Frequent Itemset Mining of Distributed Uncertain Data under User-Defined Constraints*

Alfredo Cuzzocrea[1] and Carson K. Leung[2]

[1] ICAR-CNR and University of Calabria, Italy
cuzzocrea@si.deis.unical.it
[2] University of Manitoba, Canada
kleung@cs.umanitoba.ca

Abstract. Many existing *distributed data mining* algorithms do not allow users to express the patterns to be mined according to their intention via the use of constraints. Consequently, these unconstrained mining algorithms can yield numerous patterns that are not interesting to users. Moreover, due to inherited measurement inaccuracies and/or network latencies, data are often riddled with uncertainty. These call for *constrained mining* and *uncertain data mining*. In this paper, we propose a tree-based system for mining frequent itemsets that satisfy user-defined constraints from a distributed environment such as a wireless sensor network of uncertain data.

Keywords: Data mining, knowledge discovery, constraints, distributed data, frequent itemsets, uncertain data.

1 Introduction and Related Work

Many frequent itemset mining algorithms in the early days were Apriori-based [1, 19], 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 traditional databases (DBs) containing precise data. To avoid the generate-and-test paradigm, the FP-growth algorithm [7] was proposed. Such a tree-based algorithm constructs a Frequent Pattern tree (FP-tree) to capture the contents of the DBs; it focuses on frequent pattern growth which is a restricted test-only approach.

In many real-life applications, data are riddled with uncertainty. 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 DB is uncertain. Hence, mining uncertain data [2, 11, 12, 15, 21] 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* $P(x, t_i)$ of an item x in a transaction t_i in a probabilistic DB. To mine frequent itemsets from these uncertain data, the UF-growth [13] algorithm was proposed.

* Extended Abstract

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

Many frequent itemset mining algorithms, regardless whether Apriori-based or tree-based, 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. This calls for *constrained frequent itemset mining* [9, 14], which finds frequent itemsets that satisfy user-defined constraints. For example, DCF [8] mines constrained frequent itemsets from traditional precise data.

With advances in technology, one can easily collect large amounts of data 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. This calls for *distributed mining* [4, 16–18, 20]—which discovers implicit, previously unknown, and potentially useful knowledge that might be embedded in distributed data.

Existing distributed mining algorithms—such as FDM [3] and Parallel-HFP-Leap [6]—find frequent itemsets in a distributed environment, but they all do not handle constraints nor do they mine uncertain data. In contrast, DCF finds *constrained* frequent itemsets, but they mine a centralized DB of precise data. UF-growth mines a centralized DB of *uncertain* data for all (unconstrained) frequent itemsets instead of only those constrained ones. In other words, these existing frequent itemset 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 *constraints* in a *distributed* environment? In response to this question, we propose in this paper a tree-based system for mining 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) distributed mining, (iii) uncertain data mining, with (iv) tree-based frequent itemset mining. The resulting tree-based system efficiently mines from distributed uncertain data for only those constrained frequent itemsets.

This paper is organized as follows. In the next section, we propose our non-trivial integration of tree-based frequent set mining, constrained mining, distributed mining, and uncertain mining. Experimental results are shown in Section 3. Finally, Section 4 presents the conclusions.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

2 Our Proposed Distributed Mining System

Without loss of generality, we assume to have p sites/processors and $m = m_1 + m_2 + ... + m_p$ 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 distributed mining system finds (i) constrained itemsets that are locally frequent w.r.t. site/processor P_i and (ii) those that are globally frequent w.r.t. all sites/processors in the entire wireless sensor network.

2.1 Finding Constrained Locally Frequent Itemsets

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. Here, we use the "possible world" interpretation of uncertain data. We aim to find itemsets that are both (i) locally frequent to P_i and (ii) satisfying a *succinct* constraint, to which a majority of user-defined constraints belong. A constraint C is ***succinct*** [9] if one can directly generate precisely all and only those itemsets satisfying C without generating and excluding itemsets not satisfying C. Examples of succinct constraints include $C_1 \equiv max(X.Price) \leq \$25$ (which expresses the user interest in finding every itemset X such that the maximum price of all market basket items in X is at most \$25) and $C_2 \equiv min(X.Price) \leq \$30$ (which says that the minimum price of all items in X is at most \$30). Note that many non-succinct constraints (e.g., $C_3 \equiv avg(X.Price) \leq \$30$) can be induced into weaker but succinct constraints (e.g., C_2).

Step 1: Identification of items satisfying the constraint. A succinct constraint C is also *anti-monotone* [9] if all subsets of an itemset X satisfy C whenever X satisfies C. Hence, succinct constraints can be further into (i) ***succinct anti-monotone* (*SAM*)** and (ii) ***succinct non-anti-monotone* (*SUC*)** constraints. Then, let $\texttt{Item}^{\texttt{M}}$ be the collection of *mandatory items*—i.e., the collection of domain items that individually satisfy C (e.g., SAM or SUC constraint); let $\texttt{Item}^{\texttt{O}}$ be the collection of *optional items*—i.e., the collection of domain items that individually violate C.

For any C_{SAM}, an itemset X satisfying C_{SAM} cannot contain any item from $\texttt{Item}^{\texttt{O}}$ (e.g., if an itemset X containing an item having price $>$ \$25, then X violates C_{SAM} C_1 and so does every superset of X). So, any X satisfying C_{SAM} must be generated by combining items from $\texttt{Item}^{\texttt{M}}$ (i.e., $X \subseteq \texttt{Item}^{\texttt{M}}$). 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}. Due to page limitation, please see Ref. [5] for an illustrative example.

For any 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 whether or not they satisfy C_{SUC}). Note that, if X violates C_{SUC}, there is no guarantee that all or any of its supersets would violate C_{SUC}.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

Hence, 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 Ref. [5] for an illustrative example.

Step 2: Construction of an 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 an UF-tree, which is built in preparation for mining constrained frequent itemsets from uncertain data. It does so by first scanning the DB of uncertain data once. It accumulates the *expected support* of each of the items in order to find all *frequent* domain items. The ***expected support*** [10] of an itemset X consisting of k independent items over n transaction in the DB can be computed by $expSup(X) = \sum_{i=1}^{n} \left(\prod_{x \in X} P(x, t_i) \right)$. The system discards infrequent items and only captures frequent items 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 DB the second time and inserts each transaction of the DB 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 C_{SAM}, the corresponding UF-tree captures only those frequent $\texttt{Item}^{\texttt{M}}$ items; for C_{SUC}, 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.* For an illustrative example, see Ref. [5].

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 DB for each $x \in \texttt{Item}^{\texttt{M}}$. The system does not need to form projected DBs 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 DB 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 $\{x\}$-projected DB. Such a distinction between two kinds of items is only needed for the UF-tree for the DB (for SUC constraints only) but not

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

projected UF-trees once we found at least one valid item $x \in \texttt{Item}^{\texttt{M}}$ because, for any v satisfying C_{SUC}, $v = \{x\} \cup \texttt{others}$, where (i) $x \in \texttt{Item}^{\texttt{M}}$, (ii) $\texttt{others} \subseteq (\texttt{Item}^{\texttt{M}} \cup \texttt{Item}^{\texttt{O}} - \{x\})$. After constructing these projected UF-tree 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}. Again, see Ref. [5] for an illustrative example.

2.2 Finding Constrained Globally Frequent Itemsets

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 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 worsen when TDB_i's are huge; wireless sensors can generate huge amount 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 into an UF-tree. Note that all globally frequent itemsets must be composed of only the items from this union because: (i) if an item A is globally frequent, 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, B is guaranteed to be globally infrequent.) At each site P_i, the UF-tree contains (i) items that are locally frequent w.r.t. P_i and (ii) items that are potentially globally frequent but locally infrequent items w.r.t P_i. Then, our system recursively applies the usual tree-based mining process (e.g., UF-growth) to each α-projected DB (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

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6, Copyright (c) 2012 - Edizioni Libreria Progetto and the authors

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 UP-tree keeps all potential globally frequent items).

To summarize, 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 (w.r.t. the entire distributed environment). Again, constraints are pushed inside the mining process; the computation is proportional to the selectivity of constraints. Moreover, our proposed system does not require lots of communication among processors (e.g., it does not need to transmit TDB_i).

3 Experimental Results

For experimental evaluation, we used different datasets including IBM synthetic data, real-life DBs from the UC Irvine Machine Learning Depository as well as those from the Frequent Itemset Mining Implementation (FIMI) Dataset Repository. Due to page limitation, we cite below those experimental results based on a dataset generated by the program developed at IBM Almaden Research Center [1]. The dataset contains 10M records with an average transaction length of 10 items, and a domain of 1,000 items. Unless otherwise specified, we used *min-sup* = 0.01%. We randomly 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 the first experiment, we evaluated the accuracy and flexibility 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 UF-growth [13] (which mines *unconstrained* itemsets). Experimental results showed that (i) our system is as accurate as UF-growth (and they both returned the *same* collection of frequent itemsets), and (ii) our system is more flexible than UF-growth (because the former is capable of finding frequent itemsets from *distributed* uncertain data with constraints of *any selectivity* whereas the latter is confined to those of 100% selectivity.

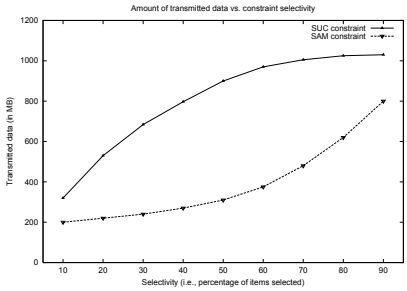
Similar observations were made when we compared our system (which mines *uncertain* data) with DCF [8] (which mines *precise* data) by using (i) a constraint and (ii) a dataset of uncertain data consisting of items *all with existential probability of 1* (indicating that all items are definitely present in the DB). We observed that (i) our system is as accurate as DCF, and (ii) our proposed system is more flexible than DCF (because the former is capable of finding frequent itemsets containing items with *various existential probability values* ranging from 0 to 1 whereas the latter is confined to those of existential probability of 1.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

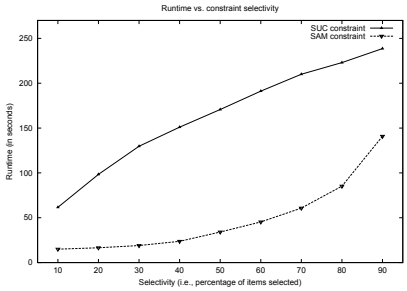
In the second experiment, we evaluated the *effectiveness of constrained mining in a distributed environment* by measuring the amount of communication/ data transmitted between the distributed sites P_i's and their centralized site Q. Fig. 1(a) shows that the amount of transmitted data decreased when the selectivity of constraints decreased. Fig. 1(b) shows the corresponding runtimes. Specifically, runtimes decreased when the selectivity of constraints decreased.

In the third experiment, we evaluated the effects of varying the number of distributed sites. 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 terms of runtime, when more sites were in the network, the runtime of our system increased slightly. This is because the extra communication time (due to extra sites) was offset by the savings in building and mining from a smaller UF-tree at each site. For example, when we doubled the number of sites (from 4 to 8 sites), the amount of communication/data transmitted was almost double (because each site produced a similar set of locally frequent itemsets—especially when TDB_i's were similar); but, the runtime just increased slightly (1.07 times; i.e., not doubled) because we built and mined from smaller FP-trees at 8 sites (rather than from bigger trees at 4 sites).

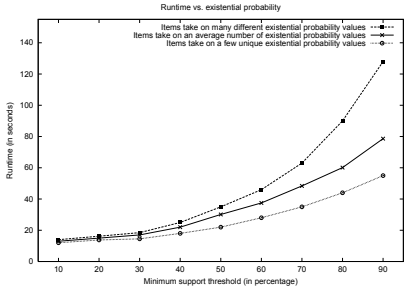
In addition, we conducted a few more experiments. For example, we tested the effect of distribution of existential probabilities of items. When items took



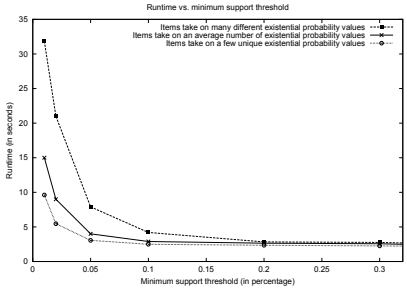
(a) Amt of transmitted data vs. selectivity



(b) Runtime vs. selectivity



(c) Runtime vs. existential probability



(d) Runtime vs. minsup

Fig. 1. Experimental results of our proposed system

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,

on a few unique existential probability values, UF-trees became smaller and thus took shorter runtimes. See Fig. 1(c). 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 experiments. See Fig. 1(d).

All these experimental results showed the importance and the benefits of using our proposed system in mining probabilistic datasets of uncertain data for frequent itemsets.

4 Conclusions

In this paper, we proposed a tree-based system for mining frequent itemsets that satisfy user-defined constraints from a distributed environment such as a wireless sensor network of uncertain data. Experimental results show effectiveness of our proposed system. As future work, we plan to extend our experiments to additional datasets.

Acknowledgement. This project is partially supported by NSERC (Canada) and University of Manitoba.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: VLDB 1994, pp. 487–499.
2. Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., Zuefle, A.: Probabilistic frequent itemset mining in uncertain databases. In: ACM KDD 2009, pp. 119–127.
3. Cheung, D.W., Han, J., Ng, V.T., Fu, A.W., Fu, Y.: A fast distributed algorithm for mining association rules. In: PDIS 1996, pp. 31–42.
4. Coenen, F., Leng, P., Ahmed, S.: T-trees, vertical partitioning and distributed association rule mining. In: IEEE ICDM 2003, pp. 513–516.
5. Cuzzocrea, A., Leung, C.K.-S.: Distributed mining of constrained frequent sets from uncertain data. In: ICA3PP 2011, Part 1. LNCS 7016, pp. 40–53.
6. El-Hajj, M., Zaïane, O.R.: Parallel leap: large-scale maximal pattern mining in a distributed environment. In: ICPADS 2006, pp. 135–142.
7. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD 2000, pp. 1–12.
8. Lakshmanan, L.V.S., Leung, C.K.-S., Ng, R.T.: Efficient dynamic mining of constrained frequent sets. ACM TODS 28(4), 337–389 (2003)
9. Leung, C.K.-S.: Frequent itemset mining with constraints. In: Encyclopedia of Database Systems, pp. 1179–1183 (2009)
10. Leung, C.K.-S.: Mining uncertain data. Wiley WIDM 1(4), pp. 316–329 (2011)
11. Leung, C.K.-S., Hao, B.: Mining of frequent itemsets from streams of uncertain data. In: IEEE ICDE 2009, pp. 1663–1670.
12. Leung, C.K.-S., Jiang, F., Hayduk, Y.: A landmark-model based system for mining frequent patterns from uncertain data streams. In: IDEAS 2011, pp. 249–250.
13. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In: PAKDD 2008. LNAI 5012, pp. 653–661.
14. Leung, C.K.-S., Sun, L.: A new class of constraints for constrained frequent pattern mining. In: ACM SAC 2012, pp. 518–523.
15. Leung, C.K.-S., Tanbeer, S.K.: Fast tree-based mining of frequent itemsets from uncertain data. In: DASFAA 2012, Part 1. LNCS 7238, pp. 272–287.
16. Park, J.S., Chen, M.-S., Yu, P.S.: Efficient parallel data mining for association rules. In: CIKM 1995, pp. 31–36.
17. Provost, F.: Distributed data mining: scaling up and beyond. In: Advances in Distributed and Parallel Knowledge Discovery, pp. 3–28 (2000)
18. Schuster, A., Wolff, R., Trock, D.: A high-performance distributed algorithm for mining association rules. Springer KAIS 7(4), pp. 458–475 (2005)
19. Toivonen, H.: Sampling large databases for association rules. In: VLDB 1996, pp. 134–145.
20. Zaki, M.J.: Parallel and distributed association mining: a survey. IEEE Concurrency 7(4), pp. 14–25 (1999)
21. Zhang, Q., Li, F., Yi, K.: Finding frequent items in probabilistic data. In: ACM SIGMOD 2008, pp. 819–832.

Nicola Ferro and Letizia Tanca (Eds.): SEBD 2012, Edizioni Libreria Progetto, Padova, Italia
ISBN: 978-88-96477-23-6,