Future Generation Computer Systems ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Mining constrained frequent itemsets from distributed uncertain data

Alfredo Cuzzocrea [a], Carson Kai-Sang Leung [b,*], Richard Kyle MacKinnon [b]

[a] ICAR-CNR and University of Calabria, Via P. Bucci, 41C, I-87036 Rende (CS), Italy

[b] Department of Computer Science, University of Manitoba, Winnipeg, MB R3T 2N2, Canada

HIGHLIGHTS

- We proposed a system for tree-based distributed uncertain frequent itemset mining.
- Our system allows users to specify constraints for expressing their interests.
- It finds frequent itemsets that satisfy succinct constraints from distributed uncertain data.
- It also handles non-succinct (e.g., inductive succinct, anti-monotone) constraints.

ARTICLE INFO

Article history:
Received 15 May 2012
Received in revised form
16 October 2013
Accepted 26 October 2013
Available online xxxx

Keywords:
Data mining
Frequent pattern mining
Advanced data-intensive computing algorithms
Constraints
Distributed computing

ABSTRACT

Nowadays, high volumes of massive data can be generated from various sources (e.g., sensor data from environmental surveillance). Many existing distributed frequent itemset mining algorithms do not allow users to express the itemsets to be mined according to their intention via the use of constraints. Consequently, these unconstrained mining algorithms can yield numerous itemsets that are not interesting to users. Moreover, due to inherited measurement inaccuracies and/or network latencies, the data are often riddled with uncertainty. These call for both *constrained mining* and *uncertain data mining*. In this journal article, we propose a data-intensive computer system for tree-based mining of frequent itemsets that satisfy user-defined constraints from a distributed environment such as a wireless sensor network of uncertain data.

1. Introduction

Data mining aims to discover implicit, previously unknown, and potentially useful information that is embedded in data. As a common data mining task, *frequent itemset mining* [1–3] looks for *itemsets* (i.e., sets of items) that are frequently co-occurring together. The mined frequent itemsets can be used in the discovery of correlation or causal relations, analysis of sequences, and formation of association rules.

Since its introduction [4], the research problem of finding frequent itemsets has been the subject of numerous studies. In early days, many algorithms were Apriori-based [5], which depends on a generate-and-test paradigm to find all frequent itemsets by first generating candidates and then checking their support (i.e., their occurrences) against the traditional databases containing precise data. To avoid the generate-and-test paradigm, the FP-growth algorithm [6] was proposed. Such a tree-based algorithm constructs an extended prefix-tree structure, called Frequent Pattern tree (FP-tree), to capture the contents of the transaction database. Rather than employing the generate-and-test strategy of Apriori-based algorithms, FP-growth focuses on frequent pattern growth [7,8]—which is a restricted test-only approach (i.e., does not generate candidates, and only tests for support).

Data in many real-life applications are riddled with uncertainty [9–11]. It is partially due to inherent measurement inaccuracies, sampling and duration errors, network latencies, and intentional blurring of data to preserve anonymity. As such, the presence or absence of items in a dataset is uncertain. Moreover, with the increasing number of uncertain objects for sensor devices and noisy data management technologies such as DUST [12] in recent years, *uncertain data mining* [13–15] is in demand. For example, a physician may highly suspect (but not guarantee) that a patient suffers from asthma. The uncertainty of such suspicion can be expressed in terms of *existential probability* of an item in a probabilistic dataset. To mine frequent itemsets from these uncertain

* Corresponding author. Tel.: +1 204 474 8677.
E-mail addresses: cuzzocrea@si.deis.unical.it (A. Cuzzocrea), kleung@cs.umanitoba.ca (C.K.-S. Leung), ummack57@cs.umanitoba.ca (R.K. MacKinnon).
URL: http://www.cs.umanitoba.ca/~kleung (C.K.-S. Leung).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.10.026

data, the U-Apriori [16] and UF-growth [17] algorithms were proposed. UF-growth captures the contents of the uncertain data in a UF-tree, from which frequent itemsets can be mined recursively.

Regardless whether they are Apriori-based or tree-based, many frequent itemset mining algorithms provide little or no support for user focus when mining precise or uncertain data. However, in many real-life applications, the user may have some particular phenomena in mind on which to focus the mining (e.g., medical analysts may want to find only those lab test records belonging to patients suspected to suffer from asthma instead of all the patients). Without user focus, the user often needs to wait for a long period of time for numerous frequent itemsets, out of which only a tiny fraction may be interesting to the user. Hence, *constrained frequent itemset mining* [18,19], which aims to find those frequent itemsets that satisfy the user-defined constraints, is needed. CAP [20], DCF [21], and $\mathcal{FIC}$ [22] are examples of algorithms that mine constrained frequent itemsets from traditional precise data.

As technology advances, one can easily collect high volumes of massive data [23,24] from not only a single source but multiple sources. For example, in recent years, sensor networks have been widely used in many application areas such as agricultural, architectural, environmental, and structural surveillance. Sensors distributed in these networks serve as good sources of data. However, sensors usually have limited communication bandwidth, transmission energy, and computational power. Thus, data are not usually transmitted to a single distant centralized processor to perform the data mining task. Instead, data are transmitted to their local (e.g., closest) processors within a distributed environment [25]. As this requires massive computing power, this calls for *parallel and distributed mining* [26–28]. For example, *parallel and distributed frequent itemset mining* [29,30] searches for implicit, previously unknown, and potentially useful frequent itemsets that might be embedded in the distributed data.

Some examples of Apriori-based distributed algorithms that find frequent itemsets in a distributed environment include Count Distribution, Data Distribution and Candidate Distribution [31], as well as FDM [32]. Similarly, Parallel-HFP-Leap [33] also finds frequent itemsets in a distributed environment, but it is a tree-based algorithm. However, regardless of whether they are Apriori-based or tree-based, all these distributed frequent itemset mining algorithms do not handle constraints nor do they mine uncertain data. On the other hand, CAP, DCF and $\mathcal{FIC}$ all find *constrained* frequent itemsets, but they mine a centralized database of precise data. Similarly, the U-Apriori and UF-growth algorithms both mine a centralized database of *uncertain* data for all (unconstrained) frequent itemsets instead of only those constrained ones. Recently, Kozawa et al. [34] used general-purpose computation on GPU in an attempt to accelerate uncertain data mining. However, they aimed to find all (unconstrained) *probabilistic-frequent* itemsets instead of constrained *frequent* ones. In other words, these existing mining algorithms fall short in different aspects. Hence, a natural question to ask is: Is it possible to mine *uncertain* data for only those frequent itemsets that satisfy user-defined *constraints* in a *distributed* environment? In response to this question, we conducted a feasibility study. Its preliminary results [35,36] show the possibility of mining constrained frequent itemsets from distributed uncertain data. In the current journal article, we propose a computer system for tree-based mining of uncertain data in a distributed environment for frequent itemsets that satisfy user-defined constraints. Here, our *key contribution* is the non-trivial integration of (i) constrained mining, (ii) parallel and distributed mining, (iii) uncertain data mining, (iv) tree-based mining and (v) frequent itemset mining. The resulting system uses a tree-based approach to efficiently mine from distributed uncertain data for only those constrained frequent itemsets. It avoids the candidate generate-and-test paradigm, handles uncertain data, pushes user constraints inside the mining process, avoids unnecessary computation, and finds only those itemsets satisfying the constraints in a distributed environment.

This journal article is organized as follows. The next section gives some background information on constrained frequent itemset mining from uncertain data. In Sections 3 and 4, we propose our non-trivial integration of tree-based frequent itemset mining, constrained mining, distributed mining, and uncertain mining. More specifically, we show in Section 3 how our proposed system finds frequent itemsets that satisfy user-defined succinct constraints. We then show in Section 4 how our proposed system handles non-succinct constraints. Experimental results are shown in Section 5. Finally, Section 6 presents the conclusions.

2. Background

In this section, we provide some background about (i) uncertain data mining and (ii) constrained mining.

2.1. Mining frequent itemsets from uncertain data

The research problem of frequent itemset mining was first introduced [4] in 1993. The corresponding algorithm – namely, Apriori – mined all frequent itemsets from a transaction database (TDB) consisting of *precise data*, in which the contents of each transaction are precisely known. Specifically, if a transaction t_i contains an item x (i.e., $x \in t_i$), then x is precisely known to be present in t_i. On the other hand, if a transaction t_i does not contain an item y (i.e., $y \notin t_i$), then y is precisely known to be absent from t_i. However, this is not the case for probabilistic databases consisting of uncertain data. A key difference between precise and uncertain data is that each transaction of the latter contains items and their *existential probabilities*. The existential probability $P(x, t_i)$ of an item x in a transaction t_i indicates the likelihood of x being present in t_i. For a real-life example, each transaction t_i represents a patient's visit to a physician's office. Each item x within t_i represents a potential disease, and is associated with $P(x, t_i)$ expressing the likelihood of a patient having that disease x in t_i (say, in t_1, the patient has a 60% likelihood of having asthma, and a 90% likelihood of catching a cold regardless of having asthma or not). With this notion, each item in a transaction t_i in datasets of precise data can be viewed as an item with a 100% likelihood of being present in t_i.

Given an item x and a transaction t_i, there are two possible worlds when using the *"possible world" interpretation* of uncertain data [37,38]:

(i) the possible world W_1 where $x \in t_i$, and
(ii) the possible world W_2 where $x \notin t_i$.

Although it is uncertain which of these two worlds is the *true world*, the probability of W_1 being the true world is $P(x, t_i)$ and that of W_2 is $1 - P(x, t_i)$.

In real-life applications, there are generally many independent items in each of the n transactions in a TDB (where $|\text{TDB}| = n$). Hence, the *expected support* of an itemset X in the TDB can be computed by summing the support of X in possible world W_j (while taking in account the probability of W_j to be the true world) over all possible worlds. When items within X are independent, such a sum can be simplified to become the sum of product of existential probabilities of $x \in X$, as follows [38]:

$$expSup(X) = \sum_{i=1}^{n} \left(\prod_{x \in X} P(x, t_i) \right). \quad (1)$$

An itemset X is *frequent* if its expected support meets or exceeds the user-specified threshold *minsup*.

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026

2.2. Mining frequent itemsets that satisfy user constraints

An existing constrained frequent itemset mining framework [20,21] allows the user to use a rich set of SQL-style constraints to specify his interest for guiding the mining process so that only those frequently occurring sets of market basket items that satisfy the user constraints are found. This avoids unnecessary computation for mining those uninteresting frequent itemsets. Besides market basket items, the set of constraints can also be imposed on items, events or objects in other domains. The following are some examples of user constraints. Constraint $C_1 \equiv max(X.Price) \leq \$25$ expresses the user interest in finding every frequent itemset X such that the maximum price of all market basket items in each X is at most \$25. Similarly, $C_2 \equiv min(X.Price) \leq \$30$ says that the minimum price of all items in an itemset X is at most \$30. For domains other than the market basket, constraint $C_3 \equiv X.Location = Winnipeg$ expresses the user interest in finding every frequent itemset X such that all events in X are held in Winnipeg, MB, Canada. Constraints $C_4 \equiv X.Symptom \supseteq \{dry\ throat,\ sneezing\}$ says that each individual in X suffers from at least dry throat and sneezing, $C_5 \equiv X.Weight \geq 32$ kg says that the weight of each object in X is at least 32 kg, and $C_6 \equiv avg(X.Rainfall) \leq 10$ mm says that the average rainfall of all selected meteorological records in X is at most 10 mm. Similarly, $C_7 \equiv sum(X.Rainfall) \leq 150$ mm says that the total rainfall of all selected meteorological records in X is at most 150 mm, and $C_8 \equiv sum(X.Rainfall) \geq 90$ mm says that the total rainfall of all selected meteorological records in X is at least 90 mm.

The above constraints can be categorized into several overlapping classes according to the properties that they possess. One of these properties is *succinctness*.

Definition 1 (*Succinctness [19]*)**.** Let `Item` be the set of domain items. Then, an itemset $SS_j \subseteq$ `Item` is a *succinct set* if SS_j can be expressed as a result of selection operation σ_p (`Item`), where σ is the usual SQL-style selection operator and p is a selection predicate. A powerset of items $SP \subseteq 2^{\texttt{Item}}$ is a *succinct powerset* if there is a fixed number of succinct sets $SS_1, \ldots, SS_k \subseteq$ `Item` such that SP can be expressed in terms of the powersets of $SS_1, \ldots, SS_k$ using set union and/or set difference operators. A constraint C is *succinct* provided that the set of itemsets satisfying C is a succinct powerset. □

It is important to note the following two observations about succinct constraints.

Observation 1. A majority of user-defined constraints are succinct. □

Among the aforementioned constraints, the first five (i.e., C_1, C_2, C_3, C_4 & C_5) are *succinct*. For any *succinct* constraints, one can directly generate precisely all and only those itemsets satisfying the constraints without generating and excluding itemsets not satisfying the constraints. Hence, one can use member generating functions [21] to precisely generate constrained itemsets. For instance, $C_1 \equiv max(X.Price) \leq \$25$ is succinct because any itemset satisfying C_1 can be expressed as a member in the succinct powerset $2^{\sigma_{Price \leq \$25}(\texttt{Item})}$. In other words, itemsets satisfying C_1 can be precisely generated by combining any market basket items having price $\leq \$25$, thereby avoiding the substantial overhead of the generation and exclusion of invalid itemsets. Similarly, itemsets satisfying $C_2 \equiv min(X.Price) \leq \$30$ can be precisely generated by combining at least one market basket item having price $\leq \$30$ with some optional items (of any price values).

Observation 2. Many non-succinct constraints can be induced into weaker constraints that are succinct. □

As an example, non-succinct constraint $C_6 \equiv avg(X.Rainfall) \leq 10$ mm can be induced into a succinct constraint $C_6' \equiv min(X.Rainfall) \leq 10$ mm as all frequent itemsets satisfying C_6 must satisfy C_6'.

Besides succinctness, there are some other properties possessed by constraints. One of them is *anti-monotonicity*.

Definition 2 (*Anti-monotonicity [19]*)**.** A constraint C is *anti-monotone* if and only if all subsets of an itemset satisfying C also satisfy C. □

With this additional property (i.e., anti-monotonicity), succinct constraints can be further divided into two subclasses:

(i) *succinct anti-monotone* (*SAM*) *constraints*, and
(ii) *succinct non-anti-monotone* (*SUC*) *constraints*.

Among the eight aforementioned constraints, three of them – namely, C_1, C_3 and C_5 – are SAM constraints. For instance, for any itemset X satisfies $C_1 \equiv max(X.Price) \leq \$25$, subsets of X formed by removing items (having either the maximum price or not) from X would not possess a higher maximum price (i.e., maximum price of all market basket items in these subsets $\leq \$25$). Note that supersets of any itemset violating the SAM constraints also violate the constraints (e.g., if an itemset X contains an item having price $> \$25$, then X violates C_1 and so does every superset of X).

In contrast, C_2 and C_4 are *SUC constraints* because they do not possess such an anti-monotonicity property. For instance, if the minimum price of all items contained within X is higher than \$30, then X violates C_2 but there is no guarantee that all supersets of X would violate C_2. As an example, let $y.Price$ be \$50 and $z.Price$ be \$10. Then, $X \cup \{y\}$ and $X \cup \{z\}$ are both supersets of X. Among them, the former still violates C_2 but the latter satisfies C_2.

3. Our proposed distributed mining system

Without loss of generality, we assume to have p sites/processors and $m = m_1 + m_2 + \cdots + m_p$ sensors in a distributed network such that m_1 wireless sensors transmit data to their closest or designated site/processor P_1, m_2 sensors transmit data to the site/processor P_2, and so on. With this setting, our proposed system finds constrained itemsets that are frequent in the entire wireless sensor network. Depending on the properties of constraints (i.e., whether the constraints are succinct or not), different procedures are carried out.

In this section, we show how our proposed system discovers frequent itemsets that satisfy *succinct* constraints. In particular, we show how the system finds (i) itemsets that satisfy succinct constraints and are locally frequent with respect to site/processor P_i (in Section 3.1) and then finds (ii) those that satisfy succinct constraints and are globally frequent with respect to all sites/processors in the entire wireless sensor network (in Section 3.2). In Section 4, we will show how our proposed system discovers frequent itemsets that do *not* satisfy succinct constraints.

3.1. Finding locally frequent itemsets that satisfy succinct constraints

Given m_i sensors transmitting data to the processor P_i, a local database TDB_i of uncertain data can be created for P_i. We aim to find itemsets that are both (i) frequent to P_i and (ii) satisfying a succinct (SAM or SUC) constraint C. For uncertain data, we use the "possible world" interpretation of uncertain data. We find constrained locally frequent itemsets from uncertain data in the following steps.

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026

3.1.1. Step 1: identification of items satisfying the constraints

Let $\texttt{Item}^{\texttt{M}}$ be the collection of mandatory items—i.e., the collection of domain items that individually satisfy the SAM or SUC constraint C; let $\texttt{Item}^{\texttt{O}}$ be the collection of optional items—i.e., the collection of domain items that individually violate C.

Then, for any SAM constraint C_{SAM}, an itemset X satisfying C_{SAM} cannot contain any item from $\texttt{Item}^{\texttt{O}}$ due to the anti-monotonicity property. So, any itemset X satisfying C_{SAM} must consist of *only* items that individually satisfy C_{SAM}. In other words, any itemset X satisfying C_{SAM} must be generated by combining items from $\texttt{Item}^{\texttt{M}}$ (i.e., $X \subseteq \texttt{Item}^{\texttt{M}}$). Due to the succinctness property, items in $\texttt{Item}^{\texttt{M}}$ can be efficiently enumerated (from the list of domain items) by selecting only those items that individually satisfy C_{SAM}. See Example 1.

Example 1. Consider the following transaction database TDB consisting of uncertain data:

Transactions	Contents
t_1	{a:0.7, b:0.8, c:0.8, d:1.0, e:0.2}
t_2	{a:0.7, b:0.8, d:1.0, e:0.1, f:0.4}
t_3	{a:0.8, c:0.5, e:0.3, f:0.4}
t_4	{b:0.8, c:0.8, d:1.0}
t_5	{c:0.8, d:1.0}

with the following auxiliary information:

Items	Price
a	\$10
b	\$20
c	\$100
d	\$50
e	\$75
f	\$25

In the above TDB of uncertain data, each transaction contains items and their corresponding existential probabilities. For example, there are five domain items a, b, c, d and e in the first transaction t_1, where the existential probabilities of these items are 0.7, 0.8, 0.8, 1.0 and 0.2 respectively. Note that (i) different items may have the same existential probabilities (e.g., the existential probabilities of two different items b and c in t_1 have the same value 0.8) but (ii) the existential probabilities of the same item may vary from one transaction to another (e.g., the existential probability of item e is 0.2 in transaction t_1 but it is 0.1 in t_2). Let constraint C_{SAM} be the SAM constraint $C_1 \equiv max(X.Price) \leq \$25$. Our proposed system checks each of the six domain items against the constraint C_{SAM}. It first enumerates the valid items a, b and f (i.e., items with individual price $\leq$ \$25). So, $\texttt{Item}^{\texttt{M}} = \{a, b, f\}$. Once we have identified the domain items that satisfy the succinct anti-monotone constraint C_{SAM}, these items serve as building blocks for all constrained frequent itemsets satisfying C_{SAM} because all constrained frequent itemsets must comprise only those $\texttt{Item}^{\texttt{M}}$ items. □

Next, for any SUC constraint C_{SUC}, any itemset X satisfying C_{SUC} is composed of mandatory items (i.e., items that individually satisfy C_{SUC}) and possibly some optional items (regardless of whether or not they satisfy C_{SUC}). Note that, although C_{SUC} possesses the succinctness property (i.e., one can easily enumerate all and only those itemsets that are guaranteed to satisfy C_{SUC}), it does not possess the anti-monotonicity property. So, if an itemset violates C_{SUC}, there is no guarantee that all or any of its supersets would violate C_{SUC}. Hence, not all itemsets satisfying C_{SUC} are composed of only domain items that individually satisfy the constraints (as for SAM constraints). Instead, any itemset X satisfying C_{SUC} must be generated by combining at least one $\texttt{Item}^{\texttt{M}}$ item and possibly some $\texttt{Item}^{\texttt{O}}$ items. Due to succinctness, items in $\texttt{Item}^{\texttt{M}}$ and in $\texttt{Item}^{\texttt{O}}$ can be efficiently enumerated. See Example 2.

Example 2. Consider the same TDB and auxiliary information in Example 1. Let constraint C_{SUC} be the SUC constraint $C_2 \equiv min(X.Price) \leq \$30$. Our proposed system checks each of the six domain items against C_{SUC}. It first enumerates the valid items a, b and f (i.e., items with individual price $\leq$ \$30), giving $\texttt{Item}^{\texttt{M}} = \{a, b, f\}$. The remaining domain items then belong to $\texttt{Item}^{\texttt{O}}$ (i.e., items with individual price $>$ \$30). Once we have classified the domain items into (i) the $\texttt{Item}^{\texttt{M}}$ items (which satisfy C_{SUC}) and (ii) the $\texttt{Item}^{\texttt{O}}$ items (which violate C_{SUC}), all these items serve as building blocks for all constrained frequent itemsets satisfying C_{SUC} because all constrained frequent itemsets must comprise at least one $\texttt{Item}^{\texttt{M}}$ item and may contain some additional $\texttt{Item}^{\texttt{M}}$ or $\texttt{Item}^{\texttt{O}}$ items. □

3.1.2. Step 2: construction of a UF-tree

Once the domain items are classified into $\texttt{Item}^{\texttt{M}}$ and $\texttt{Item}^{\texttt{O}}$ items (no $\texttt{Item}^{\texttt{O}}$ items for C_{SAM}), our system then constructs a UF-tree, which is built in preparation for mining constrained frequent itemsets from uncertain data. It does so by first scanning the TDB of uncertain data once. It accumulates the expected support of each of the items in order to find all *frequent* domain items. Among these items, the system discards those infrequent ones and only captures those frequent ones in the UF-tree. Note that any infrequent $\texttt{Item}^{\texttt{M}}$ or $\texttt{Item}^{\texttt{O}}$ items can be safely discarded because any itemset containing an infrequent item is also infrequent.

Once the frequent $\texttt{Item}^{\texttt{M}}$ and $\texttt{Item}^{\texttt{O}}$ items are found, our system arranges these two kinds of items in such a way that $\texttt{Item}^{\texttt{M}}$ items appear *below* $\texttt{Item}^{\texttt{O}}$ items (i.e., $\texttt{Item}^{\texttt{M}}$ items are closer to the leaves, and $\texttt{Item}^{\texttt{O}}$ items are closer to the root). Among all the items in $\texttt{Item}^{\texttt{M}}$, they are sorted in non-ascending order of accumulated expected support. Similarly, among all the items in $\texttt{Item}^{\texttt{O}}$, they are also sorted in non-ascending order of accumulated expected support. The system then scans the TDB the second time and inserts each transaction of the TDB into the UF-tree. Here, the new transaction is merged with a child (or descendant) node of the root of the UF-tree (at the highest support level) only if the same item *and the same expected support* exist in both the transaction and the child (or descendant) nodes.

For SAM constraints, the corresponding UF-tree captures only those frequent $\texttt{Item}^{\texttt{M}}$ items; for SUC constraints, the corresponding UF-tree captures both the frequent $\texttt{Item}^{\texttt{M}}$ items and the frequent $\texttt{Item}^{\texttt{O}}$ items. With such a tree construction process, the UF-tree possesses the property that *the occurrence count of a node is at least the sum of occurrence counts of all its child nodes*. See Example 3.

Example 3. Let us revisit Example 2, and let the user-specified support threshold *minsup* be set to 1.0. Our system builds the UF-tree that captures the frequent items satisfying the SUC constraint C_2 as follows. First, the system scans the TDB once and accumulates the expected support of each $\texttt{Item}^{\texttt{M}}$ item as well as each $\texttt{Item}^{\texttt{O}}$ item. Hence, it finds all frequent $\texttt{Item}^{\texttt{M}}$ items and sorts them in descending order of (accumulated) expected support. It also finds all frequent $\texttt{Item}^{\texttt{O}}$ items and sorts them in descending order of (accumulated) expected support. Among the two kinds of items, $\texttt{Item}^{\texttt{O}}$ are arranged on top (near the root) of $\texttt{Item}^{\texttt{M}}$ items (which are near the leaves). Specifically, our system obtains $\texttt{Item}^{\texttt{O}}$ items d, c and e (with their corresponding accumulated expected support values of 4.0, 2.9 and 0.6), which are sorted in descending order of their expected support values. Among these $\texttt{Item}^{\texttt{O}}$ items, e (having accumulated expected support of $0.6 <$ *minsup*) is removed. Then, the system represents the frequent $\texttt{Item}^{\texttt{O}}$ items and their expected support as d:4.0 and c:2.9. The expected support of each of these frequent $\texttt{Item}^{\texttt{O}}$ items $\geq$ *minsup*. Similarly, the system also obtains $\texttt{Item}^{\texttt{M}}$ items b, a and f (with their corresponding accumulated expected support values of 2.4, 2.2 and

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026

0.8), which are also sorted in descending order of their expected support values. Among these $\texttt{Item}^{\texttt{M}}$ items, f (having accumulated expected support of $0.8 < minsup$) is removed. Then, the system represents the frequent $\texttt{Item}^{\texttt{M}}$ items and their expected support as b:2.4 and a:2.2. The expected support of each of these frequent $\texttt{Item}^{\texttt{M}}$ items $\geq minsup$.

Next, our system scans the TDB the second time and inserts each transaction into the UF-tree. The system first inserts frequent items from the first transaction t_1 into the tree. It then inserts the frequent items from the second transaction t_2 into the UF-tree. Since the expected support of d in t_2 is the same as that in an existing branch (i.e., the branch for t_1), this node can be shared. So, the system increments the occurrence count for the tree node (d:1.0) to 2, and adds the remainder of t_2 – namely, $\langle(b{:}0.8){:}1, (a{:}0.7){:}1\rangle$ – as a child of the node (d:1.0):2. To capture the third transaction t_3, our system inserts $\langle(c{:}0.5){:}1, (a{:}0.8){:}1\rangle$ into the tree. For the fourth transaction t_4, the system increments the occurrence count for each tree node in an existing path $\langle(d{:}1.0){:}2, (c{:}0.8){:}1, (b{:}0.8){:}1\rangle$ by 1. Finally, our system increments the occurrence counts for the tree nodes in an existing path $\langle(d{:}1.0){:}3, (c{:}0.8){:}2\rangle$ by 1 for the fifth transaction t_5. Hence, at the end of the tree construction process, we get the UF-tree shown in Fig. 1(a) capturing the contents of the TDB of uncertain data. □

3.1.3. Step 3: mining of constrained frequent itemsets from the UF-tree

Once the UF-tree is constructed with the item-ordering scheme where $\texttt{Item}^{\texttt{O}}$ items are above $\texttt{Item}^{\texttt{M}}$ items, our proposed system extracts appropriate paths to form a projected database for each $x \in \texttt{Item}^{\texttt{M}}$. The system does not need to form projected databases for any $y \in \texttt{Item}^{\texttt{O}}$ because all itemsets satisfying C_{SUC} must be "extensions" of an item from $\texttt{Item}^{\texttt{M}}$ (i.e., all valid itemsets must be grown from $\texttt{Item}^{\texttt{M}}$ items) and no $\texttt{Item}^{\texttt{O}}$ items are kept in the UF-tree for C_{SAM}.

When forming each $\{x\}$-projected database and constructing its UF-tree, our system does not need to distinguish those $\texttt{Item}^{\texttt{M}}$ items from $\texttt{Item}^{\texttt{O}}$ items in the UF-tree for the $\{x\}$-projected database. Such a distinction between two kinds of items is only needed for the UF-tree for the TDB (for SUC constraints only) because, once we found at least one valid item $x \in \texttt{Item}^{\texttt{M}}$, for any v satisfying C_{SUC},

$$v = \{x\} \cup \text{others}, \tag{2}$$

where

(i) $x \in \texttt{Item}^{\texttt{M}}$, and
(ii) $\text{others} \subseteq \left(\texttt{Item}^{\texttt{M}} \cup \texttt{Item}^{\texttt{O}} - \{x\}\right)$.

After constructing projected UF-trees for each $x \in \texttt{Item}^{\texttt{M}}$, our proposed system mines all frequent itemsets that satisfy C_{SUC} in the same manner as it mines those satisfying C_{SAM}. See Example 4.

Example 4. Let us continue with Example 3. Once the UF-tree is constructed, our proposed system recursively mines constrained locally frequent itemsets from this tree with $minsup$ = 1.0 as follows. From the header table (from top to bottom) containing two $\texttt{Item}^{\texttt{O}}$ items d:4.0 = (4 × 1.0) and c:2.9 = (1 × 0.5) + (3 × 0.8) as well as two $\texttt{Item}^{\texttt{M}}$ items b:2.4 = (1 × 0.8) + (2 × 0.8) and a:2.2 = (1 × 0.8) + (1 × 0.7) + (1 × 0.7), the system first finds two constrained frequent itemsets $\{b\}$ and $\{a\}$ with expected support values of 2.4 and 2.2 respectively.

Then, our system recursively mines constrained frequent itemsets from this UF-tree with $minsup = 1.0$ as follows. From the UF-tree shown in Fig. 1(a), our system starts with $a \in \texttt{Item}^{\texttt{M}}$ and constructs a UF-tree for the $\{a\}$-projected database. The resulting tree, as shown in Fig. 1(b), consists of a single path—namely, $\langle(d{:}1.0){:}2, (b{:}0.8){:}2\rangle$ with the expected support of $\{a\}$ equal to **0.7** (implying that d or b occurs together with a twice in the original database). The expected support values of $\{a, b\} = 2 \times \mathbf{0.7} \times 0.8 = 1.12$ and of $\{a, d\} = 2 \times \mathbf{0.7} \times 1.0 = 1.4$. Thus, both $\{a, b\}$ and $\{a, d\}$ are frequent.

The system then extracts from this single-path tree to form a UF-tree for the $\{a, b\}$-projected database. The resulting tree, as shown in Fig. 1(c), consists of a single node (d:1.0):2 with the expected support of $\{a, b\}$ equal to $\mathbf{0.56} = 0.7 \times 0.8$ (implying that $\{d\}$ occurs together with $\{a, b\}$ twice in the original database). Itemset $\{a, b, d\}$, with its expected support equals $2 \times \mathbf{0.56} \times 1.0 = 1.12$, is frequent. This marks the end of the extensions of $\{a\}$.

Then, the system considers the next item in $\texttt{Item}^{\texttt{M}}$ (i.e., b) and constructs a UF-tree for the $\{b\}$-projected database. The resulting tree, as shown in Fig. 1(d), consists of a single path—namely, $\langle(d{:}1.0){:}3, (c{:}0.8){:}2\rangle$ with the expected support of b equal to **0.8** (implying that $\{b, d\}$ occurs three times and $\{b, c\}$ occurs twice in the original database). The expected support values of $\{b, c\} = 2 \times \mathbf{0.8} \times 0.8 = 1.28$ and of $\{b, d\} = 3 \times \mathbf{0.8} \times 1.0 = 2.4$. So, they are both frequent.

The system then extracts from this single-path tree to form a UF-tree for the $\{b, c\}$-projected database. The resulting tree, as shown in Fig. 1(e), consists of a single node (d:1.0):2 with expected support of $\{b, c\}$ equal to $\mathbf{0.64} = 0.8 \times 0.8$ (implying that d occurs together with $\{b, c\}$ twice in the original database). Itemset $\{b, c, d\}$, with its expected support equal to $2 \times \mathbf{0.64} \times 1.0 = 1.28$, is frequent.

Since no more items belong to $\texttt{Item}^{\texttt{M}}$, this marks the end of the mining process. Our proposed system recursively finds the following eight locally frequent itemsets that satisfy the SUC constraint C_2 from uncertain data: $\{a\}$:2.2, $\{a, b\}$:1.12, $\{a, b, d\}$:1.12, $\{a, d\}$:1.4, $\{b\}$:2.4, $\{b, c\}$:1.28, $\{b, c, d\}$:1.28 and $\{b, d\}$:2.4. □

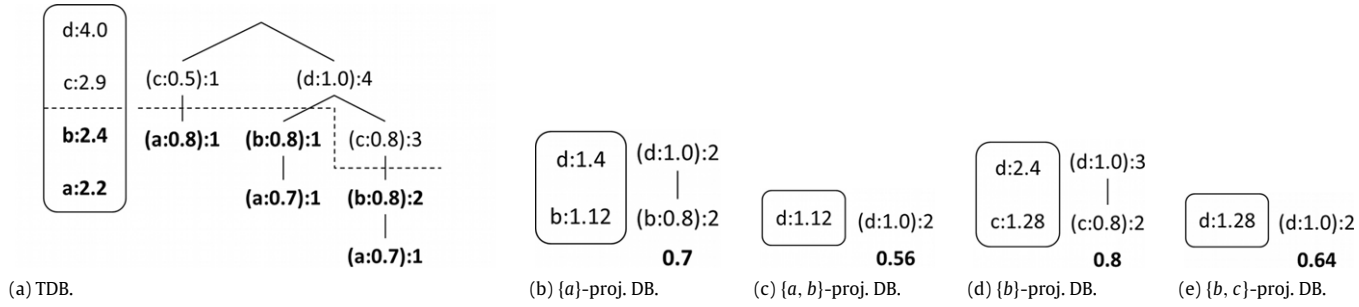
3.2. Finding globally frequent itemsets that satisfy succinct constraints

Once the constrained locally frequent itemsets are found from distributed uncertain data, the next step is to find the constrained globally frequent itemsets among those constrained locally frequent itemsets. Note that it is not a good idea to transmit all data in TDB_i from each site/processor P_i to a centralized site/processor Q, where all data are merged to form a global database $TDB = \bigcup_i TDB_i$ from which constrained globally frequent itemsets are found. The problem with such an approach is that it requires lots of communication for transmitting data from each site. This problem is worsened when TDB_i's are huge; wireless sensors can generate huge amounts of data. Moreover, such an approach does not make use of constrained locally frequent itemsets in finding constrained globally frequent itemsets.

Similarly, it is also not a good idea to ask each site to transmit all its constrained locally frequent itemsets to a centralized site, where the itemsets are merged. The merge result is a collection of global candidate itemsets. The problem is that if a constrained itemset X is locally frequent at a site P_1 but not at another site P_2, then we do not have the frequency of X at P_2. Lacking this frequency information, one may not be able to determine whether X is globally frequent or not.

Instead, our proposed system does the following. Each site/processor P_i (for $1 \leq i \leq p$) applies constraint checking and frequency checking to find locally frequent $\texttt{Item}_i^{\texttt{M}}$ items (and $\texttt{Item}_i^{\texttt{O}}$ items for C_{SUC}), which are then transmitted to a centralized site/processor Q. It takes the union of these items, and broadcasts the union to all P_i's. Each P_i then extracts these items (potentially globally frequent items) from transactions in TDB_i and puts them into a UF-tree. Note that all globally frequent itemsets must be composed of only the items from this union because of the following:

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026



(a) TDB. (b) $\{a\}$-proj. DB. (c) $\{a, b\}$-proj. DB. (d) $\{b\}$-proj. DB. (e) $\{b, c\}$-proj. DB.

Fig. 1. The UF-trees used in our proposed system (Examples 3 and 4).

(i) If an item A is globally frequent, then A must be locally frequent in at least one of P_i's;
(ii) If an item B is locally infrequent in *all* the P_i's, then B is guaranteed to be globally infrequent.

At each site P_i, the UF-tree contains (i) items that are locally frequent with respect to P_i and (ii) items that are potentially globally frequent but locally infrequent items with respect to P_i. Then, our system recursively applies the usual tree-based mining process (e.g., UF-growth) to each α-projected database (where locally frequent $\alpha \subseteq \texttt{Item}_i^{\texttt{M}}$) of the UF-tree at P_i to find *constrained locally frequent itemsets* (with local frequency information). These itemsets are then sent to Q, where the local frequencies are summed. As a result, *constrained globally frequent itemsets* can be found. If the sum of available local frequencies of a constrained itemset X meets the minimum support threshold, then X is globally frequent. For the case where a constrained itemset is locally frequent at a site P_1 but not at another site P_2, then Q sends a request to P_2 for finding its local frequency. It is guaranteed that such frequency information can be found by traversing appropriate paths in the UF-tree at P_2 (because the UF-tree keeps all potential globally frequent items).

3.3. Summary

Given p sites/processors in a distributed environment (e.g., a wireless sensor network), our system makes use of (i) the constrained locally frequent itemsets and (ii) the UF-trees that keep all potentially global frequent items to efficiently find constrained globally frequent itemsets (with respect to the entire distributed environment). Again, succinct constraints are pushed inside the mining process; the computation is proportional to the selectivity of succinct constraints. Moreover, our proposed system does not require lots of communication among processors (e.g., it does not need to transmit TDB_i).

4. Finding frequent itemsets that satisfy non-succinct constraints

In Section 3, we showed how our proposed distributed mining system finds frequent itemsets that satisfy *succinct* constraints. Recall from Section 2.2 that, although a majority of constraints are succinct, there are a few constraints that are *not* succinct. In this section, we discuss how we modify the proposed distributed mining system for finding frequent itemsets that do *not* satisfy succinct constraints.

4.1. Finding frequent itemsets that satisfy inductive succinct constraints

For a constraint C that is not succinct, our proposed distributed mining system first tests to see if such a non-succinct constraint C can be induced into a succinct constraint C'. If so, the system carries out the following actions:

Step 1. The system induces C into C';
Step 2. The system applies the same mining procedures as described in Section 3 but using the induced constraint C' (instead of the original constraint C). In other words, the system finds locally frequent itemsets that satisfy C', from which globally frequent itemsets that satisfy C' can be found; and
Step 3. For each globally frequent itemset X that satisfy C', the system tests to see if X also satisfies the original non-succinct C and returns X to users if it satisfies C.

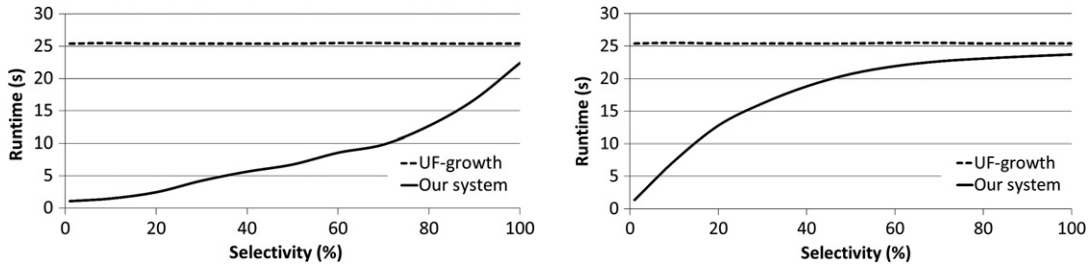
Example 5. Given a user-defined constraint $C_6 \equiv avg(X.Rainfall) \leq 10$ mm, our system induces C_6 into $C_6' \equiv min(X.Rainfall) \leq 10$ mm and finds all globally frequent itemsets that satisfy C_6'. Afterwards, the system tests these itemsets and returns only those that satisfy C_6. □

4.2. Finding frequent itemsets that satisfy anti-monotone constraints

Recall from Section 2.2 that a constraint C_{AM} is anti-monotone if all subsets of an itemset satisfying C_{AM} also satisfy C_{AM}. Hence, for a constraint C that is not succinct and cannot be induced into a succinct constraint, our proposed distributed mining system tests to see if such a non-succinct constraint C is anti-monotone. If so, the system carries out mining procedures that are similar (but not identical) to those described in Section 3. Specifically, the system takes the following steps to find constrained locally frequent itemsets:

Step 1. The system identifies frequent items satisfying C in the same way as described in Section 3.1;
Step 2. The system constructs a UF-tree in the same way as described in Section 3.1;
Step 3. The system mines those frequent itemsets that satisfy C from the UF-tree. Here, unlike the usual Step 3 in which no constraint checking is needed when forming a projected database and constructing a smaller UF-tree for a frequent itemset X, the system needs to apply additional constraint checking (to see if X satisfies C). On the one hand, if X satisfies C, the system forms an X-projected database and constructs a smaller UF-tree capturing such an X-projected database. On the other hand, if X violates C, the system does not form an X-projected database and any supersets of X can be pruned/ignored because these *supersets of* X are guaranteed to violate C (whenever X itself violates C); and
Step 4. Once the *locally* frequent itemsets that satisfy C are found, the system finds *globally* frequent itemsets that satisfy C in the same way as described in Section 3.2.

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026



(a) SAM constraint. (b) SUC constraint.

Fig. 2. Experiment 1: runtime of our system vs. existing algorithms (e.g., UF-growth [17]).

Example 6. Given a user-defined constraint $C_7 \equiv sum(X.Rainfall) \leq 150$ mm, our system constructs a UF-tree capturing all frequent items. During the mining process, the system recursively applies constraint checking to see if an itemset X satisfies C_7. If it does, the system forms an X-projected database and constructs its corresponding UF-tree for extensions of X; otherwise, the system prunes X. Consequently, the system finds all globally frequent itemsets that satisfy C_7. □

4.3. Finding frequent itemsets in a post-processing step

Finally, for a constraint C that is (i) neither succinct nor anti-monotone and (ii) cannot be induced into succinct constraints, our proposed distributed mining system carries out the following actions:

Step 1. The system applies the same mining procedures as described in Section 3, but using no constraint. In other words, the system finds locally frequent itemsets, from which globally frequent itemsets can be found; and

Step 2. For each globally frequent itemset X, the system carries out a post-processing step to test if X satisfies the original non-succinct C and returns X to users if it satisfies C.

Example 7. Given a user-defined constraint $C_8 \equiv sum(X.Rainfall) \geq 90$ mm, our system first finds all globally frequent itemsets, which include those that satisfy C_8 and those that do not. Then, the system carries out a post-processing step to test these itemsets and returns only those that satisfy C_8. □

5. Experimental evaluation

To evaluate our proposed system, we used many different datasets including IBM synthetic data, real-life databases from the UC Irvine Machine Learning Depository (e.g., mushroom data) as well as those from the Frequent Itemset Mining Implementation (FIMI) Dataset Repository. For instance, IBM synthetic datasets used in our experiments were generated by the program developed at IBM Almaden Research Center [5]. The datasets contain 100K to 10M records with an average transaction length of 10 items, and a domain of 1000 items. We assigned to each item an existential probability in the range of (0,1]. All experiments were run in a time-sharing environment in a 2.4 GHz machine. The reported figures are based on the average of multiple runs. Runtime includes CPU and I/Os for constraint checking, UF-tree construction, and frequent itemset mining steps.

In Experiment 1, we evaluated the functionality of our proposed system, which was implemented in C++. For instance, we used (i) a dataset of uncertain data and (ii) a *constraint with 100% selectivity* (so that every item is selected). With this setting, we compared our system (which mines *constrained* frequent itemsets from uncertain data) with U-Apriori [16] and UF-growth [17] (which mine *unconstrained* frequent itemsets from uncertain data). Experimental results on the IBM dataset showed that, in terms of accuracy, our system returned the *same* mining results – i.e., the *same* collection of frequent itemsets – as those returned by U-Apriori and UF-growth.

However, it is important to note that both U-Apriori and UF-growth are confined to finding frequent itemsets from a centralized dataset of uncertain data when the user-defined constraints are of a single selectivity of 100%, whereas our proposed system is more flexible as it is capable of finding frequent itemsets from *distributed* uncertain data with constraints of *any selectivity*.

As for the runtimes among these three algorithms, our system took the shortest amount of time to mine frequent itemsets because it pushes user-defined constraints into the mining process. The higher the selectivity of the constraints, the longer was the runtime for our system. Both U-Apriori and UF-growth were not designed to handle constraints, let alone pushing the constraints into the mining. To handle constraints, U-Apriori and UF-growth first ignored the constraints and found all frequent itemsets. Then, they applied constraint checking as a post-processing step to prune those infrequent itemsets. Hence, the runtime of these two existing algorithms were independent of the selectivity of the SAM and SUC constraints. See Fig. 2.

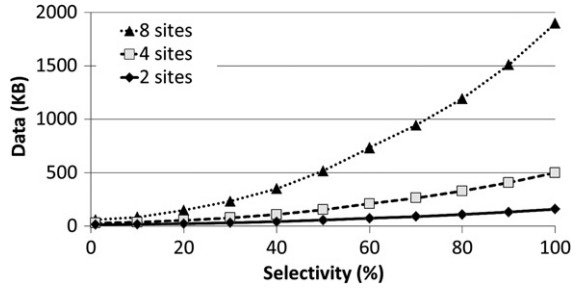
In Experiment 2, we continued with our functionality evaluation. Specifically, we used (i) a constraint and (ii) a dataset of uncertain data consisting of items *all with existential probability of* 1 (indicating that all items are definitely present in the database). With this setting, we compared our system (which mines constrained frequent itemsets from *uncertain* data) with some existing algorithms that mine constrained frequent itemsets from *precise* data (e.g., CAP [20]). We observed from the experimental results that our system returned the *same* mining results – i.e., the *same* collection of frequent itemsets – as those returned by CAP. In other words, our system is as accurate as CAP.

However, regarding the flexibility, CAP is confined to finding frequent itemsets from a centralized dataset of uncertain data when existential probability of all items is 1. In contrast, our proposed system is capable of finding frequent itemsets from distributed uncertain data containing items with *various existential probability values* ranging from 0 to 1.

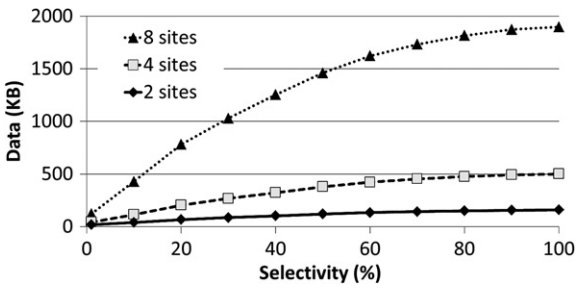
In Experiment 3, we measured the amount of communication/data transmitted between the distributed sites P_i's and their centralized site Q. Fig. 3 shows the results for both IBM synthetic dataset and UCI real-life mushroom dataset. Note that the amount of transmitted data decreased when the selectivity of constraints decreased. The reason is that, as the constraint selectivity decreased, fewer frequent itemsets satisfied the constraints. Hence, less data were transmitted.

Between the SAM and SUC constraints, frequent itemsets satisfying SAM constraints consist of only those domain items that individually satisfy the constraints. Hence, the amount of data transmitted grew exponentially when selectivity increased linearly. In contrast, frequent itemsets satisfying SUC constraints consist of (i) those domain items that individually satisfy the

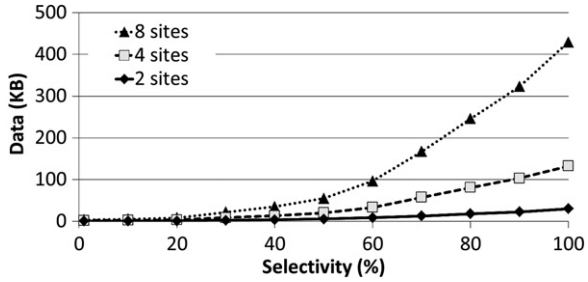
Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026



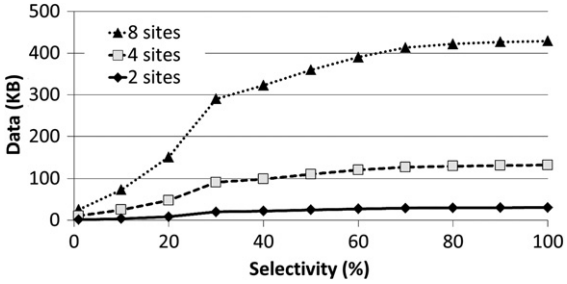
(a) SAM constraints on u100K_5L_10_100 with *minsup* = 20.



(b) SUC constraints on u100K_5L_10_100 with *minsup* = 20.

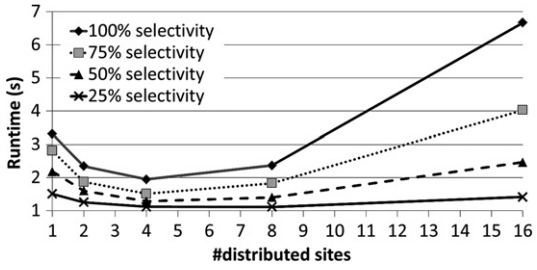


(c) SAM constraints on mushroom_50_60 with *minsup* = 700.

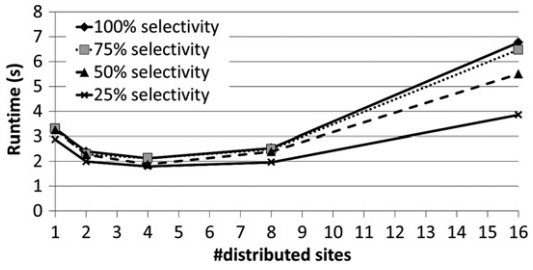


(d) SUC constraints on mushroom_50_60 with *minsup* = 700.

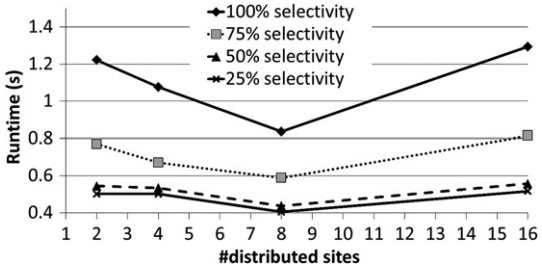
Fig. 3. Experiment 3: amount of data transmitted vs. selectivity.



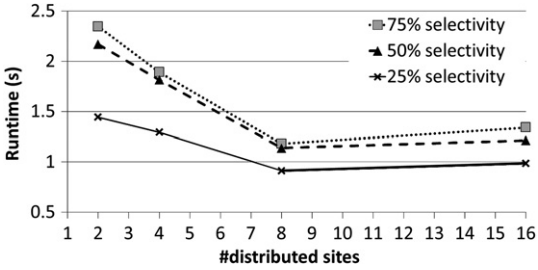
(a) SAM constraints on u100K_5L_10_100 with *minsup* = 20.



(b) SUC constraints on u100K_5L_10_100 with *minsup* = 20.



(c) SAM constraints on mushroom_50_60 with *minsup* = 700.



(d) SUC constraints on mushroom_50_60 with *minsup* = 700.

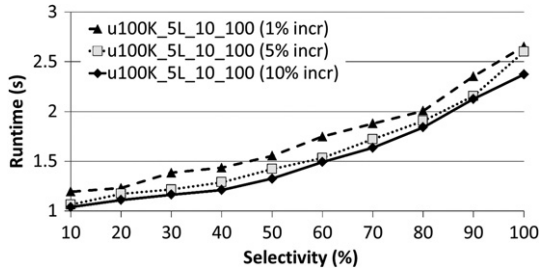
Fig. 4. Experiment 4: runtime vs. number of sites.

constraints and (ii) optional items. Hence, the amount of data transmitted grew rapidly at the beginning (i.e., when selectivity was low). When selectivity kept increasing, the amount of data transmitted gradually became stable (as domain items in frequent itemsets are either in required or optional).

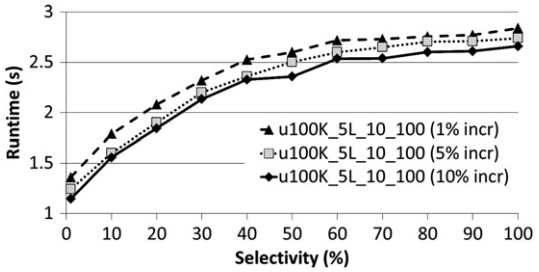
In Experiment 4, we evaluated the effects of varying the number of distributed sites. Recall from Fig. 3 that, when more sites were in the distributed network, our system transmitted more data because an addition of a site implies transmission of an additional set of locally frequent items and locally frequent itemsets. In other words, more sites in the network led to extra communication time. However, more sites led to smaller UF-trees that were built and mined at each site. Regarding runtime, there was tradeoff between the communication cost and the tree construction/mining cost. We observed from Fig. 4(a) and (b) that, when the number of sites increased from 1 to 2 and to 4, the mining on the synthetic dataset were distributed among multiple sites. As a result, less work was required at each individual site, where smaller UF-trees were built and mined. So, runtimes decreased. However, when the number of sites grew from 4 to 8 and to 16, the overhead due to extra communication cost offset the benefits of using extra sites. Consequently, runtimes increased. Fig. 4(c) and (d) show similar observation on mining the real-life mushroom databases except that (i) the runtimes decreased when the number of sites increased from 2 to 4 and to 8 and (ii) the times increased when the number of sites grew from 8 to 16.

In Experiment 5, we examined the effect of the distribution of existential probabilities of items. Recall that nodes are merged

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026



(a) SAM constraints on u100K_5L_10_100 with *minsup* = 20 & #sites = 8.



(b) SUC constraints on u100K_5L_10_100 with *minsup* = 20 & #sites = 8.

Fig. 5. Experiment 5: runtime vs. probability distribution.

in a UF-tree if they contain the same item and same existential probability values. So, we divided the precision of existential probability values into 1%, 5% and 10%. Fig. 5 shows three versions of an IBM synthetic dataset having existential probability values in the range [10%, 100%]. When the precision is 1%, all existential probability values are multiples of 1% within that range (e.g., 10%, 11%, 12%, 13%, . . . , 99%, 100%) for a total of 91 unique existential probability values. Similarly, when the precision is 5% (or 10%), there are 19 (or 10) unique existential probability values. When items took on only a few unique existential probability values, UF-trees became smaller and thus took shorter runtimes.

We also tested the effect of *minsup*. When *minsup* increased, fewer itemsets had expected support $\geq$ *minsup*, and thus shorter runtimes were required for the experiment.

All these experimental results showed the importance and the benefits of using our proposed distributed system in mining constrained frequent itemsets from uncertain data.

6. Conclusions

Amounts of uncertain data obtained from various sources such as networks of distributed wireless sensors have increased over the past few years. There are many real-life applications in which users are interested in only some subsets of all the frequent itemsets that can be mined from these high volumes of massive distributed uncertain data. To find frequent itemsets that satisfy the user-defined constraints from these distributed uncertain data, we proposed a computer system that uses a tree-based mining approach. Our system is a non-trivial integration of constrained mining, parallel and distributed mining, uncertain data mining, and tree-based frequent itemset mining. The system handles different types of user-defined constraints. For instance, our system first identifies domain items that satisfy succinct constraints at each distributed site and then constructs a UF-tree, from which constrained locally frequent itemsets can be mined recursively. To return to the user each constrained globally frequent itemset, its local frequencies at all sites are summed. Missing frequencies can be computed by traversing appropriate paths in essential UF-trees. In addition to succinct constraints, our system also handles non-succinct constraints such as inductive succinct constraints and anti-monotone constraints. Experimental results – in terms of functionality, transmitted data, and runtimes – show the effectiveness of our system in mining constrained frequent itemsets from distributed uncertain data.

Acknowledgments

This project is partially supported by NSERC (Canada) and the University of Manitoba in the form of research grants.

References

[1] J.J. Cameron, A. Cuzzocrea, F. Jiang, C.K.-S. Leung, Mining frequent itemsets from sparse data streams in limited memory environments, in: Proceedings of the WAIM 2013, Springer, pp. 51–57.
[2] E. Çem, Ö. Özkasap, ProFID: practical frequent items discovery in peer-to-peer networks, Future Gener. Comput. Syst. 29 (6) (2013) 1544–1560.
[3] C.K.-S. Leung, C.L. Carmichael, Exploring social networks: a frequent pattern visualization approach, in: Proceedings of the SocialCom 2010, IEEE Computer Society, pp. 419–424.
[4] R. Agrawal, T. Imieliński, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD 1993, pp. 207–216.
[5] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the VLDB 1994, Morgan Kaufmann, pp. 487–499.
[6] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Min. Knowl. Discov. 8 (1) (2004) 53–87.
[7] J. Ke, Y. Zhan, X. Chen, M. Wang, The retrieval of motion event by associations of temporal frequent pattern growth, Future Gener. Comput. Syst. 29 (1) (2013) 442–450.
[8] C.K.-S. Leung, S.K. Tanbeer, PUF-tree: a compact tree structure for frequent pattern mining of uncertain data, in: Proceedings of the PAKDD 2013, Part I, Springer, pp. 13–25.
[9] A. Cuzzocrea, Approximate OLAP query processing over uncertain and imprecise multidimensional data streams, in: Proceedings of the DEXA 2013, Part II, Springer, pp. 156–173.
[10] A. Cuzzocrea, Retrieving accurate estimates to OLAP queries over uncertain and imprecise multidimensional data streams, in: Proceedings of the SSDBM 2011, Springer, pp. 575–576.
[11] C.K.-S. Leung, B. Hao, Mining of frequent itemsets from streams of uncertain data, in: Proceedings of the IEEE ICDE 2009, pp. 1663–1670.
[12] S.R. Sarangi, K. Murthy, DUST: a generalized notion of similarity between uncertain time series, in: Proceedings of the ACM KDD 2010, pp. 383–392.
[13] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, A. Zuefle, Probabilistic frequent itemset mining in uncertain databases, in: Proceedings of the ACM KDD 2009, pp. 119–127.
[14] F. Jiang, C.K.-S. Leung, Stream mining of frequent patterns from delayed batches of uncertain data, in: Proceedings of the DaWaK 2013, Springer, pp. 209–221.
[15] C.K.-S. Leung, A. Cuzzocrea, F. Jiang, Discovering frequent patterns from uncertain data streams with time-fading and landmark models, LNCS Trans. Large Scale Data Knowl. Centered Syst. 8 (2013) 174–196.
[16] C.-K. Chui, B. Kao, E. Hung, Mining frequent itemsets from uncertain data, in: Proceedings of the PAKDD 2007, Springer, pp. 47–58.
[17] C.K.-S. Leung, M.A.F. Mateo, D.A. Brajczuk, A tree-based approach for frequent pattern mining from uncertain data, in: Proceedings of the PAKDD 2008, Springer, pp. 653–661.
[18] G. Grahne, L.V.S. Lakshmanan, X. Wang, Efficient mining of constrained correlated sets, in: Proceedings of the IEEE ICDE 2000, pp. 512–521.
[19] C.K.-S. Leung, Frequent itemset mining with constraints, in: L. Liu, M.T. Özsu (Eds.), Encyclopedia of Database Systems, Springer, 2009, pp. 1179–1183.
[20] R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations rules, in: Proceedings of the ACM SIGMOD 1998, pp. 13–24.
[21] L.V.S. Lakshmanan, C.K.-S. Leung, R.T. Ng, Efficient dynamic mining of constrained frequent sets, ACM Trans. Database Syst. 28 (4) (2003) 337–389.
[22] J. Pei, J. Han, L.V.S. Lakshmanan, Pushing convertible constraints in frequent itemset mining, Data Min. Knowl. Discov. 8 (3) (2004) 227–252.
[23] A. Cuzzocrea, P. Serafino, LCS-Hist: taming massive high-dimensional data cube compression, in: Proceedings of the EDBT 2009, ACM Press, pp. 768–779.
[24] C.K.-S. Leung, Y. Hayduk, Mining frequent patterns from uncertain data with MapReduce for big data analytics, in: Proceedings of the DASFAA 2013, Part I, Springer, pp. 440–455.

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026

[25] A. Bonifati, A. Cuzzocrea, Storing and retrieving XPath fragments in structured P2P networks, Data Knowl. Eng. 59 (2) (2006) 247–269.
[26] A. Faro, D. Giordano, F. Maiorana, Mining massive datasets by an unsupervised parallel clustering on a GRID: novel algorithms and case study, Future Gener. Comput. Syst. 27 (1) (2011) 711–724.
[27] J. Secretan, M. Georgiopoulos, A. Koufakou, K. Cardona, APHID: an architecture for private, high-performance integrated data mining, Future Gener. Comput. Syst. 26 (7) (2010) 891–904.
[28] M.T. Swain, C.G. Silva, N. Loureiro-Ferreira, V. Ostropytskyy, J. Brito, O. Riche, F. Stahl, W. Dubitzky, R.M.M. Brito, P-found: grid-enabling distributed repositories of protein folding and unfolding simulations for data mining, Future Gener. Comput. Syst. 26 (3) (2010) 424–433.
[29] A. Schuster, R. Wolff, D. Trock, A high-performance distributed algorithm for mining association rules, Knowl. Inf. Syst. 7 (4) (2005) 458–475.
[30] M.J. Zaki, Parallel and distributed association mining: a survey, IEEE Concurr. 7 (4) (1999) 14–25.
[31] R. Agrawal, J. Shafer, Parallel mining of association rules, IEEE Trans. Knowl. Data Eng. 8 (6) (1996) 962–969.
[32] D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, Y. Fu, A fast distributed algorithm for mining association rules, in: Proceedings of the PDIS 1996, IEEE Computer Society, pp. 31–42.
[33] M. El-Hajj, O.R. Zaïane, Parallel leap: large-scale maximal pattern mining in a distributed environment, in: Proceedings of the ICPADS 2006, IEEE Computer Society, pp. 135–142.
[34] Y. Kozawa, T. Amagasa, H. Kitagawa, GPU acceleration of probabilistic frequent itemset mining from uncertain databases, in: Proceedings of the ACM CIKM 2012, pp. 892–901.
[35] A. Cuzzocrea, C.K.-S. Leung, Distributed mining of constrained frequent sets from uncertain data, in: Proceedings of the ICA3PP 2011, Part I, Springer, pp. 40–53.
[36] A. Cuzzocrea, C.K.-S. Leung, Frequent itemset mining of distributed uncertain data under user-defined constraints, in: Proceedings of the SEBD 2012, pp. 243–250.
[37] C.C. Aggarwal, Y. Li, J. Wang, J. Wang, Frequent pattern mining with uncertain data, in: Proceedings of the ACM KDD 2009, pp. 29–37.
[38] C.K.-S. Leung, Mining uncertain data, WIREs Data Min. Knowl. Discov. 1 (4) (2011) 316–329.

Alfredo Cuzzocrea is currently a Senior Researcher at ICAR-CNR and an Adjunct Professor at University of Calabria, Italy. He covers a large number of roles in international journals (e.g., Editor-in-Chief, Associate Editor, Special Issue Editor) and international conferences (e.g., General Chair, Program Chair, Workshop Chair). His research interests include multidimensional data modeling & querying, data stream modeling & querying, data warehousing & OLAP, grid & P2P computing, as well as privacy & security of very large databases & OLAP data cubes. He has authored or co-authored more than 270 papers in international conferences, journals, and books.

Carson Kai-Sang Leung received his B.Sc.(Hons.), M.Sc., and Ph.D. from University of British Columbia, Canada. He is currently an Associate Professor with University of Manitoba, Canada. He has served as a Program Chair for C3S2E 2009 & 2010, an organizing committee member of ACM SIGMOD 2008 and IEEE ICDM 2011, and a PC member of numerous conferences including ACM KDD, ACM CIKM, and ECML/PKDD. His research interests include databases, data mining, social network mining, and distributed/parallel mining. His work has been published in *ACM TODS*, IEEE ICDE, IEEE ICDM, and PAKDD. He won the Best Paper award in SCA 2012.

Richard Kyle MacKinnon received his B.C.Sc.(Hons.) from University of Manitoba, Canada. He is currently pursuing his M.Sc. degree in Department of Computer Science at the same university under the academic supervision of Dr. C.K.-S. Leung. He is interested in conducting research in frequent itemset mining and social network mining.

Please cite this article in press as: A. Cuzzocrea, et al., Mining constrained frequent itemsets from distributed uncertain data, Future Generation Computer Systems (2013), http://dx.doi.org/10.1016/j.future.2013.10.026