Received June 10, 2020, accepted July 13, 2020, date of publication July 21, 2020, date of current version July 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3010852

Extracting User-Centric Knowledge on Two Different Spaces: Concepts and Records

JOSÉ MARÍA LUNA[1], **PHILIPPE FOURNIER-VIGER**[2], **AND SEBASTIÁN VENTURA**[1], **(Senior Member, IEEE)**
[1]Department of Computer Science and Numerical Analysis, University of Cordoba, 14071 Cordoba, Spain
[2]School of Humanities and Social Science, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China

Corresponding author: Sebastián Ventura (sventura@uco.es)

This work was supported by the Spanish Ministry of Science and Innovation, project TIN2017-83445-P, and the University of Cordoba, project UCO-FEDER 18 REF.1263116 MOD.A. Both projects were also supported by the European Fund of Regional Development.

ABSTRACT The growing demand for eliciting useful knowledge from data calls for techniques that can discover insights (in the form of patterns) that users need. Methodologies for describing intrinsic and relevant properties of data through the extraction of useful patterns, however, work on fixed input data, and the data representation, therefore, constrains the discovered insights. In this regard, this paper aims at providing foundations to make the descriptive knowledge that is extracted by pattern mining more user-centric by relying on flexible data structures defined on two different perspectives: concepts and data records. In this sense, items in data can be grouped into abstract terms through subjective hierarchies of concepts, whereas data records can also be organized based on the users' subjective perspective. A series of easy-to-follow toy examples are considered for each of the two perspectives to demonstrate the usefulness and necessity of the proposed foundations in pattern mining. Finally, aiming at experimentally testing whether classical pattern mining algorithms can be adapted to such flexible data structures, the experimental analysis comprises different methodologies, including exhaustive search, random search, and evolutionary approaches. All these approaches are based on well-known and widely recognized techniques to demonstrate the usefulness of the provided foundations for future research works and more efficient and specifically designed algorithms. Obtained insights demonstrate the importance of working with subjectivity: an item is a type of soda but belongs to a pack, including two or more soda types.

INDEX TERMS Pattern mining, space of concepts, space of records, user-centric knowledge.

I. INTRODUCTION

Over the last decade there has been a massive explosion of data collected in almost any application domain, leading to an exponentially growing interest in managing and transforming tons of facts into useful information [1]. Generally, raw data is uninteresting and an in-depth analysis is required to obtain different forms of data from which new information can be derived [2]. The process of discovering valuable insights from a collection of records has given rise to the field known as knowledge discovery in databases (KDD) [3]. KDD methods and techniques are generally user-centric as the aim is to make sense of data and decrease its uncertainty.

In general terms, the key element in the process of eliciting useful knowledge is the pattern [4], which defines subsequences, substructures or itemsets (sets of values) [5] that represent any type of homogeneity and regularity in data [6]. Patterns represent intrinsic and important properties of datasets, and these patterns are required to be novel, significant, unexpected, nontrivial and actionable [7]. Given a set of items (values or symbols) $I = \{i_1, i_2, \ldots, i_n\}$ in a database Ω, a pattern P describes valuable data features and it can be formally defined as a subset of I, i.e. $P = \{i_j, \ldots, i_k\} \subseteq I \in \Omega$, $1 \leq j, k \leq n$. The value of the discovered information lies not only in the knowledge itself but in the actions that can be taken as a result of the insights [8]. A well-known example of application domain is market basket analysis in which the extraction of patterns and analysis of correlations between items have been largely studied [9]. Here, the aim is to understand consumer purchasing habits to design successful marketing strategies. For instance, discovering that someone who acquires a product A has a high probability of also buying

The associate editor coordinating the review of this manuscript and approving it for publication was Marta Cimitile.

a product *B* provides valuable insights to managers, who may then co-promote these products by launching a tailored advertising campaign.

Nowadays, actions based on data insights are more and more relevant in any application field and users' requirements are getting increasingly complex. These intricate expectations can be hardly fulfilled using insights discovered by conventional pattern mining approaches [7]. Hence, novel mechanisms that enable extracting more valuable knowledge are required [10]. In this regard, there are some progresses in supplying existing pattern mining approaches [11] with methods to extract more actionable insights [6]. For example, some approaches enable restricting the search space with various constraints [12]; others are ready to use on new and more flexible forms of information [13]; and finally, some methods can handle context-sensitive concepts to avoid discriminative behavior [14]. Nevertheless, even when there is an increasing awareness of the importance of extracting an appropriate knowledge type, it is still far from accomplishing the users' aim, and a more in-depth analysis is usually required to fully understand the discovered insights.

It is obvious that the knowledge discovered by any pattern mining approach highly depends on the stored information so different insights can be produced only from disparate information. Nevertheless, the same data can be analysed from contrasting perspectives or views that will produce completely different results (comprising the same information but expressed differently), and these results may be useful for a specific purpose but useless for another. This is similar to the fact that two people may describe the same thing from two different angles and both be right. Based on this idea, this paper aims to provide the foundations to make the descriptive knowledge that is extracted by pattern mining more user-centric [15] by relying on flexible data structures defined on two different spaces or perspectives: concepts and records. In this regard, the contribution of this research work can be summarized as follows:

1) The idea of flexibility in the input data is provided by defining items (concepts in the domain at hand) on different levels of abstraction through a context-free grammar, which minimizes the limitations of using homogeneous tree structures to represent the subjective knowledge.
2) The proposed use of context-free grammars enables the subjective knowledge to be modelled as inhomogeneous taxonomies (not all the concepts include abstractions in every level) and considering ambiguous concepts (each node is not unequivocally identified by an ancestor node).
3) Because data organization is key to produce the right insights, different ways of handling such arrangement are provided. Any pattern analysed from the whole dataset point of view will produce an insight that highly differs from that obtained when the same pattern is analysed on records grouped by the same object, e.g. the same customer.
4) The proposed methodology is able to cope with either objective and subjective information to group records. Additionally, it is able to consider different levels of ambiguity, which is essential in many situations (for example, descriptive analysis considering different customers and periods of time).

The final aim of this paper is to set the bases for further research studies on the ideas here presented, so the algorithmic solutions presented for each of the two perspectives are just adaptations of well-known algorithms demonstrating that all the proposed ideas are feasible to be carried out. Here, user-centring insights can be obtained, which are easily applicable (improving understanding) and focused on the task at hand (increasing actionability). The rest of the paper is organized as follows. Sections II and III describe different to handle data so more user-centric descriptions and insights can be produced from two different views: concepts and data records. Then, some classic approaches are adapted to the proposed foundations (see Section IV), which are then experimentally analyzed (see Section V). Finally, a lesson learned is described in Section VI and some concluding remarks are outlined in Section VII.

II. SPACE OF CONCEPTS

The space of concepts in pattern mining is a key element in the discovery of insights from data. Heretofore, the amount of concepts or items has been mainly considered as an invariant number so data are generally represented either as a number of transactions including a series of items or as a tabular representation with a fixed number of columns (one per item). These representations in isolation cannot include all forms of information since sets of items might appear because of some abstract information not defined in such input data. For example, information about smoking habits is only applicable for smokers and such blended information cannot be considered as missing data for non-smokers. It is therefore hardly identifiable whether a record belongs to a smoker or non-smoker unless any subjective information out of the input data is provided. Aiming at providing ways of extracting more powerful, applicable, actionable and user-centric insights impossible to be extracted with primitive concepts included in traditional input data, this section proposes to enrich such primitive concepts with subjective knowledge through a taxonomy of concepts. More specifically, a context-free grammar is considered to represent different forms of taxonomies depending on the final requirements. This section provides two different visions of the problem: inhomogeneous taxonomies and ambiguous concepts.

A. INHOMOGENEOUS TAXONOMIES

Traditional pattern mining [9] works on a fixed data structure where the number of both items and transactions is predetermined. Sometimes, however, this data structure is not enough to extract useful information from data and more abstract as well as subjective knowledge is required to be provided. In this regard, hierarchical pattern mining [16], [17] was proposed to extract patterns at different levels of abstraction always combining items from the same level. Here, the level of abstraction for a specific item is encoded through a taxonomy of concepts represented in a homogeneous tree shape (any primitive concept has an abstract concept in each level of abstraction). Items are accordingly encoded through a fixed-length string of digits that represents from the primitive ideas to higher and more abstract concepts [18]. It is noteworthy that this encoding might be performed during the data collection process and no extra encoding pass is therefore required. According to Han and Fu [13], this encoding procedure is more convenient than using standard SQL queries which are generally related to only a portion of the transactional database [19]. When considering a variable number of concepts, however, it is not easy to follow an encoding based on a fixed-length string of digits since not all the concepts include abstractions in every level. Thus, working with homogeneous trees (fixed-length strings of digits) is not an option in real-problems and new ways of handling concepts at any level of abstraction are needed. In other words, existing methods do not support all forms of knowledge so it is crucial to provide new mechanisms capable of producing adaptable insights that meet the increasingly complex expectations of the users.

Taking all the above into consideration, this paper proposes the use of context-free grammars to represent concept hierarchies and to produce the right insights. Here, the paths from the root to the primitive concepts in the hierarchy are analysed. Two elements (items or more abstract concepts) cannot be combined (appear in the same pattern) if the groups of elements obtained by traversing their paths are included one in another. The final idea is to avoid combining items that produce the same insights. This methodology is really promising since it not only enables elements from different levels of abstraction to be combined, but it also enables heterogeneous trees to represent subjective knowledge. In other words, it is not restricted to regular tree structures where every primitive concept has a higher abstract concept at every level of the hierarchy.

As a matter of clarification, let us consider a toy dataset related to a market basket as shown in Table 1. Let us also consider a sample concept taxonomy represented through a context-free grammar (see Figure 1) in which not every concept has the same number of abstraction levels. As it is illustrated, items {BumGenius}, {Huggies} and {Pampers} belong to the concept of diapers. Additionally, the concept of beer is divided into two subtypes, that is, regular and light. Here, items {Luvs} and {SamuelAdams} belong to the first subtype, whereas {Budweiser} and {Heineken} belong to the second subtype. As a result, not every single item is represented with the same number of abstraction levels.

G = $(\Sigma_N, \Sigma_T, P, S)$ with:
S = <Product>
Σ_N = {<Product>, <Diapers>, <Beer>, <Regular>, <Light>}
Σ_T = {Huggies, BumGenius, Pampers, Luvs, SamuelAdams, Budweiser, Heineken}
P = {<Product> ⇐ <Diapers> | <Beer>;
<Beer> ⇐ <Regular> | <Light> ;
<Diapers> ⇐ BumGenius | Huggies | Pampers ;
<Regular> ⇐ Luvs | SamuelAdams ;
<Light> ⇐ Budweiser | Heineken ;
}

FIGURE 1. Context-free grammar defined to represent a hierarchy and expressed in extended BNF notation.

Following the proposed methodology based on paths of the hierarchy (context-free grammar), the following patterns can be obtained (absolute frequency is denoted in brackets) from Table 1 and considering the hierarchy denoted by Figure 1:

- {Beer}(9)
- {Diapers}(9)
- {Huggies}(8)
- {Beer, Diapers}(8)
- {Light}(7)
- {Huggies, Beer}(7)
- {Diapers, Light}(6)
- {Regular}(5)
- {Luvs}(5)
- {Pampers}(5)
- {Diapers, Regular}(5)
- {Budweiser}(5)
- {Pampers, Beer}(5)
- . . .
- {Diapers, Luvs}(4)
- {Diapers, Budweiser}(4)
- {Huggies, Pampers}(4)
- . . .
- {SamuelAdams}(3)
- {Heineken}(3)
- {BumGenius}(3)
- {Diapers, SamuelAdams}(3)
- . . .

As it was already described, it is possible to combine items belonging to different levels of abstraction. For example, the pattern {Diapers, Light} reveals that 6 people bought diapers (no matter the brand) and light beer (no matter the brand) together. Similarly, the pattern {Diapers, Luvs} indicates that 4 people bought, at the same time, beer of the brand Luvs (a regular beer) and any kind of diapers. This

TABLE 1. Transactional dataset for a toy example of a market basket analysis.

Items
{BumGenius, Huggies, Luvs, SamuelAdams}
{Huggies, Pampers, Luvs, Budweiser}
{Huggies, Pampers, Heineken}
{BumGenius, Huggies, Pampers, Budweiser}
{Huggies, Pampers, Budweiser}
{Huggies, Luvs, Heineken}
{Huggies}
{BumGenius, Pampers, Luvs, SamuelAdams}
{Huggies, SamuelAdams, Budweiser}
{Luvs, Budweiser, Heineken}

G = $(\Sigma_N, \Sigma_T, P, S)$ with:
S = <Product>
Σ_N = {<Product>, <Diapers>, <Beer>, <WeekendPack>, <atLeast2BeerBrands>, <Regular>, <Light>}
Σ_T = {Huggies, BumGenius, Pampers, Luvs, SamuelAdams, Budweiser, Heineken}
P = {<Product> $\Leftarrow$ <Diapers> | <Beer> | <WeekendPack> ;
<Diapers> $\Leftarrow$ BumGenius | Huggies | Pampers ;
<Beer> $\Leftarrow$ <Regular> | <Light> | <atLeast2BeerBrands>;
<Regular> $\Leftarrow$ Luvs | SamuelAdams ;
<Light> $\Leftarrow$ Budweiser | Heineken ;
<atLeast2BeerBrands> $\Leftarrow$ Luvs, SamuelAdams | Luvs, Budweiser | Luvs, Heineken | SamuelAdams, Budweiser | Budweiser, Heineken |
SamuelAdams, Heineken | Luvs, SamuelAdams, Budweiser | Luvs, SamuelAdams, Heineken |
Luvs, Budweiser, Heineken | SamuelAdams, Budweiser, Heineken |
Luvs, SamuelAdams, Budweiser, Heineken;
<WeekendPack> $\Leftarrow$ Regular, Diapers;
}

FIGURE 2. Context-free grammar defined to represent a concept hierarchy including ambiguous concepts expressed in extended BNF notation.

ability to combine items belonging to any level of abstraction according to a context-free grammar provides a really useful tool for users to obtain accurate and actionable knowledge. Besides, the methodology follows the same process previously described to discard patterns by analysing their paths in the hierarchy. Thus, the pattern {Beer, Heineken} is a useless pattern and it is required to be discarded since Heineken is also a beer.

B. AMBIGUOUS CONCEPTS

As previously described, the ability to extract specific information on different levels of abstraction is essential for many companies to design successful marketing strategies [20]. In traditional hierarchical pattern mining, each concept of an abstraction level is unequivocally defined by a single ancestor in its upper level of abstraction. In some situations, however, a concept might be ambiguously defined by more than a unique abstract concept (two or more ancestors in its upper level). This problem can be seen as that of mining disjunctive relations among items [21], but existing algorithms [22] for this kind of relations do not consider any subjective knowledge so they only work on items within data. Some additional authors [16], [17] have partially solved the problem by including twice the brand concept, that is, one for each supertype in the hierarchy. Nevertheless, these proposals are unable to produce information about the percentage of the abstract concepts bought by customers. Considering the same market basket analysis shown in Table 1, let us consider a weekend pack that holds regular beer (no matter the brand) and a pack of diapers (no matter the brand). Now, regular beer is a subtype of beer, but it is also a subtype of the weekend pack. The main problem is that existing proposals consider a rigid tree shape (each node is unequivocally identified by an ancestor node) so ambiguity in terms of abstraction levels is impossible. Additionally, none of the existing proposals is able to define abstract concepts that depend on multiple combinations of elements. For example, imagine that it is required to analyse the buying habits considering packs of two or more packs of beers (no matter the brands). This ambiguous combination may be somehow related to the purchase of other products and it can be hardly extracted by existing pattern mining proposals.

The use of a context-free grammar to represent the hierarchy of concepts is essential to consider the aforementioned ambiguities. Similarly, the analysis of the paths (from the root to the primitive concepts) is key to avoid useless patterns, that is, those that produce the same insights. With all of this in mind, and considering the same market basket dataset shown in Table 1, let us consider the context-free grammar illustrated in Figure 2. This grammar includes two different ambiguous terms. First, regular beer might be a kind of beer and a part of the concept <WeekendPack>. Hence, this term is not unequivocally represented by an upper abstract concept. Second, the abstract concept <atLeast2BeerBrands> represents any combination of 2 or more types of beer brands. It might represent the pack formed by Luvs and Samuel Adams, as well as the pack formed by all the brands at the same time: Luvs, Samuel Adams, Budweiser, Heineken.

Following the proposed methodology based on paths of the hierarchy, the following patterns can be obtained (absolute frequency is denoted in brackets) from Table 1 and considering the hierarchy denoted by Figure 2:

- {Beer}(9)
- {Diapers}(9)
- {Beer, Diapers}(8)
- {Light}(7)
- {Huggies, Beer}(7)
- …
- {atLeast2BeerBrands}(6)
- {atLeast2BeerBrands, Diapers}(5)
- {WeekendPack}(5)
- …
- {WeekendPack, Light}(3)
- {SamuelAdams}(3)
- {Heineken}(3)
- {BumGenius}(3)
- {Diapers, SamuelAdams}(3)
- …

As it was shown, it is possible to combine items belonging to different levels of abstraction and considering ambiguity within data. For example, the pattern {atLest2BeerBrands} indicates that 6 people bought two or more packs of beer (no matter the brand). Similarly, the pattern {atLest2BeerBrands, Diapers} indicates that 5 people bought, at the same time, two or more packs of beer (no matter the brand) and any brand of diapers. Finally, it is also possible to extract the pattern {WeekendPack, Light}(3), denoting that 3 customers usually buy the weekend pack (diapers and any brand of regular beer) and they also buy light beer. Maybe this information provides clues about the necessity of considering not only regular but also light beer in the weekend pack (some family members will prefer light beer). Thus, to summarize, this ability to combine items belonging to any level of abstraction according to a context-free grammar provides a really useful tool for users to obtain accurate and actionable knowledge.

Last but not least, it is important to remark that, as already described, this methodology is able to discard some patterns, that is, those which items share paths in the hierarchy. For example, the pattern {Beer, atLeast2BeerBrands} is a useless pattern since <Beer> is a predecessor of <atLeast2BeerBrands> and, therefore, does not provide any useful insight. At this point it is important to remark two important things. First, the context-free grammar definition is key to produce the right insights and it is the responsability of the users. For example, if <atLeast2BeerBrands> concept is not a kind of <Beer> in the hierarchy, then both concepts would be compatible and might appear, therefore, in the same pattern. Second, when a specific concept has multiple paths to the root, all of them are taken into account. For example, <Regular> is a type of beer but it is also part of the weekend pack so none of the following patterns are useful: {Regular, WeekendPack} and {Regular, WeekendPack}.

III. SPACE OF RECORDS

Traditional datasets organize the information by transactions where each of these transactions is clearly defined by a single record including a set of elements or items. Sometimes, though, data information is ambiguous in the sense that a specific concept may be described by an undefined number of different records like, for example, a market basket dataset where each customer (the concept) is represented by multiple records (one per purchase). This ambiguity cannot be handled by traditional frequent pattern mining approaches, which provide a general description of homogeneity and regularity in the whole dataset. The state-of-the-art in descriptive analysis includes some studies [23] to deal with such ambiguity, aiming at extracting insights or patterns that describe groups of specific features. However, these studies are in an immature stage and they do not consider different views that may produce a completely different characterization of the results through considering either objective and subjective information. Besides, no studies related to different levels of ambiguity can be found, which is essential in many situations (for example, descriptive analysis considering different customers and periods of time). Hence, new and more flexible data representations on the space of records are required to meet the users' expectation and to obtain the right insights. In this section, new and different ways in which the space of records should be treated are grouped into two main groups (single and multiple level of ambiguity) as it is described below. It is also important to highlight that any of the proposed analyses of the space of data records is fully compatible with the analysis of the space of concepts described in the previous section.

A. SINGLE LEVEL OF AMBIGUITY

Data ambiguity that is inherent to many application domains was first studied by Dietterich *et al.* [24] in 1997 in the context of drug activity prediction, giving rise to the multiple-instance learning problem. Here, authors aimed at predicting whether a given molecule is a good drug molecule, which was measured by its ability to bind to a given target. Each molecule was represented as a bag of transactions (a set of instances or data records), and each one matched to a different conformation (molecular structure) of a particular compound [25]. Thus, each bag was associated with an outcome, that is, a discrete or real-valued label, whereas it was only known that each transaction belonged to a specific bag (no outcome was associated with a transaction but with its bag). Finally, the number of transactions belonging to each bag was not defined and it might vary from bag to bag (see Table 2 including m features and n bags).

TABLE 2. Structure of a sample multi-instance dataset with *n* bags.

Bag	Feature$_1$	Feature$_2$	Feature$_3$	...	Feature$_m$	Label
B_1	$F_{1,1,1}$	$F_{1,1,2}$	$F_{1,1,3}$	...	$F_{1,1,m}$	L_1
	$F_{1,2,1}$	$F_{1,2,2}$	$F_{1,2,3}$	...	$F_{1,2,m}$	
	...	...	...	...	...	
	$F_{1,i,1}$	$F_{1,i,2}$	$F_{1,i,3}$	...	$F_{1,i,m}$	
B_2	$F_{2,1,1}$	$F_{2,1,2}$	$F_{2,1,3}$	...	$F_{2,1,m}$	L_2
	...	...	...	...	...	
	$F_{2,j,1}$	$F_{2,j,2}$	$F_{2,j,3}$	...	$F_{2,j,m}$	
...	...	...	...	...	...	...
B_n	$F_{n,1,1}$	$F_{n,1,2}$	$F_{n,1,3}$	...	$F_{n,1,m}$	L_n
	...	...	...	...	...	
	$F_{n,k,1}$	$F_{n,k,2}$	$F_{n,k,3}$	...	$F_{n,k,m}$	

In traditional approaches for mining frequent patterns, there is no ambiguity in data and each transaction is clearly defined by a single record. However, as it was proposed by Luna *et al.* [23], there are some scenarios where patterns of interest are required to be extracted on inherently ambiguous domains. Here, a specific pattern occurs in a bag if it appears in at least one of its transactions. As a matter of example, let us consider the toy market basket dataset (see Table 1) used in the previous section but grouping the transactions by customers (each transaction is identified by a customer ID as shown in Table 3). Considering the single concept ID as the one for forming groups of transactions (those data records with the same ID are grouped under the same bag), four different bags are considered each one gathering a different number of data records. On this new scenario and given a database Ω comprising a set of n bags $\Omega = \{B^1, B^2, \ldots, B^n\}$, each bag B^j comprises an undetermined number of transactions, i.e.

TABLE 3. Transactional dataset for a toy example of a market basket analysis including customers' ID.

ID	Items
1	{BumGenius, Huggies, Luvs, SamuelAdams}
1	{Huggies, Pampers, Luvs, Budweiser}
1	{Huggies, Pampers, Heineken}
2	{BumGenius, Huggies, Pampers, Budweiser}
2	{Huggies, Pampers, Budweiser}
3	{Huggies, Luvs, Heineken}
3	{Huggies}
3	{BumGenius, Pampers, Luvs, SamuelAdams}
3	{Huggies, SamuelAdams, Budweiser}
4	{Luvs, Budweiser, Heineken}

$B^j = \{t_1^j, \ldots, t_p^j\}$. A pattern P occurs in a bag B^j if and only if P is a subset of at least one transaction t_i^j from B^j, i.e. $\exists t_i^j : t_i^j \in B^j \wedge P \subseteq t_i^j$. The frequency of occurrence of P in Ω, also known as $support(P)$ [26], is defined as the number of bags in which P satisfies at least one transaction, i.e. $support(P) = |\forall B^j \in \Omega, \exists t_i^j : t_i^j \in B^j \wedge P \subseteq t_i^j|$. Additionally, the relative support is denoted as $support_r(P) = support(P)/|\Omega|$. In this toy dataset, the pattern P ={Budweiser} is satisfied by all the four bags so $support(P) = 4$ and its relative support is $support_r(P) = 1.00$. This pattern denotes that 100% of the customers have bought beer of the brand Budweiser at least once in all their purchases. This knowledge highly differs from the one obtained on traditional data representation (see Table 1) where the relative support of the same pattern is $support_r(P) = 0.50$, denoting that only 50% of the transactions describe a purchase including beer of the brand Budweiser. This data representation can seen as one of sequential pattern mining [27] including multiple records per customers but considering single itemsets.

Heretofore, the analysis of patterns in bags through the flexibility in the space of records has stated that a pattern occurs in a bag if it appears in at least one of its transactions. It may provoke that a specific pattern could be satisfied in every transaction of a particular bag B^j and, at the same time, it may occur in just a single transaction of another bag B^k. The mere fact that a specific bag B^k includes a high number of transactions provokes a higher possibility of finding the pattern in such a bag. Hence, it may be of high interest for the usefulness of the insights to determine the significance of the pattern within each bag. In this regard, it might be valuable to determine a minimum percentage of transactions to be satisfied to consider that the pattern is fully described by the bag. Formally, and considering a database Ω comprising a set of n bags $\Omega = \{B^1, B^2, \ldots, B^n\}$, each bag B^j comprises an undetermined number of transactions, i.e. $B^j = \{t_1^j, \ldots, t_p^j\}$. A pattern P is significant in a bag B^j if and only if P is a subset of at least minimum percentage (min_p) of transactions in B^j, i.e. $|\forall t_i^j : t_i^j \in B^j \wedge P \subseteq t_i^j|/|B^j| \geq min_p$. The frequency of occurrence of P in Ω, also known as $support(P)$, is defined as the number of bags in which P satisfies at least a significance min_p, i.e. $support(P) = |\forall B^j, \forall t_i^j : B^j \in \Omega \wedge t_i^j \in B^j \wedge |P \subseteq t_i^j|/|B^j| \geq min_p|$. Additionally, the relative support is denoted as $support_r(P) = support(P)/|\Omega|$, similarly to the one already proposed in [28]. As a matter of example, let us consider again the view of the database organized by customers, which was illustrated in Table 3. A product bought by a customer is of high significance for him/her if it appears in a high number of transactions. Diapers of the brand Pampers is included in every purchase of the customer with ID #2, in 75% of the purchases of the customer with ID #1, and only in 25% of the purchases of the customer with ID #3. It demonstrates that the same item can be really important for specific customers and not much too important for others. Taking a minimum value of $min_p = 0.75$ (a pattern will occur in a bag if and only if it appears in at least 75% of its transactions), it is obtained now that the pattern P ={Budweiser} is satisfied by two bags (customers with IDs #2 and #4) so $support(P) = 2$ and its relative support is $support_r(P) = 0.50$. This pattern denotes that 50% of the customers usually buy beer of the brand Budweiser in their purchases. This knowledge highly differs from the one previously obtained (100% of the customers have bought beer of the brand Budweiser at least once in all their purchases).

Flexibility in the space of records allows the same dataset to be described by an undefined number of views according to the users' expectations. Each of these views implies a different organization of the bags in data, producing a completely different characterization of the results. Hence, it is crucial to apply the right view to achieve the expectations and the same dataset that was previously organized by customers' IDs (see Table 3) can also be organized by a different concept within data (or any abstract concept according to a hierarchy of concepts), enabling general trends for all the pre-fixed concepts to be described.

B. MULTIPLE LEVELS OF AMBIGUITY

The space of records in data has been grouped into different ways to obtain different views that provide the right insights (according to the users' expectations). Up to this moment, data records have been grouped according to a single level or concept (either subjective or objective concepts may be considered). However, this assemblage of data records based on a single level is useless in many situations and not a single but multiple levels are required to carry out the grouping procedure. As a matter of clarification, let us consider that experts want to know which products are usually bought together at least once for most of the customers regardless the season (some products are seasonal and their sales highly increase in a specific season). The fact of defining the bags of records through a single concept does not solve the problem since the season in which the product was bought is also required. Now, each bag of records is also divided into sub-bags of records and the number of sub-bags, as well as records per sub-bag, is completely different from a bag and another. Formally, given a database Ω comprising a set of n bags $\Omega = \{B^1, B^2, \ldots, B^n\}$, each bag B^j comprises an undetermined number of sub-bags $S_1^j, S_2^j, \ldots, S_m^j$ and each

TABLE 4. Transactional dataset for a toy example of a market basket analysis including customers' ID and season.

ID	Season	Items
1	A	{BumGenius, Huggies, Luvs, SamuelAdams}
		{Huggies, Pampers, Luvs, Budweiser}
	B	{Huggies, Pampers, Heineken}
2	A	{BumGenius, Huggies, Pampers, Budweiser}
		{Huggies, Pampers, Budweiser}
3	B	{Huggies, Luvs, Heineken}
	C	{Huggies}
		{BumGenius, Pampers, Luvs, SamuelAdams}
	D	{Huggies, SamuelAdams, Budweiser}
4	C	{Luvs, Budweiser, Heineken}

sub-bag S_k^j gathers an undetermined number of transactions, i.e. $S_k^j = \{t_1^k, \ldots, t_p^k\}$. In general terms, a pattern P occurs in a bag B^j if and only if P is a subset of at least one transaction t_i^k from S_k^j and it happens for every sub-bag in such bag, i.e. $\forall S_k^j \in B^j$. The frequency of occurrence of P in Ω, also known as $support(P)$, is defined as the number of bags in which P is satisfied, i.e. $support(P) = |\{\forall B^j \in \Omega : (\forall S^k \in B^j, \exists t_i^k : t_i^k \in S^k \land P \subseteq t_i^k)\}|$. Additionally, the relative support is denoted as $support_r(P) = support(P)/|\Omega|$. As a matter of clarification, let us consider the toy example for a market basket dataset where the customers' ID and the season in which the purchases were carried out are considered (see Table 4). Considering multiple concepts, it is obtained that the pattern $P = \{\text{Pampers}\}(2)$ is satisfied for two customers (ID #1 and #2) in any of the seasons. The relative support for this pattern is therefore $support_r(P) = 0.5$, meaning that 50% of the customers buy diapers of the brand Pampers at least once per season (denoting that this is not a seasonal product for half of the customers). Here, it is important to highlight that the concept of season may be an abstract concept obtained by a date. In other words, the same date may be considered as a period A or B depending on the meaning the users want to consider (seasons, semesters, holiday periods, etc).

The use of multiple concepts in the space of records provides novel powerful mechanisms to extract what users really need, considering new views in which patterns are mined (completely new insights are therefore obtained). It should be highlighted that the number of levels in which the space of records is organized is related to the probability of finding frequent patterns (a higher number of levels will hamper the chance of mining a pattern that frequently occurs for all the levels). However, the extraction of frequent patterns on flexible space of records produces knowledge that is more in connection to the users' background as well as in accordance with their expectations. The concepts used to organize data records highly depends on the users (and their expectations), and the percentage of records satisfied per sub-bag is also a pre-requisite that can be modified by the users. As a matter of example, let us consider now the same toy example organized by customers and seasons (see Table 4). Additionally, let us consider that a bag B^j is satisfied if and only if most of its sub-bags are also satisfied ($\geq$50% of the sub-bags include at least one transaction that satisfies the pattern). In this scenario, it is obtained that $P = \{\text{Luvs}\}(3)$, stating that 75% of the customers buy beer of the brand Luvs at least once per season in most of the seasons (more than 50% of the seasons). Thus, it is not only important to consider how data records are grouped but also the thresholds that patterns must satisfy for a bag and sub-bag.

IV. APPROACHES

To demonstrate the usefulness of the proposed ideas and how they can be accomplished, disparate methodologies including exhaustive search, random search and evolutionary approaches are proposed. All these approaches are just adaptations of well-known and widely recognised techniques with the aim of serving as a demonstration of the usefulness of the provided foundations for future research works as well as more efficient and specifically designed algorithms. Adapted algorithms belong to two different methodologies (exhaustive search and heuristic-based approaches).

A. SPACE OF CONCEPTS

Different adaptations based on contrasting methodologies are proposed in this section to deal with flexibility in the space of concepts. First, let us propose an exhaustive search approach (see Algorithm 1), based on the well-known Apriori algorithm [29], in which valid patterns are analysed after the mining procedure. In this first proposal, the exhaustive search process is applied to a transformed database according to the provided grammar. Each path and subpath within the grammar should be therefore transformed into an item within the new database (see lines 3 to 6, Algorithm 1). After that, the proposal generates frequent itemsets as traditionally Apriori does, the main difference is the final process (see lines 24 to 28, Algorithm 1) in which the proposal analyses each solution and checks whether the solution is valid or not (according to the proposed grammar).

The previous proposal has the main drawback that any pattern, valid or not, is previously mined. This issue hinders the mining process since the number of items of the transformed database is usually much greater than that of the original database. In this regard, a modified approach is also proposed so the mining of candidate patterns takes into account not only the support of the patterns but also whether it is a valid pattern or not. It implies a reduction in the number of patterns to be analysed and, therefore, a more efficient algorithm. Similarly to the previous approach, the mining process is applied to a transformed database according to the provided grammar. Each path and subpath within the grammar should be therefore transformed into an item within the new database (see lines 3 to 6, Algorithm 2). Then, the proposed methodology generates candidate itemsets (see line 15, Algorithm 2) from the previous set of frequent itemsets, and these candidates $\mathcal{C}_j$ need to satisfy two restrictions: *a*) $\nexists i \in \mathcal{C}_j : i_x \in i, i_y \in i,\ path(i_x) \subseteq path(i_y)$. In other words, there is no pair of items in $i \in \mathcal{C}_j$ in which elements in the path of one of the items

Algorithm 1 Exhaustive Search Approach in Which Valid Patterns Are Analysed After the Mining Procedure

```
Require: $\mathcal{D}$,G, $\alpha$   ▷ Dataset $\mathcal{D}$, grammar $G$ and support threshold $\alpha$
Ensure: $\mathcal{R}$   ▷ Set $\mathcal{R}$ of discovered patterns
1: $\mathcal{L} \leftarrow \emptyset$
2: $\Omega \leftarrow \emptyset$   ▷ Encoded dataset
3: for all path $p \in G$ do   ▷ Analyse each path $p$ within the grammar $G$
4:     $t \leftarrow$ getTransactions($p$, $\mathcal{D}$)   ▷ Transactions satisfied by $p$ in $\mathcal{D}$
5:     $\Omega \leftarrow \Omega \cup < p, t >$   ▷ The path $p$ and transactions $t$ are added to $\Omega$
6: end for
7: $\mathcal{L}_1 \leftarrow \{1 - itemsets\}$   ▷ Generate all the single itemsets
8: for all items $i \in \mathcal{L}_1$ do
9:     if support($i$, $\Omega$)$\geq \alpha$ then   ▷ Check whether $i$ is frequent
10:        $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup i$
11:    end if
12: end for
13: $j = 2$
14: while $\mathcal{L}_{j-1} \neq \emptyset$ do   ▷ Generate all the $j$-itemsets from the set of $j-1$
15:     $\mathcal{C}_j \leftarrow$ generateCandidates($\mathcal{L}_{j-1}$)
16:     for all patterns $p \in \mathcal{C}_j$ do
17:         if support($p$, $\Omega$)$\geq \alpha$ then   ▷ Check if $p$ is a frequent pattern
18:             $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup p$
19:         end if
20:     end for
21:     $j \leftarrow j + 1$
22: end while
23: $\mathcal{R} \leftarrow \emptyset$
24: for all patterns $p \in \mathcal{L}$ do   ▷ Check common paths
25:     if $\nexists i_x, i_y \in p : path(i_x) \subseteq path(i_y)$ then
26:         $\mathcal{R} \leftarrow \mathcal{R} \cup p$
27:     end if
28: end for
29: return $\mathcal{R}$
```

Algorithm 2 Exhaustive Search Approach in Which Valid Patterns Are Analysed During the Mining Procedure

```
Require: $\mathcal{D}$,G, $\alpha$   ▷ Dataset $\mathcal{D}$, grammar $G$ and support threshold $\alpha$
Ensure: $\mathcal{L}$   ▷ Set $\mathcal{L}$ of discovered patterns
1: $\mathcal{L} \leftarrow \emptyset$
2: $\Omega \leftarrow \emptyset$   ▷ Encoded dataset
3: for all path $p \in G$ do   ▷ Analyse each path $p$ within the grammar $G$
4:     $t \leftarrow$ getTransactions($p$, $\mathcal{D}$)   ▷ Transactions satisfied by $p$ in $\mathcal{D}$
5:     $\Omega \leftarrow \Omega \cup < p, t >$   ▷ The path $p$ and transactions $t$ are added to $\Omega$
6: end for
7: $\mathcal{L}_1 \leftarrow \{1 - itemsets\}$   ▷ Generate all the single itemsets
8: for all items $i \in \mathcal{L}_1$ do
9:     if support($i$, $\Omega$)$\geq \alpha$ then   ▷ Check whether $i$ is frequent
10:        $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup i$
11:    end if
12: end for
13: $j = 2$
14: while $\mathcal{L}_{j-1} \neq \emptyset$ do   ▷ Generate all the $j$-itemsets from the set of $j-1$
15:     $\mathcal{C}_j \leftarrow$ generateCandidates($\mathcal{L}_{j-1}$)
16:     for all patterns $p \in \mathcal{C}_j$ do   ▷ Check common paths
17:         if $\nexists i_x, i_y \in p : path(i_x) \subseteq path(i_y)$ then
18:             if support($p$, $\Omega$)$\geq \alpha$ then   ▷ Check whether $p$ is frequent
19:                 $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup p$
20:             end if
21:         end if
22:     end for
23:     $j \leftarrow j + 1$
24: end while
25: return $\mathcal{L}$
```

constitute a subset of the elements in the path of the other item (see line 17, Algorithm 2); *b*) the candidate itemset $i \in \mathcal{C}_j$ should be frequent —based on the number of transactions in which it appears— according to a minimum threshold value (see line 18, Algorithm 2).

A third proposal is based on a random search methodology, which belongs to the field of stochastic optimization [30] whose strategy is to sample solutions from across the entire search space using a uniform probability distribution. Each future sample is independent of the samples that come before it through *ite* iterations. The proposed algorithm (see Algorithm 3) works therefore on *ite* iterations and each iteration is responsible for generating M random patterns (see Lines 7 to 24, Algorithm 3). When generating a random pattern, the algorithm produces a random value l which maximum value is the number of attributes in Ω. Then, the algorithm randomly chooses items from the dataset Ω till the number l is reached (see Lines 11 to 15, Algorithm 3) and, finally, the new pattern p is evaluated according to its support (see Line 18) previously checking if it is a valid pattern (see Line 17). This new pattern p will be included in the resulting set $\mathcal{P}$ if it is good enough according to its support (only the best n solutions are kept in $\mathcal{P}$. The main advantage of this proposal with regard to the two exhaustive search approaches already described is the reduction of the computational cost since it does not require to explore the whole search space. On the contrary, its major downside is related to its inability to explore the whole search space, especially in high-dimensional data, not guaranteeing that the global optimum is reached.

Algorithm 3 Random Search Algorithm to Cope With Flexibility in the Space of Concepts

```
Require: $n, ite, M, \mathcal{D}, \alpha$
Ensure: $\mathcal{P}$
 1: $\Omega \leftarrow \emptyset$                                   ▷ Encoded dataset
 2: for all path $p \in G$ do      ▷ Analyse each path $p$ within the grammar $G$
 3:     $t \leftarrow$ getTransactions($p$, $\mathcal{D}$)   ▷ Transactions satisfied by $p$ in $\mathcal{D}$
 4:     $\Omega \leftarrow \Omega \cup < p, t >$   ▷ The path $p$ and transactions $t$ are added to $\Omega$
 5: end for
 6: $\mathcal{P} \leftarrow \emptyset$                             ▷ Form the population $\mathcal{P}$
 7: for $i$ from 1 to $it$ do ▷ Iterate $ite$ times seeking solutions
 8:     for $j$ from 1 to $M$ do  ▷ Generate $M$ patterns in each iteration
 9:         Random number $l \in [1, k]$   ▷ $k$ is the number of attributes in $\Omega$
10:         $p \leftarrow \emptyset$
11:         for $j$ from 1 to $l$ do  ▷ Iterate $l$ times to generate a pattern $p$
12:             Select a random attribute $a_k \in \Omega$
13:             $d_j$ $\leftarrow$ Select a random (uniform) discrete value for $a_k$
14:             $p \leftarrow p \cup d_j$
15:         end for                            ▷ Check common paths
16:         if $\nexists i_x, i_y \in p : path(i_x) \subseteq path(i_y)$ then
17:             if support($p$, $\Omega$)$\geq \alpha$ then ▷ Check whether $p$ is frequent
18:                 $\mathcal{P} \leftarrow$ take best $n$ solutions from $\mathcal{P} \cup p$
19:             end if
20:         end if
21:     end for
22: end for
23: return $\mathcal{P}$
```

A fourth proposal (see Algorithm 4) is based on an evolutionary computation methodology, which unlike random search methods, guides the search process through promising areas of the search space. The proposal follows a well-known generational schema where, in each generation of the evolutionary process, solutions are crossed and mutated, and new offspring are obtained. The algorithm starts by encoding patterns through a process similar to the already described random search approach, that is, it produces a random number l between 1 and k (number of attributes in data) to determine the length of the solution (number of items in the pattern) and attributes/items are randomly chosen from the dataset Ω till the number l is reached (see Lines 4 to 10, Algorithm 4). Finally, a generational schema is carried out through G generations (see Lines 19 to 32, Algorithm 4), producing new solutions thanks to crossover and mutation and finally returning the set $\mathcal{E}$ comprising those best solutions found along the evolutionary process. A fitness function is proposed to define how good or promising a solution p is, in such a way that it is defined as the support of p in Ω (see Lines 12 and 24, Algorithm 4). In order to obtain new solutions along the evolutionary process, two simple genetic

Algorithm 4 Evolutionary Algorithm to Cope With Flexibility in the Space of Concepts

```
Require: $n, G, M, \Omega, \alpha$
Ensure: $\mathcal{E}$
 1: $\mathcal{P} \leftarrow \emptyset$                              ▷ General population
 2: $\mathcal{E} \leftarrow \emptyset$  ▷ Elite population with the best solutions found so far
 3: for $j$ from 1 to $M$ do     ▷ Generate $M$ random solutions (patterns)
 4:     Random number $l \in [1, k]$          ▷ $k$ is number of attributes in $\Omega$
 5:     $p \leftarrow \emptyset$
 6:     for $j$ from 1 to $l$ do     ▷ Iterate $l$ times to generate a pattern $p$
 7:         Select a random attribute $a_k \in \Omega$
 8:         $d_j$ $\leftarrow$ Select a random (uniform) discrete value for $a_k$
 9:         $p \leftarrow p \cup d_j$
10:     end for                          ▷ Check common paths
11:     if $\nexists i_x, i_y \in p : path(i_x) \subseteq path(i_y)$ then
12:         support($p$, $\Omega$)   ▷ Calculate the fitness of $p$ as its support in $\Omega$
13:         if support($p$, $\Omega$)$\geq \alpha$ then   ▷ Check whether $p$ is frequent
14:             $\mathcal{E} \leftarrow$ take best $n$ solutions from $\mathcal{E} \cup p$
15:         end if
16:     end if
17:     $\mathcal{P} \leftarrow \mathcal{P} \cup p$
18: end for
19: for $g$ from 1 to $G$ do              ▷ Iterate $G$ times seeking solutions
20:     $parents$ $\leftarrow$ apply parent selector on $\mathcal{P}$ ▷ Tournament size is 2
21:     $offspring$ $\leftarrow$ apply crossover and mutation on $parents$
22:     for all patterns $p \in offspring$ do   ▷ Check common paths
23:         if $\nexists i_x, i_y \in p : path(i_x) \subseteq path(i_y)$ then
24:             support($p$, $\Omega$)  ▷ Calculate the fitness of $p$ as its support in $\Omega$
25:             if support($p$, $\Omega$)$\geq \alpha$ then   ▷ Check whether $p$ is frequent
26:                 $\mathcal{E} \leftarrow$ take best $n$ solutions from $\mathcal{E} \cup p$
27:             end if
28:         end if
29:     end for
30:     $\mathcal{P}$ $\leftarrow$ update the general population considering the set $offspring$
31: end for
32: return $\mathcal{E}$
```

operators have been considered. The crossover genetic operator just swaps two random items from two patterns. The mutator genetic operator, on the contrary, takes a random item from a parent and modifies its value. Similarly to the random search proposal, the evolutionary proposal provides a major advantage with regard to the exhaustive search approach, it reduces the computational cost since it does not require to explore the whole search space. On the contrary, its major disadvantage is its inability to explore the whole search space, especially in high-dimensional data. An advantage of this proposal with regard to the random search proposal is that it is able to guide the search process towards promising areas, whereas the random search approach produces completely new solutions each time (each future sample is independent of the samples that come before it).

Finally, let us analyse the size of the search space, which highly depends on the taxonomy provided by the user. In the best case, the search space is equal to that of existing methodologies for pattern mining like, for example, Apriori [29]. In these methodologies, the number of items is equal to the primitive concepts so the search space includes $2^k - 1$ solutions when k primitive concepts and no taxonomy are considered. In the proposed approaches, this number of solutions varies with the taxonomy since both abstract elements and relationships among elements are considered. For the sample market basket dataset (see Table 1), the search space for a traditional pattern mining algorithm (no taxonomy is considered) is $2^7 - 1$ since 7 different items are considered (BumGenius, Huggies, Pampers, Luvs, SamuelAdams, BudWeiser and Heineken). On the contrary, considering the taxonomy shown in Figure 1, 11 items are considered and a total of $2^{11} - 1$ solutions can be found if no restriction is included. This number of solutions, however, should be reduced depending on the taxonomy restrictions so, for example, {Beer} cannot appear together with any of the following items: {Regular}, {Light}, {Luvs}, {SamuelAdams}, {Budweiser} and {Heineken}.

B. SPACE OF DATA RECORDS

Traditional pattern mining works on datasets where each transaction or data record is unequivocally described by a row or list of features [6], [9]. For example, in a market basket analysis, each transaction includes information for a specific purchase. Sometimes, however, the final aim of the analysis may require the whole set of transactions to be grouped by customers so a specific customer may include a high number of purchases (data records). At this point, the extracted knowledge might be completely different when data records are considered as a whole to the insights obtained when data are grouped by a specific concept (customers in this example). With the aim of avoiding inaccurate insights to be extracted (information that is biased against occasional customers), some previous attempts [23] have considered the organization of data records into bags of records. This, however, did not consider the possibility of analysing the dataset into different views (different organizations of the records according to the users' requirements). Thus, the existing models do not support all forms of knowledge and it is therefore crucial to provide new mechanisms that produce adaptable insights that meet the increasingly complex expectations of the users. The aim of this sub-section is to describe different approximations to demonstrate that the ideas proposed in this section are feasible. The algorithmic solutions provide new mechanisms for dealing with datasets organized into groups of records considering either a single or multiple levels of assemblage.

The first proposal is an adaptation of the already proposed Apriori-MI [23] considering new features (records organized into multiple-levels, different ways in which a bag is denoted as satisfied, etc). In this approach (see Algorithm 5) the evaluation process (see lines 4 and 12, Algorithm 5) is the most important part, each of the records within a bag of records are analysed so the bag B^j will be satisfied for a pattern P if and only if $|\forall t_i^j : t_i^j \in B^j \land P \subseteq t_i^j| \geq \alpha$. As described in previous sections, this α value can be a fixed value or a value depending on the number of records of B^j (percentage of records from B^j satisfied by P). In situations where bags are also organized into sub-bags, then it is required to check whether a sub-bag $S^k \in B^j$ is satisfied or not (according to a new α_1 value for bags and α_2 for sub-bags).

Algorithm 5 Exhaustive Search Approach to Cope With Flexibility in the Space of Data Records

```
Require: $\Omega, \alpha$                  ▷ Dataset $\Omega$ organized into bags
Ensure: $\mathcal{L}$                      ▷ Set $\mathcal{L}$ of discovered patterns
 1: $\mathcal{L} \leftarrow \emptyset$
 2: $\mathcal{L}_1 \leftarrow \{1 - itemsets\}$      ▷ Generate all the single itemsets
 3: for all elements $i \in \mathcal{L}_1$ do      ▷ Check whether $i$ is a frequent item
 4:     if support($i$, $\Omega$)$\geq \alpha$ by considering the bags and sub-bags then
 5:         $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup i$
 6:     end if
 7: end for
 8: $j = 2$
 9: while $\mathcal{L}_{j-1} \neq \emptyset$ do    ▷ Generate all the $j$-itemsets from the set of $j-1$
10:     $\mathcal{C}_j \leftarrow$ generateCandidates($\mathcal{L}_{j-1}$)
11:     for all elements $p \in \mathcal{C}_j$ do    ▷ Check whether $p$ is a frequent pattern
12:         if support($p$, $\Omega$)$\geq \alpha$ by considering the bags and sub-bags then
13:             $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup p$
14:         end if
15:     end for
16:     $j \leftarrow j + 1$
17: end while
18: return $\mathcal{L}$
```

Algorithm 6 Random Search Algorithm to Cope With Flexibility in the Space of Data Records

```
Require: $n, ite, M, \Omega, \alpha$    ▷ Dataset $\Omega$ organized into bags
Ensure: $\mathcal{P}$
 1: $\mathcal{P} \leftarrow \emptyset$    ▷ Form the population $\mathcal{P}$
 2: for $i$ from 1 to $it$ do    ▷ Iterate $ite$ times seeking solutions
 3:     for $j$ from 1 to $M$ do    ▷ Generate $M$ patterns in each iteration
 4:         Random number $l \in [1, k]$    ▷ $k$ number of attributes in $\Omega$
 5:         $p \leftarrow \emptyset$
 6:         for $j$ from 1 to $l$ do    ▷ Iterate $l$ times to generate a pattern $p$
 7:             Select a random attribute $a_k \in \Omega$
 8:             $d_j$ $\leftarrow$ Select a random (uniform) discrete value for $a_k$
 9:             $p \leftarrow p \cup d_j$
10:         end for    ▷ Check whether $p$ is a frequent pattern
11:         if support($p$, $\Omega$)$\geq \alpha$ by considering the bags and sub-bags then
12:             $\mathcal{P} \leftarrow$ take best $n$ solutions from $\mathcal{P} \cup p$
13:         end if
14:     end for
15: end for
16: return $\mathcal{P}$
```

A second proposal (random search, see Algorithm 6) is based on stochastic optimization [30] whose strategy is to randomly obtain new solutions from across the entire search space and each solution is independent of those already obtained in previous iterations. The proposed approach works on *ite* iterations and each iteration is responsible for generating M random solutions or patterns (see Lines 3 to 14, Algorithm 6). To produce a new random solution (see Lines 4 to 10, Algorithm 6), the algorithm randomly chooses a value l in the interval $[1, k]$, which is the maximum number of attributes included in Ω, and produces l random items from Ω. The resulting pattern is finally evaluated according to its support (see Line 11). The main advantage of this proposal with regard to the exhaustive search approach is the reduction of the computational cost since it does not require to explore the whole search space. On the contrary, its major downside is related to its inability to explore the whole search space, especially in high-dimensional data, not guaranteeing that the global optimum is reached.

A final proposal (see Algorithm 7) is based on an evolutionary computation methodology, guiding the search process through promising areas of the search space thanks to a fitness function based on the support of a solution p in Ω. Unlike the random search proposal previously described, this evolutionary approach is able to guide the search process towards promising areas, that is, each future sample highly

Algorithm 7 Evolutionary Algorithm to Cope With Flexibility in the Space of Data Records

```
Require: $n, G, M, \Omega, \alpha$    ▷ Dataset $\Omega$ organized into bags
Ensure: $\mathcal{E}$
 1: $\mathcal{P} \leftarrow \emptyset$    ▷ General population
 2: $\mathcal{E} \leftarrow \emptyset$    ▷ Elite population with the best solutions found so far
 3: for $j$ from 1 to $M$ do    ▷ Generate $M$ random solutions (patterns)
 4:     Random number $l \in [1, k]$    ▷ $k$ number of attributes in $\Omega$
 5:     $p \leftarrow \emptyset$
 6:     for $j$ from 1 to $l$ do    ▷ Iterate $l$ times to generate a pattern $p$
 7:         Select a random attribute $a_k \in \Omega$
 8:         $d_j$ $\leftarrow$ Select a random (uniform) discrete value for $a_k$
 9:         $p \leftarrow p \cup d_j$
10:     end for
11:     if support($p$, $\Omega$)$\geq \alpha$ then    ▷ Check whether $p$ is a frequent pattern
12:         $\mathcal{E} \leftarrow$ take best $n$ solutions from $\mathcal{E} \cup p$
13:     end if
14:     $\mathcal{P} \leftarrow \mathcal{P} \cup p$
15: end for
16: for $g$ from 1 to $G$ do    ▷ Iterate $G$ times seeking solutions
17:     $parents$ $\leftarrow$ apply parent selector on $\mathcal{P}$    ▷ The tournament size is 2
18:     $offspring$ $\leftarrow$ apply crossover and mutation on $parents$
19:     for all patterns $p \in offspring$ do
20:         if support($p$, $\Omega$)$\geq \alpha$ then    ▷ Check whether $p$ is frequent
21:             $\mathcal{E} \leftarrow$ take best $n$ solutions from $\mathcal{E} \cup p$
22:         end if
23:     end for
24:     $\mathcal{P} \leftarrow$ update the general population considering the set $offspring$
25: end for
26: return $\mathcal{E}$
```

depends on the samples that come before it. This proposal is based on a well-known generational schema where existing solutions are crossed and mutated to produce new solutions in each generation of the evolutionary process. At the beginning of the algorithm new solutions are produced from scratch, that is, following a similar process of the already described random search approach —it produces a random number l between 1 and k (number of attributes in data) to determine the length of the solution (number of items in the pattern) and attributes/items are randomly chosen from the dataset Ω till the number l is reached (see Lines 3 to 15, Algorithm 7). Finally, a generational schema is carried out through G generations (see Lines 16 to 25, Algorithm 7), producing new solutions thanks to crossover and mutation and finally returning the set $\mathcal{E}$ comprising those best solutions found along the evolutionary process. Here, whereas the crossover genetic operator just swaps two random items from two patterns, the mutator genetic operator takes a random item from a parent and modifies its value. The proposed evolutionary proposal reduces the computational cost since it does not require to explore the whole search space, which is a major advantage with regard to the exhaustive search approach. Nevertheless, its major disadvantage is its inability to explore the whole search space, especially in high-dimensional data.

Finally, it is important to highlight that the search space when no restriction is considered is the same as any traditional pattern mining algorithm on rigid data representation [29], that is, $2^k - 1$ for a dataset comprising k elements. However, even when the search space remains the same, the computational complexity of the proposed algorithmic solutions could be much higher/lower when records are organized into objective concepts (concepts included in data). Here, each data record may be analysed for more than a single bag and, therefore, the time required to analyse the whole dataset highly depends on the data organization as well as the users' requirements.

V. EXPERIMENTAL ANALYSIS

The aim of this analysis is to compare the proposed approaches in terms of runtime and solution quality (heuristic-based approaches vs exhaustive search procedures). Here, it is also interesting to analyse how the extracted knowledge (set of solutions) varies according to the provided grammar and/or the way in which data records are organized. The experimental analysis is therefore divided into two main studies: space of concepts and space of data records. Finally, some solutions are analysed with/without a hierarchy to demonstrate the power of the presented foundations.

The experimental study was performed using some synthetic datasets comprising information about up to 23 items that anyone can buy in a supermarket and including the purchasing habits of between 100 and 10,000 different customers. Values for these items were uniformly distributed. Additionally, random concept hierarchies were considered so comparisons are as fair as possible and not depending on the hierarchy. These random hierarchies produce different search spaces when they are applied to the same dataset and, therefore, different results as it will be proved later on. It is important to highlight that results obtained by this analysis is useless in terms of insights and it was only performed to check how the algorithms perform and how different the results are when different taxonomies are considered.

The following parameters were considered through the experimental analysis. These parameters were fixed after a previous analysis aiming to be as fair as possible so all the proposed approaches work on similar circumstances, e.g. a similar number of evaluated solutions. Authors are aware that these parameters cannot be the optimal ones for any dataset/problem and should be therefore adjusted for further applications. Exhaustive search approaches work on a support threshold value of 0.3 (the same threshold is used either on the space of concepts and data records). As for the random search approach, a total of 5000 random solutions are considered. Regarding the evolutionary algorithm, it runs on 100 generations, considering a population size of 50 and a tournament size of 3. The genetic operators' probabilities were set up to 0.85 for crossover and 0.4 for mutation. Finally, in any of the analysed approaches, the best 20 discovered patterns are returned. Nevertheless, some extra studies are also considered for a different number of returned patterns as it will be explained. The experiments were run on an Intel(R) Core(TM) i7 CPU at 2.67GHz with 12GB main memory, running CentOS 5.4.

A. SPACE OF CONCEPTS

In this analysis, it is interesting to analyse first how the runtime varies according to the size of the search space (the number of instances remains the same). Taking a synthetic dataset (23 items uniformly distributed and 100 instances/customers) different concept hierarchies are applied so different search spaces are produced on the same dataset. The four proposed approaches are analysed: AprioriPost (analysis after the mining process), AprioriIn (analysis during the mining process), Random Search and Evolutionary Algorithm. Figure 3 illustrates the variation in runtime for each algorithm on a logarithmic scale, demonstrating that exhaustive search approaches are appropriate for small search spaces. Both exhaustive search proposals present a runtime that is exponential with regard to the search space, AprioriPost behaving worse than AprioriIn —around four orders of magnitude lower for a search space of 10^4 solutions. When large search spaces are analysed, then heuristic-based proposals (random search and evolutionary algorithm) perform much better. In fact, these two approaches are run on an almost constant runtime. Here, it is interesting to highlight that the random

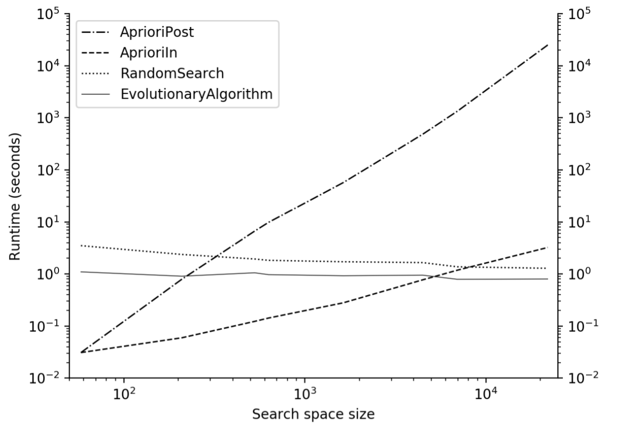

FIGURE 3. Variation of the runtime when the search space increases (the number of instances is the same).

search approach slightly reduces its runtime when the search space increases. This behaviour is caused by the huge number of invalid solutions (those that do not satisfy the hierarchy or those that are not present in data) which gives rise to a lower probability of having to try adding a solution into the elite population (set of best solutions found so far). This behaviour, on the contrary, does not appear in the evolutionary algorithm since it guides the search process through promising areas avoiding solutions that are not present in data.

In this study, it is also interesting to analyse whether the number of instances has a huge impact on the runtime. To this aim, the same concept hierarchy (search space including 21,963 solutions) with a variation in the number of instances is considered (this number varies from 100 to 10,000). The results of this analysis are shown in Figure 4, where it is demonstrated that the number of instances to be evaluated is not a key issue in the runtime. Again, both the evolutionary approach and the random search proposal achieve the best results in runtime, whereas AprioriPost highly differs (around four orders of magnitude higher).

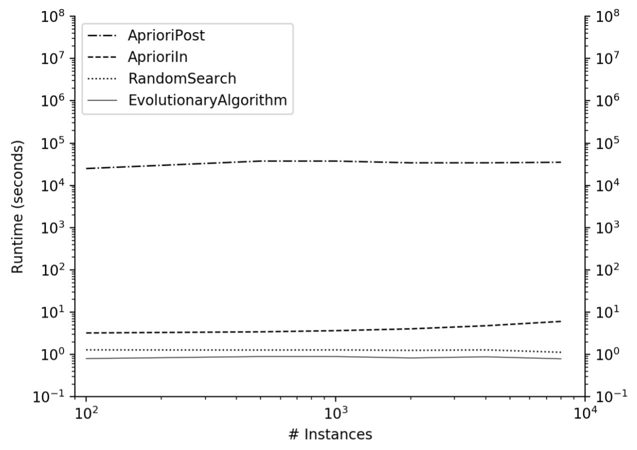

FIGURE 4. Variation of the runtime when the number of instances increases (the search space is the same).

Once the runtime has been analysed, it is of high interest to demonstrate how good heuristic-based solutions are. In this sense, and based on the fact that exhaustive search approaches obtain the whole set of solutions, the aim is to analyse whether the resulting set of solutions provided by heuristic-based proposals is close to the real optimum (the set of top solutions provided by exhaustive search approaches). Table 5 shows the percentage, in per unit basis, of the proximity between the average support value of the top solutions found by both exhaustive search and heuristic-based approaches. Values close to 1 represent a similar average, whereas values close to 0 indicate completely dissimilar average support values. As shown in that table, the higher the number of top solutions to be considered, the more difficult it is to reach the optimum. Nevertheless, in general terms, results demonstrate that the evolutionary algorithm (EA) widely outperforms the random search (RS) approach. In fact, and taking into account the previous analyses (see Figure 3 and Figure 4) it can be asserted that the evolutionary algorithm is a really good option for large search spaces since it obtains really good solutions (close to the optimum) in an acceptable runtime. On the other hand, exhaustive search approaches, and more specifically AprioriIn, are useful in small search spaces.

TABLE 5. Comparative of the average support values when they are compared to the optimum (provided by exhaustive search approches).

	Top 10		Top 20		Top 30	
SearchSpace	RS	EA	RS	EA	RS	EA
58	0.8965	0.9945	0.7890	0.9525	0.8088	0.9418
209	0.8444	1.0000	0.7824	0.9488	0.7060	0.9863
528	0.9308	1.0000	0.7994	1.0000	0.7056	1.0000
630	0.8419	1.0000	0.7359	1.0000	0.7115	0.9565
1625	0.8626	1.0000	0.7190	1.0000	0.6121	0.9961
4475	0.8304	1.0000	0.7873	0.9968	0.5991	0.9918
6949	0.8869	1.0000	0.6587	1.0000	0.6086	0.9905
21963	0.8960	1.0000	0.7289	0.9979	0.6203	0.9935

B. SPACE OF DATA RECORDS

In this second analysis, and similarly to the previous analysis carried out on the space of data records, it is interesting to analyse first how the runtime varies according to the size of the search space. This study is carried out with the three proposed approaches: AprioriIn (exhaustive search approach), Random Search and Evolutionary Algorithm. Figure 5 illustrates the variation in runtime for each algorithm, demonstrating that the exhaustive search approach is appropriate for small search spaces, presenting a runtime that is exponential with regard to the search space. On the contrary, heuristic-based proposals (random search and evolutionary algorithm) perform much better when large search spaces are considered. In fact, the evolutionary algorithm is run on an almost constant runtime. Here, and similarly to the results obtained in the analysis on the space of concepts, it is also interesting to highlight that the random search approach slightly reduces its runtime when the search space increases and this reduction is even higher than the one obtained in the

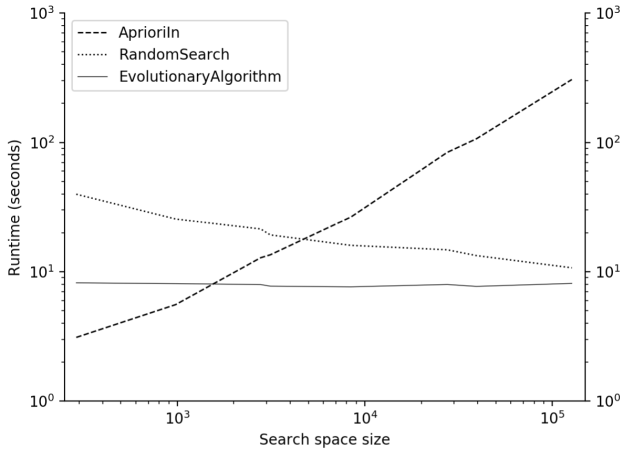

FIGURE 5. Variation of the runtime when the search space increases.

space of concepts. This behaviour is caused by the huge number of solutions having a support value lower than the predefined minimum threshold, which implies a lower probability of having to try adding a solution into the elite population (set of best solutions found so far). This threshold value is much more restrictive when records are organized into bags and this is why the behaviour slightly differs from the one obtained when analysing the space of concepts. This behaviour is not observed for the evolutionary algorithm since it guides the search process through promising areas avoiding solutions that are not present in data.

The following study is related to whether the number of bags (sets of instances) has a huge impact on the runtime. To this aim, the same search space is considered but the number of bags is varied, providing the results illustrated in Figure 6. Taking the heuristic-based proposals (random search and evolutionary algorithm), it is observed that the number of bags does not have a huge impact on these algorithms' runtime. The exhaustive search approach, on the contrary, presents a huge increment in the runtime when the number of bags also increases. An additional analysis must also be performed, considering a fixed search space and a fixed number of bags while varying data density. This density is quantified according to the average number of instances per bag in data. Figure 7 illustrates how the three proposed algorithms behave when density is varied. In general terms, the density does not influence much performance, all the algorithms running in an almost similar time.

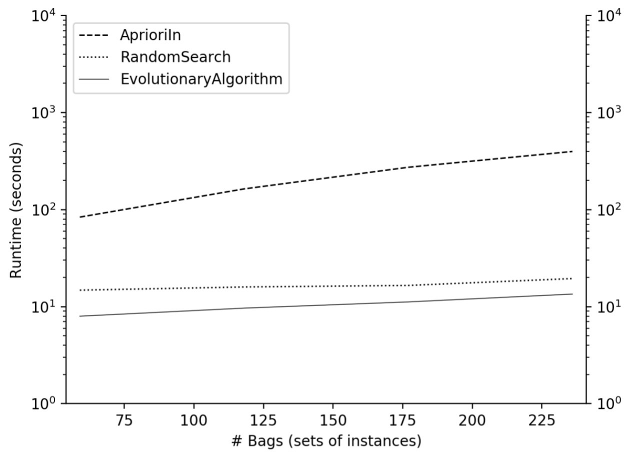

FIGURE 6. **Variation of the runtime when the number of bags (groups of instances) increases (the search space is the same).**

Finally, and once the runtime has been analysed, it is of high interest to evaluate how good heuristic-based solutions are when considering flexibility in the space of data records. In this sense, and based on the fact that AprioriIn obtains the whole set of solutions, the aim is to analyse whether the resulting set of solutions provided by heuristic-based proposals is close to the real optimum (the set of top solutions provided by the exhaustive search approach). Table 6 shows the percentage, in per unit basis, of the proximity between the average support value of the top solutions found by both

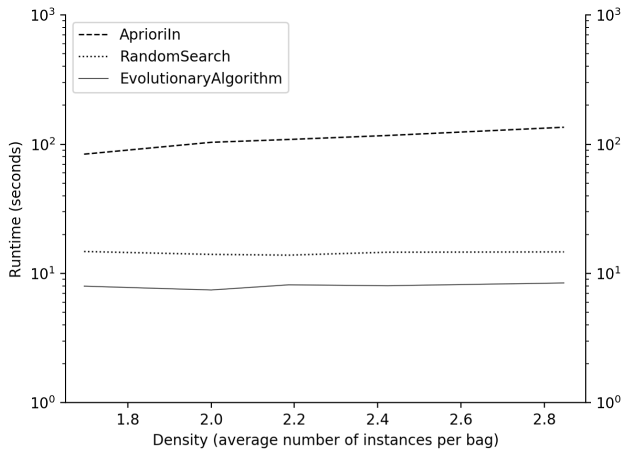

FIGURE 7. **Variation of the runtime when the density (average number of instances per bag) increases (the search space is the same).**

TABLE 6. **Comparative of the average support values when they are compared to the optimum (provided by exhaustive search approches).**

	Top 10		Top 20		Top 30	
SearchSpace	RS	EA	RS	EA	RS	EA
289	0.8885	0.9831	0.7790	0.9211	0.7961	0.8905
979	0.8354	0.9801	0.7911	0.9309	0.7214	0.9201
2784	0.9091	1.0000	0.8224	1.0000	0.7001	0.9667
3135	0.8392	0.9905	0.7298	0.9877	0.6987	0.9261
8427	0.8442	0.9935	0.8221	0.9801	0.7552	0.9709
27549	0.8100	0.9863	0.7998	0.9668	0.6954	0.9526
39400	0.8409	0.9961	0.7594	0.9888	0.6201	0.9518
127149	0.8789	1.0000	0.8559	1.0000	0.6003	0.9511

exhaustive search and heuristic-based approaches. Values close to 1 represent a similar average, whereas values close to 0 indicate completely dissimilar average support values. In general terms, results reveal that as the number of top solutions is increased, it is more difficult to reach the optimum. Additionally, the evolutionary algorithm (EA) widely outperforms the random search (RS) approach, achieving results close to the maximum in any case. Taking into account the previous analyses (see Figure 5, Figure 6 and Figure 7) it can be asserted that the evolutionary algorithm is a really good option for large search spaces since it obtains really good solutions (close to the optimum) in an acceptable runtime. For really small search spaces, on the contrary, an exhaustive search is much more appropriate.

C. ANALYSIS OF THE INSIGHTS

This study aims to analyse the solutions found by the proposed approaches to assess their usefulness and their ability to extract user-centric knowledge. With this aim, let us consider first a synthetic dataset comprising information about 23 items that anyone can buy in a supermarket. This dataset includes the purchasing habits of 100 different customers. Each customer has bought, on average, 11.43 items (standard deviation of 2.26). Applying a traditional pattern mining approach, where no flexibility is added to the input dataset, it is possible to obtain solutions such as those illustrated in Table 7. Here the product that is most frequently bought is item14, almost 60% of the customers have bought this item. If we analyse patterns of length two (patterns with #Id

TABLE 7. Set of solutions obtained on a sample market basket dataset (rigid data structured). Solutions are ranked (top to bottom, left to right) according to their support in per unit basis.

#Id	Patterns	Support	#Id	Patterns	Support	#Id	Patterns	Support
1	item14	0.59	21	item18	0.45	41	item13, item14	0.31
2	item1	0.58	22	item16	0.44	42	item6, item19	0.31
3	item7	0.58	23	item12	0.40	43	item0, item1	0.30
4	item4	0.55	24	item1, item7	0.37	44	item14, item20	0.30
5	item3	0.54	25	item7, item14	0.36	45	item0, item3	0.30
6	item19	0.53	26	item14, item19	0.35	46	item14, item18	0.30
7	item17	0.52	27	item14, item22	0.34	47	item14, item17	0.30
8	item21	0.52	28	item1, item14	0.34	48	item14, item15	0.30
9	item2	0.50	29	item1, item4	0.34	49	item1, item9	0.30
10	item22	0.50	30	item7, item11	0.33	50	item11, item14	0.30
11	item13	0.49	31	item1, item22	0.33	51	item6, item14	0.30
12	item15	0.49	32	item1, item3	0.33	52	item4, item22	0.30
13	item6	0.49	33	item10, item14	0.33	53	item4, item17	0.30
14	item10	0.48	34	item3, item4	0.32	54	item3, item6	0.30
15	item20	0.48	35	item4, item7	0.32	55	item2, item21	0.30
16	item0	0.47	36	item1, item19	0.32	56	item2, item14	0.30
17	item11	0.46	37	item18, item21	0.32	57	item1, item15	0.30
18	item8	0.46	38	item7, item19	0.32	58	item19, item21	0.30
19	item5	0.46	39	item7, item22	0.32			
20	item9	0.45	40	item4, item14	0.32			

G = (Σ_N, Σ_T, P, S) with:
S = <Product>
Σ_N = {<Product>, <dairy>, <cheeses>, <eggs>, <milk>, <yogurt>, <butter>, <personalCare>, <personalCarePacks>, <shampoo>, <soap>, <handsoap>, <beverages>, <beer>, <soda>, <sodaPack>}
Σ_T = {item0, item1, item2, item3, item4, item5, item6, imte7, ..., item18, item19, item20, item21, item22}
P = {<Product> ⇐ <diary> | <PersonalCare> | <beverages> ;
<dairy> ⇐ <cheeses> | <eggs> | <milk> | <yogurt> | <butter> ;
<cheeses> ⇐ item13 | item14 | item15 ;
<eggs> ⇐ item16 | item17 ;
<milk> ⇐ item18 | item19 ;
<yogurt> ⇐ item20 | item21 ;
<butter> ⇐ item22 ;
<personalCare> ⇐ <shampoo> | <soap> | <handsoap> | <PersonalCarePacks> ;
<personalCarePacks> ⇐ item8, item10 | item8, <handsoap> | item9, item10 | item9, <handsoap> ;
<shampoo> ⇐ item8 | item9 ;
<soap> ⇐ item10 | <handsoap> ;
<handsoap> ⇐ item11 | item12 ;
<beverages> ⇐ <beer> | <soda> ;
<beer> ⇐ item0 | item1 | item2 | item3 | item4 ;
<soda> ⇐ item5 | item6 | item7 | <sodaPack> ;
<sodaPack> ⇐ item5, item6 | item5, item7 | item6, item7 | item5, item6, item7 ;
}

FIGURE 8. Context-free grammar defined to represent subjective information on a dataset related to customers' purchasing habits. The grammar is expressed in extended BNF notation.

from 24 to 58), the one with the highest support is #Id 24, which represents that 37% of the customers bought item1 and item7 together.

Continuing with this dataset, let us consider adding some conceptual information to extract user-centric knowledge. For that, a hierarchy of concepts is provided (see Figure 8) in which items are grouped to produce abstract concepts. Here, it is indicated that items 0 to 4 are different types of beer, whereas items 5 to 7 are types of soda. Additionally, there exist a promotion of packs of soda, including two or more items that belong to soda (any type of soda). This and further information is properly represented/defined through a context-free grammar (see Figure 8). Applying the proposed approaches on the dataset together with the defined hierarchy, a set of solutions is obtained (see Table 8) which is, obviously, larger than the one obtained without hierarchy. A total of 2806 solutions are found, including all the previous solutions (Table 7) and much more.

Analysing the set of solutions, it is found that anyone that goes to the supermarket buys a dairy product (cheese, egg, milk, yoghurt, butter). Similarly, 100% of customers usually buy beverages (beer or soda). These insights are quite interesting and actionable since they are dealing with multiple products (from item13 to item22) as a whole (dairy products) and, therefore, this information can be used for launching the right advertising campaign. Of course, this knowledge is impossible to extract from traditional and rigid data representations used by existing pattern mining approaches. Using a hierarchy of concepts does not imply that items (in the lowest abstraction level) cannot be extracted as traditional pattern mining approaches do. For example, the pattern with #Id 124 denotes that 59% of the customers usually buy item14, which is the same pattern obtained with no grammar (see Table 7). These primitive items can also be combined with abstract concepts as pattern #Id 566, which represents that 44% of the customers generally buy item3 (a specific

TABLE 8. Set of solutions obtained on a sample market basket dataset and considering the hierarchy of concepts (flexible data structured) defined in Figure 8. Solutions are ranked (top to bottom, left to right) according to their support in per unit basis.

#Id	Patterns	Support	#Id	Patterns	Support
1	dairy	1.00	122	personalCarePacks	0.59
2	beverages	1.00	123	shampoo, personalCarePacks, soap	0.59
3	dairy, beverages	1.00	124	item14	0.59
4	beer	0.98	...	...	...
5	dairy, beer	0.98	566	personalCare, soda, item3	0.44
6	personalCare, dairy	0.94	567	beverages, item16	0.44
7	personalCare, beverages	0.94	568	beer, item16	0.44
8	personalCare	0.94	569	personalCare, item20	0.44
9	personalCare, dairy, beer	0.92	570	cheeses, butter, beer, personalCare	0.44
10	personalCare, beer	0.92	571	personalCare, dairy, soda, item3	0.44
11	soda	0.90	572	yogurt, eggs, cheeses, personalCare, beverages	0.44
12	dairy, soda	0.90	573	personalCare, cheeses, item1, soda	0.44
13	cheeses	0.89	574	cheeses, soda, item3	0.44
14	beverages, cheeses	0.89	575	milk, cheeses, item1	0.44
15	beer, soda	0.88	576	personalCare, cheeses, item3	0.44
...	...	...	...	...	...
118	yogurt, beer, personalCare, soda	0.59	2803	cheeses, soap, beer, item7, shampoo	0.30
119	dairy, shampoo, soap	0.59	2804	eggs, sodaPack, personalCare, beer, item7	0.30
120	eggs, personalCare, soda	0.59	2805	sodaPack', 'cheeses', 'soap', 'beer', 'item7	0.30
121	beverages, personalCarePacks	0.59	2806	personalCare, dairy, item1, item3	0.30

type of beer), soda (no matter which one) and a personal care product (no matter which one). Finally, it is of high interest to demonstrate that the hierarchy is perfectly satisfied and invalid patterns are not obtained. This is easily demonstrable by analysing the five first patterns (#Id 1 to #Id 5). Here, beer appears like a really frequent product (98% of the customers usually buy it), and it is combined with dairy (100% of the customers usually buy a dairy product) in pattern #Id 5, but it is not combined with beverages (100% of the customers usually buy a beverage). Beer and beverages are incompatible items since beer is, by definition, a type of beverage so the resulting insight is useless.

Finally, and continuing with the same dataset, let us consider now a completely different conceptual information (see Figure 9). Here, it is represented that items from 0 to 7 are different types of beverages, not being important whether it is a beer or a soda. Items from 8 to 12 represent personal care product, and it does not matter the product type. Finally, items from 13 to 22 represent dairy products in general. Applying the previous dataset together with the defined hierarchy on the already proposed approaches, a set of solutions is obtained (see Table 9) which is, obviously, larger than the one obtained without hierarchy and smaller than the previous one (the hierarchy is now much more simple or abstract). As shown, results for this hierarchy were already obtained in the previous hierarchy. The main difference is that results are simplified because there are less abstract concepts.

To sum up, the use of a hierarchy of concepts allows encoding much richer information in the data. Results with and without a hierarchy are completely different except for the primitive concepts, which are obtained in any case. If the hierarchy is slightly modified, the results also change, being more general or specific depending on the hierarchy. As it has been demonstrated, the use of a hierarchy is essential to obtain useful and user-centric insights.

Finally, and considering the same hierarchy defined in Figure 9, it is possible to organize data records into groups or bags of instances. This analysis is of high interest in situations where all the purchases of the same customer must be analysed at the same time, avoiding inaccurate insights to be extracted (information that is biased against occasional customers). Results of this analysis are shown in Table 10, which highly differ from those obtained in Table 9 even when the hierarchy is the same. As shown, the support values are different and some of the variations are quite interesting. For example, item14 in isolation has a support value of 0.71 but when it appears together with dairy (a support of 100%) produces a not too high support value (see pattern #Id 543, support value of 0.37). It is caused by the fact that a pattern that appears in 100% of the bags does not necessarily appear together with others. It may happen that the two items appear

TABLE 9. Set of solutions obtained on a sample market basket dataset and considering the hierarchy of concepts (flexible data structured) defined in Figure 9. Solutions are ranked (top to bottom, left to right) according to their support in per unit basis.

#Id	Patterns	Support
1	dairy	1.00
2	beverages	1.00
3	dairy, beverages	1.00
4	beverages, personalCare	0.94
5	beverages, personalCare, dairy	0.94
6	personalCare	0.94
7	personalCare, dairy	0.94
8	beverages, item14	0.59
9	item14	0.59
10	item7	0.58
11	item1, diary	0.58
12	item7, diary	0.58
13	item1	058
14	personalCare, beverages, item14	0.57
15	personalCare, item14	0.57
...	...	...
100	item1, item7	0.37
101	item7, item1, dairy	0.37
102	item14, item7	0.36
...	...	...
171	item15, beverages, item14	0.30
172	beverages, item14, item17	0.30
173	item4, item22	0.30

G = $(\Sigma_N, \Sigma_T, P, S)$ with:
S = <Product>
Σ_N = {<Product>, <dairy>, <personalCare>, <beverages>}
Σ_T = {item0, item1, item2, item3, item4, item5, item6, imte7, ..., item18, item19, item20, item21, item22}
P = {<Product> $\Leftarrow$ <diary> | <PersonalCare> | <beverages> ;
<dairy> $\Leftarrow$ item13 | item14 | item15 | item16 | item17 | item18 | item19 | item20 | item21 | item22 ;
<personalCare> $\Leftarrow$ item8 | item9 | item10 | item11 | item12 ;
<beverages> $\Leftarrow$ item0 | item1 | item2 | item3 | item4 | item5 | item6 | item7 ;
}

FIGURE 9. Context-free grammar defined to represent a completely different subjective information on the same dataset. The grammar is expressed in extended BNF notation.

TABLE 10. Set of solutions obtained on a sample market basket dataset organized into bags (space of data records) and considering the hiearchy of concepts (flexible data structured) defined in Figure 9. Solutions are ranked (top to bottom, left to right) according to their support in per unit basis.

#Id	Patterns	Support
1	dairy	1.00
2	beverages	1.00
3	dairy, beverages	1.00
4	beverages, personalCare	0.98
5	beverages, personalCare, dairy	0.98
6	personalCare	0.98
7	personalCare, dairy	0.98
8	item14	0.71
8	beverages, item14	0.71
...	...	...
543	item14, dairy	0.37
...	...	...
762	item1, item4, item22	0.30
763	item1, item7, item14	0.30
764	item14, item20, item22, beverages, personalCare	0.30

in different transactions/records within the same bag. In other words, a product may always be bought by all the customers but it does not mean that this product is bought at the same time (in the same transaction) as others.

VI. LESSON LEARNED

The growing demand for eliciting useful insights from data on different domains has resulted in a growing need for extracting rich information that satisfies the requirements of users. Existing methodologies for describing intrinsic and important properties of data through the extraction of useful patterns, however, work on fixed input data and the discovered insights are therefore restricted by the data structure. These approaches were not designed for discovering adaptable and user-centric knowledge, which can be obtained simply by analysing data from contrasting perspectives or views. Each of these views comprises the same information but expressed differently and, therefore, the obtained results might be useful for some specific aims but useless for others. Based on this idea, it is possible to extract more user-centric knowledge using flexible data structures in two different spaces or perspectives: concepts and records.

The first way in which flexible data structures can be applied is related to the space of concepts, dealing with the importance of defining concepts at different abstraction levels. Concepts (elements or items in data) are the key piece to produce the final insights. Hence, with such concepts on different abstraction levels plays a crucial role in providing powerful, applicable and actionable information that was inaccessible to traditional approaches based only on primitive concepts. Thanks to the use of a context-free grammar, it is possible to achieve heterogeneous hierarchies (not all the items or concepts have an abstract concept in every level of abstraction) and including concepts that produce two or more different generalizations of such concepts (every concept is formally defined by a single abstract concept in the existing methodologies). The first analysis was related to how the runtime varies when different concept hierarchies are considered (different search spaces as a result) by using the same dataset. As it was illustrated in Figure 3, exhaustive search methods exponentially grow with the search space, whereas heuristic-based methods remain almost constant. Focusing on exhaustive search methods, it is faster to apply pruning during the mining process than considering this pruning after the mining process. Additionally, when considering the same hierarchy but increasing the number of instances (see Figure 4), heuristic-based solutions remain almost the same whereas exhaustive search methods linearly increase (noted that y-axis is in logarithmic scale). Finally, when considering the ability to extract the top n solutions, it is obvious that exhaustive search approaches are impossible to be outperformed since they always produce any feasible result. However, the experimental analysis has shown (see Table 5) that evolutionary computation approaches are really good and much better than a random search approach.

In a second analysis, the necessity of handling ambiguous descriptions on the space of records was considered. As it has been studied, the importance of patterns in data highly depends on the way in which data is organized, in such a way that the same pattern may produce completely different results when it is analysed on a dataset organized by rows (each one represents a transaction) and on a dataset organized by bags of rows (each bag comprises a variable number of transactions). This analysis is of high interest in many situations such as the market basket analysis where all the purchases of the same customer are required to be analysed together to avoid extracting inaccurate insights (information that is biased against occasional customers). The analysis carried out in this space of concepts has demonstrated the utility of grouping records at different levels, considering either objective and subjective concepts, and taking into account the significance of a pattern within each bag of data records. The very same analysis as the one performed in the space of concepts was performed, considering different concept hierarchies for the same dataset (see Figure 6), different bags of instances for the same hierarchy of concepts (see Figure 6) and, finally, whether evolutionary approaches are much better than random search methods to extract the best n solutions found by exhaustive search (see Table 6).

In any of the spaces in which data structures can be modelled to achieve the right insights, the obtained results will be more suitable to the background or subjective knowledge provided by users. In other words, if the concept hierarchy is not well defined, the results cannot produce the right insights. Something similar happens when working on the space of data records since transactions are organized by a concept (or multiple concepts) provided by the expert and, therefore, the extracted knowledge highly depends on such organization. The analysis carried out on two different spaces (concepts and records) has giving rise to interesting ways in which the pattern mining task can be improved (according to the users' expectations), and that will serve as the basis for future research studies on this field.

VII. CONCLUSION

This paper has settled the basis for the extraction of more user-centric patterns, easily applicable (improving the understanding), and focused on the task at hand (increasing the actionability). The idea is that the same data can be analyzed from contrasting perspectives or views that will produce completely different results (comprising the same information but expressed differently), and these results may be useful for a specific aim but useless for another. It is similar because two people may describe the same thing from two different angles, and both be right. In this regard, flexible data structures have been described based on two different spaces or perspectives: concepts and records. In each of these spaces, examples to understand the importance of the proposed foundations have been included. It is important to note that the final aim is to set the bases for further research studies on the ideas here presented, so the algorithmic solutions included in each of the two perspectives are just adaptations of well-known and widely used algorithms to demonstrate that all the proposed ideas are feasible and, therefore, further research studies could provide high-performance approaches in this field.

For future work, several research directions can be explored. A possibility is to consider Web Ontology Language (OWL) to represent rich and complex concepts and groups of concepts and relations. This computational logic-based language could provide advanced operators and semantic definitions to improve the understanding and increase the actionability of the extracted insights. This is a natural extension of this work that brings new technological and performance challenges to be faced.

REFERENCES

[1] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Reading, MA, USA: Addison-Wesley, 2005.

[2] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2000.

[3] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds., *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996.

[4] A. Belhadi, Y. Djenouri, J. C.-W. Lin, C. Zhang, and A. Cano, "Exploring pattern mining algorithms for hashtag retrieval problem," *IEEE Access*, vol. 8, pp. 10569–10583, 2020.

[5] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 7, no. 4, p. e1207, Jul. 2017.

[6] S. Ventura and J. M. Luna, *Pattern Mining with Evol. Algorithms*. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-33858-3.

[7] C. Zhang and S. Zhang, *Association Rule Mining: Models and Algorithms*. Berlin, Germany: Springer, 2002.

[8] X. Han, X. Liu, J. Chen, G. Lai, H. Gao, and J. Li, "Efficiently mining frequent itemsets on massive data," *IEEE Access*, vol. 7, pp. 31409–31421, 2019.

[9] C. C. Aggarwal and J. Han, *Frequent Pattern Mining*. Cham, Switzerland: Springer, 2014.

[10] J. M. Luna, "Pattern mining: Current status and emerging topics," *Prog. Artif. Intell.*, vol. 5, no. 3, pp. 165–170, Aug. 2016.

[11] J. M. Luna, P. Fournier-Viger, and S. Ventura, "Frequent itemset mining: A 25 years review," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 9, no. 6, Nov. 2019.

[12] J. M. Luna, J. R. Romero, and S. Ventura, "Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules," *Knowl. Inf. Syst.*, vol. 32, no. 1, pp. 53–76, Jul. 2012, doi: 10.1007/s10115-011-0419-z.

[13] J. Han and Y. Fu, "Mining multiple-level association rules in large databases," *IEEE Trans. Knowl. Data Eng.*, vol. 11, no. 5, pp. 798–804, Sep./Oct. 1999, doi: 10.1109/69.806937.

[14] S. Ruggieri, D. Pedreschi, and F. Turini, "Data mining for discrimination discovery," *ACM Trans. Knowl. Discovery from Data*, vol. 4, no. 2, pp. 1–40, May 2010, doi: 10.1145/1754428.1754432.

[15] S. Ventura and J. M. Luna, *Supervised Descriptive Pattern Mining*. Cham, Switzerland: Springer, 2018.

[16] P. Gautam and K. R. Pardasani, "Algorithm for efficient multilevel association rule mining," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 5, pp. 1700–1704, 2010.

[17] Y. Wang, L. Yu, Q. Wen, and C. Liu, "Improved multi-level association rule in mining algorithm based on a multidimensional data cube," in *Proc. 3rd Int. Conf. Consum. Electron., Commun. Netw.*, Nov. 2013, pp. 355–358.

[18] V. V. Ramana, M. V. Rathnamma, and A. R. M. Reddy, "Methods for mining cross level association rule in taxonomy data structures," *Int. J. Comput. Appl.*, vol. 7, no. 3, pp. 28–35, Sep. 2010.

[19] J. Han, Y. Cai, and N. Cercone, "Data-driven discovery of quantitative rules in relational databases," *IEEE Trans. Knowl. Data Eng.*, vol. 5, no. 1, pp. 29–40, 1993, doi: 10.1109/69.204089.

[20] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *Proc. 21th Int. Conf. Very Large Data Bases VLDB*, Zurich, Switzerland, 1995, pp. 420–431.

[21] A. A. Nanavati, K. P. Chitrapura, S. Joshi, and R. Krishnapuram, "Mining generalised disjunctive association rules," in *Proc. 10th Int. Conf. Inf. Knowl. Manage. CIKM*, 2001, pp. 482–489.

[22] R. Vimieiro and P. Moscato, "Disclosed: An efficient depth-first, top-down algorithm for mining disjunctive closed itemsets in high-dimensional data," *Inf. Sci.*, vol. 280, pp. 171–187, Oct. 2014.

[23] J. M. Luna, A. Cano, V. Sakalauskas, and S. Ventura, "Discovering useful patterns from multiple instance data," *Inf. Sci.*, vol. 357, pp. 23–38, Aug. 2016, doi: 10.1016/j.ins.2016.04.007.

[24] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, nos. 1–2, pp. 31–71, Jan. 1997.

[25] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D. Sánchez, and S. Vluymans, *Multiple Instance Learning—Foundations and Algorithms*. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-47759-6.

[26] J. M. Luna, M. Ondra, H. M. Fardoun, and S. Ventura, "Optimization of quality measures in association rule mining: An empirical study," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 1, pp. 59–78, Nov. 2018.

[27] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Proc. 5th Int. Conf. Extending Database Technol. EDBT*, Avignon, France, 1996, pp. 3–17.

[28] P. Fournier-Viger, Z. Li, J. C.-W. Lin, R. U. Kiran, and H. Fujita, "Efficient algorithms to identify periodic patterns in multiple sequences," *Inf. Sci.*, vol. 489, pp. 205–226, Jul. 2019, doi: 10.1016/j.ins.2019.03.050.

[29] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data SIGMOD*, 1993, pp. 207–216.

[30] K. Lee, *Modern Heuristic Optimization Techniques With Applications To Power Systems*. Hoboken, NJ, USA: Wiley, 2005.

• • •