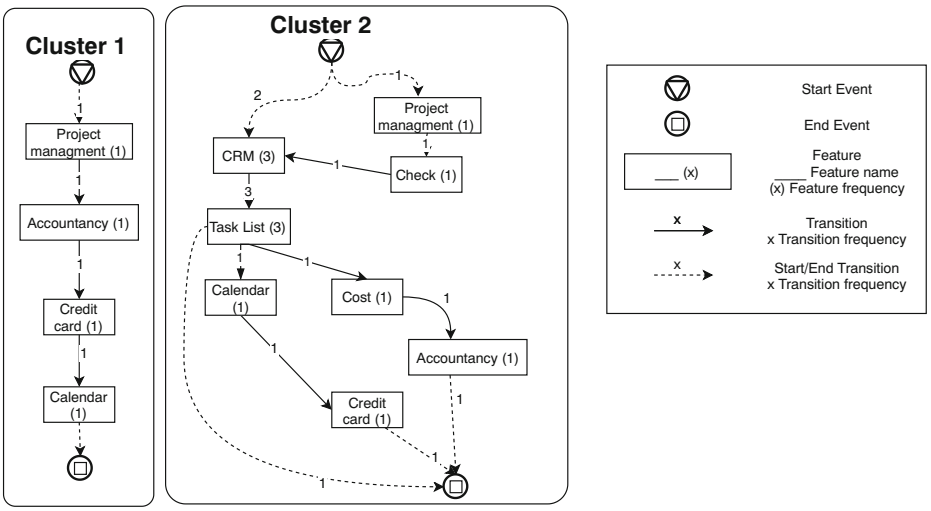


Fig. 5 Clustering for the ERP excerpt using the *Entropy-features*

consider as transition any edge present in the workflow out of an activity. In the case of Entropy-features, in the first cluster, the result is 0, since there is only one trace and there is no possibility of comparing the features. In the second cluster, we found six features that only occur in one of the two traces (*Project Management, Check, Calendar, Cost, Accountancy and Credit Card*), out of the eight that are totalled. Similarly, the presence of a single trace makes Entropy-transitions 0 in the first cluster, while in the second cluster we count nine transitions that only occur in one of the two traces (*Project Management - Check, Check - CRM, Task List - End, Task List - Calendar, Task List - Cost, Calendar - Credit Card, Cost - Accountancy, Credit Card - End, Accountancy - End*), among the ten possible ones.

Note that the range of the entropy is [0..1]. The values of entropy that are close to zero represent more similar traces, whilst when they are close to one represent that there are different features involved in the traces of the cluster. The best configuration of clusters obtained from a set of configurations traces is the one that has been partitioned into as many clusters as indicated by the optimal value of $k\text{-}cluster$, and which, in turn, minimises the summation of the entropy of all clusters. The challenge is how to obtain the best configuration of clusters as a pre-processing of process discovery.

To find out the best configuration traces divided into clusters, minimising the entropy of the resulting clusters, different algorithms can be used. In accordance with Jain et al. (1999) *clustering* provides an unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). *Clustering* (Kobren et al. 2017) brings together a large set of algorithms that can be classified in different ways according to the point of view necessary in the case of study.

Table 2 Entropies for the clusters of the Fig. 5

	Entropy-features	Entropy- transitions
Cluster 1	$\frac{0}{4} = 0$	$\frac{0}{4} = 0$
Cluster 2	$\frac{6}{8} = 0,75$	$\frac{9}{10} = 0,9$

Ideally, all clustering algorithms seek to group the information into clusters as homogeneous as possible. This implies that the distances between elements in the same cluster must be minimal and that, in turn, the distances between elements in a different cluster must be maximum. In order to carry out this operation, clustering algorithms usually generate, during their execution, what is known as distance matrices. These distance matrices are calculated according to different criteria which elements should be grouped and which should not. In this area, we find widely used methodologies to create a distance matrix, such as euclidean, manhattan, etc. (Grabusts et al. 2011). But, because our approach is based on minimising the sum of entropies, we need the clustering algorithm to be aware of this when grouping information. For this purpose, the entropy matrix is constructed previously and not during the execution of the algorithm, being just a square matrix containing the entropy value between each trace pair following the definition of entropy above. It is equivalent to the distance between each pair of traces according to our criteria. For this reason, our entropy matrix will be used as a distance matrix in our clustering process when necessary.

However, as previously mentioned, before the application of the clustering algorithms, it is necessary to determine the optimal number of clusters. Classically, the algorithms to determine the number of clusters (optimal $k\text{-}cluster$) also build distance matrices. For this reason, our approach will also use the entropy matrix as the distance matrix to calculate the optimal $k\text{-}cluster$. To find it, we obtain the optimal value of $k\text{-}cluster$ for each of the 17 indicators. Nevertheless, the values of $k\text{-}cluster$ derived from the different indicators obtained could be very dissimilar in some cases, therefore, frequently the optimal $k\text{-}cluster$ is selected from the most frequent value, in other words, the most voted $k\text{-}cluster$ is the number of optimal clusters by the indicators. This is the reason why we use a dendrogram to help approximate the optimal $k\text{-}cluster$, since if after the votes made by the indicators, a clear consensus is not reached, the user can use the dendrogram to be able to visually interpret how the groups would turn out. So, if for example, the indicators propose to divide the information into 2 or 3 clusters, without consensus, the user can decide, based on what is observed in the dendrogram, to work with a $k\text{-}cluster$ value of 2. Figure 9 shows an example of dendrogram.

In our proposal, we compute the different values of $k\text{-}cluster$ in a range between [0-10] for each indicator using the hierarchical algorithm explained in Section 2.2.3 and when the most voted, for a specific case study and entropy, is selected, that $k\text{-}cluster$ value is used by all the clustering algorithms in our proposal. For the example presented in Fig. 3, we have obtained an optimal $k\text{-}cluster$ value of 2.

Obviously, the entropy matrix and the $k\text{-}cluster$ determinations represent two of the weak points of our proposal, since, without an entropy matrix, it is not possible to determine the $k\text{-}cluster$ and without it, it is not possible to execute any clustering algorithm. In fact, as it will be seen in Section 3, some results have not been possible to obtain due to the impossibility of computing the entropy matrix, as is the case of Smart Farm with entropy by transitions.

The selection of the most suitable distribution of traces among clusters to discover the later process, and therefore, the value of $k\text{-}cluster$, is crucial since an incorrect distribution of the traces will produce non-understandable processes. In addition, there exists a high computational complexity to find out the optimal distribution. As a first approach, we propose to use a trivial algorithm (greedy), a complete algorithm (backtracking), and two approximation algorithms (genetic and hierarchical). For this reason, the comparison of the application of four algorithms greedy, backtracking, hierarchical, and genetic algorithm is proposed to ascertain the most suitable distribution. In general, to find a solution

to a described problem, each algorithm must define how solutions are created. This implies the definition of the following five components:

- **A candidate set** used to create a solution, this is the list of disordered traces that confirm the problem that will be assigned to a cluster.
- **A selection function**, which chooses the best candidate among the candidate set to be added to the solution. It chooses the cluster to be assigned for a trace.
- **An objective function**, which assigns a value to a solution, or a partial solution used to compare the adjustment of the solutions. The objective function aims to minimise the entropy of the traces assignment among the cluster.
- **A solution function**, which indicates when a complete and most appropriate solution is discovered. It implies the assignment of every trace to a cluster.

Greedy algorithm Greedy algorithms are frequently applied in the case which the computational complexity of the problem is very high, as is in the distribution of thousands of traces in various clusters. A greedy algorithm is a strategy that evaluates each decision to reach the optimal solution only taking into account the current state, with the goal of this eventually leading to a globally optimum solution. This implies that the greedy algorithm selects the best solution in each moment without regard for consequences in a far future, only regarding the next candidate. The use of a greedy algorithm cannot assure to achieve the best solution, however, the solution can be found in a short time.

We propose the ordered assignation of each trace to a cluster that minimises the global defined entropy.

Let be $\{t_1, \ldots, t_i, \ldots, t_n\}$ the set of traces and $\{c_1, \ldots, c_e, \ldots, c_k\}$ the set of clusters that represent the candidate set for each trace. The selection function choose the best cluster c_e, that minimises the entropy after the assignment (objective function). At the start, the entropy is zero, when a new trace, t_i, is assigned to a cluster, for instance c_e, a new entropy e_i is obtained. In each step of the algorithm a trace is assigned to a cluster, if the set of traces $\{t_1, \ldots, t_{i-1}\}$ have been already assigned to any clusters and being the current entropy e_{i-1}, when a new trace (t_i) is assigned to any cluster (c_e) a new entropy $e_i(t_i, c_e)$ will be obtained.

The trace t_i is assigned to c_e iff $\forall c_d \mid c_d \in \{c_1, \ldots, c_e, \ldots, c_k\}\ e_i(t_i, c_d) \geq e_i(t_i, c_e)$

Firstly, applying the proposed greedy algorithm to the ERP domain example with two clusters, the traces distribution obtained the processes as shown in Fig. 6. Two of the four traces have been grouped in each cluster according to $S_{features}$. Secondly, three traces have been grouped in one cluster and one trace in another concerning the $S_{transitions}$. The second distribution is similar to the one shown in Fig. 5. The process models depicted in the figure are in BPMN[1] format.

Table 3 shows the resulting entropies for each algorithm, entropy type and distribution. The entropy value present in the table for each pair of entropy type and algorithm is calculated as the mean of the partial entropies obtained by each cluster. As previously mentioned, the entropies help to understand how similar are the configuration traces grouped into the clusters concerning features and transitions. The values close to zero are the most desirable. In this case, the backtracking and hierarchical with the entropy of features returned the best results of distributions. Note that, the drawback of backtracking is affordable due to the small size of the problem.

[1]BPMN: Business Process Model and Notation

Table 3 Entropy per algorithm and entropy for the ERP domain example of Fig. 5

Greedy Algorithm		Backtracking Algorithm		Genetic Algorithm		Hierarchical Algorithm	
Entropy features	Entropy trans.	Entropy features	Entropy trans.	Entropy features	Entropy trans.	Entropy features	Entropy trans.
0.625	0.9166	0.4375	0.9166	0.5	1	0.15	0.83

Backtracking algorithm A technique to analyse every possible solution is backtracking algorithms, they solve problems recursively to building a solution incrementally, one decision at a time, such as the greedy algorithm but selection every assignment and removing those solutions that fail to optimise the function.

The complete analysis of every solution implies an exponential computation time, making backtracking algorithms a high time-consuming technique. For this reason, we applied several improvements to reduce the possibilities:

- Apply a branch and bound mechanism to avoid the exploration of some branches, where several solutions from the greedy algorithm are used to bound.
- Avoid symmetric solutions, to analyse equivalent solutions (assignments) with the same number of traces together but in different clusters.

Let be $\{t_1, \ldots, t_i, \ldots, t_n\}$ the set of traces and $\{c_1, \ldots, c_e, \ldots, c_k\}$ the set of clusters that represent the candidate set for each trace. The selection function chooses every cluster c_e (if more than one has no traces assigned yet, only one of them will be analysed to avoid the creation of equivalent solutions), avoiding those that generate a higher entropy that the best solution found until the moment. At the beginning, the entropy is the one obtained from the greedy algorithm, when a new trace t_i is assigned to a cluster c_e, a new entropy e_i is

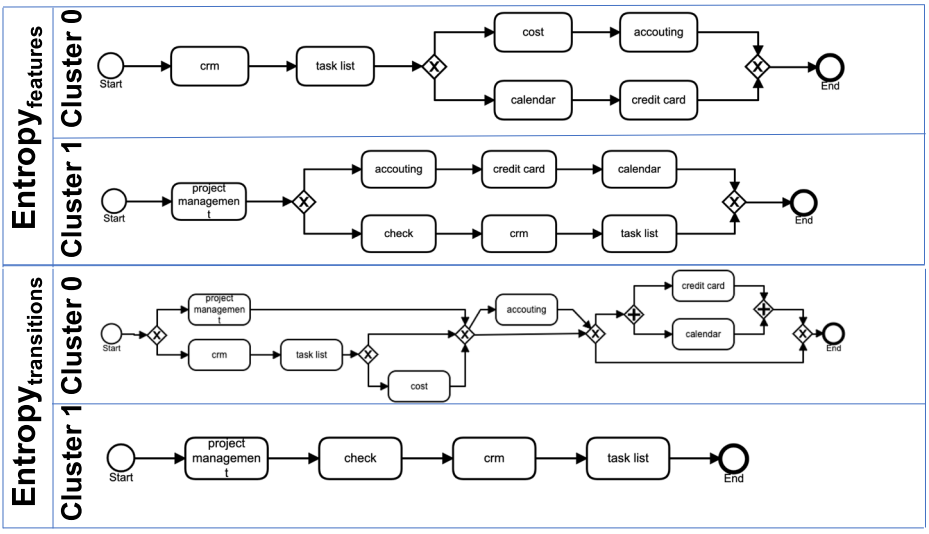


Fig. 6 Processes obtained per clusters and entropies using a greedy algorithm

obtained. In each step of the backtracking algorithm, a trace is assigned to a cluster, then if $\{t_1, \ldots, t_{i-1}\}$ have been already assigned to the clusters and being the current entropy, e_{i-1}, when a new trace (t_i) is assigned to a cluster (c_e), a new entropy $e_i(t_i, c_e)$ will be obtained. If the new entropy $(e_i(t_i, c_e))$ is a worse entropy than the best found until the moment $(e_{bestSol})$, the candidate will be skipped, a new trace will be assigned otherwise.

For all cluster c_d a solution is created where a trace t_i is assigned to c_d iff $e_{bestSol} > e_i(t_i, c_e)$. Each new created solution will be used recursively in the backtracking algorithm.

Applying the proposed backtracking algorithm to the ERP domain example, with two clusters, the traces distribution obtained is shown in Fig. 7. In this case, the models obtained for both entropy are the same, for this reason, only the models for $S_{features}$ are shown. The *Cluster 0* grouped one trace and the *Cluster 1* grouped the rest. Regarding the entropy of features, this is the one of best distribution (i.e., assignment of traces to clusters). This distribution is not obtained by the greedy solution, but the backtracking ensures that it is the optimum solution in terms of the objective function.

However, even applying the aforementioned improvements, it has not been possible to obtain results in our study cases with this algorithm, since, due to their size, the search could not finish in an acceptable time as we will show in Section 3.

Genetic algorithm Genetic algorithms provide a trade-off between high computational requirements and the necessity to find the best solution. Genetic algorithms are commonly used to generate high-quality solutions that optimise the problems by relying on bio-inspired operators such as mutation, crossover and selection by using a population of candidate solutions, where it is necessary to define:

- A genetic representation of the solution domain is a list of assignments, l_i. Starting from a set of traces $\{t_1, \ldots, t_i, \ldots, t_n\}$ and from a set of clusters $\{c_1, \ldots, c_i, \ldots, c_n\}$, it is possible to build another list whose size is equal to the number of traces, in which each l_i value will imply that the t_i trace has been assigned to the c_i cluster. For instance, let be $\{t_1, t_2, t_3, t_4, t_5\}$ a set of traces and 2 the number of clusters to distribute them. A possible assignment would be [0, 1, 0, 1, 1], which means that the traces t_1 and t_3 belong to cluster 0 and traces $\{t_2, t_4, t_5\}$ to cluster 1.
- A fitness function to evaluate the solution domain seeks to group the maximum number of traces according to their entropy and is a maximisation function based on the entropy value of the assignment. It is calculated as the inverse of entropy (*Entropy-features or Entropy-transitions*) plus one to which we add three weights according to the advantages of the assignment. Each of the weights refers to one goodness of allocation: (g_1)

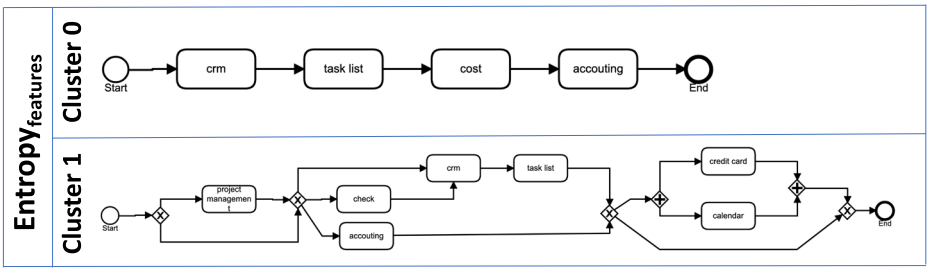


Fig. 7 Processes obtained per clusters and entropies using a backtracking algorithm

there are not empty clusters; (g_2) the number of traces assigned to each cluster is balanced, and; (g_3) the similarity between the traces within each group is also balanced. With this fitness function the individual will be better, the higher fitness value it gets. We preserve the same objective function for all algorithms, which is the entropy, but in the case of the genetic, just the entropy as the objective function is insufficient, because there are situations in which clusters can be empty or unbalanced. To adjust better the solution, we included some weights. These weights are discrete values and allow us to penalise or favour a certain individual regarding its characteristics. These weights are not necessary for the rest of the algorithms applied, because, by their construction, they take care that the events considered in functions g_1, g_2, g_3 do not occur. For instance, the backtracking algorithm includes mechanisms to avoid those solutions that include empty clusters or in which the clusters are unbalanced. However, the genetic algorithm we apply does not possess these abilities and requires the objective function to explicitly penalise candidates that would not be acceptable solutions. As these weights are discrete values, they are customisable and adaptable to the context of each problem and user, so that the user can decide which properties want to enhance or soften in the allocations.

$$fitness = \frac{1}{1 + entropy} + g_1 + g_2 + g_3 \tag{3}$$

- Different rates will determine the evolution of successive populations during the execution of the algorithm, and therefore, of the final solution. the maximum population growth has been limited to 600,000 generations, each one with a size of 80 individuals. We have used an elitism rate of 30%, a crossover rate of 80% and a mutation rate of 70%. This selection of the parameters is due to different trial and error tests, in which this parameterization has yielded the best results. The trial and error tests were performed with all the cases to find the best general parameterization. Once the best combination of parameters was determined, the same one was used for all the experimentation. The designed test consists of the execution of each case several times with different parameters configuration and the calculation of the average execution time (in seconds) spent in each.
- Crossover policy: one-point crossover, where a random crossover point is selected and the first part from each parent is copied to the corresponding child, and the second parts are copied crosswise.
- Mutation policy: binary mutation, in which randomly one gene is changed.

Applying the proposed genetic algorithm to the ERP domain example with two clusters, the traces distribution obtained the next processes as shown in Fig. 8. In this case, the genetic algorithm seems to balance quite well each cluster considering both entropy distributions. In both cases, each cluster groups two traces. However, as will be seen later in Section 3, it is not the one that offers the best results in terms of entropy.

Hierarchical algorithm The well-known *hierarchical agglomerative clustering* algorithm is based on the work by Ward (1963) as the combination of hierarchical and agglomerative clustering.

As it was previously stated, we use this algorithm with our entropy matrix as the distance matrix. From here, it is possible to decide which grouping criteria the algorithm must follow when creating clusters. Examples of this methods are *single-linkage*, *complete-linkage* (Hubert 1974), *average-linkage* (Murtagh 1983), and *Ward* (Ward 1963). Being the last one, the one used in our solution, since it is one of the most used in practice and it has proven

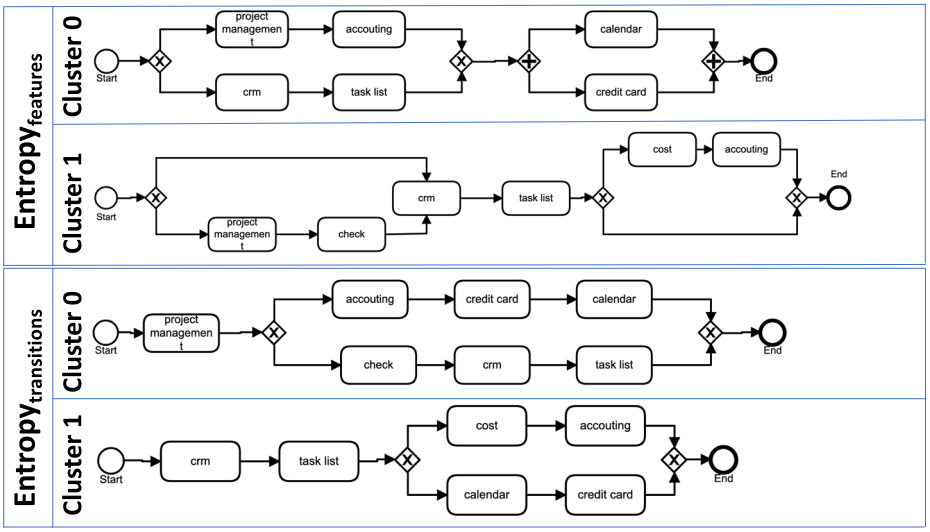


Fig. 8 Processes obtained per clusters and entropies using a genetic algorithm

to be more accurate obtaining the optimal distribution, than the previous ones and avoiding that the partition in groups distorts the original information (Kuiper and Fisher 1975).

On the first hand, *hierarchical clustering* is defined as a procedure to form hierarchical groups of mutually exclusive subsets, each of which has members that are maximally similar concerning the specified characteristics. In the same study, authors define the process as "*assuming we start from n sets, it permits their reduction to* $n-1$ *mutually exclusive sets by considering the union of all possible* $\frac{n(n-1)}{2}$ *pairs and selecting a union having a maximal value for the objective function*".

On the other hand, *agglomerative clustering* is an algorithm that starts from the assumption that each element constitutes a cluster by itself (singleton) and it successively merges these singletons forming clusters until a stopping criterion is satisfied which is also determined by the objective function.

The elements used in our solution are based on both entropies presented (features and transitions), combined with the objective function Ward's minimum variance method (Ward 1963). This function aims to minimise the sum of the squared differences within all clusters, which means, a variance-minimising approach. Figure 9 shows the obtained dendrogram[2] for the example in Fig. 3 by using Entropy-features. In the mentioned dendrogram, a horizontal line that intersects perpendicularly with two lines, appears, symbolising partitioning into two groups: the first one, corresponding to the first cluster, which contains only one trace, and the second, corresponding to the second cluster, which contains the three remaining traces. The positioning of the horizontal line is determined by the optimal $k\text{-}cluster$ value since this line must be placed at a height where it intersects with as many vertical lines like the $k\text{-}cluster$ value is.

Once the clusters have been generated, it is possible to create a new log for each one of the clusters and to discover the workflows corresponding to them. In the same figure, the

[2] *Dendrogram* that is a branching diagram which represents the arrangement of the clusters produced by the corresponding analyses

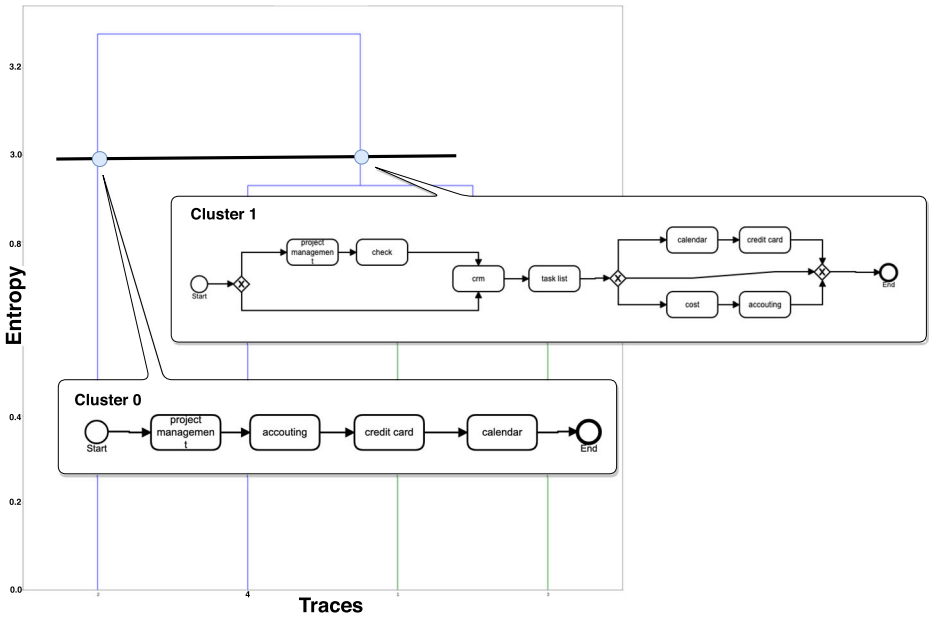


Fig. 9 Clustering for the ERP excerpt using the *Entropy-features* in an agglomerative clustering algorithm

process models obtained according to the clustering division based on entropy-feature and transition have been included, since the same models are obtained using both entropies.

2.2.4 Configuration workflow discovery with process mining

As it could be seen in Fig. 2, once the clusters have been obtained, the next step is to generate the logs for each cluster and execute the process mining on each one of them.

Process mining is a family of techniques based on event logs that can be categorised as process discovery, conformance checking and enhancement (van der Aalst 2016). In this paper, we are focused on the use of *process mining* to analyse the configuration logs for the discovering of configuration workflows based on user experiences. Process discovery in process mining brings together a set of algorithms to generate a workflow process model that covers the traces of activities observed in an organisation (Maruster et al. 2002). The evolution of algorithms during last decades has allowed the discovery of complex models that are able to involve not only the activities executed in the daily work of companies but also the persons who execute them and the used resources.

Process mining is an important topic that has been well received by the enterprises, bringing about the evolution of the research solution tools (e.g., ProM (van Dongen et al. 2005)) to commercial solutions (e.g., Disco[3] and Celonis[4]). This facilitates its applicability to several contexts and areas, although variability has been out of the scope of these techniques before our proposal.

[3] https://fluxicon.com/disco/
[4] https://www.celonis.com/

The purpose of obtaining these configuration workflows is to assist the user who performs the configuration. So that, thanks to this help, it is possible to know what actions must be taken before others, which of them can be carried out in parallel, with which activity should start the configuration, what task should be performed after the current one in order to do it optimally, etc. In this way, following our example, a user who is facing the configuration of an ERP system could know that he could start by configuring the CRM, after that, the task list, followed by the calendar, ending with the credit card.

Process discovery in process mining uses a set of traces similar to the configuration log shown in Fig. 3, to obtain a model that covers the possible traces. Figure 4 shows the process discovered by Disco tool-suite of the example, which covers every possibility configuration trace. The relational patterns among the definition of the features become part of the model. For example, two features can be the first in the traces (*CRM* or *Project management*) or after *CRM* always *Task List* is selected. Figure 4 also shows the number of traces that are represented by each transition as labels of the edges, giving information about the importance of each part of the traces in the obtained model.

2.3 COLOSSI implementation

COLOSSI is supported by the implementation of a framework which is made up of one module for each activity of the process shown in Fig. 2.

1. *Configuration log extractor* is a piece of a software module which takes a set of raw configuration log (including timestamps) in a semi-structured format and returns an XES file. It corresponds to the activity with the same name in Fig. 2.
2. *Configuration log handler* is another piece of software which takes a FM and an XES log as input. First, apply a set of operations over the FM as described in Section 2.2.2. Then, a data cleaning is carried out over the XES log to get a filtered configuration log. The output of this connector is a new XES log with the filtered configuration log. It corresponds to the activity with the same name in Fig. 2.
3. *Cluster generator* is a Python/R module which takes an XES log file which is translated into a matrix. This matrix enables the entropy calculation and the optimal $k\text{-}cluster$ that will be used when some of the clustering algorithms are applied to determine the clusters. A new XES log file is generated for each cluster that composed the final output of this component. This module is represented in Fig. 2 by the activity called Configuration log cluster generator, the different clustering algorithms that can be applied and the generation of new the logs.
4. *Discovery connector* is a piece of software which gets the XES file logs of each cluster and automatically feed the ProM to discover the process models utilizing the *Inductive Miner*. The output of this component is a process model in Petri-net or BPMN format. This module is depicted in Fig. 2 by the Process Discovery activity.

The reason for using ProM in our proposal is because it is the free framework most used by the academy. In it, the Process Mining community adds its contributions as plug-ins, and therefore, it is always up-to-date with new solutions to problems that are under research. On the other hand, it is a very powerful tool that contains all the Process Discovery algorithms, and its construction allows it to be used inside of another software. The selection of the Inductive Miner algorithm is reasoned by the great capacity of this algorithm to fully adjust to the behaviour observed in the log. As explained in Section 3.4, it is one of the most robust algorithms, which produces better results when facing non-synthetic logs, ensuring the correctness of the models obtained. Its parameterization has been adjusted to 100% of

fitness, in such a way that all the configuration workflows obtained will completely cover all the traces contained in the log, representing all the possible behaviours that have been collected.

All the resources, thus, configuration logs, the XES files, the workflows discovered, the source code of the COLOSSI framework (i.e., git repository), and a Jupyter notebook that are employed in this work are freely available at[5] http://www.idea.us.es/empiricalsoftware/. The notebook is self-explanatory and allows users to work interactively by executing step-by-step instructions to get the clusters.

3 Evaluation

In this section, we present the evaluation of COLOSSI. Concretely, the evaluation of the algorithms of clustering detailed in Section 2 to different configuration logs obtained from three real scenarios. Moreover, the suitability of each algorithm according to a set of metrics are analysed.

3.1 Experimentation data

In order to analyse the applicability of our example in a configuration real scenario, we propose three different scenarios where the creation of the configuration workflow can be obtained from a set traces: a real ERP (enterprise resource planning) configuration, smart farming, and a computer configurator. Please note that the number of possible products depicted by the models is an interesting metric to understand the usefulness of this approach when performing variability reduction. However, our approach relays on the number of real configurations to obtain the configuration workflows. This is further explained in Section 3.5.

3.1.1 Enterprise resource planning

This dataset (Pereira et al. 2016b) reflects the information of a real ERP variability among a set of configuration logs. The ERP feature model has 1920 features and 59044 cross-tree constraints. Also, the configuration log consists of 35193 event occurrences that represent a total of 170 different configuration traces with an average of 207 features per configuration trace.

3.1.2 Smart farming

This dataset represents several e-commerce transactions from the agribusiness domain (Pereira et al. 2016a). Concretely this model consists of 2008 features. It contains features targeting final customers (around 10%) and business to business (around 90%). Each log consists of a real configuration developed for a concrete user or business having a total of 5749 logs and up to 10^9 possible configuration due to the lack of cross-tree constraints.

[5] https://doi.org/10.5281/zenodo.3574053

3.1.3 Computer configuration

This dataset represents the variability existing in a Dell laptop (Pereira et al. 2018a). It reflects features such a processor or display, among others. Concretely this feature model represents 68 features with seven cross-tree constraints that encode up to 10^9 configurations according to their creators.[6] The configuration log is composed of 42 configurations of such a model.

3.1.4 COLOSSI setup

In this section, we detail each task of the framework presented in Fig. 2.

Configuration log extractor The input data of the configuration examples (i.e., ERP, farming and computers) are represented in CSV files with two elements (columns), the configuration id and the $feature$ that is configured. Note that a feature can appear in one or more traces, but no more than once in the same trace. Then, the timestamp required to extract the traces was taken by the line number in which the features were appearing throughout the file in sequential order. This is, we assume that the timestamps were implicit based on the order of appearance (i.e., line numbers). In all cases, we iterated over all configurations extracting the orders. Finally, we transformed them into a more standard format for traces. Concretely we use in our solution the IEEE Standard for eXtensible Event Stream (XES) 2016. This is a standard to serialise, store and exchange events data, that is commonly used in process mining tools.

Configuration log handler To clean up the set of configurations retrieved by the extractor we decided to consider only valid partial and full configurations. This filtering operation is performed by using the FaMa framework (Felfernig et al. 2018), with the intention of keeping only valid configurations in the log, to avoid introducing noise into the results as a consequence of the use of invalid traces.

After filtering, the valid partial configurations using automated analysis (Galindo et al. 2018), for each case are: for ERP ended up considering 61 configuration traces from the initial set of 170, Farming example initially has 5749 traces and 919 after the filtering, and dell configuration contained the full 42 of the initial set (i.e. all traces were valid).

Determining the number of clusters As commented previously, before applying any grouping algorithm, the optimal number of clusters (i.e., optimal $k\text{-}cluster$) must be determined for each use case and entropy by applying the indicators established in Section 2.2.3.

For the three examples, as it was previously stated, we propose to analyse a range between [0-10] for $k\text{-}cluster$ since the obtained clusters will be used to create configuration workflows later used by humans. In this context of the application and the number of traces of the examples, we consider this range proper on the bases of the results shown in Table 4, that summarises the number of clusters calculated by using each case and entropy.It is important to note that due to the impossibility of computing the entropy matrix for the Smart Farm case with entropy transitions, results for this case will not appear in that table.

As described previously, the distribution of the traces among the cluster is a complex activity, and the best assignment is not a trivial task. For this reason, COLOSSI framework

[6] https://wwwiti.cs.uni-magdeburg.de/~jualves/PROFilE/datasets-download/Dell-Laptop_readme.txt

Table 4 Number of clusters per use case and entropy

Config. Workflow	Entropy	N. of Clusters ($k\text{-}cluster$)
ERP	$S_{features}$	5
	$S_{transitions}$	3
Smart Farming	$S_{features}$	6
	$S_{transitions}$	-
Config. Computer	$S_{features}$	2
	$S_{transitions}$	8

provides a set of techniques to be applied once the traces have been generated from the raw data. The objective of each algorithm is the same to distribute the traces among a determined number of clusters to minimise the summing of the entropy of every cluster for a later application of discovery process algorithms for each cluster.

Nevertheless, there are no approach helping to determine the optimal $k\text{-}cluster$ in algorithms such as backtracking or genetic, which is why, in our proposal, the optimal number of clusters is always determined using the hierarchical with the entropy matrix. Once an optimal $k\text{-}cluster$ value has been found for a case study and an entropy type, this value is used in all grouping algorithms for that case study and that entropy.

3.2 Analysis of clustering

As introduced in previous sections, two different entropies are applied to infer the clustering (i.e., *Entropy-features* and *Entropy-transitions*). The two entropy formulas can help to understand the quality of the future workflow. Thus, a lower value of entropy more quality of the cluster, hence, the workflow has more quality.

Table 5 shows the results obtained for each case and each algorithm taking into account the two entropies defined (feature and transitions). For a better comparison, the metrics associated with the clusters are aggregated as arithmetic means. For the sake of the results, the entropy of features are the best distributions in all the cases except for *Computer Configuration* in the hierarchical algorithm. These results will be confirmed with complexity results obtained with the metrics in Section 3.4.

It is important to emphasise that no results have been obtained for the backtracking algorithm and some of the cases for the entropy of transitions, caused by the huge exponential complexity that the examples imply. Because of this, two drawbacks of our approach have been identified in the application of algorithms and the determination of entropy. Regarding the backtracking algorithm, the use of a complete algorithm requires to explore all the space of solutions hence in most of the cases it requires exponential time depending on the size of the problem in terms of features, traces, and the number of configurations. The high number of features, traces, and configurations in the data used for experimentation made impossible the application of backtracking algorithm in an acceptable time since its executions in the simplest cases took more than 24 hours without results. Also, it is necessary to build an entropy matrix that acts as a distance matrix to determine the number of clusters. Due to the complexity of some scenarios, the creation of the entropy matrix was computationally impossible, disabling the calculation of the $k\text{-}cluster$ and consequently, the execution of the clustering algorithms. This is the case of *Smart Farming* where the entropy matrix with entropy-transitions was unapproachable in linear time, taking more than 10 hours.

Table 5 Entropy per algorithm and example

	Greedy Algorithm		Genetic Algorithm		Hierarchical Algorithm	
	Entropy features	Entropy transitions	Entropy features	Entropy transitions	Entropy features	Entropy transitions
ERP	0.2170	0.802	0.4609	0.793	0.1759	0.8585
Smart Farm	0.5137	–	–	–	0.5990	–
Computer Configuration	0.7830	0.9313	0.7264	0.9492	0.2566	0.2352

As previously highlighted, the characteristics and size of the configurations data will condition the achievement of results with clear limitations of resources and time. Therefore, it is necessary to compare the execution times of each algorithm for each example. Table 6 shows the duration in minutes of executions from when the clustering algorithm starts until it reaches a solution. Something important to highlight in these execution times is the evidence that the hierarchical algorithm is much faster than the rest, mostly, because the task that requires the longest computation time is the construction of the distance matrix (in COLOSSI, entropy matrix) and this had previously been done to calculate the optimal $k\text{-}cluster$. In the rest of the algorithms, the entropy matrix does not apply and they build the distances during their executions, this is the reason why their computation time is longer.

3.3 Statistical analysis of results

To check if the running algorithms have an actual impact on the Entropy performance indicators we used the *Null Hypothesis Statistical Test* in which two contrary hypotheses are formulated. On the first hand, the null hypothesis (H0) states that the selected techniques do not influence the obtained results (i.e., the algorithm has no impact on the entropy of the retrieved models). On the other hand, the alternative hypothesis states that the selected algorithm impacts the obtained results significantly (i.e., selecting a greedy or a hierarchical algorithm impacts the entropy obtained).

We decided to fix such hypothesis to understand if our techniques to improve the entropy of the resulting clusters were affected by the technique, and thus, can be useful in variability-aware scenarios. This is, testing this hypothesis intend to check whether the application of techniques coming from other contexts (e.g. process mining) provide meaningful results on variability-intensive systems.

Table 6 Execution times in minutes per algorithm and example

	Greedy Algorithm		Genetic Algorithm		Hierarchical Algorithm	
	Entropy features	Entropy transitions	Entropy features	Entropy transitions	Entropy features	Entropy transitions
ERP	1.76	0.75	105	12	0.016	0.016
Smart Farm	152	–	–	–	0.016	–
Computer Configuration	0.026	0.041	15	15	0.016	0.016

Such executed tests provide a probability value (called *p-value*) which ranges from 0 to 1. The lower the *p-value* of a test, the more likely that the null hypothesis is false, and the alternative hypothesis is correct. It is established that *p-values* under 0.05 or 0.01 are so-called statistically significant, which let us assume that the alternative hypothesis is likely true.

In this analysis, we only checked instances of the greedy and the hierarchical algorithm. This is motivated by the lack of results for all models of the genetic and backtracking algorithms for each case study. Note that this unavailability points out that the results are dependent on the techniques.

The test we relied on for performing the statistical analysis and obtain the *p-values* depends on the properties of the data (Derrac et al. 2011). Concretely, we executed the Wilcoxon test (Wilcoxon 1946) and were not able to prove that our data follow normal distribution so, we had to rely on non-parametric techniques. We executed Friedman's tests for both entropy metrics, obtaining a *p-value* of 0.0455, and a statistics of four in the case of the feature-based entropy. Thus, we have to reject the null hypothesis and then, accept the alternative one. Therefore, for the case of feature-based entropy, the selection of the appropriated algorithm impacts the quality of results. Secondly, we obtained a *p-value* of 0.5637, with a statistic of 0.333 for the case of transition-based entropy's which prevents us from rejecting the null hypothesis.

These results provide two main insights. First, we observe that for the feature-based entropy is heavily dependent on the method used. Second, we can not determine if the transition-based entropy is being affected by the method.

3.4 Analysis of discovered configuration workflows

To evaluate how the application of different clustering algorithms can improve the configuration workflows obtained by COLOSSI, in this section, for each case study and algorithm, we compare the models discovered by using a set of metrics. For each algorithm, entropy and case study, the workflows corresponding to (1) the *filtered* version of the same log including only valid configurations (i.e., after applying *configuration log handler*) and; (2) the set of clusters obtained after applying *cluster generator* explained in Section 2.2.3. The mentioned metrics helps to compare with each other, in terms of complexity and understandability, the configuration workflow of the filtered log and those extracted after the clustering. This means that we measure the quality of the resulting configuration workflow, without taking into account the input data of the feature model.

The analysis is carried out following two different perspectives: (1) the analysis of the discovered configuration workflows and (2) the analysis of the set of configuration traces involved in each cluster used in the process discovery.

First, we highlight that inductive process discovery techniques used by COLOSSI ensure the soundness and correctness of the process models obtained (Leemans et al. 2015). Thus, an analysis of the soundness and correctness of the configuration workflows are unnecessary since processes discovered is always complete, have a proper completion, and have no dead transitions.

However, the complexity of the configuration models are affected by the number of features, the number of configuration traces and the number of transitions. The filtering of the configuration traces or the division of the logs will bring about simpler configuration workflows. Table 7 depicts comparatively the number of features, configuration traces and transitions of the set of configuration logs using in each case study. The data of features,

Table 7 Characteristics of the configuration logs

				Greedy Algorithm		Genetic Algorithm		Hierarchical Algorithm	
		Original	Reduction Ratio	Entropy Features	Entropy Transitions	Entropy Features	Entropy Transitions	Entropy Features	Entropy Transitions
ERP	Features	425	48.66	129.40	253.67	211.40	333.67	146	229.67
	Transitions	2,028	64.20	260.40	863.67	806.40	1,521.33	296.40	462.33
	Traces	61	72.77	12.20	20.33	12.40	20.33	12.40	20.33
Smart Farm	Features	1,420	60.01	723.33	–	–	–	712	–
	Transitions	2,844	8.95	3,394.33	–	–	–	1,794.17	–
	Traces	919	83.33	153.17	–	–	–	153.17	–
Computer Configuration	Features	53	39.68	41.40	22.38	39	30	35	24
	Transitions	205	43.15	174	65.38	151	102.25	106.50	59
	Traces	42	66.86	21	5.25	21	5.25	21	10

transitions, and configuration traces of the clusters have been aggregated per use case, clustering algorithm and type of entropy to show the average for every cluster obtained as a solution after the application of the algorithms of clustering. In addition, the reduction ratio column contains the average rate by which each characteristic has been reduced with respect to the value presented by the original log.

The number of features and configuration traces grouped in the different clusters are decreased in comparison with the original case since they are distributed among the clusters. Regarding the transitions, these values are affected by the entropy and the algorithm used. In general, the number of transitions is reduced, but it could happen, that because the grouped traces are formed by non-common features, the number of transitions can increase, as is the case of smart farming in the greedy algorithm.

In conclusion, clusters reduce the complexity of configuration workflow discovered by reducing the configuration traces involved in the same configuration workflow. However, the question is in what level the quality of the obtained workflow is improved, and which distribution of cluster-entropy works better.

In the literature, several metrics are used to measure how "good" is a design of a business process model (Mendling 2008; Pérez-Castillo et al. 2019; Cardoso 2005). Discovered configuration workflows are also processes with features instead of activities, therefore, these metrics can be adapted to measure the quality of our obtained configuration workflows. The next set of metrics is adapted to measure the understandability and the complexity of the configuration workflows to compare the discovered configuration workflows:

- *Density*: the ratio of transitions divided by the maximum number of possible transitions. The lower the value of density, the higher the understandability.
- *Cyclomatic number* (CC): the number of paths needed to visit all features. The cyclomatic can be seen as a complexity metric, thus, the lower the value of CC, the lower the level of complexity.
- *Coefficient of connectivity* (CNC): the ratio of transitions to features. The greater the value of CNC, the greater the complexity of configuration workflows. Although, the

authors in Mendling (2008) remark that models with the same CNC value might differ in complexity regarding this parameter.

– *Control Flow Complexity* (CFC) enables to measure the complexity in terms of the potential transitions after a split depending on its type. The greater the value of the *CFC*, the greater the overall structural complexity of a workflow.

These four metrics help us to know the complexity and understandability of the configuration workflows from the design perspective and the elements in the model. Nevertheless, these metrics used to measure the quality of the workflow are inconclusive to measure the real usefulness and quality of the discovered workflows applied to the context of the variability management.

As previously mentioned, the uselessness of the quality metrics related to the workflows leads us to define a new custom metric which enables to establish the quality level of the workflow by relating the number of features and their occurrence within the discovered workflow of a cluster. Thus, a metric that enables us to measure how spaghetti is the workflow obtained. Our custom-quality metric is defined as follows:

– Quality ($\Delta_{\equiv}$) measures the difference between the total number of features and the ratio of the sum of the number of times that a feature is selected for each configuration trace and the number of configuration traces.

Formally, given a workflow based on a set of configuration traces (CT) and a set of features ($Features$), the quality can be determined as the following formula:

$$\Delta_{\equiv} = |Features| - \sum_{f \in Workflow} \frac{occurrences(f)}{|CT|} \tag{4}$$

The range of the quality is $[0..|Features|]$, the lower value of quality a better configuration workflow is indicated. The number of features and configuration traces are grouped into the most similar workflow, therefore, it brings about that the quality is near to 0.

To make the application of metric more understandable, we use the first example of Fig. 3 and the *Cluster 2* in Fig. 5. The number included in the rectangle, next to the name of the feature, corresponds to the number of traces within the cluster where the feature appears. Hence, the quality for the *Cluster 2* can be determined by applying the formula as follows:

$$\Delta_{\equiv} = 8 - \left(\frac{3}{3} + \frac{3}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3} + \frac{1}{3}\right) \approx 4 \tag{5}$$

The results obtained for these five metrics are shown using box-plot charts represented for each scenario in (Figs. 10, 11 and 12) to compare distribution by metric, algorithm and the type of entropy for each case study. The y axis represents, on a logarithmic scale, the results of each metric for the configuration workflows obtained using a specific algorithm. Detailed values for each case and metrics can be consulted in the Appendix section.

In the case of *ERP* scenario, Fig. 10 shows that the values for *CC* and *CFC* metrics are equal and significantly higher, which implies that the connectivity of the workflow is higher, increasing its complexity. In addition, in the case of the genetic algorithm, it is important to highlight the importance of the type of entropy, which will determine whether the clusters generated after the assignment have similar levels of connectivity or not. For *Density* and *CNC* metrics, all algorithms have similar behaviour regardless of the entropy used. In terms of *Quality*, it is clearly observed how the use of entropy by transitions produces considerably more complex workflows.

In the *Smart Farm* example, the greedy algorithm produces slightly more complex workflows but also more balanced clusters as shown in Fig. 11. Also, the hierarchical clustering

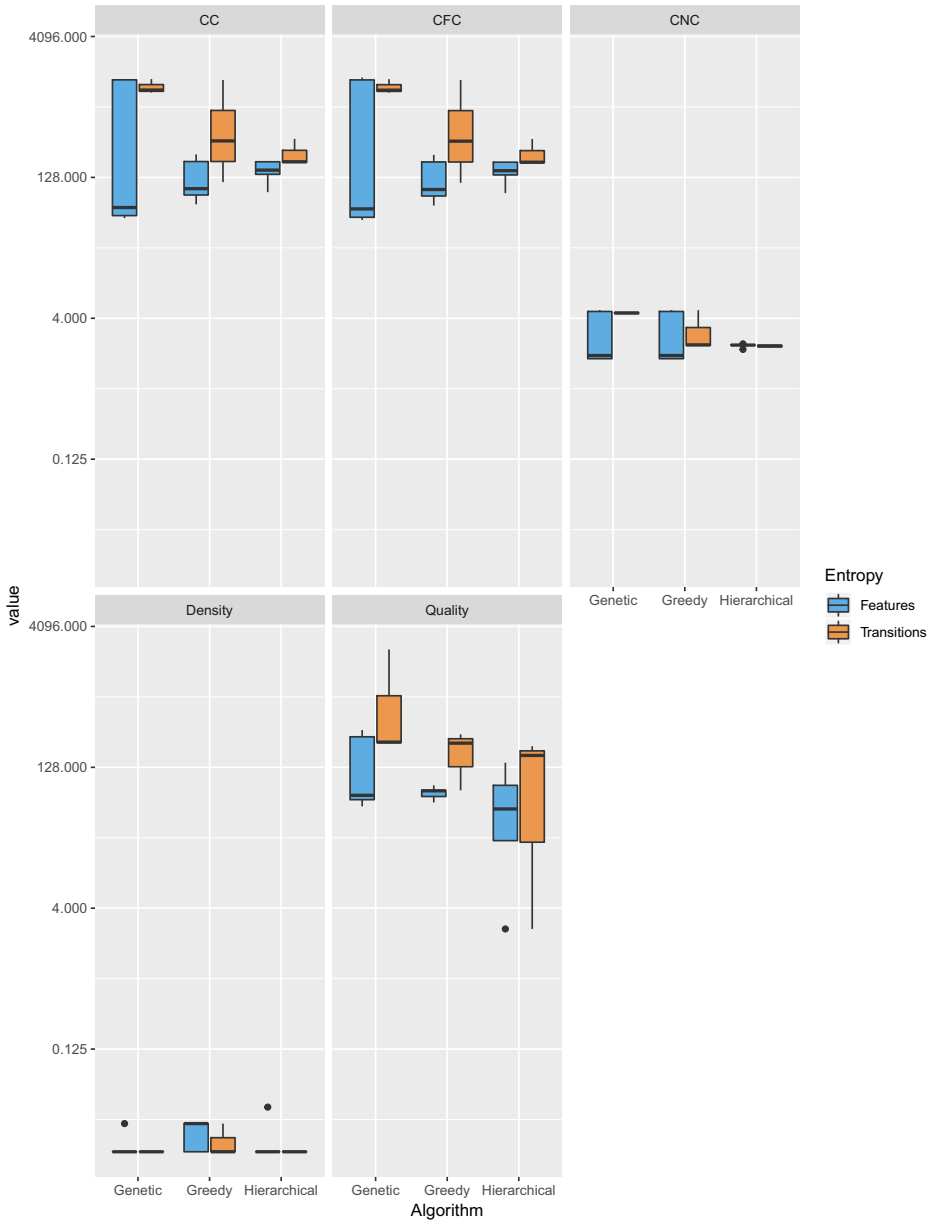


Fig. 10 Metrics distribution of ERP case study

generates less similar clusters, which reduces the complexity and the connectivity, as it is clearly observable for the case of *Quality* metric.

Finally, there are no remarkable differences between the three algorithms for the case of *Computer Configuration*, since they all produce very similar results as shown in Fig. 12. Exclusively, the use of the hierarchical algorithm with transitions entropy could be distinguished as the solutions with less complex configuration workflows. Nevertheless and in

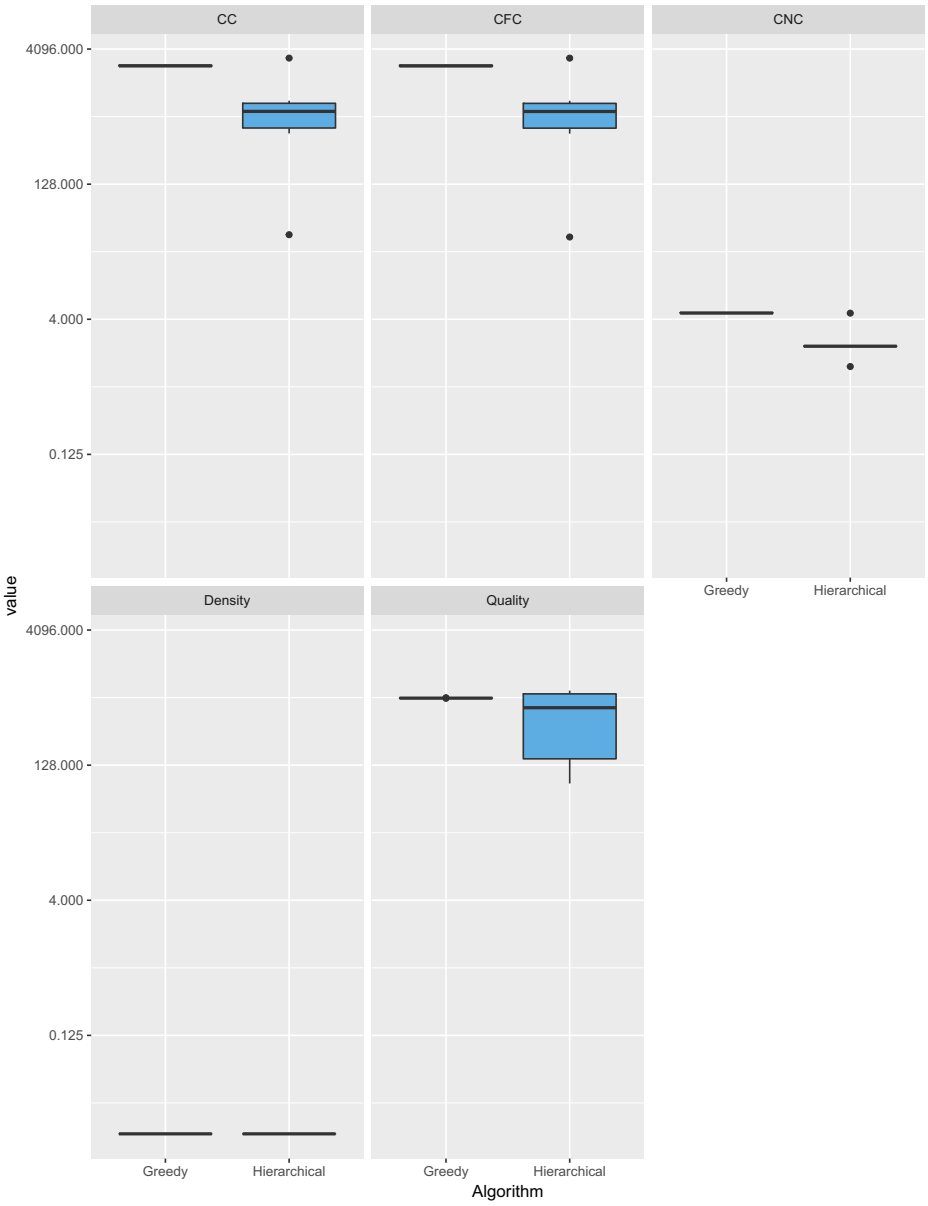


Fig. 11 Metrics distribution of Smart Farm case study

contrary to the other cases, in this particular scenario better general results of *Quality* are obtained using the entropy of transitions.

To compare all the cases with each other, we propose Fig. 13. This figure represents the average values of each metric for each case study in aggregate. In this way, we intend to compare the three examples in terms of complexity, taking a holistic view of the average values obtained by each for each metric. Thus, it is noteworthy that the mean values for three

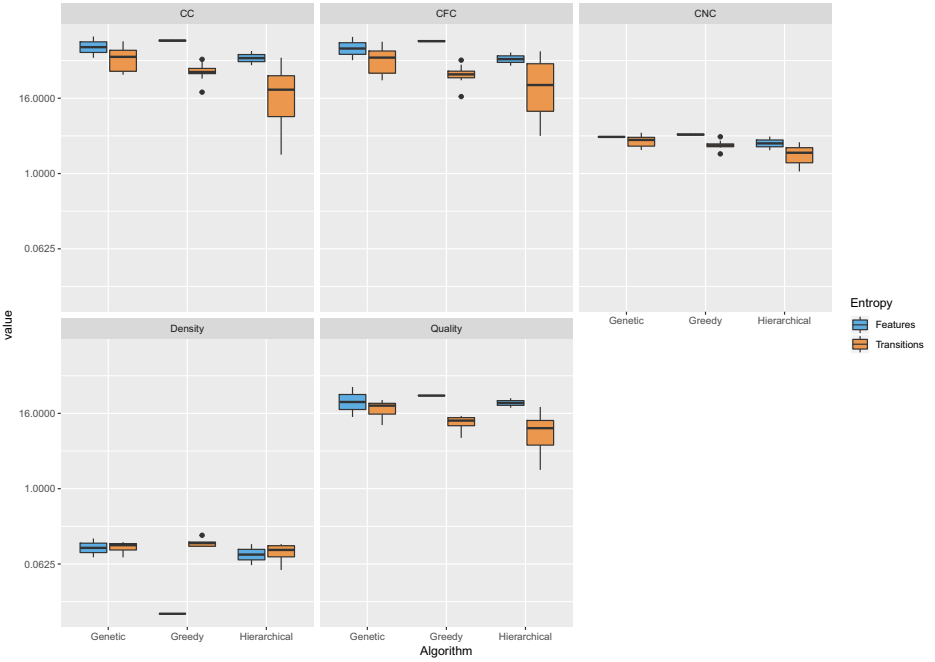


Fig. 12 Metrics distribution of Computer Configuration case study

of the metrics for the *Computer Configuration* case are significantly smaller than in the other cases, which can mean that the workflows obtained from this example are less complex. On the other hand, in light of the results shown in the chart, we can realise that, for these cases, the *Density* and *CNC* metrics do not help to discriminate the real level of complexity of the workflows produced with COLOSSI, since they present the same average value for the three cases. Nevertheless, the *Quality*, *CC* and *CFC* metrics present the highest values for the *Smart Farm* case, which could lead to higher levels of complexity and connectivity. It can also be inferred that *Computer Configuration* may be the most balanced case since all its edges have approximately the same length. In the case of *ERP*, it is remarkable how all its complexity can be focused on its high degree of connectivity, reaching the maximum of our evaluation. The fact that the *Smart Farm* example occupies a larger area between the Quality and CFC metrics denotes that, probably, this case contains a much greater number of features, transitions and splits than the other cases.

3.5 Threats to validity

Although the experiments presented in this paper provide evidence that demonstrates the validity of the proposed solution, in this section, we discuss the different threats to validity that affect the evaluation, derived from the assumptions that we made.

External validity The inputs used for the experiments presented in this paper were either realistic or designed to mimic realistic feature models. However, we do not control the development process and it may have errors and not encode every configuration for all case studies.

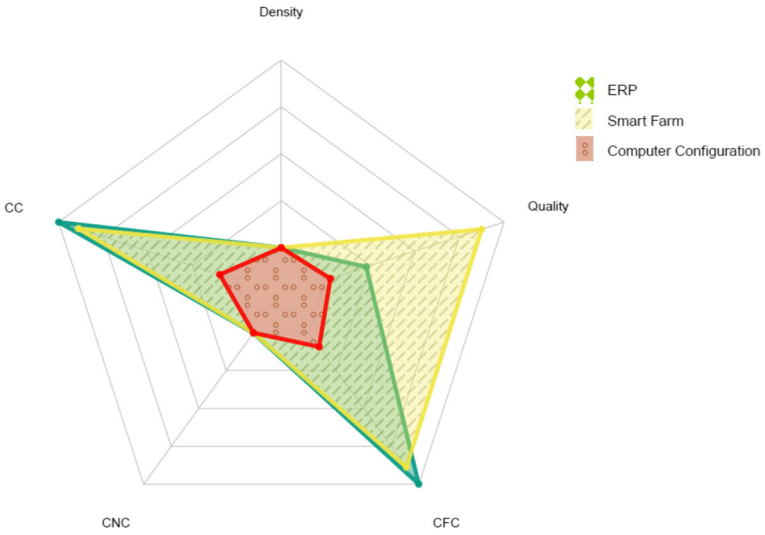


Fig. 13 Average of metrics of the configuration workflows

The major threats to external validity are:

- *Population validity*: the three examples that we used do not represent all configuration traces. Note that all of the models were provided after an anonymisation process. Moreover, the timestamps used to derive the traces were relying on the appearance within the input file without an explicit enumeration. To reduce these threats to validity, we chose some large models that were used in different studies in the literature. Also, we were not directly involved in the development of such models which
- *Ecological validity*: while external validity, in general, is focused on the generalisation of the results to other contexts (e.g., using other models), the ecological validity if focused on possible errors in the experiment materials and tools used. To avoid as much as possible such threats, we relied on previously existing algorithms to perform the process discovery.
- *Limitations depending on the input data*: another external validity problem lies with the shape and size of the input data. As previously stated, one of the important bottlenecks of this solution is the construction of the entropy matrix, since, without the matrix, it will not be possible to determine the optimal $k\text{-}cluster$, and therefore, to apply our solution.
- *Limitations on the use of feature attributes*: as mentioned in Section 2.1, features may contain attributes with more information, which has not been taken into account in our solution. Ignoring this data means that the selection of the same feature with different values in its attributes cannot be differentiated in the resulting workflows. This represents an important weakness of the approach since the use of this information would mean that different paths would be generated in the workflow for the same feature with different values in its attributes. While in our solution, it is considered as the same one.

Internal validity We developed several algorithms and metrics that reveal different properties of the workflows. To mitigate this threat, we have relied on a diversity of approaches. However, there might be characteristics of such workflows that are not revealed and further research should be developed. Also, another major threat to internal validity was the short number of models and configuration available in which we were able to test our techniques. In this case, we tried to cover all the models we found in the literature.

3.6 Examples of discovered configuration workflows

Derived from the high number of implementations and test cases tackled in this paper, every configuration workflow cannot be included in the document. However, for the sake of illustration, two configuration workflows of the ERP are shown in Figs. 14 and 15. Both figures represent the obtained clusters according to the dendrogram built by means of the entropy of features and transitions analysis respectively. In the case of entropy of features (cf., Fig. 14), five clusters are obtained. On the other hand, when the entropy of transition is used, three clusters are derived to split the configuration logs into simpler configuration workflows. In the following subsections, the details about the obtained configuration workflows and clusters are analysed. Moreover, every obtained cluster distribution for each algorithm, case study and type of entropy are available at http://www.idea.us.es/empiricalsoftware/.

4 Related work

This paper combines different research areas, for this reason, this section is structured to cover the main ones: configuration workflows, application for process mining and the variability in process mining.

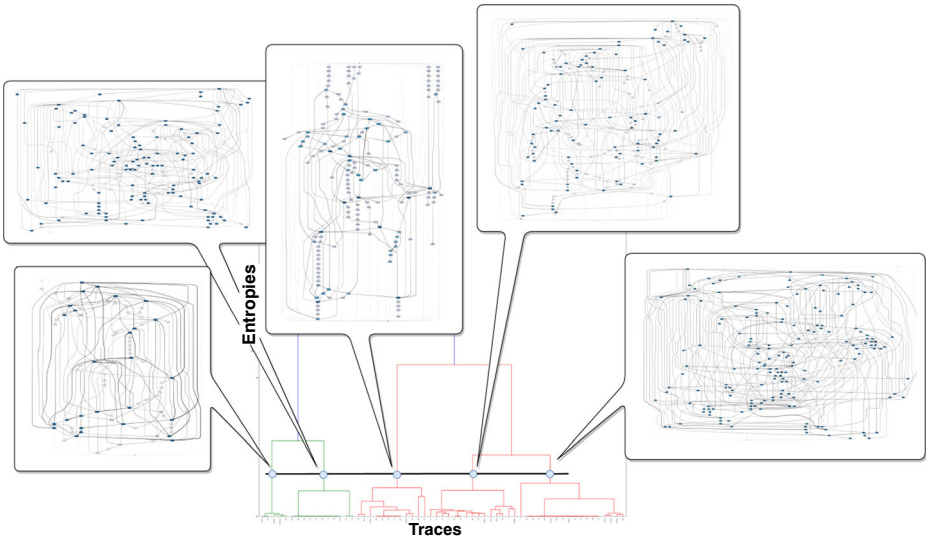


Fig. 14 Clustering for the ERP example using the *Entropy-features*

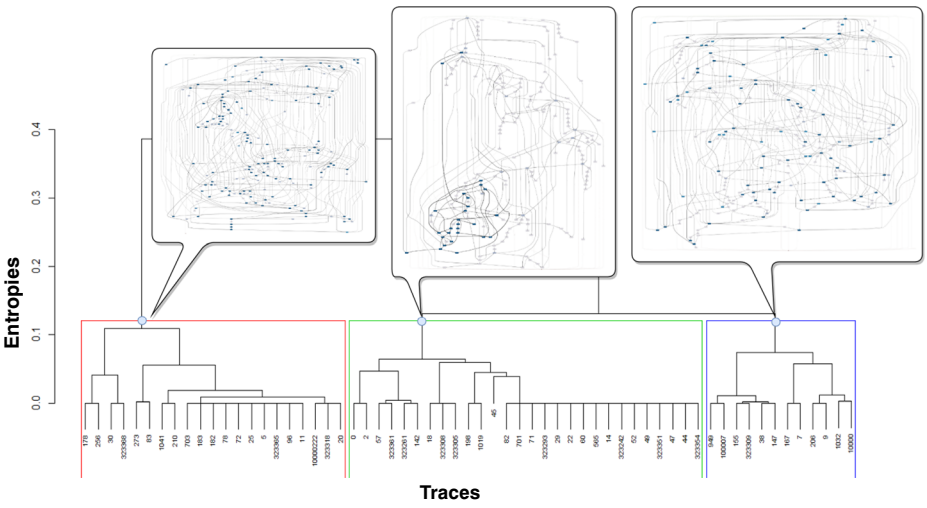


Fig. 15 Clustering for the ERP example using the *Entropy-transitions*

4.1 Configuration workflows

A formal description of configuration workflows is given in Hubaux et al. (2009). However, a configuration workflow is a bit different from our definition. The activity of the configuration workflow can be mapped to more than just a feature as in our case. However, our approach is complementary because in the handling process we can group different features as well. Furthermore, although formal semantics and automated support for configuration workflows are presented, no automated mechanism is developed to automatically generate configuration workflows from existing configuration logs. In that sense, our approach complements theirs.

Different possible feature orders are defined in Galindo et al. (2015). Those orders are used to build web–based configurators hiding the details of the concrete variability model flavours (e.g., OVM, FMs, CVL, etc.). The orders are built from the structure of the variability model. For instance, in the case of FMs, pre-order, post-order or in-order can be used to determine the feature order in which features are presented to the user. COLOSSI differs from this approach because we use as input configuration logs to automatically derive and cluster configuration workflows. Our approach can be complementary to Galindo et al. (2015) because different existing workflows could be also measured using process alignment metrics to determine what is the best feature order to be used.

There exist other approaches (Wang and Tseng 2011, 2014) focused on the field of product configurator design in which configuration workflows has been tackled from the perspective of machine learning. Neither of these two approaches uses clustering or process mining. Both are based on probabilistic estimations aided by a recommendation system that recalculates and guides the user through each step. This means that the user can lose the holistic view of the entire configuration process since from the step n he/she can only know the different possibilities that he/she can do in the step $n + 1$. With our proposal, the user has, from the beginning, an overall perspective of the entire configuration process from start to finish.

4.2 Application of process mining in different contexts

In order to discover the processes followed by users or systems analysing event logs, process mining has been applied in several scenarios. Depending on the scenario, different are the points of view that could be used to discover a process, such as the activities executed, persons involved, the resources used, the location where the actions occur, etc. The versatility of process mining techniques has brought about its application to several scenarios (Dakic et al. 2018), healthcare (Mans et al. 2009; Rozinat et al. 2009; Perimal-Lewis et al. 2016) and IT (Sahlabadi et al. 2014; Măruşter and van Beest 2009; Pérez-Álvarez et al. 2018; Fernández-Cerero et al. 2019) being the most active areas.

The case studies where event logs are produced by human behaviour interactions are especially complex, derived from the free-will capacity of the persons that is not always possible to be modelled. This is the context of this paper, where configuration tasks describe the interaction of users with systems. Previous examples in previous scenarios have been developed, such as Astromskis et al. (2015), to analyse how the users interact with an enterprise resource planning software, or the applicability of software scenarios analysing how the users interact with software to promote improvements about functional specifications or usability aspects (Rubin et al. 2014). Software development has also provided a complex scenario where process mining can provide a mechanism to improve and optimise the known as software process mining (Rubin et al. 2007). However, configurability issue has not been analysed before with process mining.

4.3 High variability in process mining

When there is a high human interaction, as in configuration processes, spaghetti and lasagne processes tend to be obtained. The occurrence of infrequent activities or non-repeated sequence of activities in the analysed log events brings about the necessity to apply frequency-based filtering solutions (Conforti et al. 2017) and other based on the discovery of a chaotic set of activities that can be frequent (Tax et al. 2019).

The infrequence patterns in process discovery are frequently treated as noise (Ly et al. 2012), being removed from the log traces to discover a process that represents the most frequent behaviour (Sani et al. 2017). Different types of filtering can be performed: (i) filtering the events that are not belong to the mainstream behaviour (Conforti et al. 2017; Sani et al. 2017); (ii) integrating the filtering as a part of the discovery (Leemans et al. 2014; Maruster et al. 2006; Weijters and Ribeiro 2011; vanden Broucke and Weerdt 2017); (iii) filtering traces, in an unsupervised (Ghionna et al. 2008) or supervised way (Cheng and Kumar 2015), and; (iv) including a previous steps for clustering the problem, facilitating the discrimination of traces according to different points of view or dividing different types of behaviour (de Leoni et al. 2016; Song et al. 2009; de Medeiros et al. 2007). In Sani et al. (2019) clustering techniques have been used to improve the quality of process models, but not to distribute the traces in different clusters.

To our knowledge, this paper, which is as an extension of Varela-Vaca et al. (2019a), is the first solution for workflow retrieval in SPL-related contexts. It is also an achievement the application of process mining techniques in new fields. This paper aims at promoting synergies between these two areas of study, consider different types of algorithms, metrics or entropies, and they are not oriented towards the consideration of the characteristics of configuration workflows.

Our contribution described in this paper intends to promote synergies between process mining and variability management and software product lines. We consider different types of algorithms, metrics and entropies.

5 Concluding remarks & future work

In this paper, we have coped with the problem of extracting the actual workflows used by SPL configurators by analysing configuration logs. To discover configuration workflows, we decided to rely on process mining techniques. Moreover, we proposed to apply clustering to improve the resulting configuration workflows reducing the complexity and improving their understandability. From our research on configuration workflows, we learned the following important lessons:

1. **Reduce the complexity of the configuration workflows.** We have defined a mechanism based on clustering to divide the configuration logs into smaller configuration groups to facilitate the understanding of the configuration workflows inferred from configuration logs.
2. **Quality measurement.** We have defined a set of metrics adapted from business process literature, to measure the quality of the obtained clusters and configuration workflows.
3. **Improving decisions about configurators.** The clustering creation and the analysis of the obtained configuration workflows provide information to expert users about the features that used to be configured together or sequential lists of features that could be integrated into a single feature. In this way, thanks to the structure of the workflow, the user will know that everything that appears in a parallel can be done simultaneously, while those that appear sequentially will indicate precedence relationships and restrictions.
4. **The selection of the clustering method depends mostly on the input data.** Generally, hierarchical clustering provides good solutions, in terms of quality, velocity and efficiency. However, it has an important bottleneck: the construction of the entropy matrix, which will determine the time and resources necessary to achieve the results. For the cases in which the construction of the entropy matrix becomes very hard, less accurate algorithms, such as greedy or genetic algorithms may be more suitable. The greedy algorithm provides very acceptable results consuming less time, and the genetic algorithm is positioned at an intermediate level for those cases in which results very close to the optimum, are needed. Finally, backtracking ensures optimal entropy minimisation and traces distribution. Unfortunately, the time and resources needed will increase exponentially with the size of the data of the case study. Lamentably, the use of these algorithms without the previous computation of the entropy matrix and the optimal $k\text{-}cluster$, will imply that the number of clusters chosen and the distribution of the traces, probably, will lead to a non-optimal assignment.
5. **The impact of the size and the morphology of the case study data on time and metrics.** Based on the results of the metrics in different examples, and the time and resources that are necessary to get the solutions of each one, there seems to exists a direct relationship among (i) features and the way in which these are related, (ii) the time and resources that are necessary to compute the entropy matrix and the

optimal $k - cluster$, (iii) the time consumed for each algorithm until the clusters are achieved, (iv) and the range of values in which the results of the metrics will be.

In future work, we plan to develop new variability-oriented metrics that can show the impact of the numbers of features within the workflows, trying to incorporate characteristics of the feature models into the clustering and process discovery. Additionally, we would like to incorporate the information present in the attributes of the features in the feature model to achieve more realistic workflows. Moreover, we would like to apply this technique to more scenarios and datasets to complement the validation of our proposal, including in the analysis of other methods to tackle spaghetti processes. Further, we consider interesting to investigate a proper way to obtain the best distribution clusters automatically for a defined number of clusters. In addition, the search of optimum $k - cluster$ is a very difficult task such as shown in our work, therefore it is necessary to find out a way to discover an approximate $k-cluster$ that ensures the achieved assignments not only are correct but they also are potential solutions. We propose to study new formulas of seeking $k - cluster$ based on the structure (i.e., number of task and/or gateways) of initial workflows before clustering. From our point of view, it is also relevant to propose multiple uses of the resulting workflows to help in different areas such as reverse engineering or SPL testing. Another proposal is to incorporate incorrect configuration to analyse the misalignment with the expected feature model.

Acknowledgements This work has been partially by the Ministry of Science and Technology of Spain through ECLIPSE (RTI2018-094283-B-C33) and OPHELIA (RTI2018-101204-B-C22) projects; the TASOVA network (MCIU-AEI TIN2017-90644-REDT); and the Junta de Andalucía via METAMORFOSIS projects, the European Regional Development Fund (ERDF/FEDER), and the MINECO Juan de la Cierva postdoctoral program.

Appendix : Quality metrics results

This appendix contains in the Tables 9, 10, 11, 12, and 13, the metric data represented in Figs. 10, 11, 12. To facilitate the interpretation of the data, the values of the metrics have been normalised in each metric, so that, all the results are between 0 and 1, allowing comparisons to be made. In addition, Table 8 is included to show the metric values for the original logs. With this, it can be seen how, in most cases, their values are closer to 0 after clustering, meaning that the resulting configuration workflows are also less complex. Still, it is important to note that it is very difficult to determine a generalisation regarding this data, since they are too domain-specific.

Table 8 Metrics for the initial logs of each case study

Case Study	Density	CC	CNC	CFC	Quality
ERP	0.05	0.49	0.96	0.49	0.26
Smart Farm	0	0.43	0.24	0.43	1
Computer Configuration	0.39	0.04	0.73	0.04	0.03

Table 9 Metrics for ERP case with entropy-features

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 1	Greedy	0.05	0.05	0.26	0.05	0.09
	Hierarchical	0.16	0.02	0.24	0.02	0.05
	Genetic	0.05	0.43	0.95	0.43	0.36
Cluster 2	Greedy	0.11	0.01	0.16	0.01	0.08
	Hierarchical	0.05	0.04	0.20	0.04	0.16
	Genetic	0.05	0.01	0.10	0.01	0.07
Cluster 3	Greedy	0.11	0.02	0.24	0.02	0.07
	Hierarchical	0.05	0.05	0.26	0.05	0.09
	Genetic	0.11	0.01	0.13	0.01	0.06
Cluster 4	Greedy	0.11	0.02	0.20	0.02	0.08
	Hierarchical	0.05	0.05	0.25	0.05	0.002
	Genetic	0.05	0.01	0.08	0.01	0.05
Cluster 5	Greedy	0.05	0.06	0.28	0.06	0.06
	Hierarchical	0.05	0.04	0.25	0.04	0.02
	Genetic	0.05	0.43	1	0.45	0.31

Table 10 Metrics for ERP case with entropy-transitions

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 1	Greedy	0.11	0.03	0.24	0.03	0.08
	Hierarchical	0.05	0.10	0.24	0.10	0.24
	Genetic	0.05	0.33	0.91	0.33	0.27
Cluster 2	Greedy	0.05	0.09	0.26	0.09	0.26
	Hierarchical	0.05	0.05	0.22	0.05	0.19
	Genetic	0.05	0.31	0.88	0.31	0.26
Cluster 3	Greedy	0.05	0.43	0.99	0.43	0.33
	Hierarchical	0.05	0.05	0.25	0.05	0.002
	Genetic	0.05	0.44	0.93	0.44	0.37

Table 11 Metrics for Smart Farm case with entropy-features

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 1	Greedy	0.05	0.80	0.92	0.80	0.82
	Hierarchical	0	0.33	0.23	0.33	0.12
	Genetic	–	–	–	–	–
Cluster 2	Greedy	0.05	0.84	0.96	0.84	0.83
	Hierarchical	0	0.25	0.25	0.25	0.90
	Genetic	–	–	–	–	–

Table 11 (continued)

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 3	Greedy	0.05	0.82	0.93	0.82	0.82
	Hierarchical	0	0.14	0.24	0.14	0.46
	Genetic	–	–	–	–	–
Cluster 4	Greedy	0.05	0.81	0.94	0.81	0.82
	Hierarchical	0	0.25	0.24	0.25	0.92
	Genetic	–	–	–	–	–
Cluster 5	Greedy	0.05	0.80	0.94	0.80	0.82
	Hierarchical	0.05	1	0.94	1	1
	Genetic	–	–	–	–	–
Cluster 6	Greedy	0.05	0.83	0.96	0.83	0.82
	Hierarchical	0.05	0.01	0.02	0.01	0.09
	Genetic	–	–	–	–	–

Table 12 Metrics for Computer Configuration case with entropy-features

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 1	Greedy	0.55	0.03	0.77	0.03	0.03
	Hierarchical	0.72	0.02	0.73	0.02	0.02
	Genetic	0.88	0.02	0.71	0.02	0.01
Cluster 2	Greedy	0.61	0.04	0.85	0.04	0.03
	Hierarchical	0.33	0.01	0.33	0.01	0.03
	Genetic	0.44	0.04	0.74	0.04	0.04

Table 13 Metrics for Computer Configuration case with entropy-transitions

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 1	Greedy	0.77	0.01	0.46	0.01	0.01
	Hierarchical	0.33	0.02	0.47	0.02	0.02
	Genetic	0.77	0.01	0.43	0.009	0.01
Cluster 2	Greedy	0.77	0.01	0.46	0.01	0.01
	Hierarchical	0.27	0.02	0.36	0.02	0.01
	Genetic	0.66	0.03	0.89	0.03	0.03
Cluster 3	Greedy	0.66	0.005	0.25	0.005	0.007
	Hierarchical	0.72	0.01	0.54	0.01	0.01
	Genetic	0.61	0.02	0.70	0.02	0.02
Cluster 4	Greedy	0.72	0.01	0.59	0.01	0.01
	Hierarchical	0.61	0.002	0.11	0.002	0.005
	Genetic	0.44	0.01	0.34	0.01	0.02

Table 13 (continued)

	Algorithm	Density	CC	CNC	CFC	Quality
Cluster 5	Greedy	0.66	0.01	0.42	0.01	0.01
	Hierarchical	0.5	0	0	0	0
	Genetic	0.77	0.02	0.79	0.02	0.02
Cluster 6	Greedy	1	0.02	0.73	0.02	0.01
	Hierarchical	0.55	0.001	0.07	0.001	0.002
	Genetic	0.72	0.02	0.73	0.02	0.02
Cluster 7	Greedy	0.66	0.01	0.44	0.01	0.01
	Hierarchical	0.66	0.08	0.34	0.008	0.01
	Genetic	0.5	0.02	0.5	0.02	0.02
Cluster 8	Greedy	0.77	0.009	0.39	0.009	0.01
	Hierarchical	0.72	0.004	0.22	0.004	0.006
	Genetic	0.72	0.01	0.41	0.01	0.01

References

Alférez M, Acher M, Galindo JA, Baudry B, Benavides D (2019) Modeling variability in the video domain: language and experience report. Softw Qual J 27(1):307–347

Astromskis S, Janes A, Mairegger M (2015) A process mining approach to measure how users interact with software: an industrial case study. In: Proceedings of the 2015 international conference on software and system process. ICSSP 2015. ACM, New York, pp 137–141

Augusto A, Conforti R, Dumas M, Rosa ML, Maggi FM, Marrella A, Mecella M, Soo A (2019) Automated discovery of process models from event logs: review and benchmark. IEEE Trans Knowl Data Eng 31(4):686–705. https://doi.org/10.1109/TKDE.2018.2841877

Baker FB, Hubert LJ (1975) Measuring the power of hierarchical cluster analysis. J Am Stat Assoc 70(349):31–38

Ball GH, Hall DJ (1965) Isodata a novel method of data analysis and pattern classification. Tech. rep. Stanford Research Inst, Menlo Park

Bosch J (2018) The three layer product model: an alternative view on spls and variability. In: Proceedings of the 12th international workshop on variability modelling of software-intensive systems, VAMOS 2018, Madrid, Spain, February 7–9, 2018, p 1. https://doi.org/10.1145/3168365.3168366

Caliński T, Harabasz J (1974) A dendrite method for cluster analysis. Commun Stat-Theory Methods 3(1):1–27

Cardoso J (2005) Control-flow complexity measurement of processes and weyuker's properties. In: 6th International enformatika conference, vol 8, pp 213–218

Cheng H, Kumar A (2015) Process mining on noisy logs—can log sanitization help to improve performance? Decis Support Syst 79:138–149. https://doi.org/10.1016/j.dss.2015.08.003

Conforti R, Rosa ML, ter Hofstede AHM (2017) Filtering out infrequent behavior from business process event logs. IEEE Trans Knowl Data Eng 29(2):300–314. https://doi.org/10.1109/TKDE.2016.2614680

Dakic D, Stefanovic D, Cosic I, Lolic T, Medojevic M (2018) Business application: a literature review. In: 29th DAAAM international symposium on intelligent manufacturing and automation. https://doi.org/10.2507/29th.daaam.proceedings.125

Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell (2):224–227

de Leoni M, van der Aalst WMP, Dees M (2016) A general framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf Syst 56:235–257. https://doi.org/10.1016/j.is.2015.07.003

de Medeiros AKA, Guzzo A, Greco G, van der Aalst WMP, Weijters AJMM, van Dongen BF, Saccà D (2007) Process mining based on clustering: a quest for precision. In: Business process management workshops, BPM 2007 international workshops, BPI, BPD, CBP, ProHealth, RefMod, semantics4ws, Brisbane, Australia, September 24, 2007, Revised Selected Papers, pp 17–29. https://doi.org/10.1007/978-3-540-78238-4_4

De Weerdt J, vanden Broucke S, Vanthienen J, Baesens B (2013) Active trace clustering for improved process discovery. IEEE Trans Knowl Data Eng 25(12):2708–2720
Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
Duda RO, Hart PE et al (1973) Pattern classification and scene analysis, vol 3. Wiley, New York
Dunn JC (1974) Well-separated clusters and optimal fuzzy partitions. J Cybern 4(1):95–104
Durán A, Benavides D, Segura S, Trinidad P, Ruiz-Cortés A (2017) Flame: a formal framework for the automated analysis of software product lines validated by automated specification testing. SOSYM 16(4):1049–1082. https://doi.org/10.1007/s10270-015-0503-z
Felfernig A, Walter R, Galindo JA, Benavides D, Erdeniz SP, Atas M, Reiterer S (2018) Anytime diagnosis for reconfiguration. J Intell Inf Syst 51(1):161–182. https://doi.org/10.1007/s10844-017-0492-1
Fernández-Cerero D, Varela-Vaca ÁJ, Fernández-Montes A, Gómez-López MT, Alvárez-Bermejo JA (2019) Measuring data-centre workflows complexity through process mining: the google cluster case. J Supercomput. https://doi.org/10.1007/s11227-019-02996-2
Ferreira DR, Alves C (2011) Discovering user communities in large event logs. In: Daniel F, Barkaoui K, Dustdar S (eds) Business process management workshops—BPM 2011 international workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I, Springer, Lecture Notes in Business Information Processing, vol 99, pp 123–134. https://doi.org/10.1007/978-3-642-28108-2_11
Frey T, Van Groenewoud H (1972) A cluster analysis of the d2 matrix of white spruce stands in saskatchewan based on the maximum-minimum principle. J Ecol 60(3):873–886
Galindo J, Turner H, Benavides D, White J (2014a) Testing variability-intensive systems using automated analysis: an application to android. Softw Qual J 1–41. https://doi.org/10.1007/s11219-014-9258-y
Galindo JA, Alférez M, Acher M, Baudry B, Benavides D (2014b) A variability-based testing approach for synthesizing video sequences. In: International symposium on software testing and analysis, ISSTA '14, San Jose, CA, USA—July 21–26, 2014, pp 293–303
Galindo J, Dhungana D, Rabiser R, Benavides D, Botterweck G, Grünbacher P (2015) Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. Inf Softw Technol 62(1):78–100
Galindo JA, Benavides D, Trinidad P, Gutiérrez-Fernández AM, Ruiz-Cortés A (2018) Automated analysis of feature models: Quo vadis? Computing 101:387–433
Ghionna L, Greco G, Guzzo A, Pontieri L (2008) Outlier detection techniques for applications. In: Foundations of intelligent systems. Springer, Berlin, pp 150–159
Grabusts P et al (2011) The choice of metrics for clustering algorithms. In: Proceedings of the 8th international scientific and practical conference, vol 2, pp 70–76
Greco G, Guzzo A, Pontieri L, Sacca D (2006) Discovering expressive process models by clustering log traces. IEEE Trans Knowl Data Eng 18(8):1010–1027
Halkidi M, Vazirgiannis M, Batistakis Y (2000) Quality scheme assessment in the clustering process. In: European conference on principles of data mining and knowledge discovery. Springer, pp 265–276
Hartigan JA (1975) Clustering algorithms, 99th, John Wiley & Sons, Inc., USA
Hompes BFA, Verbeek HMW, van der Aalst WMP (2015) Finding suitable activity clusters for decomposed process discovery. In: Ceravolo P, Russo B, Accorsi R (eds) Data-driven process discovery and analysis. Springer International Publishing, Cham, pp 32–57
Hompes BFA, Buijs JCAM, van der Aalst WMP, Dixit PM, Buurman J (2017) Detecting changes in process behavior using comparative case clustering. In: Ceravolo P, Rinderle-Ma S (eds) Data-driven process discovery and analysis. Springer International Publishing, pp 54–75
Hubaux A, Classen A, Heymans P (2009) Formal modelling of feature configuration workflows. In: Proceedings of the 13th international software product line conference, Carnegie Mellon University, Pittsburgh, PA, USA, SPLC '09, pp 221–230. http://dl.acm.org/citation.cfm?id=1753235.1753266
Hubaux A, Heymans P, Schobbens PY, Deridder D, Abbasi E (2013) Supporting multiple perspectives in feature-based configuration. SOSYM 12(3):641–663. https://doi.org/10.1007/s10270-011-0220-1. http://www.scopus.com/inward/record.url?eid=2-s2.0-84879788174&partnerID=40&md5=dee1ff6a27f859c32d424a1528d81ada
Hubert L (1974) Approximate evaluation techniques for the single-link and complete-link hierarchical clustering procedures. J Am Stat Assoc 69(347):698–704
Hubert LJ, Levin JR (1976) A general statistical framework for assessing categorical clustering in free recall. Psychol Bull 83(6):1072
Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

Kobren A, Monath N, Krishnamurthy A, McCallum A (2017) A hierarchical algorithm for extreme clustering. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '17. ACM, New York, pp 255–264

Krzanowski WJ, Lai Y (1988) A criterion for determining the number of groups in a data set using sum-of-squares clustering. Biometrics 44(1):23–34

Kuiper FK, Fisher L (1975) 391: a Monte Carlo comparison of six clustering procedures 777–783. Biometrics 31(3):777–783

Lebart L, Morineau A, Piron M (2000) Statistique exploratoire multidimensionnelle, Dunod, Paris, France

Leemans SJJ, Fahland D, van der Aalst WMP (2014) Discovering block-structured process models from incomplete event logs. In: Petri Nets, Springer, Lecture Notes in Computer Science, vol 8489, pp 91–110

Leemans SJJ, Fahland D, van der Aalst WMP (2015) Scalable process discovery with guarantees. In: Gaaloul K, Schmidt R, Nurcan S, Guerreiro S, Ma Q (eds) Enterprise, business-process and information systems modeling. Springer International Publishing, Cham, pp 85–101

Lettner M, Rodas-Silva J, Galindo JA, Benavides D (2019) Automated analysis of two-layered feature models with feature attributes. J Comput Lang 51:154–172

Ly LT, Indiono C, Mangler J, Rinderle-Ma S (2012) Data transformation and semantic log purging for process mining. In: CAiSE, Springer, Lecture notes in computer science, vol 7328, pp 238–253

MacKay DJC (2002) Information theory inference & learning algorithms. Cambridge University Press, New York

Makanju A, Brooks S, Zincir-Heywood AN, Milios EE, Safavi-Naini R (2008) Logview: visualizing event log clusters. In: Korba L, Marsh S (eds) Sixth annual conference on privacy, security and trust, PST 2008, October 1–3, 2008. IEEE Computer Society, Fredericton, pp 99–108. https://doi.org/10.1109/PST.2008.17

Makanju A, AN Zincir-Heywood, Milios EE (2009) Clustering event logs using iterative partitioning. In: IV JFE, Fogelman-Soulié F, Flach PA, Zaki MJ (eds) Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, Paris, France, June 28–July 1, 2009. ACM, pp 1255-1264. https://doi.org/10.1145/1557019.1557154

Mans RS, Schonenberg MH, Song M, van der Aalst WMP, Bakker PJM (2009) Application of process mining in healthcare—a case study in a dutch hospital. In: Fred A, Filipe J, Gamboa H (eds) Biomedical engineering systems and technologies. Springer, Berlin, pp 425–438

Măruşter L, van Beest NRTP (2009) Redesigning business processes: a methodology based on simulation and techniques. Knowl Inf Syst 21(3):267. https://doi.org/10.1007/s10115-009-0224-0

Maruster L, Weijters AJMM, van der Aalst WMP, van den Bosch A (2002) Process mingin: discovering direct successors in process logs. In: Discovery Science, 5th international conference, DS 2002, Lübeck, Germany, November 24–26, 2002, Proceedings, pp 364–373. https://doi.org/10.1007/3-540-36182-0_37

Maruster L, Weijters AJMM, van der Aalst WMP, van den Bosch A (2006) A rule-based approach for process discovery: dealing with noise and imbalance in process logs. Data Min Knowl Discov 13(1):67–87

McClain JO, Rao VR (1975) Clustisz: a program to test for the quality of clustering of a set of objects. JMR. J Market Res (pre-1986) 12(000004):456

Mendling J (2008) Metrics for business process models. Springer, Berlin, pp 103–133

Milligan GW (1980) An examination of the effect of six types of error perturbation on fifteen clustering algorithms. Psychometrika 45(3):325–342

Milligan GW (1981) A monte carlo study of thirty internal criterion measures for cluster analysis. Psychometrika 46(2):187–199

Murtagh F (1983) A survey of recent advances in hierarchical clustering algorithms. Comput J 26(4):354–359. https://doi.org/10.1093/comjnl/26.4.354. http://oup.prod.sis.lan/comjnl/article-pdf/26/4/354/1072603/26-4-354.pdf

Pereira JA, Matuszyk P, Krieter S, Spiliopoulou M, Saake G (2016a) A feature-based personalized recommender system for product-line configuration. In: Proceedings of the international conference on generative programming: concepts and experiences. ACM, pp 120–131

Pereira JA, Matuszyk P, Krieter S, Spiliopoulou M, Saake G (2016b) A feature-based personalized recommender system for product-line configuration. In: Proceedings of the international conference on generative programming: concepts and experiences. ACM, pp 120–131

Pereira JA, Schulze S, Figueiredo E, Saake G (2018a) N-dimensional tensor factorization for self-configuration of software product lines at runtime. In *Proceedings of the 22nd International Systems and Software Product Line Conference - Volume 1 (SPLC '18)*. Association for Computing Machinery, New York, NY, USA, 87–97. https://doi.org/10.1145/3233027.3233039

Pereira JA, Matuszyk P, Krieter S, Spiliopoulou M, Saake G (2018b) Personalized recommender systems for product-line configuration processes. Comput Lang Syst Struct 54:451–471

Pérez-Álvarez JM, Maté A, López MTG, Trujillo J (2018) Tactical business-process-decision support based on kpis monitoring and validation. Comput Ind 102:23–39
Pérez-Castillo R, Fernéndez-Ropero M, Piattini M (2019) Business process model refactoring applying ibuprofen. An industrial evaluation. J Syst Softw 147:86–103
Perimal-Lewis L, Teubner D, Hakendorf P, Horwood C (2016) Application of process mining to assess the data quality of routinely collected time-based performance data sourced from electronic health records by validating process conformance. Health Inform J 22(4):1017–1029
Ratkowsky D, Lance G (1978) Criterion for determining the number of groups in a classification Vol. 44, No. 1, pages 23-34
Rodas-Silva J, Galindo JA, García-Gutiérrez J, Benavides D (2019) Selection of software product line implementation components using recommender systems: an application to wordpress. IEEE Access 7:69226–69245
Rohlf FJ (1974) Methods of comparing classifications. Annu Rev Ecol System 5(1):101–113
Rozinat A, de Jong ISM, Günther CW, van der Aalst WMP (2009) Process mining applied to the test process of wafer scanners in ASML. IEEE Trans Syst Man Cybern Part C 39(4):474–479
Rubin V, Günther CW, van der Aalst WMP, Kindler E, van Dongen BF, Schäfer W (2007) Process mining framework for software processes. In: Wang Q, Pfahl D, Raffo DM (eds) Software process dynamics and agility. Springer, Berlin, pp 169–181
Rubin VA, Mitsyuk AA, Lomazova IA, van der Aalst WMP (2014) Process mining can be applied to software too! In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement. ESEM '14. ACM, New York, pp 57:1–57:8
Sahlabadi M, Muniyandi R, Shukur Z (2014) Detecting abnormal behavior in social network websites by using a process mining technique. J Comput Sci 10(3):393–402. https://doi.org/10.3844/jcssp.2014.393.402
Sani MF, van Zelst SJ, van der Aalst WMP (2017) Improving process discovery results by filtering outliers using conditional behavioural probabilities. In: Business process management workshops—BPM 2017 international workshops, Barcelona, Spain, September 10–11, 2017, Revised Papers, pp 216–229. https://doi.org/10.1007/978-3-319-74030-0_16
Sani MF, Boltenhagen M, van der Aalst W (2019) Prototype selection based on clustering and conformance metrics for model discovery. https://arxiv.org/pdf/1912.00736.pdf
Schobbens P, Heymans P, Trigaux J, Bontemps Y (2007) Generic semantics of feature diagrams. Comput Netw 51(2):456–479. https://doi.org/10.1016/j.comnet.2006.08.008
She S, Lotufo R, Berger T, Wasowski A, Czarnecki K (2010) The variability model of the linux kernel. In: VAMOS, vol 10, pp 45–51
Song M, Günther CW, van der Aalst WMP (2008) Trace clustering in process mining. In: Ardagna D, Mecella M, Yang J (eds) Business process management workshops, BPM 2008 international workshops, Milano, Italy, September 1–4, 2008. Revised Papers, Springer, Lecture Notes in Business Information Processing, vol 17, pp 109–120. https://doi.org/10.1007/978-3-642-00328-8_11
Song M, Günther CW, van der Aalst WMP (2009) Trace clustering in. In: Ardagna D, Mecella M, Yang J (eds) Business Process Management Workshops. Springer, Berlin, pp 109–120
Tax N, Sidorova N, van der Aalst WMP (2019) Discovering more precise process models from event logs by filtering out chaotic activities. J Intell Inf Syst 52(1):107–139. https://doi.org/10.1007/s10844-018-0507-6
Thüm T, Apel S, Kästner C, Schaefer I, Saake G (2014) A classification and survey of analysis strategies for software product lines. ACMCS 47(1). https://doi.org/10.1145/2580950
Valencia-Parra A, Ramos-Gutiérrez B, Varela-Vaca AJ, López MTG, Bernal AG (2019a) Enabling process mining in aircraf manufactures: extracting event logs and discovering processes from complex data. In: Proceedings of the industry forum at BPM 2019 co-located with 17th international conference on business process management (BPM 2019), Vienna, Austria, September 1–6, 2019, pp 166–177
Valencia-Parra Á, Varela-Vaca ÁJ, Gómez-López MT, Ceravolo P (2019b) CHAMALEON: framework to improve data wrangling with complex data. In: Proceedings of the 40th international conference on information systems, ICIS 2019, Munich, Germany, December 15–18, 2019
van der Aalst WMP (2011) Analyzing "spaghetti processes". Springer, Berlin
van der Aalst WMP (2016) Process mining–data science in action, 2nd edn. Springer, Berlin
van Dongen BF, de Medeiros AKA, Verbeek HMW, Weijters AJMM, van der Aalst WMP (2005) The prom framework: a new era in process mining tool support. In: Applications and theory of Petri nets 2005, 26th international conference, ICATPN 2005, Miami, USA, June 20–25, 2005, Proceedings, pp 444–454. https://doi.org/10.1007/11494744_25
vanden Broucke SKLM, Weerdt JD (2017) Fodina: a robust and flexible heuristic process discovery technique. Decis Support Syst 100:109–118. https://doi.org/10.1016/j.dss.2017.04.005

Varela-Vaca AJ, Gasca RM (2013) Towards the automatic and optimal selection of risk treatments for business processes using a constraint programming approach. Inf Softw Technol 55(11):1948–1973
Varela-Vaca ÁJ, Galindo JA, Ramos-Gutiérrez B, Gómez-López MT, Benavides D (2019a) Process mining to unleash variability management: discovering configuration workflows using logs. In: Proceedings of the 23rd International Systems and Software Product Line conference, SPLC 2019, Volume A, Paris, France, September 9–13, 2019, pp 37:1–37:12
Varela-Vaca ÁJ, Gasca RM, Ceballos R, Gómez-López MT, Torres PB (2019b) Cyberspl: a framework for the verification of cybersecurity policy compliance of system configurations using software product lines. Applied Sciences 9(24). https://doi.org/10.3390/app9245364. https://www.mdpi.com/2076-3417/9/24/5364
Wang Y, Tseng MM (2011) Adaptive attribute selection for configurator design via shapley value. Artif Intell Eng Des Anal Manuf 25(2):185–195. https://doi.org/10.1017/S0890060410000624
Wang Y, Tseng M (2014) Attribute selection for product configurator design based on gini index. Int J Prod Res 52(20):6136–6145. https://doi.org/10.1080/00207543.2014.917216
Ward JH Jr (1963) Hierarchical grouping to optimize an objective function. J Am Stat Assoc 58(301):236–244
Weijters AJMM, Ribeiro JTS (2011) Flexible heuristics miner (FHM). In: CIDM. IEEE, pp 310–317
Wilcoxon F (1946) Individual comparisons of grouped data by ranking methods. J Econ Entomol 39(2):269–270
XES (2016) IEEE Standard for eXtensible Event Stream (XES) for achieving interoperability in event logs and event streams. IEEE Std 1849-2016 pp 1–50. https://doi.org/10.1109/IEEESTD.2016.7740858

Affiliations

Belén Ramos-Gutiérrez[1] · Ángel Jesús Varela-Vaca[1] · José A. Galindo[1] · María Teresa Gómez-López[1] · David Benavides[1]

Ángel Jesús Varela-Vaca
ajvarela@us.es

José A. Galindo
jagalindo@us.es

María Teresa Gómez-López
maytegomez@us.es

David Benavides
benavides@us.es

[1] Data-Centric Computing Research Hub (IDEA), Universidad de Sevilla, Seville, Spain