Itemset generalization with cardinality-based constraints

Luca Cagliero*, Paolo Garza

Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca degli Abruzzi 24, 10129, Torino, Italy

Abstract

Generalized itemset mining is an established data mining technique that focuses on discovering high-level correlations among large databases. By exploiting a taxonomy built over the data items, items are aggregated into higher level concepts and, thus, data correlations at different abstraction levels can be discovered. However, since a large number of patterns can be extracted, the result of the mining process is often not easily manageable by domain experts.

We propose a novel approach to discovering a compact subset of generalized itemsets from structured data. To guarantee model conciseness and readability, a set of itemsets that has a common generalization is generated only when its cardinality is so small that its manual inspection is practically feasible. Furthermore, generalizations are generated only when their knowledge is covered by a large number of low-level descendant itemsets, and the generalizations are worth considering in place of their many low-level descendants only in these cases.

*Corresponding author. Tel.: +39 011 090 7084. Fax: +39 011 090 7099.
Email addresses: luca.cagliero@polito.it (Luca Cagliero),
paolo.garza@polito.it (Paolo Garza)

Preprint submitted to

Experiments performed on synthetic, benchmark, and real data taken from a mobile application scenario demonstrate the effectiveness and efficiency of the proposed approach.

Keywords: Data Mining and knowledge discovery, Generalized itemset mining, Mobile data analysis

1. Introduction

Frequent generalized itemset mining is an exploratory data mining technique that focuses on discovering recurrent high-level correlations that are hidden in large databases. A pioneering work in this research field has been presented in [3]. This contribution extends the traditional itemset mining problem, which was first introduced in [1] in the context of market basket analysis, to addressing data that have been enriched with taxonomies (i.e., is-a hierarchies). Specifically, to discover data correlations at different abstraction levels, a taxonomy is used to aggregate low-level data items (e.g., market basket items) into higher level concepts (e.g., item categories), called generalized items. Frequent generalized itemsets are sets of items or generalized items for which the frequency of occurrence (support) in the analyzed data is above a given threshold.

In recent years, frequent generalized itemsets have successfully been adopted to analyze data that arises from diverse application domains (e.g., context-aware data analysis [9, 14, 34], network traffic analysis [6, 8]). As a drawback, many frequent itemset mining approaches have been challenged because of the low manageability of the generated models. In fact, the result of the mining process using real-world data is usually a large set of patterns that is neither practical to manage nor easy for domain experts to interpret. To overcome this issue, post-pruning steps are usually applied, which have the following aims: (i) ensuring that only valid and useful itemsets are incorporated into the decision support system and (ii) making the model that is composed of the selected itemsets both manageable and interpretable. To reduce the number of extracted patterns, a support constraint is commonly enforced to prune the itemsets that occur rarely in the analyzed data. However, in some contexts the rare itemsets represent interesting information that is worth considering for decision making. Furthermore, since the support constraint does not ensure a limited pattern set size the readability of the extracted patterns is not guaranteed. When addressing data that are supplied with taxonomies, the number of mined patterns further increases because the high-level itemsets are extracted as well as low-level ones. Nevertheless, a generalized itemset can be considered to be a high-level representative of a subset of low-level descendant itemsets, which might potentially represent more specific (low-level) knowledge in a compact way. Hence, generating compact and manageable generalized itemset-based models that are characterized by a good balance between itemset specialization and generalization is desirable.

This paper proposes a novel approach to generating compact models that are composed of frequent generalized itemsets; the proposed method is both efficient and effective. To generate itemset-based models that can be easily managed by domain experts, without the need for ad-hoc post-pruning phases, we propose to place two novel mining constraints, which are the descendant and ancestor cardinality-based constraints, into the generalized itemset mining process. The aim is two-fold: (i) to prevent the generation of large sets of sibling itemsets that are descendants of the same high-level ancestor and (ii) to report the corresponding generalization (ancestor), which concisely represents the knowledge that is associated with a large set of low-level descendant itemsets. The descendant cardinality-based constraint focuses on pruning those itemset subsets that are not suitable for manual inspection. More specifically, this constraint states that a set of itemsets that have a common generalization is generated only when its manual inspection becomes practically feasible, i.e., its cardinality is smaller than a (analyst-provided) maximum cardinality max_card. During a preliminary analysis, analysts typically focus their attention on subsets of low-level itemsets (i.e., the most fine-grained patterns). Hence, if an itemset set has a humanly manageable size, then experts might deem it useful for decision making. Otherwise, they could examine the corresponding (higher level) generalizations. The ancestor cardinality-based constraint states that only the generalizations whose knowledge is covered by a very large number of lower level itemsets, i.e., the number of itemsets is larger than a minimum (user-provided) cardinality min_card, are generated because they are worth considering in place of their many lower level descendants. Hence, this approach aims at selecting the high-level itemsets that cover the knowledge that is pruned by the descendant cardinality-based constraint. Although minimum and maximum cardinality thresholds might be set independently by domain experts, the use of a unique threshold value allows the itemset-based model to best cover the analyzed data and, thus, to achieve the best trade-off between knowledge specialization and generalization. Note that the proposed approach is applicable to data that come from any applications context in which meaningful taxonomies might be inferred.

We also propose a novel generalized itemset mining algorithm, namely CarGeMi (CARdinality-based GEneralized itemset MIner), in which the newly proposed constraints are incorporated into the mining process to ensure the conciseness and readability of the generated model without performing the traditional itemset mining process followed by post-pruning.

The experiments, which were conducted on datasets coming from a real mobile context-aware applications scenario, demonstrate the effectiveness and usefulness of the proposed approach. The actionability of the discovered patterns for supporting advanced analysis has been validated by a domain expert. Furthermore, the CarGeMi performance and scalability have been evaluated on benchmark and synthetic data.

This paper is organized as follows. Section 2 compares our work with previous approaches. Section 3 introduces preliminary notions about generalized itemset mining. Section 4 formally states the generalized itemset mining problem with cardinality-based constraints, whereas Section 5 describes the CarGeMi algorithm. Section 6 thoroughly describes the experimental evaluation. Finally, Section 7 draws conclusions and discusses future work.

2. Related work

Generalized itemsets have been fruitfully exploited in many research contexts (e.g., context-aware systems [9, 14, 34], network traffic analysis [6, 8]). For this reason, in recent years, many efficient mining algorithms that integrate constraints or taxonomy-based filtering techniques have been proposed (e.g., [4, 5, 7, 20, 24, 27, 3, 30, 31]).

The generalized itemset mining problem was first addressed in [3] in the context of market basket analysis. The authors proposed a generalized frequent itemset mining algorithm that generates itemsets by considering, for each item, its parents in the hierarchy. Hence, candidate frequent itemsets are generated by exhaustively evaluating the taxonomy and, thus, by producing a large number of redundant patterns. More recent approaches address generalized itemset mining complexity reduction by preventing the generation of uninteresting candidate patterns. For example, in [30], subset-superset and parent-child relationships in the lattice of generalized patterns are exploited to constrain the mining process. Novel optimization strategies applied to generalized itemset mining have also been proposed [7, 20, 24, 31]. For example, in [24, 31], an attempt to mine closed and maximal itemsets [29] in the presence of taxonomies has been conducted. In contrast, in [20], the authors propose an optimization that is based on a top-down hierarchy traversal. This method identifies in advance itemsets that cannot be frequent in a transactional dataset by exploiting the Apriori principle [1]. More recently, an opportunistic-driven approach has been introduced [7]. This approach avoids exhaustive taxonomy evaluation followed by post-pruning by generalizing an itemset only if its support value is below the minimum support threshold. Similarly, we also propose a novel and efficient algorithm for generalized itemset mining. However, instead of pruning itemsets based on their main quality indexes, we analyze the cardinality of the pattern sets related to the same upper level generalization to drive the itemset extraction process. Specifically, only manageable sets of descendant itemsets are kept because they are worth considering during a manual inspection. Furthermore, generalizations are kept only when their descendant set is not practically manageable. Another recently proposed approach [10] analyzes itemset correlation flippings while generalizing items at higher abstraction levels. However, the above-mentioned approach does not consider cardinality-based constraints. The proposed constraints could be classified, at first glance, as global constraints [18]. Instead of pruning itemsets based on their main quality indexes, global constraints prune (non-generalized) itemsets by comparing each of them with a set of related (non-generalized) itemsets. To the best of our knowledge, global constraints have neither been defined nor applied in the context of generalized itemsets.

A related issue is the discovery and selection of relevant subsets of (non-generalized) itemsets. Notable examples of relevant subsets are maximal and closed frequent itemsets. Maximal frequent itemsets are frequent itemsets whose supersets are all infrequent [1]. In contrast, a frequent itemset is closed if none of its immediate supersets has the same support [29]. Efficient algorithms for mining both maximal and closed frequent itemsets have been proposed (e.g., [35, 36]). In parallel, other approaches have adopted probabilistic approaches to select the subset of most informative yet non-redundant frequent itemsets. Some of these approaches (e.g., [13, 23]) compared the observed frequency (i.e., the support) of each itemset against a null hypothesis (e.g., its expected frequency) to evaluate its interestingness; other approaches (e.g., [26, 33]) also considered the previously selected patterns in itemset evaluation to reduce the model redundancy. However, the above-mentioned approaches did not address pattern mining in the presence of taxonomies and also did not propose cardinality-based constraints to generate compact and easily manageable pattern sets.

3. Preliminaries

In the context of structured data [32], a dataset is composed of a set of records. Each record is a set of items, where an item is a couple (attribute_name, value). While attribute_name is the description of a data feature, value represents the associated information and belongs to the corresponding attribute domain. Since continuous attribute values are typically not suitable for use in itemset mining, they are preliminary discretized [32].

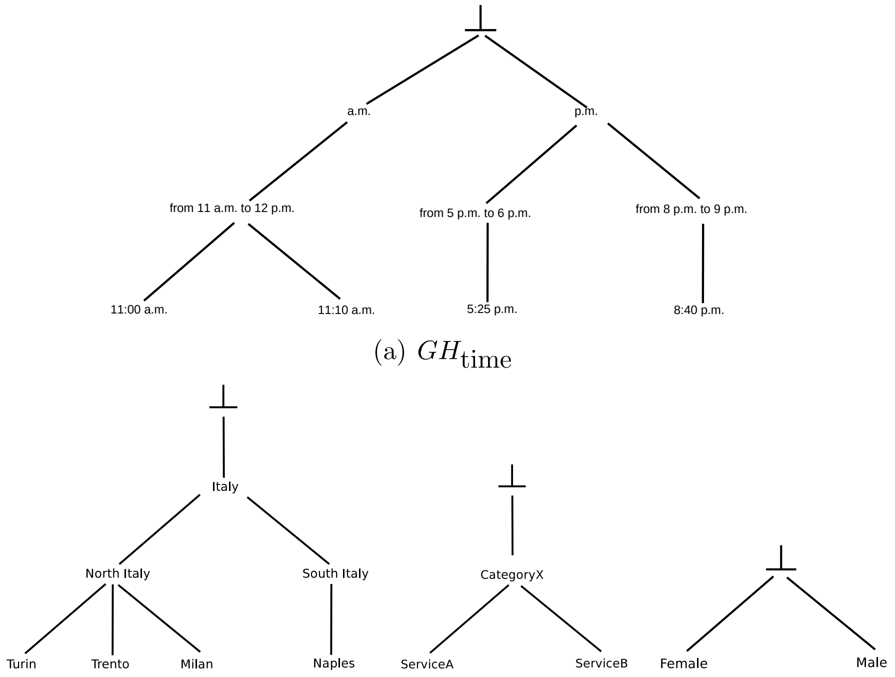
Table 1 shows the structured dataset that is exploited below as a running example. This dataset is composed of 5 records, and each record corresponds to a user request that was submitted to a mobile application through her/his mobile phone. This request is characterized by the following attributes: user gender, time, location, and service description. To generalize items that are contained in a structured dataset at a higher abstraction level, a taxonomy can be defined. A taxonomy [21] is a forest of generalization hierarchies, in which each tree represents a hierarchy of aggregations that are defined on an attribute domain. In the following, Γ and GH_i represent a taxonomy and its generalization hierarchy corresponding to the i-th dataset attribute, respectively. Figure 1 reports an example taxonomy that was built over the running example dataset. This taxonomy is composed of 4 generalization hierarchies, one for each dataset attribute. As an example, consider the generalization hierarchy corresponding to the *location* attribute (see Figure 1(b)). Leaf nodes are labeled with values in the *location* attribute domain, whereas non-leaf nodes are leaf node aggregations that are labeled with distinct values that are not in the attribute domain. Root nodes are labeled with the special value $\perp$. Although a taxonomy could potentially include an arbitrary number of generalization hierarchies per attribute, for the sake of simplicity in this paper, we will consider taxonomies that contain exactly one generalization hierarchy per attribute.

Table 1: Example dataset $\mathcal{D}$.

Time	**User gender**	**Location**	**Service description**
11:00 a.m.	Male	Milan	ServiceA
11:10 a.m.	Male	Milan	ServiceA
8:40 p.m.	Female	Turin	ServiceA
11:00 a.m.	Female	Trento	ServiceA
5:05 p.m.	Female	Naples	ServiceB

In the presence of taxonomies, itemsets are sets of data items that belong to distinct dataset attributes and whose values are taxonomy node labels (disregarding the root label). Items can be mapped to the corresponding taxonomy nodes. Itemsets that contain at least one item that is mapped to a non-leaf taxonomy node (i.e., a generalized item) are called *generalized* itemsets [3]. The generalization level of an item with respect to a taxonomy Γ is defined as the height of the subtree rooted in the corresponding node. The level $\mathrm{L}[X,\Gamma]$ of a generalized itemset X with respect to Γ is the maximum among its item levels. For example, {(Service, CategoryX), (Location, Italy)} is a generalized itemset of level 3.

A k-itemset I (i.e., a set of k items) covers a given record r if all of its items are either contained in r or in ancestors of an item in r with respect to the given taxonomy [3]. Given a structured dataset $\mathcal{D}$, the support of I



(a) GH_{time}

(b) GH_{location} (c) GH_{service} (d) GH_{gender}

Figure 1: Taxonomy built over the data items in $\mathcal{D}$

in $\mathcal{D}$ is defined as the ratio between the number of records in $\mathcal{D}$ that are covered by I and the total number of records in $\mathcal{D}$. (Generalized) itemsets whose support is above or equal to a given support threshold are said to be *frequent* [3]. For example, the generalized itemset {(Service, ServiceA), (Location, North Italy)} has support $\frac{4}{5}$ in $\mathcal{D}$ because it covers 4 out of 5 dataset records (See Table 1).

Given two generalized itemsets X and Y, X is said to be an ancestor of Y with respect to a given taxonomy Γ if L[X,Γ]>L[Y,Γ] and for every (generalized) item $y_i \in Y$ there exists a (generalized) item $x_i \in X$ such that either $x_i = y_i$ or x_i is an ancestor of y_i in $GH_i \in \Gamma$. If X is an ancestor of Y, then Y is a descendant of X, i.e., $Y \in Desc[X, \Gamma]$. Given an arbitrary

Table 2: Generalized itemsets extracted from $\mathcal{D}$. min_sup = 1.

Id	**Generalized itemsets**	**Sup**	**Level**	**Lower level descendants**	**Itemsets that satisfy the cardinality constraints (max_card=3, min_card=3)**
					Selected: X Pruned: -
1	{Turin}	1	1	-	-
2	{Naples}	1	1	-	X
3	{Trento}	1	1	-	-
4	{ServiceB}	1	1	-	X
5	{Milan}	2	1	-	-
6	{ServiceA}	4	1	-	X
7	{Naples,ServiceB}	1	1	-	X
8	{Turin,ServiceA}	1	1	-	-
9	{Trento,ServiceA}	1	1	-	-
10	{Milan,ServiceA}	2	1	-	-
11	{South Italy}	1	2	{Naples}	-
12	{North Italy}	4	2	{Milan} {Turin} {Trento}	X
13	{CategoryX}	5	2	{ServiceA} {ServiceB}	-
14	{North Italy,ServiceA}	4	2	{Milan,ServiceA} {Turin,ServiceA}	-
15	{South Italy,ServiceB}	1	2	{Naples,ServiceB}	-
16	{Turin,CategoryX}	1	2	{Turin,ServiceA}	-
17	{Milan,CategoryX}	2	2	{Milan,ServiceA}	-
18	{Trento,CategoryX}	1	2	{Trento,ServiceA}	-
19	{Naples,CategoryX}	1	2	{Naples,ServiceB}	-
20	{South Italy,CategoryX}	1	2	{Naples,ServiceB}	-
21	{North Italy,CategoryX}	4	2	{Turin,ServiceA} {Milan,ServiceA} {Trento,ServiceA}	-
22	{Italy}	5	3	{North Italy} {South Italy}	-
23	{Italy,ServiceA}	4	3	{North Italy,ServiceA}	-
24	{Italy,ServiceB}	1	3	{South Italy,ServiceB}	-
25	{Italy,CategoryX}	5	3	{North Italy,CategoryX} {South Italy,CategoryX} {North Italy,ServiceA} {South Italy,ServiceB}	X

generalized itemset Z, for our purposes we will denote as $\phi(Z) \subseteq$ Desc[Z, Γ] the set of descendants of Z whose generalization level is L[Z,Γ]-1. For example, according to the taxonomy that is reported in Figure 1, Italy is an ancestor of North Italy and ϕ(Italy)={North Italy, South Italy}.

Given a structured dataset $\mathcal{D}$, a taxonomy, and a minimum support threshold min_sup, the frequent generalized itemset mining problem [3] entails discovering all of the frequent (generalized) itemsets from $\mathcal{D}$, i.e., all of the frequent (generalized) itemsets that satisfy min_sup.

4. Generalized itemset mining with cardinality-based constraints

The support threshold is a widely used itemset mining constraint; a support threshold prevents the extraction of itemsets that occur rarely in the source data [1]. Unfortunately, the frequent itemset mining process might still generate a large set of patterns that is hardly manageable by domain experts, especially when very low support thresholds are enforced [4]. This situation motivates the need for enforcing new constraints during the mining process with the goal of making the result easily manageable by domain experts.

We propose a new type of constraint called *cardinality-based constraints*. The proposed constraints are placed in the generalized itemset mining process with the aim of producing compact generalized itemset sets, which can be inspected manually by domain experts. Unlike many traditional constraints (e.g., the minimum support constraint), which evaluate the itemset significance based solely on the characteristics of the itemset, the newly proposed constraints evaluate the itemset interestingness by comparing it with a set of other itemsets. Specifically, these constraints consider, for each generalized itemset I of level $l > 1$, the cardinality of the set of its low-level descendants of level $l - 1$, to decide whether I and its descendants might be suitable for manual inspection. The cardinality-based constraints can be specialized into two subcategories: the *descendant cardinality-based constraint* and the *ancestor cardinality-based constraint*. Their formal definitions are given in the following.

The descendant cardinality-based constraint selects manageable subsets of sibling generalized itemsets, i.e., itemsets that (i) have the same generalization level and (ii) are descendants of the same high-level itemset. More specifically, a subset of sibling generalized itemsets satisfies the descendant cardinality-based constraint if its manual inspection is practically feasible, i.e., when its cardinality is smaller than a (analyst-provided) maximum cardinality threshold.

Definition 4.1. Descendant cardinality-based constraint. *Let $\mathcal{D}$ be a structured dataset, let Γ be a taxonomy, let X be a generalized itemset, and let max_card be a non-negative integer number. The generalized itemsets in $\phi(X)$ satisfy the descendant cardinality-based constraint if $\nexists\ Y \subseteq X$ such that $|\phi(Y)| \geq$ max_card.*

Enforcing the descendant cardinality-based constraint prevents the generation of large descendant sets. The key idea is that large sets of itemsets are barely manageable by domain experts and, hence, their extraction is prevented. The constraint prunes X's descendants if X, or any of its subsets $Y \subseteq X$, has more than max_card descendants.

Consider again the running example (see Table 1 and Figure 1). Table 2 reports the itemsets that were mined by a traditional approach [3] from the location and service attributes in $\mathcal{D}$ by exploiting the taxonomy that is reported in Figure 1 and by enforcing an absolute support threshold min_sup equal to 1. Itemset support values (i.e., the observed frequencies), generalization levels with respect to the given taxonomy, and lower level descendants are also reported. Note that the enforcement of low support thresholds often entails generating a very large number of patterns that might become difficult to consider. Even when coping with very small datasets, such as the dataset that is reported in Table 1, and when considering only two of its attributes, enforcing low support thresholds leads to the extraction of a very large number of patterns (25). According to the given taxonomy, Turin, Milan, and Trento are generalized as North Italy. By enforcing a maximum cardinality threshold max_card = 3, the expert deems that itemset sets that include three or more itemsets are not manageable for subsequent analysis. Hence, {(Location,Turin)}, {(Location,Milan)}, and {(Location,Trento)} are discarded because they are all descendants of the upper level generalization {(Location,North Italy)}. When considering combinations of the requested service description and the requested location, i.e., patterns with schema {(Service=value, Location=value)}, analysts should expect a similar trend and, thus, might no longer be interested in considering itemsets that include the items {(Location,Turin)}, {(Location,Milan)}, and {(Location,Trento)}; instead, they would immediately consider itemsets that include their generalization {(Location,North Italy)}.

The descendant cardinality-based constraint has a notable property, called the anti-monotonicity property, which allows the search space to be pruned early.

Property 1. Anti-monotonicity property of the descendant cardinality-based constraint. *Let $\mathcal{D}$ be a structured dataset, and let Γ be a taxonomy. Let G be the set of generalized itemsets that are mined from $\mathcal{D}$ by evaluating the taxonomy Γ and by enforcing no minimum support threshold (i.e., min_sup = 1). Let $\mathcal{S} \in G$ be the subset of generalized itemsets that satisfy the descendant cardinality-based constraint. The anti-monotonicity of the descendant cardinality-based constraint states that, if a generalized itemset X*

does not satisfy the descendant cardinality-based constraint, i.e., $X \notin \mathcal{S}$*, then every itemset* Y *such that* $X \subset Y$ *does not satisfy that constraint.*

Proof 1. *Let* X *be a generalized itemset that does not satisfy the descendant cardinality-based constraint. By definition, there exists a generalized itemset* $Z \subseteq X$ *such that* $|\phi(Z)| \geq max\_card$*. Because* $Z \subseteq X$*, for every itemset* Y *such that* $X \subset Y$*,* $Z \subset Y$*. Hence,* Y *does not satisfy the descendant cardinality-based constraint.*

An algorithm that exploits the above-mentioned property to efficiently perform generalized itemset mining with cardinality-based constraints is presented in Section 5.

The ancestor cardinality-based constraint selects, among the upper-level generalizations, the most valuable generalizations based on the cardinality of their lower level descendant set.

Definition 4.2. Ancestor cardinality-based constraint. *Let* $\mathcal{D}$ *be a structured dataset, let* Γ *be a taxonomy, let* X *be a generalized itemset, and let* min_card *be a non-negative integer number. The generalized itemset* X *satisfies the ancestor cardinality-based constraint if* $|\phi(X)| \geq min\_card$*.*

Consider again the previous example. The domain expert is asked to set the minimum cardinality of the set of patterns related to the same region for which an upper generalization is worth considering in place of its (large) descendant set. By enforcing a minimum cardinality threshold min_card = 3, the generalized itemset {(Location,North Italy)} is extracted because it has at least 3 descendants (Turin, Milan, and Trento) and, hence, satisfies the constraint. The high-level itemset related to "North Italy" is chosen as a representative of the knowledge that is covered by its corresponding (large) descendant set.

The enforcement of both the descendant and the ancestor cardinality-based constraints during the generalized itemset mining process produces a concise pattern-based model that is suitable for domain expert analysis. Although the minimum and maximum cardinality thresholds can be set independently, the use of a unique threshold value (i.e., min_card = max_card) is advisable because it makes the mined pattern-based model representative of all of the analyzed data and, thus, especially suitable for targeted analysis. Specifically, when min_card = max_card (also denoted in the following as the *standard configuration*), unmanageable descendant sets are pruned and the corresponding ancestors are kept. Hence, all of the dataset records that are covered by the discarded descendants are still covered by an upper level generalization. We formalize this notable property as follows.

Property 2. Coverage of a descendant set. *Let $\mathcal{D}$ be a structured dataset, let Γ be a taxonomy, let X be a generalized itemset, and let min_card (= max_card) be a non-negative integer number. Let $\phi(X) \subseteq$ Desc[X, Γ] be the set of descendants of X whose generalization level is L[X,Γ]-1 such that all of its itemsets do not satisfy the descendant cardinality-based constraint max_card. X satisfies the ancestor cardinality-based constraint min_card (= max_card) and covers all of the records in $\mathcal{D}$ that are covered by any itemsets in $\phi(X)$.*

Proof 2. *Because by Definition 4.1 there exists $\Theta \subseteq \phi(X)$ such that $|\Theta| \geq$ max_card, it follows that $\phi(X) \geq$ max_card=min_card. Hence, X satisfies the* *ancestor cardinality-based constraint. Furthermore, because $\phi(X) \subseteq$ Desc[X, Γ], it follows that all of the records in D that are covered by itemsets in $\phi(X)$ are also covered by X.*

The above property confirms that setting a unique value for min_card and max_card prevents the expert from disregarding potentially relevant data correlations because discarded low-level correlations are still maintained at a higher abstraction level.

In Table 2, Column (6) reports the set of generalized itemsets that are mined by CarGeMi by setting the minimum and the maximum cardinality thresholds to three (min_card = max_card = 3). Generalized itemsets that satisfy the above constraints are the itemsets that have (i) three lower level descendants or more, and (ii) at most two siblings. Readers could notice that the generated model (i.e., the set of generalized itemsets that satisfy both constraints) is a compact and easy-to-read set, which is composed of only 6 out of the 25 patterns that would be extracted without enforcing any cardinality-based constraint.

A thorough experimental analysis of the impact of the proposed constraints is reported in Section 6.

5. The Cardinality-based Generalized Itemset Miner

Given a structured dataset $\mathcal{D}$, a taxonomy Γ that is built over the data items in $\mathcal{D}$, and a minimum and a maximum cardinality threshold min_card and max_card, respectively, the CarGeMi (CARdinality-based GEneralized itemset MIner) algorithm addresses the extraction of all of the itemsets, generalized and not, that satisfy both the descendant and the ancestor cardinality-based constraints (Cf. Definitions 4.1 and 4.2) and occur at least once in $\mathcal{D}$ (i.e., min_sup = 1). A pseudo-code of the CarGeMi algorithm is reported in Algorithm 1.

Similar to *Cumulate* [3], CarGeMi is a level-wise algorithm that follows an *Apriori*-like approach to itemset mining [2]. At each iteration, the algorithm generates all of the generalized itemsets of a given length. Specifically, at an arbitrary iteration k, *Cumulate* performs two main steps: (i) candidate generation, in which all of the generalized k-itemsets are generated from the selected generalized $(k-1)$-itemsets and (ii) candidate evaluation and pruning, to discard candidate generalized itemsets that do not satisfy the mining constraints. Unlike *Apriori* and *Cumulate*, CarGeMi addresses the itemset mining problem with cardinality-based constraints and it does not enforce any support constraint, i.e., min_sup is set to 1. In detail, at an arbitrary iteration k CarGeMi performs the following steps: (i) generation of the sets $C[l,k]$ of candidate generalized k-itemsets of increasing level l, (ii) pruning of the set of descendants of every generalized itemset for which (a) the support is equal to zero, i.e., the itemset never occurs in $\mathcal{D}$ (see lines 9-10) or (b) the cardinality is higher than or equal to the maximum descendant cardinality threshold max_card (lines 11-20), (iii) generation of the set $C[l,k+1]$ of generalized $(k+1)$-itemsets of level l (see lines 21-25), and (iv) pruning of the set of generalized itemsets that have a level greater than l and whose descendant set has a cardinality that is lower than min_card (lines 26-34). The pruning step (ii).(b) exploits the anti-monotonicity property of the descendant cardinality-based constraint (see Property 1).

The candidate set $C[1,1]$ is initialized with the set of items that oc-

Algorithm 1 The CARdinality-based GEneralized itemset MIner algorithm

```
Input: structured dataset $\mathcal{D}$, taxonomy $\Gamma$, maximum descendant cardinality threshold max_card, minimum ancestor cardinality threshold min_card
Output: $\mathcal{L}$, the set of generalized itemsets that satisfy both the ancestor and descendant constraints
/* initializations */
$\mathcal{L} = \emptyset$
/* $C[l,k]$ is the set of candidate generalized $k$-itemsets of level $l$ */
$C[1,1]$ = {domain of the items in $\mathcal{D}$}
$C[l,1]$ = {generalized items of level $l$ in $\Gamma$} $\forall\ l > 1$
$k = 1$
while $\exists\ l \mid C[l,k]$ is not empty do
    /* Generate candidate k-itemsets of increasing level and select the ones that satisfy the descendant cardinality-based constraint and have a support higher than or equal to 1 */
    scan $D$ and count the support of $c_i\ \forall\ c_i \in C[l,k]$
    discard $c_i \in C[l,k]$ that never occurr in $\mathcal{D}$
    for $l$ from 2 to maxlevel do
        for all $c \in C[l,k]$ do
            $c$.desc = {$it \in C[l-1,k] \mid it$ is a descendant of $c \in C[l,k]$ of level $l-1$ in $\Gamma$}
            /* Select the level-($l-1$) descendants of $c$ */
            if $|c.\text{desc}| \geq$ max_card then
                /* Discard the descendants that do not satisfy max_card */
                remove $c$.desc from $C[l-1,k]$
            end if
        end for
    end for
    for $l$ from 1 to maxlevel do
        /* Generation of level-l candidate $(k+1)$-itemsets */
        $C[l,k+1]$ = candidate_generation( $C[l,k]$)
        /* Apriori-based generation step of candidates $C[l,k+1]$ by self-joining of the candidate set $C[l,k]$ */
    end for
    for $l$ from 1 to maxlevel do
        for all $c \in C[l,k]$ do
            /* Select the level-l generalized k-itemsets that satisfy the ancestor cardinality constraint max_card */
            if L[c,Γ]>1 and $|c.\text{desc}| \geq$ min_card then
                insert $c$ into $\mathcal{L}[l,k]$
                /* Update of the output set $\mathcal{L}$ */
            end if
        end for
    end for
    $k = k + 1$
end while
return $\mathcal{L}$
```

cur in $\mathcal{D}$. Because we address generalized itemset mining in the context of structured datasets, CarGeMi exploits the characteristics of the datasets to prune candidates that include multiple items per attribute early. Candidates that do not satisfy the descendant cardinality-based constraint are discarded from the set $C[l, k]$ of candidate generalized k-itemsets of level l and are not considered in the generation of the upper level k-itemsets. Candidates that fulfill both the ancestor and the descendant constraints are included in the output set $\mathcal{L}$.

In the worst case, CarGeMi requires up to $n \cdot h$ dataset scans, where n is the number of record attributes and h is the height of the used taxonomy. An empirical analysis of the CarGeMi scalability is given in Section 6.4.

6. Experimental results

We performed a large set of experiments on real-life, benchmark, and synthetic datasets to evaluate the efficiency and effectiveness of the proposed approach. Specifically, we analyzed (i) the impact of the cardinality-based constraints on the characteristics of the mined patterns (see Section 6.1), (ii) the performance comparison between CarGeMi and the state-of-the-art approaches (see Section 6.2), (iii) the usefulness of the patterns that were mined from data that were acquired from real-life mobile applications (see Section 6.3), and (iv) the scalability of the CarGeMi algorithm (see Section 6.4).

Table 3 summarizes the main characteristics of the real and benchmark datasets. All of the evaluated datasets and taxonomies, the synthetic data generation, the Python version of the code for CarGeMi and the other

Table 3: Dataset characteristics

Dataset	Number of records	Number of attributes	Taxonomy height
Recs	5,688	4	4
TeamLife	1,197	4	4
Nursery (UCI)	12,960	9	2
Shuttle (UCI)	43,500	10	2

evaluated approaches (i.e., GenIO [7] and Cumulate [3]) are available at [19]. A more detailed description of the analyzed datasets and taxonomies is given in the following.

Mobile datasets. The analyzed real datasets have been collected by Telecom Italia Lab[1] and are related to a variety of mobile applications. The considered applications, called *Recs* and $TeamLife$, provide users with a set of services (e.g., weather forecasting, restaurant recommendations, and photo and movie uploads) through mobile devices (e.g., smartphones and tablet PCs). We collected the application service requests that come from each application in separate log files (i.e., datasets).

Recs. The *Recs* application is a recommender system that provides recommendations to users on restaurants, museums, movies, and other entertainment activities. Each user can request a recommendation, vote for an item (i.e., an entertainment center), update a vote, and upload a file or a photo to provide useful information about an item (i.e., a restaurant or a museum) or to post a comment. Hence, a set of services is provided to the end users to perform the described operations/services. The dataset contains

[1] Telecom Italia Lab is the research hub of the Telecom Italia Group, an international leader in the telecommunications area.

the user requests that were submitted and that were obtained by logging the user requests over the time period of three months. For the *Recs* dataset, the following generalization hierarchies have been considered:

- date → month → trimester → year
- time stamp → hour → time slot (2-hour time slots)→ day period (AM/PM)
- user → gender
- service → service category

TeamLife. The *TeamLife* dataset was generated by logging the activities of the users of the *TeamLife* application. *TeamLife* allows users to upload files, photos, and videos and to share them with other system users. Four different types of services are offered. The dataset collects the user requests that were submitted over a time period of three months. For *TeamLife*, we used a taxonomy that was similar to the taxonomy previously described for the *Recs* dataset.

UCI benchmark datasets. Two representative UCI benchmark datasets [12], namely *Shuttle* and *Nursery*, have also been tested to evaluate the CarGeMi performance on data from different domains. Specifically, the NASA shuttle dataset contains 9 continuous attributes that describe the positions of radiators in the Space Shuttle and one class attribute that is useful for classification purposes. The nursery dataset contains data that was used to rank applications for nursery schools. Applications are ranked at 5 possible levels: recommended, very recommended, priority, special priority, and not recommended.

To build generalization hierarchies over the continuous attributes, we applied several 10-bin equi-depth discretization steps with finer granularities [32]. The finest discretized values are considered to be the data item values and, thus, become the taxonomy leaves, whereas coarser discretizations are exploited to aggregate the corresponding lower level values into higher level values. Generalization hierarchies over the categorical attributes are analyst-provided. For example, for the Nursery attribute *Child's nursery* "critical" and "very critical" priority values are generalized as "critical priority".

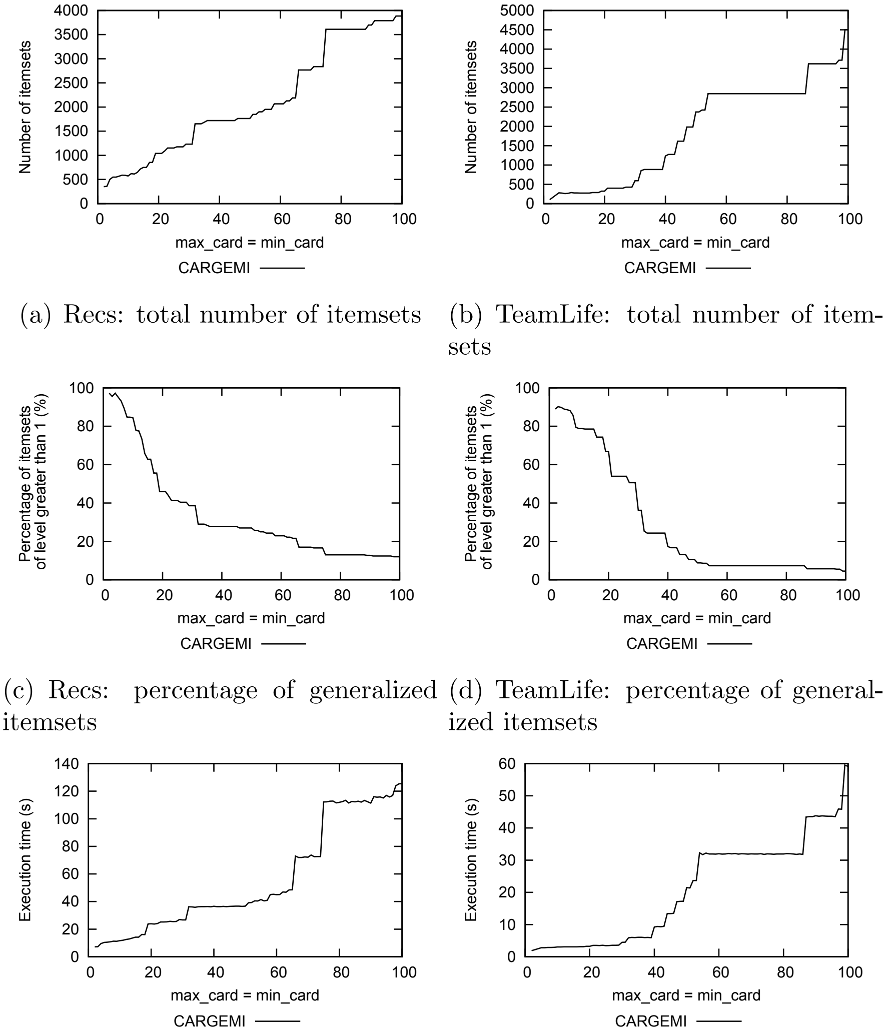
Synthetic datasets. We exploited the IBM synthetic dataset generator [22] to evaluate CarGeMi scalability. The data generator automatically produces structured datasets that are composed of a user-specified number of records and attributes. To automate the taxonomy generation procedure over the generated datasets, we extended the generator source code as follows. For each attribute, the item values are treated as taxonomy leaves; they are sorted into lexicographical order and are grouped by an aggregation factor f. Any item group produces an upper level item that aggregates all of the group members. The procedure iterates until all of the items are clustered in a unique group (i.e., the root node). To keep the ratio between the level-l and level-(l-1) item set cardinalities constant, we set f to $\lceil \sqrt[\text{(h - 1)}]{n} \rceil$, where n is the attribute domain size and h is the user-specified taxonomy height. The extended generator version is available for research purposes at [19].

6.1. Impact of the cardinality-based constraints

We performed a set of experiments to evaluate the impact of the cardinality-based constraints on the characteristics of the mined patterns. Figure 2 reports the number of itemsets (generalized and not), the percentage of generalized itemsets, and the CarGeMi execution time that were achieved on the real-life mobile datasets by setting the standard configuration (i.e., max_card=min_card) and by varying the values of max_card and min_card in the range [2,100].

As expected, the total number of extracted itemsets increases when the value of max_card (= min_card) increases because the descendant cardinality-based constraint becomes less selective and, thus, a higher number of low-level itemsets is, on average, extracted (see Figures 2(a) and 2(b)). However, the percentage of mined generalized itemsets decreases when max_card increases (see Figures 2(c) and 2(d)) because high-level pattern extraction is prevented as a result of the selection of the corresponding descendant sets.

The curve slopes in Figure 2 depend on the cardinality of the considered attribute domains. In fact, the larger the domain of an attribute A is, the higher, on average, the number of generated low-level itemsets that refer to A (i.e., the itemsets that include non-generalized items of A) becomes. Hence, enforcing the descendant cardinality-based constraint on datasets with attributes that are characterized by a high domain cardinality yields, on average, more severe low-level itemset pruning. In contrast, the pruning selectivity of the ancestor cardinality-based constraint on high-level itemsets, on average, decreases when addressing data with similar characteristics. Because the CarGeMi execution time is mainly affected by the number of generated



(a) Recs: total number of itemsets

(b) TeamLife: total number of itemsets

(c) Recs: percentage of generalized itemsets

(d) TeamLife: percentage of generalized itemsets

(e) Recs: execution time

(f) TeamLife: execution time

Figure 2: CarGeMi: performance analysis

itemsets, its trend is analogous to the one of the previously described curves (see Figures 2(e) and 2(f)).

Note that experts can also enforce different values for max_card and min_card. However, in general, this choice is suboptimal in terms of model compactness because the extraction of a manageable set of sibling itemsets might not prevent the extraction of the corresponding ancestor itemset. Because we focus on generating compact generalized itemset models, we deem this type of parameter setting to be less interesting for our research purposes. Hence, the corresponding curves have been omitted.

6.2. Comparison between CarGeMi *and state-of-the-art approaches*

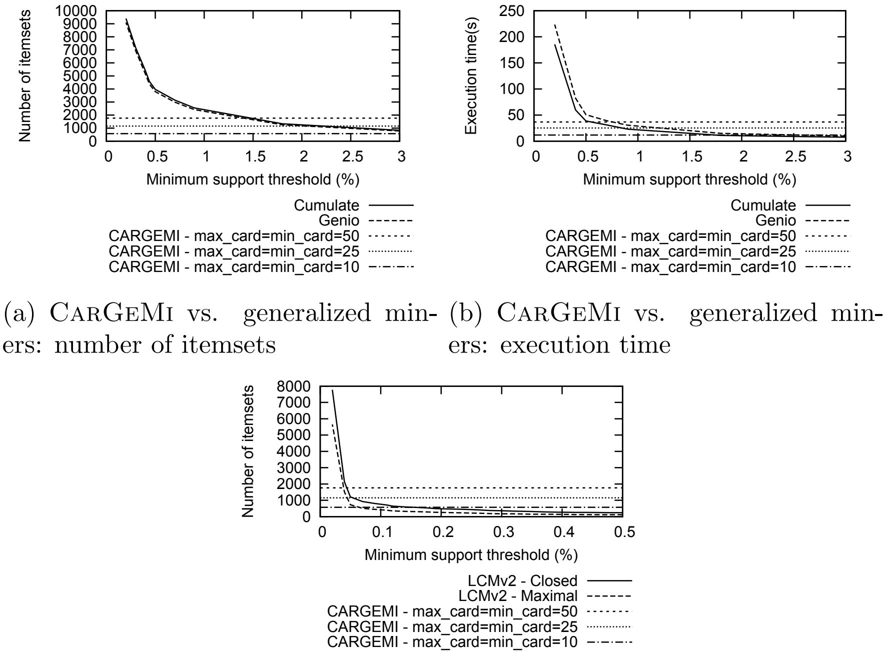
We compared CarGeMi mining results with those achieved using (i) two representative generalized itemset mining algorithms, i.e., the traditional Cumulate algorithm [3] and the recently proposed GenIO algorithm [7] and (ii) a maximal and closed itemset miner [35].

6.2.1. Comparison with previous generalized itemset mining algorithms

To evaluate the compactness and manageability of the subset of patterns that were generated using our approach, we compared them with the results from two representative generalized itemset mining algorithms, Cumulate [3] and GenIO [7]. Cumulate and GenIO, which are similar to all of the previous generalized itemset mining algorithms, enforce a minimum support threshold (i.e., a minimum frequency of occurrence of the generated patterns) to reduce the number of mined (generalized) itemsets. However, the enforcement of a minimum support threshold could also prune rare but interesting knowledge [13]. Unlike [7, 3], our approach exploits the cardinality-based constraints to make the set of mined patterns manageable by domain experts and does not enforce any minimum support threshold.

To compare CarGeMi with Cumulate and GenIO, we first performed several mining sessions with both Cumulate and GenIO by varying the minimum support threshold, which is the only parameter of both algorithms, in the range [0.2%, 3%]. Then, we compared the obtained results, in terms of the number of extracted itemsets and the execution time, with the results achieved by our approach. We considered the following settings, which are taken as representatives among all of the tested settings: min_card = max_card = 10, min_card = max_card = 25, and min_card = max_card = 50. Figures 3(a) and 3(b) summarize the results that were achieved on the Recs dataset. Similar results were obtained for the other datasets.

Because CarGeMi does not enforce any support constraint, the number of itemsets mined by CarGeMi remains constant by varying the support threshold. The number of itemsets mined by CarGeMi is always less than the number of patterns extracted by Cumulate when max_card is set to 10. A similar result was obtained by comparing CarGeMi with GenIO. The number of (generalized) itemsets mined by Cumulate and GenIO becomes comparable or slightly less than the number extracted by CarGeMi only when relatively high support thresholds are enforced (e.g., min_sup>1.5% for both Cumulate and GenIO) and max_card is set to 50 for CarGeMi. Because the execution time of all of the three considered algorithms is strictly related to the number of generated combinations, when low minimum support thresholds are enforced CarGeMi performs significantly better than both Cumulate and GenIO in terms of the execution time (see Figure 3(b)).



(a) CarGeMi vs. generalized miners: number of itemsets

(b) CarGeMi vs. generalized miners: execution time

(c) CarGeMi vs. closed/maximal: number of itemsets

Figure 3: Recs dataset: Comparison between CarGeMiand its competitors

To additionally analyze data that have different distributions and that come from diverse contexts, we compared CarGeMi and GenIO performance on the UCI datasets. While the GenIO itemset mining process from Shuttle takes more than 10 hours to complete successfully, even when relatively high support threshold values (higher than 30%) are enforced, CarGeMi always succeeds and generates 593 generalized itemsets when max_card (= min_card) is set to 2 and 70,209 generalized itemsets when max_card (= min_card) is equal to 10. The former extraction takes 83 seconds, whereas the latter takes approximately 1.5 hours. When addressing the Nursery dataset, GenIO takes less than 10 hours to successfully complete only when the minimum support threshold is higher than or equal to 10%. For example, when min_sup=10%, GenIO generates 94,902 itemsets and takes approximately 3.5 hours to succeed. In contrast, by setting max_card (= min_card) to 15, the number of generalized itemsets mined by CarGeMi is two orders of magnitude less than the number mined by GenIO (1,243), and the execution time is approximately 42 seconds. As discussed in Section 6.1, when lowering max_card, the model size and execution time further reduce.

The obtained results show that, in most cases, CarGeMi generates more compact pattern sets than its competitors and that it is, in general, more efficient in terms of the execution time. Furthermore, unlike all of the previous approaches, CarGeMi does not require the enforcement of a minimum support constraint to successfully complete the mining task.

6.2.2. Comparison between CarGeMi *and closed and maximal itemset mining algorithms*

Closed [29] and maximal [11] itemsets are two types of itemsets that are frequently used to concisely represent the main characteristics of the analyzed data. We compared the pruning effectiveness that is achieved by CarGeMi with the effectiveness of a non-generalized closed and maximal itemset mining algorithm, which was derived from the LCM algorithm implementation [35]. We considered frequent non-generalized closed and maximal itemsets because (i) to the best of our knowledge, no implementation of a frequent generalized closed itemset mining algorithm is publicly available and (ii) non-generalized frequent closed/maximal itemsets are a subset of the itemsets that are extracted by any generalized itemset miner. Hence, their cardinality represents a lower bound estimate.

Figure 3(c) reports the results that were achieved for the Recs dataset when the minimum support threshold is varied; the support threshold is the only parameter of the closed and maximal itemset mining algorithm and is set in the range [0.02%, 0.5%]. The number of generalized and non-generalized itemsets mined by our approach is less than the number of frequent non-generalized closed and maximal itemsets when medium and low support thresholds are enforced. Indeed, in most of the cases in which a significant number of potentially relevant patterns are extracted, the pruning selectivity of the cardinality-based constraint (not combined with any support constraint, i.e., min_sup=1) is higher than the selectivity that is achieved by closed/maximal itemset selection. Similar results have been obtained for the TeamLife dataset. To perform a fair comparison, we also tested the closed/maximal itemset miner [35] without enforcing any support threshold. As expected, the gap between our approach and its competitors increases significantly. Specifically, the number of mined closed/maximal itemsets becomes approximately one order of magnitude higher than the number of generalized itemsets mined by CarGeMi.

6.3. Examples of real-life applications

We analyzed the usefulness and applicability of the proposed approach in a real-life mobile context with the help of a domain expert.

The expert is interested in performing targeted advertising based on the analyzed data. To analyze the characteristics of the system user requests and to tailor service provision to the actual user needs, she initially focused her attention on the subset of itemsets that are characterized by the schema {user=value, service=value}. Columns 1 and 2 in Table 4 report the itemsets

Table 4: Generalized itemsets with schemata {user=value, service=value} and {time=value}. min_card=max_card=20.

{user=value, service=value}		**{time=value}**	
Itemset	**support (%)**	**Itemset**	**support (%)**
{(user,guest), (service,post)}	6.27%	{(time,14PM-15PM)}	16.96%
{(user,guest), (service,photo)}	0.17%	{(time,09AM-10AM)}	13.03%
{(user,carmen), (service,file)}	9.94%	{(time,18PM-19PM)}	12.53%
{(user,carmen), (service,photo)}	1.34%	{(time,16PM-17PM)}	11.95%
{(user,carmen), (service,post)}	0.17%	{(time,12PM-13PM)}	8.44%
{(user,anna), (service,photo)}	1.84%	{(time,11AM-12AM)}	8.27%
{(user,anna), (service,post)}	0.67%	{(time,20PM-21PM)}	7.18%
{(user,anna), (service,file)}	0.50%	{(time,22PM-23PM)}	4.43%
{(user,cristina), (service,photo)}	2.59%	{(time,01AM-02AM)}	4.09%
{(user,cristina), (service,file)}	0.33%	{(time,05AM-06AM)}	3.84%
{(user,nicoletta), (service,photo)}	0.50%	{(time,03AM-04AM)}	3.59%
		{(time,07AM-08AM)}	3.26%
		{(time,23PM-00AM)}	2.42%

and their corresponding support values and were extracted from TeamLife by setting max_card and min_card to 20. The expert selected a relatively low value for max_card and min_card because she usually prefers to address compact models (i.e., small sets of itemsets), from which fruitful knowledge can be inferred through manual inspection. Note that the number of extracted patterns remains constant when setting max_card=min_card in the range of [19, 26]. Moreover, when setting higher threshold values (e.g., in the range of [27, 40]), the model size increases slightly (up to 44 itemsets), whereas the model size slightly reduces when setting the thresholds to values in [10, 18] (approximately 10 itemsets). Hence, the obtained results are suitable for manual inspection for a relatively large set of cardinality-based constraint values.

Two of the mined itemsets reported in Column 1 of Table 4 are related to the user *guest*, whereas all of the others are related to female users (i.e., Carmen, Anna, Cristina, and Nicoletta). The extracted itemsets can be deemed relevant by domain experts for performing targeted advertising. For example, based on the collected results, the analyst determines that user Nicoletta exclusively requests a photo service. Hence, a personalized advertisement can recommend to her correlated services (e.g., a post service) for cross-selling purposes. Because itemsets that are related to male users never occur in the mined set, it turns out that the number of male users who request system services can be so high as to successfully tailor personalized advertisements to each of them. Similarly, the number of frequent patterns with schema {user=value, service=value} concerning male users exceeds the analyst-provided threshold max_card=10 for every considered level of abstraction. Thus, the corresponding pattern sets are not generated because they are deemed to be hardly manageable through manual inspection.

Now consider the 1-itemsets with schema {time=value} reported in Table 4. These patterns can be deemed to be useful for profiling system usage and scheduling bandwidth allocation. Columns 3 and 4 in Table 4 report the corresponding itemsets that are mined from TeamLife. All of the selected patterns refer to 2-hour time slots. The compactness of the model generated by CarGeMi allows the analyst to determine the time slots at which the system is mostly used and to shape the system resources accordingly. In contrast, traditional generalized itemset miners, which do not enforce any cardinality-based constraint, generate a larger number of patterns, among which the most significant and promptly usable patterns remain hidden. For example, the support of two of the itemsets that are reported in Table 4 is 0.17%. These two itemsets can be mined by Cumulate and GenIO only by enforcing a minimum support threshold that is equal to or less than 0.17%.

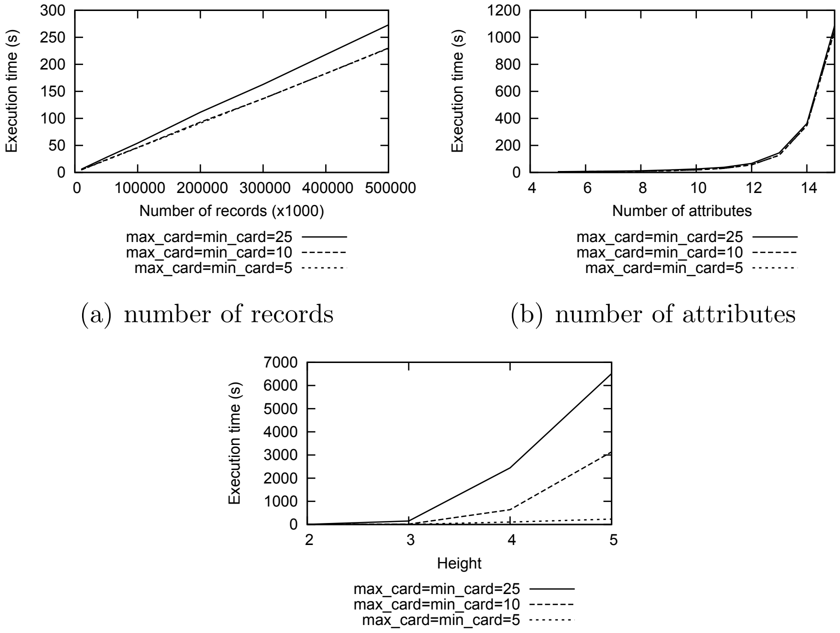
However, by enforcing min_sup=0.17%, Cumulate and GenIO mine 8,656 and 7,777 itemsets, respectively. These larger sets of itemsets can cause confusion rather than useful knowledge for the domain expert. However, by enforcing higher support threshold values, many obvious high-level or less interesting itemsets are mined by Cumulate and GenIO. Indeed, the actionability of their mined patterns remains limited.

6.4. Scalability

We evaluated the scalability of the proposed algorithm on synthetic datasets with (i) the number of records, (ii) the number of attributes, and (iii) the taxonomy height (i.e., the number of generalization levels).

Figure 4(a) reports the CarGeMi execution time by varying the number of records from 10,000 to 500,000 on datasets that were characterized by 5 attributes and by exploiting taxonomies with 2 levels of generalization. Three representative constraint settings are considered. Similar to previous Apriori-based algorithms, CarGeMi scales roughly linearly with the number of records. Curves with slightly different slopes were obtained by using different constraint settings. As already discussed in Section 6.1, the CarGeMi execution time increases when higher max_card values are enforced because of the lower pruning selectivity of the descendant cardinality-based constraint. In spite of the fact that CarGeMi does not enforce any support constraint in the itemset mining process, it appears to be able to efficiently cope with quite large and complex datasets (e.g., the execution time is approximately 270 seconds with min_card = max_card = 10).

We also analyzed the impact of the number of dataset attributes and the taxonomy height on the CarGeMi execution time. The scalability of



(a) number of records

(b) number of attributes

(c) taxonomy height

Figure 4: CarGeMi scalability analysis

CarGeMi with the number of attributes was performed on datasets characterized by 10,000 records, taxonomies with 2 levels, and the number of attributes in the range of 4 to 15. A set of datasets with 10,000 records, 5 attributes, and a taxonomy height varying from 2 to 5 have been used to analyze the CarGeMi scalability with the taxonomy height. The results, which are reported in Figures 4(b)-4(c), show that, similar to previous level-wise approaches (e.g., [7, 3]), CarGeMi scales super-linearly with both the number of attributes and the taxonomy height because of the combinatorial increase in the number of generated combinations.

7. Conclusions and future work

This paper presents a novel approach to discovering generalized itemsets with constraints from data supplied with taxonomies. Specifically, this work proposes two novel constraints that allow generating concise itemset-based models, in which high-level itemsets are used to represent large and, thus, not easily manageable descendant sets. Furthermore, the paper also proposes a novel algorithm in which the newly proposed constraints are incorporated into the mining process.

The effectiveness of the proposed approach has been evaluated on real datasets that were acquired from the mobile context, whereas the algorithm performance and scalability have been evaluated on benchmark and synthetic datasets.

The proposed approach relies on analyst-provided taxonomies. For future research, we plan to extend this work by integrating automatic taxonomy inference procedures. Furthermore, to analyze sequential data from different viewpoints, we will investigate the extension of the proposed approach to temporal generalized sequence mining [15, 16, 25]. Last but not least, an interesting future research direction will be the application of the proposed approch to role mining [17, 28]. In such context, control data can be analyzed at different levels of abstraction to elicit a set of meaningful roles that simplify access management. For example, user permissions in an enterprise can be aggregated into higher level categories and analyzed at different granularity levels in order to achieve optimal security administration based on the role that each individual plays within the organization.

References

[1] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, ACM Press, 1993, pp. 207–216.

[2] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of the 20th VLDB conference, ACM Press, 1994, pp. 487–499.

[3] R. Agrawal, R. Srikant, Mining generalized association rules, in: VLDB 1995, Morgan Kaufmann, 1995, pp. 407–419.

[4] R. Agrawal, R. Srikant, Mining association rules with item constraints, in: KDD 1997, ACM Press, 1997, pp. 67–73.

[5] S. Ayubi, M.K. Muyeba, A. Baraani, J. Keane, An algorithm to mine general association rules from tabular data, Information Sciences 179 (2009) 3520 – 3539.

[6] M. Baldi, E. Baralis, F. Risso, Data mining techniques for effective and scalable traffic analysis, in: International Symposium on Integrated Network Management, IEEE Computer Society, 2005, pp. 105–118.

[7] E. Baralis, L. Cagliero, T. Cerquitelli, V. D'Elia, P. Garza, Support driven opportunistic aggregation for generalized itemset extraction, in: 5th IEEE International Conference on Intelligent Systems, IEEE Computer Society, 2010, pp. 102–107.

[8] E. Baralis, L. Cagliero, T. Cerquitelli, P. Garza, Generalized association rule mining with constraints, Information Sciences 194 (2012) 68–84.

[9] E. Baralis, L. Cagliero, T. Cerquitelli, P. Garza, M. Marchetti, CAS-MINE: Providing personalized services in context-aware applications by means of generalized rules, Knowl. Inf. Syst. (2010) 283–310.

[10] M. Barsky, S. Kim, T. Weninger, J. Han, Mining flipping correlations from large datasets with taxonomies, Proc. VLDB Endow. 5 (2011) 370–381.

[11] R.J. Bayardo, Efficiently mining long patterns from databases, in: L.M. Haas, A. Tiwary (Eds.), SIGMOD 1998, ACM Press, 1998, pp. 85–93.

[12] C. Blake, C. Merz, Uci repository of machine learning databases, 2012. Available at http://archive.ics.uci.edu/ml. Last accessed: 20/07/2012.

[13] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: Generalizing association rules to correlations, in: SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, ACM Press, 1997, pp. 265–276.

[14] L. Cagliero, Discovering temporal change patterns in the presence of taxonomies, IEEE Transactions on Knowledge and Data Engineering 99 (2011).

[15] E. Chen, H. Cao, Q. Li, T. Qian, Efficient strategies for tough aggregate constraint-based sequential pattern mining, Information Sciences 178 (2008) 1498 – 1518.

[16] Y.L. Chen, S.Y. Wu, Y.C. Wang, Discovering multi-label temporal patterns in sequence databases, Information Sciences 181 (2011) 398 – 418.

[17] A. Colantonio, R. Di Pietro, A. Ocello, N.V. Verde, A formal framework to elicit roles with business meaning in RBAC systems, in: Proceedings of the 14^{th} ACM Symposium on Access Control Models and Technologies, SACMAT '09, pp. 85–94.

[18] B. Crémilleux, A. Soulet, Discovering knowledge from local patterns with global constraints, in: Computational Science and Its Applications - ICCSA 2008, International Conference, Springer, 2008, pp. 1242–1257.

[19] DBDMG, Database and Data Mining Web Site, Politecnico di Torino, 2012. Available at http://dbdmg.polito.it/wordpress/research/cardinality-based-generalized-itemset-miner. Last accessed: 15/12/2012.

[20] J. Han, Y. Fu, Mining multiple-level association rules in large databases, IEEE Transactions on knowledge and data engireering 11 (1999) 798–805.

[21] P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies, Chapman & Hall/CRC, 2009.

[22] IBM, IBM Quest Synthetic Data Generation Code, 2009.

[23] S. Jaroszewicz, D.A. Simovici, Interestingness of frequent itemsets using bayesian networks as background knowledge, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press, 2004, pp. 178–186.

[24] D. Kunkle, D. Zhang, G. Cooperman, Mining frequent generalized itemsets and generalized association rules without redundancy, J. Comput. Sci. Technol. 23 (2008) 77–102.

[25] Y. Li, S. Zhu, X.S. Wang, S. Jajodia, Looking into the seeds of time: Discovering temporal patterns in large transaction sets, Information Sciences 176 (2006) 1003 – 1031.

[26] M. Mampaey, N. Tatti, J. Vreeken, Tell me what I need to know: succinctly summarizing data with itemsets, in: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '11, ACM Press, 2011, pp. 573–581.

[27] G. Mansingh, K.M. Osei-Bryson, H. Reichgelt, Using ontologies to facilitate post-processing of association rules by domain experts, Information Sciences 181 (2011) 419 – 434.

[28] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. Calo, J. Lobo, Mining roles with multiple objectives, ACM Transactions on Information and System Security (TISSEC) 13 (2010) 1–35.

[29] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, in: Proceedings of the 7th International Conference on Database Theory, ICDT '99, Springer-Verlag, 1999, pp. 398–416.

[30] K. Sriphaew, T. Theeramunkong, A new method for finding generalized frequent itemsets in association rule mining, in: Proceeding of the VII

International Symposium on Computers and Communications, IEEE Computer Society, 2002, pp. 1040–1045.

[31] K. Sriphaew, T. Theeramunkong, Fast algorithms for mining generalized frequent patterns of generalized association rules., IEICE Transactions on Information and Systems 87 (2004) 761–770.

[32] P.N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.

[33] N. Tatti, Probably the best itemsets, in: Proceedings of the 16th ACM International Conference on Knowledge discovery and data mining, ACM Press, 2010, pp. 293–302.

[34] S. Tseng, C. Tsui, Mining multilevel and location-aware service patterns in mobile web environments, IEEE Transactions on Systems, Man, and Cybernetics 34 (2004) 2480–2485.

[35] T. Uno, M. Kiyomi, H. Arimura, Lcm ver.3: collaboration of array, bitmap and prefix tree for frequent itemset mining, in: Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations, OSDM '05, ACM Press, 2005, pp. 77–86.

[36] M.J. Zaki, C.J. Hsiao, Efficient algorithms for mining closed itemsets and their lattice structure, IEEE Trans. Knowl. Data Eng. 17 (2005) 462–478.