Infrequent Weighted Itemset Mining Using Frequent Pattern Growth

Luca Cagliero and Paolo Garza

Abstract—Frequent weighted itemsets represent correlations frequently holding in data in which items may weight differently. However, in some contexts, e.g., when the need is to minimize a certain cost function, discovering rare data correlations is more interesting than mining frequent ones. This paper tackles the issue of discovering rare and weighted itemsets, i.e., the infrequent weighted itemset (IWI) mining problem. Two novel quality measures are proposed to drive the IWI mining process. Furthermore, two algorithms that perform IWI and Minimal IWI mining efficiently, driven by the proposed measures, are presented. Experimental results show efficiency and effectiveness of the proposed approach.

Index Terms—Clustering, classification, and association rules, data mining

1 Introduction

Itemset mining is an exploratory data mining technique widely used for discovering valuable correlations among data. The first attempt to perform itemset mining [1] was focused on discovering frequent itemsets, i.e., patterns whose observed frequency of occurrence in the source data (the support) is above a given threshold. Frequent itemsets find application in a number of real-life contexts (e.g., market basket analysis [1], medical image processing [2], biological data analysis [3]). However, many traditional approaches ignore the influence/interest of each item/transaction within the analyzed data. To allow treating items/transactions differently based on their relevance in the frequent itemset mining process, the notion of weighted itemset has also been introduced [4], [5], [6]. A weight is associated with each data item and characterizes its local significance within each transaction.

Consider, as an example, the data set reported in Table 1. It includes six transactions (identified by the respective tids), each one composed of four distinct items weighted by the corresponding degree of interest (e.g., item a has weight 0 in tid 1, and 100 in tid 4). In the contexts of data center resource management and application profiling, transactions may represent CPU usage readings collected at a fixed sampling rate. For instance, tid 1 means that, at a fixed point of time (1), CPU b works at a high usage rate (weight 100), CPUs c and d have an intermediate usage rate (weights 57 and 71, respectively), while CPU a is temporarily idle (weight 0). The itemsets mined from the example data set can be exploited by a domain expert to profile system usage in order to perform resource allocation and system resizing.

• *The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129, Torino, Italy. E-mail: {luca.cagliero, paolo.garza}@polito.it.*

Manuscript received 16 Apr. 2012; revised 19 Oct. 2012; accepted 23 Apr. 2013; date of publication 29 Apr. 2013; date of current version 18 Mar. 2014. Recommended for acceptance by J. Freire.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2013.69

The significance of a weighted transaction, i.e., a set of weighted items, is commonly evaluated in terms of the corresponding item weights. Furthermore, the main itemset quality measures (e.g., the support) have also been tailored to weighted data and used for driving the frequent weighted itemset mining process. For instance, when evaluating the support of $\{a,b\}$ in the example data set reported in Table 1, the occurrence of b in tid 1, which represents a highly utilized CPU, should be treated differently from the one of a, which represents an idle CPU at the same instant. In [4], [5], [6] different approaches to incorporating item weights in the itemset support computation have been proposed. Note that they are all tailored to frequent itemset mining, while this work focuses on infrequent itemsets.

In recent years, the attention of the research community has also been focused on the infrequent itemset mining problem, i.e., discovering itemsets whose frequency of occurrence in the analyzed data is less than or equal to a maximum threshold. For instance, in [7], [8] algorithms for discovering *minimal* infrequent itemsets, i.e., infrequent itemsets that do not contain any infrequent subset, have been proposed. Infrequent itemset discovery is applicable to data coming from different real-life application contexts such as (i) statistical disclosure risk assessment from census data and (ii) fraud detection [7], [8], [9]. However, traditional infrequent itemset mining algorithms still suffer from their inability to take local item interestingness into account during the mining phase. In fact, on the one hand, itemset quality measures used in [4], [5], [6] to drive the frequent weighted itemset mining process are not directly applicable to accomplish the infrequent weighted itemset (IWI) mining task effectively, while, on the other hand, state-of-the-art infrequent itemset miners are, to the best of our knowledge, unable to cope with weighted data.

This paper addresses the discovery of infrequent and weighted itemsets, i.e., the infrequent weighted itemsets, from transactional weighted data sets. To address this issue, the IWI-support measure is defined as a weighted frequency of occurrence of an itemset in the analyzed data. Occurrence weights are derived from the weights

TABLE 1
Example of Weighted Transactional Data Set

Tid	CPU usage readings
1	$\langle a,0\rangle\ \langle b,100\rangle\ \langle c,57\rangle\ \langle d,71\rangle$
2	$\langle a,0\rangle\ \langle b,43\rangle\ \langle c,29\rangle\ \langle d,71\rangle$
3	$\langle a,43\rangle\ \langle b,0\rangle\ \langle c,43\rangle\ \langle d,43\rangle$
4	$\langle a,100\rangle\ \langle b,0\rangle\ \langle c,43\rangle\ \langle d,100\rangle$
5	$\langle a,86\rangle\ \langle b,71\rangle\ \langle c,0\rangle\ \langle d,71\rangle$
6	$\langle a,57\rangle\ \langle b,71\rangle\ \langle c,0\rangle\ \langle d,71\rangle$

associated with items in each transaction by applying a given cost function. In particular, we focus our attention on two different IWI-support measures: (i) The IWI-support-min measure, which relies on a minimum cost function, i.e., the occurrence of an itemset in a given transaction is weighted by the weight of its *least* interesting item, (ii) The IWI-support-max measure, which relies on a maximum cost function, i.e., the occurrence of an itemset in a given transaction is weighted by the weight of the *most* interesting item. Note that, when dealing with optimization problems, minimum and maximum are the most commonly used cost functions. Hence, they are deemed suitable for driving the selection of a worthwhile subset of infrequent weighted data correlations. Specifically, the following problems have been addressed:

A. IWI and Minimal IWI mining driven by a maximum IWI-support-min threshold, and
B. IWI and Minimal IWI mining driven by a maximum IWI-support-max threshold.

Task (A) entails discovering IWIs and minimal IWIs (MIWIs) which include the item(s) with the least local interest within each transaction. Table 2 reports the IWIs mined from Table 1 by enforcing a maximum IWI-support-min threshold equal to 180 and their corresponding IWI-support-min values. For instance, $\{a,b\}$ covers the transactions with tids 1, 2, 3, and 4 with a minimal weight 0 (associated with a in tids 1 and 2 and b in tids 3 and 4), while it covers the transactions with tids 5 and 6 with minimal weights 71 and 57, respectively. Hence, its IWI-support-min value is 128. In the context of system usage profiling, IWIs in Table 2 represent sets of CPUs which contain *at least* one underutilized or idle CPU at each sampled instant. As shown in Section 5, real-life system malfunctioning or underutilization may arise when the workload is not allocated properly over the available CPUs. For instance, considering CPUs a and b, recognizing a suboptimal usage rate of at least one of them may trigger targeted actions, such as system resizing or resource sharing policy optimization. As an extreme case, $\{a,b,c\}$ has IWI-support-min equal to 0 because at every sampled point of time at least one between a, b, or c (not necessarily the same at each instant) is idle, possibly due to system oversizing.

TABLE 2
IWIs Extracted from the Data Set in Table 1

IWI	IWI-support-min	IWI	IWI-support-min
{c}	172 (Minimal)	{a,b,c}	0 (Not Minimal)
{a,b}	128 (Minimal)	{a,b,d}	128 (Not Minimal)
{a,c}	86 (Not Minimal)	{a,c,d}	86 (Not Minimal)
{b,c}	86 (Not Minimal)	{b,c,d}	86 (Not Minimal)
{c,d}	172 (Not Minimal)	{a,b,c,d}	0 (Not Minimal)

Maximum threshold $\xi = 180$.

TABLE 3
IWIs Extracted from the Data Set in Table 1

IWI	IWI-support-max	IWI	IWI-support-max
{a}	286 (Minimal)	{a,c}	372 (Not Minimal)
{b}	285 (Minimal)	{b,c}	371 (Not Minimal)
{c}	172 (Minimal)		

Maximum IWI-support-max threshold $\xi = 390$.

Considering minimal IWIs allows the expert to focus her/his attention on the *smallest* CPU sets that contain at least one underutilized/idle CPU and, thus, reduces the bias due to the possible inclusion of highly weighted items in the extracted patterns. In Table 2 IWIs are partitioned between minimal and not, as indicated next to each itemset IWI-support-min value (minimal/not minimal).

Task (B) entails discovering IWIs and MIWIs which include item(s) having maximal local interest within each transaction by exploiting the IWI-support-max measure. Table 3 reports the IWIs mined from Table 1 by enforcing a maximum IWI-support-max threshold equal to 390. They may represent sets of CPUs which contain *only* under-utilized/idle CPUs at each sampled time instant. Note that, in this context, discovering large CPU combinations may be deemed particularly useful by domain experts, because they represent large resource sets which could be reallocated.

To accomplish tasks (A) and (B), we present two novel algorithms, namely Infrequent Weighted Itemset Miner (IWI Miner) and Minimal Infrequent Weighted Itemset Miner (MIWI Miner), which perform IWI and MIWI mining driven by IWI-support thresholds. IWI Miner and MIWI Miner are FP-Growth-like mining algorithms [10], whose main features may be summarized as follows: (i) Early FP-tree node pruning driven by the maximum IWI-support constraint, i.e., early discarding of part of the search space thanks to a novel item pruning strategy, and (ii) cost function-independence, i.e., they work in the same way regardless of which constraint (either IWI-support-min or IWI-support-max) is applied, (iii) early stopping of the recursive FP-tree search in MIWI Miner to avoid extracting non-minimal IWIs. As shown in Section 4, Property (ii) makes tasks (A) and (B) equivalent, from an algorithmic point of view, as long as a preliminary data transformation step, which adapts data weights according to the selected aggregation function, is applied before accomplishing the mining task.

Experiments, performed on both synthetic and real-life data sets, show efficiency and effectiveness of the proposed approach. In particular, they show the characteristics and usefulness of the itemsets discovered from data coming from benchmarking and real-life multi-core systems, as well as the algorithm scalability.

This paper is organized as follows. Section 2 discusses and compares related work with the proposed approach. Section 3 introduces preliminary definitions and notations as well as formally states the IWI and MIWI mining tasks addressed by this paper. Section 4 describes the proposed algorithms, while Section 5 evaluates efficiency and effectiveness of the proposed approach. Finally, Section 6 draws conclusions and presents future work.

2 Previous Work

Frequent itemset mining is a widely used data mining technique that has been introduced in [1]. In the traditional itemset mining problem items belonging to transactional data are treated equally. To allow differentiating items based on their interest or intensity within each transaction, in [4] the authors focus on discovering more informative association rules, i.e., the weighted association rules (WAR), which include weights denoting item significance. However, weights are introduced only during the rule generation step after performing the traditional frequent itemset mining process. The first attempt to pushing item weights into the itemset mining process has been done in [5]. It proposes to exploit the anti-monotonicity of the proposed weighted support constraint to drive the Apriori-based itemset mining phase. However, in [4], [5] weights have to be preassigned, while, in many real-life cases, this might not be the case. To address this issue, in [6] the analyzed transactional data set is represented as a bipartite hub-authority graph and evaluated by means of a well-known indexing strategy, i.e., HITS [11], in order to automate item weight assignment. Weighted item support and confidence quality indexes are defined accordingly and used for driving the itemset and rule mining phases. This paper differs from the above-mentioned approaches because it focuses on mining infrequent itemsets from weighted data instead of frequent ones. Hence, different pruning techniques are exploited.

A related research issue is probabilistic frequent itemset mining [12], [13]. It entails mining frequent itemsets from uncertain data, in which item occurrences in each transaction are uncertain. To address this issue, probabilistic models have been constructed and integrated in Apriori-based [12] or projection-based [14] algorithms. Although probabilities of item occurrence may be remapped to weights, the semantics behind probabilistic and weighted itemset mining is radically different. In fact, the probability of occurrence of an item within a transaction may be totally uncorrelated with its relative importance. For instance, an item that is very likely to occur in a given transaction may be deemed the least relevant one by a domain expert. Furthermore, this paper differs from the above-mentioned approaches as it specifically addresses the infrequent itemset mining task.

A parallel effort has been devoted to discovering rare correlations among data, i.e., the infrequent itemset mining problem [7], [8], [9], [15], [16], [17]. For instance, in [7], [8] a recursive algorithm for discovering minimal unique itemsets from structured data sets, i.e., the shortest itemsets with absolute support value equal to 1, is proposed. They extend a preliminary algorithm version, previously proposed in [18], by specifically tackling algorithm scalability issues. The authors in [9] first addressed the issue of discovering minimal infrequent itemsets, i.e., the itemsets that satisfy a maximum support threshold and do not contain any infrequent subset, from transactional data sets. More recently, in [17] an FP-Growth-like algorithm for mining minimal infrequent itemsets has also been proposed. To reduce the computational time the authors introduce the concept of residual tree, i.e., an FP-tree associated with a generic item i that represents data set transactions obtained by removing i. Similarly to [17], in this paper we propose an FP-tree-based approach to mining infrequent itemsets. However, unlike all of the above-mentioned approaches, we face the issue of treating items differently, based on their relative importance in each transaction, in the discovery of infrequent itemsets from weighted data. Furthermore, unlike [17], we adopt a different item pruning strategy tailored to the traditional FP-tree structure to perform IWI mining efficiently. An attempt to exploit infrequent itemsets in mining positive and negative association rules has also been made in [15], [16]. Since infrequent itemset mining is considered an intermediate step, their focus is radically different from that of this paper.

3 Problem Statement

This paper addresses the problem of mining infrequent itemsets from transactional data sets. Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of data items. A transactional data set $T = \{t_1, t_2, \ldots, t_n\}$ is a set of transactions, where each transaction t_q ($q \in [1, n]$) is a set of items in $\mathcal{I}$ and is characterized by a transaction ID (tid).

An itemset I is a set of data items [1]. More specifically, we denote as k-itemset a set of k items in $\mathcal{I}$. The support (or occurrence frequency) of an itemset is the number of transactions containing I in T. An itemset I is infrequent if its support is less than or equal to a predefined maximum support threshold ξ. Otherwise, it is said to be frequent [1]. An infrequent itemset is said to be *minimal* if none of its subsets is infrequent [7]. Given a transactional data set T and a maximum support threshold ξ, the infrequent (minimal) itemset mining problem entails discovering all infrequent (minimal) itemsets from T [9].

Unfortunately, using the traditional support measure for driving the itemset mining process entails treating items and transactions equally, even if they do not have the same relevance in the analyzed data set. To treat items differently within each transaction we introduce the concept of weighted item as a pair $\langle i_k, w_k^q \rangle$, where $i_k \in \mathcal{I}$ is an item contained in $t_q \in T$, while w_k^q is the weight associated with i_k that characterizes its local interest/intensity in t_q [4]. Concepts of weighted transaction and weighted transactional data set are defined accordingly as sets of weighted items and weighted transactions, respectively.

Definition 1 (Weighted transactional data set). *Let $\mathcal{I} = \{i_1, i_2, \ldots, i_m\}$ be a set of items. A weighted transactional data set T^w is a set of weighted transactions, where each weighted transaction t_q^w is a set of weighted items $\langle i_k, w_k^q \rangle$ such that $i_k \in \mathcal{I}$ and w_k^q is the weight associated with i_k in t_q^w.*

Note that, in general, weights could be either positive, null, or negative numbers. Itemsets mined from weighted transactional data sets are called *weighted itemsets*. Their expression is similar to the one used for traditional itemsets, i.e., a weighted itemset is a subset of the data items occurring in a weighted transactional data set. The problem of mining itemsets by considering weights associated with each item is known as the *weighted itemset mining problem* [4]. For the sake of simplicity, by convenient abuse of

notation weighted itemsets will be denoted by itemsets whenever it is clear from the context. For the same reason, a generic weighted data set and transaction are denoted by T and t_q, respectively, throughout the paper.

Consider again the example data set reported in Table 1. It is a weighted transactional data set T composed of 6 transactions, each one including four weighted items. Since, for instance, the weight of item a in tid 1 (0) is significantly lower than the ones of b (100) and d (71) then a, b, and d should be treated differently during the mining process.

This paper focuses on considering item weights in the discovery of infrequent itemsets. To this aim, the problem of evaluating itemset significance in a given weighted transactional data set is addressed by means of a two-step process. Firstly, the weight of an itemset I associated with a weighted transaction $t_q \in T$ is defined as an aggregation of its item weights in t_q. Secondly, the significance of I with respect to the whole data set T is estimated by combining the itemset significance weights associated with each transaction.

In traditional itemset mining, an itemset I is said to *cover* a given transaction t_q if $I \subseteq t_q$. For our purposes, we define two different weighting functions, i.e., the minimum and the maximum functions, which associate the minimum and the maximum weight relative to items in I with each covered transaction t_q. As discussed in the following, minimum and maximum are weighting functions which are deemed suitable for performing different targeted analysis.

Definition 2 (Weighting functions). *Let $t_q = \{\langle i_1, w_1^q\rangle, \langle i_2, w_2^q\rangle, \ldots, \langle i_l, w_l^q\rangle\}$ be a weighted transaction, and $IS(t_q) = \{i_k | \langle i_k, w_k^q\rangle \in t_q$ for some $w_k^q\}$ the set of items in t_q. Let I be an itemset covering t_q, i.e., $I \subseteq IS(t_q)$. The minimum weighting function is defined by $W_{\min}(I, t_q) = \min_{j|\, i_j \in I} w_j^q$. The maximum weighting function is defined by $W_{\max}(I, t_q) = \max_{j|\, i_j \in I} w_j^q$.*

Notice that weighting functions are defined only when a transaction is *covered* by the given itemset. Selecting the minimum item weight within each transaction allows the expert to focus her/his attention on the rare itemsets that contain *at least* one lowly weighted item (e.g., an underutilized/idle CPU). On the other hand, using the maximum weighting function allows considering rare itemsets that contain *only* lowly weighted items.

Similarly to the traditional absolute support measure,[1] the IWI-support of an itemset is defined as its weighted observed frequency of occurrence in the source data, where for each transaction itemset occurrences are weighted by the output of the chosen weighting function.

Definition 3 (IWI-support). *Let I be an itemset, T a weighted transactional data set, $IS(t_q)$ the set of items in $t_q \in T$, and W_f a minimum or maximum weighting function. The IWI-support of I in T is defined as follows.*

$$\text{IWI-support}(I, T) = \sum_{t_q \in T|\ I \subseteq IS(t_q)} W_f(I, t_q).$$

1. The absolute support of an itemset is defined as the number of occurrences of the itemset in the source data [1].

If $W_f = W_{\min}$ then the IWI-support measure is denoted as *IWI-support-min*. Otherwise (i.e., in case $W_f = W_{\max}$), it is denoted as *IWI-support-max*.

Consider again the data set in Table 1. The IWI-support-min of $\{a,b\}$ is 128, because its weights referring to the transactions with tids 1-6 are 0, 0, 0, 0, 71, and 57. Instead, the IWI-support-max of $\{a,b\}$ is 443, because the assigned weights are 100, 43, 43, 100, 86, and 71.

The IWI-support measures are characterized by the following notable properties.

Property 1 (Equivalence between the IWI-support measure and the traditional support measure). *Let T be a weighted transactional data set that exclusively contains items with weight 1, and T^u its corresponding unweighted version. Let W_f be an arbitrary aggregation function. The IWI-support value of an itemset I in T corresponds to its traditional support value in T^u, i.e., IWI-support(I, T) = support(I, T^u).*

Proof. Since any item in T has a weight equal to 1 then $W_f(I, t_q) = 1$ for any $t_q \in T$ covered by I, where $W_f(I, t_q)$ may be either $W_{\min}(I, t_q)$ or $W_{\max}(I, t_q)$. Thus, by Definition 3, the IWI-support is equal to the number of the transactions covered by I. Hence, it follows that IWI-support(I, T) = support(I, T^u). □

The maximum IWI-support-min constraint is also characterized by the monotonicity property.

Property 2 (Monotonicity property of the maximum IWI-support-min constraint). *Let T be a weighted transactional data set and $\preceq$ a precedence relation holding between pairs of weighted itemsets X and Y, such that $X \preceq Y$ holds if and only if $X \subseteq Y$. Let ξ be a maximum IWI-support threshold. The maximum IWI-support constraint IWI-support-min $(X, T) \leq \xi$ is monotone with respect to $\preceq$.*

Proof. Let X and Y be two arbitrary weighted itemsets such that $X \preceq Y$. Since $X \subseteq Y$ the transactions covered by X in T are a subset of those covered by Y. Moreover, given an arbitrary weighted transaction t_q covered by both X and Y, it trivially follows from Definition 2 that $W_{\min}(X, t_q) \geq W_{\min}(Y, t_q)$. Hence, the following inequality holds. IWI-support-min$(X, T) = \sum_{t_q \in T|\ X \subseteq IS(t_q)} W_{\min}(X, t_q) \geq \sum_{t_q \in T|\ Y \subseteq IS(t_q)} W_{\min}(Y, t_q)$ = IWI-support-min(Y, T), where $IS(t_q)$ is the set of items in t_q.

It follows that the maximum IWI-support constraint is monotone with respect to the precedence relation $\preceq$. □

Problem statement. Given a weighted transactional data set T, an IWI-support measure based on a weighting function W_f (let it be either the minimum or the maximum weighting function), and a maximum IWI-support threshold ξ, this paper addresses the following tasks:

A. discovering all IWIs that satisfy ξ in T,
B. discovering all MIWIs that satisfy ξ in T.

While task (A) entails discovering all IWIs (minimal and not), task (B) selects only minimal IWIs, which represent the *smallest* infrequent item combinations satisfying the constraints.

If W_f is the minimum weighting function then the IWI-support-min measure is considered and tasks (A) and (B) select all IWIs/MIWIs that include *at least one* lowly

weighted item within each transaction. Otherwise, i.e., in case W_f is the maximum weighting function, the IWI-support-max measure is considered and, thus, tasks (A) and (B) select all IWIs/MIWIs that include *only* lowly weighted items within each transaction.

4 The Algorithms

This section presents two algorithms, namely Infrequent Weighted Itemset Miner and Minimal Infrequent Weighted Itemset Miner, which address tasks (A) and (B), stated in Section 3, respectively. The proposed algorithms are FP-Growth-like miners whose main characteristics may be summarized as follows: (i) The use of the equivalence property, stated in Property 3, to adapt weighted transactional data to traditional FP-tree-based itemset mining, and (ii) the exploitation of a novel FP-tree pruning strategy to prune part of the search space early. This section is organized as follows. Section 4.1 formally states the weighted transaction equivalence property and describes the FP-tree pruning strategy, while IWI Miner and MIWI Miner algorithms are thoroughly described in Sections 4.2 and 4.3.

4.1 Weighted Transaction Equivalence

The weighted transaction equivalence establishes an association between a weighted transaction data set T, composed of transactions with arbitrarily weighted items within each transaction (Cf. Definition 1), and an equivalent data set TE in which each transaction is exclusively composed of equally weighted items. To this aim, each weighted transaction $t_q \in T$ corresponds to an *equivalent weighted transaction set* $TE_q \subseteq TE$, which is a subset of TE's transactions $\{te_1, \ldots, te_k\}$. Item weights in t_q are spread, based on their relative significance, among their equivalent transactions in TE_q. The proposed transformation is particularly suitable for compactly representing the original data set by means of an FP-tree index [10]. As shown in Sections 4.2 and 4.3, the generated FP-tree will be used to tackle the (M)IWI mining problem effectively and efficiently. The equivalent weighted transaction set is defined as follows.

Definition 4 (Equivalent weighted transaction set). *Let T be a weighted transactional data set and TE its corresponding equivalent data set. Let t_q be a weighted transaction in T, $\{\langle i_1, w_1^q\rangle, \langle i_2, w_2^q\rangle, \ldots, \langle i_l, w_l^q\rangle\}$ the enumeration of all weighted items in t_q and W_f a minimum or maximum weighting function. Let $\mathcal{W}_q = \{w_1^q, w_2^q, \ldots, w_l^q\}$ be the set of weights associated with items in t_q and $\overline{w}_{p,W_f}^q$ the pth distinct least $w_j^q \in \mathcal{W}_q$ in case W_f is minimum, or the pth distinct highest $w_j^q \in \mathcal{W}_q$ if W_f is maximum. The equivalent weighted transaction set $TE_q = \{te_1, \ldots, te_k\}$ associated with t_q is a set of k weighted transactions te_p ($p \in [1,k]$) such that for all $1 \le p \le k$, $te_p \in TE$. Each te_p includes items with weight wt_p and is defined as follows:*

$$te_p = \begin{cases} \{\langle i_j, wt_p\rangle | \langle i_j, w_j^q\rangle \in t_q \wedge w_j^q \ge \overline{w}_{p,W_f}^q\} & \text{if } W_f = W_{\min}, \\ \{\langle i_j, wt_p\rangle | \langle i_j, w_j^q\rangle \in t_q \wedge w_j^q \le \overline{w}_{p,W_f}^q\} & \text{if } W_f = W_{\max} \end{cases}$$

where

$$wt_p = \begin{cases} \overline{w}_{1,W_f}^q & \text{if } p = 1, \\ \overline{w}_{p,W_f}^q - \overline{w}_{(p-1),W_f}^q & \text{otherwise.} \end{cases}$$

TABLE 4
Equivalent Weighted Transaction Associated with the Transaction with Tid 1 in the Example Data Set

Tid	Equivalent Weighted transaction	Original transaction
Minimum weighting function		
1.a	$\langle a,0\rangle\langle b,0\rangle\langle c,0\rangle\langle d,0\rangle$	$\langle a,0\rangle\langle b,100\rangle\langle c,57\rangle\langle d,71\rangle$
1.b	$\langle b,57\rangle\langle c,57\rangle\langle d,57\rangle$	
1.c	$\langle b,14\rangle\langle d,14\rangle$	
1.d	$\langle b,29\rangle$	
Maximum weighting function		
1.a	$\langle a,100\rangle\langle b,100\rangle\langle c,100\rangle\langle d,100\rangle$	$\langle a,0\rangle\langle b,100\rangle\langle c,57\rangle\langle d,71\rangle$
1.b	$\langle a,-29\rangle\langle c,-29\rangle\langle d,-29\rangle$	
1.c	$\langle a,-14\rangle\langle c,-14\rangle$	
1.d	$\langle a,-57\rangle$	

The equivalent version TE of a weighted transactional data set T (Cf. Definition 1) is the union of all equivalent transactional sets associated with each weighted transaction. Consider, for instance, the example data set reported in Table 1. The equivalent versions of the transaction with tid 1 obtained by using the minimum and the maximum weighting functions are reported in the left-hand side of Table 4, where the original transaction and its equivalent versions are put side by side for convenience.

Readers can notice that each transaction in the equivalent data sets only includes equally weighted items. In both cases, the transaction with tid 1 in Table 1 is mapped to the equivalent transactions with tids 1.a, 1.b, 1.c, and 1.d. When using the minimum weighting function, the equivalence procedure first considers the lowest among the weights occurring in the original transaction as current reference weight w_{ref} (e.g., the weight 0 associated with item a in tid 1) and generates an equivalent transaction of equally weighted items (tid 1.a). Next, an iterative procedure only considers, for the subsequent steps, the set S of items contained in the original transaction and having weight strictly higher than w_{ref} (e.g., items b, c, and d in tid 1). Items in S are combined in a new equivalent transaction (tid 1.b). At this stage, the new value of reference weight w_{ref} for tid 1.b is equal to the minimum weight among the items in S reduced by the previous reference weight value (e.g., the minimum between 100 (100-0), 57 (57-0), and 71 (71-0)). Next, set S is further pruned by excluding items with weight w_{ref} once more. The above procedure is iterated until S is empty. In case the maximum weighting function is adopted, the procedure is analogous, but the highest transaction weight is selected at each step instead of the lowest one. Note that reducing item weights by the local maximum weight may yield negatively weighted equivalent transactions.

The IWI-support of a weighted itemset in a weighted transactional data set corresponds to the one evaluated on the equivalent data set. We denote this property as the equivalence property.

Property 3 (Equivalence property). *Let T be a weighted transactional data set and TE its equivalent version. The IWI-support values of an itemset I in T and TE are equal.*

Proof. Let $t_q \in T$ be a weighted transaction covered by I and $TE_q = \{te_1, \ldots, te_k\}$ its equivalent transaction set. Let $matched = \{\langle i_j, w_j^q\rangle \in t_q |\ i_j \in I\}$ be the set of matched weighted items and $IS(t_q)$ and $IS(te_p)$ the set of items in

t_q and te_p, respectively. Consider the IWI-support-min measure first. Let $\langle i_l, w_l^q \rangle \in matched$ be the least weighted item in $matched$. By Definition 2, the following equality holds: $W_{\min}(I, t_q) = w_l^q$. Furthermore, by Definition 4, any transaction $te_p \in TE_q$ containing i_l includes all the other items in $matched$ as well. Thus, the IWI-support-min of I in TE_q could be rewritten as follows: $\sum_{te_p | i_l \in IS(te_p)} W_{min}(I, te_p) = w^q = W_{min}(I, t_q)$. Hence, IWI-support-min(I,TE) $= \sum_{TE_q \in TE}$ IWI-support-min(I, TE_q) $= \sum_{te_p \in TE_q \mid I \subseteq IS(te_p) \wedge TE_q \in TE} W_{\min}(I, te_p) =$ $\sum_{t_q \in T | I \subseteq IS(t_q)} W_{\min}(I, t_q) =$ IWI-support-min(I, T).

Consider now the IWI-support-max measure. Let $\langle i_h, w_h^q \rangle \in matched$ be the maximally weighted item in $matched$. By Definition 2, the following equality holds: $W_{\max}(I, t_q) = w_h^q$. Furthermore, by Definition 4, $\sum_{te_p | i_h \in IS(te_p)} W_{\max}(I, te_p) = w_h^q = W_{\max}(I, t_q)$. Hence, IWI-support-max $(I, TE) =$ IWI-support-max(I,T) follows analogously to the former case. □

Complexity analysis. The data set transformation procedure generates, for each transaction, a number of equivalent transactions at most equal to the original transaction length. A lower number of equivalent transactions is generated when two or more items have the same weight in the original transaction. The product of the original data set cardinality and its *longest* transaction length can be considered a preliminary upper bound estimate of the equivalent data set cardinality. However, in real data sets many transactions are usually shorter than the longest one and many items have equal weight in the same transaction. This reduces the number of generated equivalent transactions significantly. As confirmed by the experimental results achieved on real and synthetic data (see Section 5), the scaling factor becomes actually lower than the *average* transaction length, which could be considered a more realistic upper bound estimate.

IWI Miner and MIWI Miner exploit the equivalence property to address tasks (A) and (B), stated in Section 3, efficiently and effectively. In the following section a thorough description of the proposed algorithms is given.

4.2 The Infrequent Weighted Itemset Miner Algorithm

Given a weighted transactional data set and a maximum IWI-support (IWI-support-min or IWI-support-max) threshold ξ, the Infrequent Weighted Itemset Miner algorithm extracts all IWIs whose IWI-support satisfies ξ (Cf. task (A)). Since the IWI Miner mining steps are the same by enforcing either IWI-support-min or IWI-support-max thresholds, we will not distinguish between the two IWI-support measure types in the rest of this section.

IWI Miner is a FP-growth-like mining algorithm [10] that performs projection-based itemset mining. Hence, it performs the main FP-growth mining steps: (a) FP-tree creation and (b) recursive itemset mining from the FP-tree index. Unlike FP-Growth, IWI Miner discovers infrequent weighted itemsets instead of frequent (unweighted) ones. To accomplish this task, the following main modifications with respect to FP-growth have been introduced: (i) A novel pruning strategy for pruning part of the search space early and (ii) a slightly modified FP-tree structure, which allows storing the IWI-support value associated with each node.

To cope with weighted data, an equivalent data set version is generated (Cf. Definition 4) and used to populate the FP-tree structure. The FP-tree is a compact representation of the original data set residing in main memory [10]. Unlike the traditional FP-tree creation, items in the FP-tree header table are sorted by their IWI-support value instead of by their traditional support value. Furthermore, the insertion of an equivalent weighted transaction te_p, whose items are all characterized by the same weight wt_p, requires increasing the weights associated with the covered tree nodes by wt_p rather than 1.

To reduce the complexity of the mining process, IWI Miner adopts an FP-tree node pruning strategy to early discard items (nodes) that could never belong to any itemset satisfying the IWI-support threshold ξ. In particular, since the IWI-support value of an itemset is at least equal to the one associated with the leaf node of each of its covered paths, then the IWI-support value stored in each leaf node is a lower bound IWI-support estimate for all itemsets covering the same paths. Hence, an item (i.e., its associated nodes) is pruned if it appears *only* in tree paths from the root to a leaf node characterized by IWI-support value greater than ξ. The pruning property could be formalized as follows.

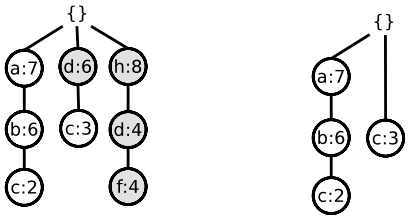
Property 4 (Pruning property). *Let T be a weighted transactional data set and FP_T the FP-tree associated with T. Let i be an arbitrary item and $N_i = \{ni_1, , ni_k\}$ the set of nodes associated with i in FP_T. If for each path from $ni_j \in N_i$ to a leaf in FP_T the leaf node is characterized by an IWI-support value greater than ξ then i cannot be contained in any IWI satisfying the IWI-support constraint.*

Proof. Suppose that i is an item such that, for each path from $ni_j \in N_i$ to a leaf, the leaf node is characterized by an IWI-support greater than ξ. Hence, for every path pi_l from the FP-tree root to a leaf through $ni_j \in N_i$ the leaf node is characterized by an IWI-support higher than ξ. Consider now a generic itemset Ii_l composed of items in pi_l. Since the IWI-support of the leaf node of pi_l is greater than ξ then the IWI-support of Ii_l is greater than ξ, i.e., Ii_l does not satisfy the maximum IWI-support constraint. Thus, every itemset containing i does not satisfy ξ. □

Consider, for example, the FP-tree in Fig. 1a and suppose discovering IWIs by enforcing an IWI-support-min threshold ξ equal to 2.5. Item d is included in the paths {d,c} and {h,d,f}, whose leaf nodes have IWI-support equal to 3 and 4, respectively (see Fig. 1a). Since d is contained only in paths whose leaf nodes have an IWI-support value greater than ξ, it can be pruned. The same consideration holds for f and h. Instead, b is contained in a path ({a,b,c}) associated with a leaf node having an IWI-support value lower than ξ. Thus, it should be kept. In fact, {a,b,c} is an IWI characterized by an IWI-support-min equal to 2. The result of the pruning step is reported in Fig. 1b.

4.2.1 Algorithm Pseudocode

In Algorithm 1 the IWI Miner pseudocode is reported. The first steps (lines 2-9 of Algorithm 1) generate the FP-tree



(a) FP-tree before pruning. (b) FP-tree after pruning.

Fig. 1. Example of node pruning. Maximum IWI-support threshold $\xi = 2.5$.

associated with the input weighted data set T. Then, the recursive mining process is invoked on the constructed FP-tree (line 10). The FP-tree is initially populated with the set of equivalent transactions generated from T. For each weighted transaction $t_q \in T$ the equivalent set (line 5) is generated by applying function *equivalentTransactionSet*, which implements the transactional data set equivalence transformation described in Section 4.1.

Algorithm 1 IWI-Miner(T, ξ)

Input: T, a weighted transactional dataset
Input: ξ, a maximum IWI-support threshold
Output: $\mathcal{F}$, the set of IWIs satisfying ξ
1: $\mathcal{F}=\emptyset$ /* *Initialization* */
/* *Scan T and count the IWI-support of each item* */
2: countItemIWI-support(T)
3: $Tree \leftarrow$ a new empty FP-tree; /* *Create the initial FP-tree from T* */
4: **for all** weighted transaction t_q in T **do**
5: $TE_q \leftarrow$ equivalentTransactionSet(t_q)
6: **for all** transaction te_j in TE_q **do**
7: insert te_j in $Tree$
8: **end for**
9: **end for**
10: $\mathcal{F} \leftarrow$ IWIMining($Tree$, ξ, null)
11: **return** $\mathcal{F}$

Algorithm 2 IWIMining($Tree$, ξ, $prefix$)

Input: $Tree$, a FP-tree
Input: ξ, a maximum IWI-support threshold
Input: $prefix$, the set of items/projection patterns with respect to which *Tree* has been generated
Output: $\mathcal{F}$, the set of IWIs extending $prefix$
1: $\mathcal{F}=\emptyset$
2: **for all** item i in the header table of $Tree$ **do**
3: $I = prefix \cup \{i\}$ /* *Generate a new itemset I by joining $prefix$ and i with IWI-support set to the IWI-support of item i* */
/* *If I is infrequent store it* */
4: **if** IWI-support(I)$\leq \xi$ **then**
5: $\mathcal{F} \leftarrow \mathcal{F} \cup \{I\}$
6: **end if**
/* *Build I's conditional pattern base and I's conditional FP-tree* */
7: $condPatterns \leftarrow$ generateConditionalPatterns($Tree$, I)
8: $Tree_I$ = createFP-tree($condPatterns$)
/* *Select the items that will never be part of any infrequent itemset* */
9: $prunableItems \leftarrow$ identifyPrunableItems($Tree_I$, ξ)
/* *Remove from $Tree_I$ the nodes associated with prunable items* */
10: $Tree_I \leftarrow$ pruneItems($Tree_I$, $prunableItems$)
11: **if** $Tree_I \neq \emptyset$ **then**
12: $\mathcal{F} \leftarrow \mathcal{F} \cup$ IWIMining($Tree_I$, ξ, I) /* *Recursive mining* */
13: **end if**
14: **end for**
15: **return** $\mathcal{F}$

Once a compact FP-tree representation of the weighted data set T has been created, the recursive itemset mining process is executed (Algorithm 1, line 10). A pseudocode of the mining procedure is given in Algorithm 2. Since IWI Miner relies on a projection-based approach [10], items belonging to the header table associated with the input FP-tree are iteratively considered (lines 2-14). Initially, each item is combined with the current prefix to generate a new itemset I (line 3). If I is infrequent, then it is stored in the output IWI set $\mathcal{F}$ (lines 4-6). Then, the FP-tree projected with respect to I is generated (lines 7-8) and the IWIMining procedure is recursively applied on the projected tree to mine all infrequent extensions of I (line 12). Unlike traditional FP-Growth-like algorithms [10], IWI Miner adopts a different pruning strategy (see lines 9-10). According to Property 4, *identifyPrunableItems* procedure visits the FP-tree and identifies items that are only included in paths whose leaves have an IWI-support above ξ. Since they cannot be part of any IWI, they are pruned (line 10).

4.3 The Minimal Infrequent Weighted Itemset Miner Algorithm

Given a weighted transactional data set and a maximum IWI-support (IWI-support-min or IWI-support-max) threshold ξ, the Minimal Infrequent Weighted Itemset Miner algorithm extracts all the MIWIs that satisfy ξ.

The pseudocode of the MIWI Miner algorithm is similar to the one of IWI Miner, reported in Algorithm 1. Hence, due to space constraints, the pseudocode is not reported. However, in the following, the main differences with respect to IWI Miner are outlined. At line 10 of Algorithm 1, the *MIWIMining* procedure is invoked instead of *IWIMining*. The MIWIMining procedure is similar to IWIMining. However, since MIWI Miner focuses on generating only minimal infrequent patterns, the recursive extraction in the MIWIMining procedure is stopped as soon as an infrequent itemset occurs (i.e., immediately after line 5 of Algorithm 2). In fact, whenever an infrequent itemset I is discovered, all its extensions are not minimal.

5 Experiments

We evaluated IWI Miner and MIWI Miner performance by means of a large set of experiments addressing the following issues: (i) Expert-driven validation from real weighted data (Section 5.2), (ii) algorithm performance analysis (Section 5.3), and (iii) algorithm scalability analysis (Section 5.4). All the experiments were performed on a 3.0 GHz Intel Xeon system with 4 GB RAM, running Ubuntu 10.04 LTS. The IWI Miner and MIWI Miner algorithms were implemented in the C++.

5.1 Data Set Description

The characteristics of the evaluated data sets are summarized in the following.

Real-life data sets. To validate the usefulness of the proposed algorithms we analyzed 10 collections, each one composed of 31 real-life weighted data sets. Each collection was obtained by measuring the CPU usage of a multicore system when executing a different benchmark, among the ones available at http://www.dacapobench.org/. The

benchmarks (e.g., avrova, batik) are exploited to analyze multi-core system performance by running a number of standard programs. Each data set reports the per-core usage rate of a multi-core machine during the execution of a benchmark. In particular, each weighted transaction corresponds to a distinct reading sampled at a fixed point of time, where its weighted items represent the per-core usage rate measures. Data sets have been populated by performing tests on a IBM Power7 machine equipped with four CPUs with 8 cores each and enabling/disabling the available cores. More specifically, for each collection, each of the considered data sets represents the CPU usage collected with a different multi-core setting, i.e., the first data set is relative to a 2-core setting (i.e., 2 out of 32 cores were enabled, while all the others were temporarily idle), the second one to a 3-core setting, etc. Each data set is characterized by a number of items per transaction equal to the number of enabled cores, i.e., the transaction length is equal to the number of enabled cores. The sampling rate is around 1 s for the 2-core setting, while it decreases when a larger number of cores is enabled. Hence, the data set cardinality is usually higher for highly parallelized test settings.

Synthetic data sets. We also exploited a synthetic data set generator to evaluate algorithm performance and scalability. The data generator is based on the IBM data generator [19]. It allows generating transactional synthetic data sets by setting (i) the data set cardinality, (ii) the average transaction length, and (iii) the item correlation factor. To assign weights to data generated by the IBM generator we integrated a synthetic weight generator. The newly proposed data generator version may assign to each data item a weight according to two different distributions, chosen as representative among all the possible data distributions, i.e, the uniform data distribution and the Poisson distribution. When not otherwise specified, in the following experiments item weights are selected in the range [1, 100]. The synthetic weighted data generator is publicly available at http://dbdmg.polito.it/∼paolo/.

5.2 Knowledge Discovery from Real Benchmark Data Sets

Analysis and monitoring of multi-core system usage is commonly devoted to (i) detecting system malfunctioning, (ii) optimizing computational load balancing and resource sharing, and (iii) performing system resizing. To address these issues we focus on analyzing and validating, with the help of a domain expert, the usefulness of the patterns extracted by IWI Miner and MIWI Miner from the real-life data sets described in Section 5.1.

Since item weights occurring in the analyzed data sets represent core load rates, IWIs and MIWIs that satisfy a maximum IWI-support value represent combinations of underutilized or idle cores. More specifically, when setting an IWI-support-min threshold, MIWIs represent the smallest core combinations that contain *at least* one underutilized core. Instead, when setting the IWI-support-max threshold, MIWIs represent the smallest core combinations that contain *only* underutilized cores.

TABLE 5
Number of MIWIs Mined by MIWI Miner by Setting Different Values of Maximum IWI-Support-Min and IWI-Aupport-Max Threshold ξ

Dataset	Usage $\leq$10%			Usage $\leq$30%		
		Num. MIWIs			Num. MIWIs	
	ξ	IWI-sup-min	IWI-sup-max	ξ	IWI-sup-min	IWI-sup-max
D2cores	790	0	0	2340	0	0
D3cores	780	0	0	2250	0	0
D4cores	750	0	0	2310	0	0
D5cores	770	0	0	2340	1	0
D6cores	780	0	0	2370	10	0
D7cores	790	10	0	2400	15	1
D8cores	800	20	0	2460	6	5
D9cores	820	33	0	2850	9	6
D10cores	950	45	0	3000	12	6
D11cores	1000	54	0	2790	10	9
D12cores	930	52	3	2940	11	11
D13cores	980	39	3	3150	12	11
D14cores	1050	59	2	3150	13	12
D15cores	1050	62	3	3180	15	15
D16cores	1060	58	4	3150	15	14
D17cores	1050	50	6	3300	16	15
D18cores	1100	59	6	3300	17	17
D19cores	1100	43	9	3300	18	17
D20cores	1100	61	9	3360	19	19
D21cores	1120	79	8	3360	21	21
D22cores	1120	66	10	3360	22	22
D23cores	1120	84	10	3390	23	23
D24cores	1130	50	14	3450	23	23
D25cores	1150	74	13	3480	25	25
D26cores	1160	73	14	3570	25	25
D27cores	1190	74	15	3480	27	27
D28cores	1160	58	18	3420	28	28
D29cores	1140	51	20	3510	28	27
D30cores	1170	69	19	3480	29	29
D31cores	1160	76	19	3160	30	30
D32cores	1180	84	20	3120	32	32

Avrova benchmark.

Consider the avrova benchmark first. In Table 5 the number of MIWIs mined by MIWI Miner by setting different values of the maximum IWI-support-min and IWI-support-max thresholds (relative to 10 and 30 percent core usage rates, respectively) is reported. Since core usage rates range from 0 to 100, for each data set the absolute threshold relative to the x percent usage rate is given by $|T| \cdot x$ percent, where $|T|$ is the data set cardinality (i.e., the number of samples). The corresponding values are given in columns 2 and 5 in Table 5. A preliminary analysis of the MIWIs extracted by enforcing different IWI-support-max thresholds shows that, unexpectedly, some cores become underutilized when the workflow is allocated over several cores. For instance, when seven or more cores are simultaneously enabled some of them have an average usage rate lower than 30 percent in all sampled points of time, i.e., at least one MIWI is mined. Similarly, when 12 cores are simultaneously active, there exist three core combinations composed of cores with average usage rate lower than 10 percent. Their inspection allows the expert to drive the process of system resizing.

To detect system malfunctioning or analyze the maximum application parallelism, the expert may also perform a worst-case analysis by discovering situations in which the simultaneous enabling of multiple cores may cause one or more cores to remain, possibly in an alternate fashion, completely idle (i.e., usage rate $= 0$). This situation may be due to either specific scheduling problems (e.g., the scheduler may fail to consider some of the available cores for its internal allocation policy) or

TABLE 6
Avrova Benchmark

Dataset	Num. of original trans. $\|T\|$	Num. of equivalent trans. $\|TE\|$	Size increase ratio $\|TE\|/\|T\|$	Num. of MIWIs with ξ=0	Num. of IWIs with ξ=0
D2cores	79	155	1.96	0	0
D3cores	78	221	2.83	0	0
D4cores	75	291	3.88	0	0
D5cores	77	370	4.81	0	0
D6cores	78	437	5.60	0	0
D7cores	79	511	6.47	0	0
D8cores	80	579	7.24	0	0
D9cores	82	606	7.39	0	0
D10cores	95	723	7.61	0	0
D11cores	100	769	7.69	11	70
D12cores	93	726	7.81	27	781
D13cores	98	756	7.71	42	1,908
D14cores	105	823	7.84	21	9,592
D15cores	105	821	7.82	187	12,931
D16cores	106	824	7.77	135	57,797
D17cores	105	811	7.72	190	121,724
D18cores	110	856	7.78	208	244,546
D19cores	110	882	8.02	175	513,470
D20cores	110	892	8.11	166	1e06
D21cores	112	942	8.41	552	2e06
D22cores	112	947	8.46	484	4e06
D23cores	112	993	8.87	562	8e06
D24cores	113	953	8.43	444	16e06
D25cores	115	1,086	9.44	576	33e06
D26cores	116	1,060	9.14	428	67e06
D27cores	119	1,064	8.94	636	134e06
D28cores	116	1,123	9.68	1513	267e06
D29cores	114	969	8.50	627	536e06
D30cores	117	1,051	8.98	448	1e09
D31cores	116	1,027	8.85	594	2e09
D32cores	118	816	6.92	1,816	4e09

Number of IWIs and MIWIs mined by setting the maximum IWI-support-min threshold ξ to 0.

running application constraints, which may limit the maximum system parallelism. Table 6 reports the number of IWIs and MIWIs mined from the avrova benchmark data sets by setting IWI-support-min to 0. To give a more detailed insight into the characteristics of the extracted MIWIs, in Fig. 2 we also reported the per-length distribution of the MIWIs mined from three representative data sets relative to the avrova benchmark (i.e., the 16-, 24-, and 32-core settings) by enforcing an IWI-support-min threshold equal to 0. The extracted IWIs compactly represent the information that, at each sampling time, at least one of the included cores is idle. The extraction of IWIs with length 1 from highly parallel settings (i.e., the 24- and 32-core settings) confirms the hypothesis of suboptimal resource utilization and suggests disabling/reallocation of specific cores. On the

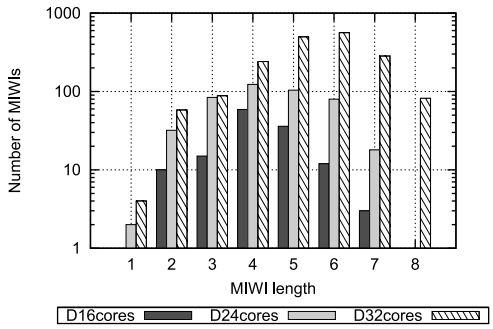


Fig. 2. Per-length MIWI distribution. Maximum IWI-support-min threshold $\xi = 0$. Avrova benchmark.

TABLE 7
Workflow Parallelization Analysis

Benchmark	Max. number of exploited cores		Benchmark	Max. number of exploited cores	
	no idle	no usage rate $\leq$10%		no idle	no usage rate $\leq$10%
avrova	10	7	lusearch	32	32
batik	5	6	pmd	32	18
h2	23	12	sunflow	32	32
jython	8	7	tomcat	29	17
luindex	7	6	xalan	27	25

other hand, longer IWIs may suggest the presence of an unbalanced load allocation when enabling specific core combinations. Note that longer IWIs are discovered even when lowly parallelized settings (e.g., 16-core) are analyzed. As a drawback, IWIs with IWI-support-min equal to 0 may also contain highly loaded cores. To focus the attention on the smallest potentially relevant core combinations, the expert chooses to look into the subset of minimal IWIs. Although the mined IWI set may be hardly manageable for manual inspection, the number of mined MIWIs is, in most cases, orders of magnitude lower (see Table 6).

The results confirm that the system has parallelized the computational workflow to lower extent than expected. In fact, idle core situations appear when sampling the system usage with a 14-core setting or higher. Similar results were achieved by evaluating system performance with other benchmarks. Table 7 reports, for each tested benchmark, the maximum number of cores yielding an optimal system parallelization in terms of the following criteria: (A) no idle core is detected, and (B) no core with average usage rate lower than 10 percent is detected. Condition (A) is verified if no MIWI with IWI-support-min equal to 0 is mined, whereas condition (B) holds when no MIWI with IWI-support-max lower than the 10 percent usage rate is extracted. The results show that in seven out of 10 benchmarks there exists at least one core combination for which one or more cores remain idle (not necessarily the same at each instant), and in eight cases out of 10 there is a suboptimal system resource usage.

5.3 Performance Analysis

We analyzed IWI Miner and MIWI Miner performance on standard synthetic and real data sets. In particular, we analyzed: (i) The impact of the equivalence procedure on the data set size (see Section 5.31), (ii) the impact of the IWI-support thresholds on both the number of mined patterns and the algorithm execution time (see Section 5.32), and (iii) the comparison, in terms of execution time, between MIWI Miner and MINIT, a state-of-the-art minimal infrequent (unweighted) itemset miner [9] (see Section 5.33).

5.3.1 Impact of the Equivalence Procedure

To enable the IWI mining process from weighted data, an equivalence procedure, described in Section 4.1, is preliminary applied to the original data set in order to suit weighted data to the subsequent FP-Growth-like mining step. This section reports the results of an experimental evaluation of the impact of the equivalence procedure on the resulting data set size.

TABLE 8
Comparison between Original and Equivalent Data Sets

Dataset	Num. of trans. $\|T\|$	Max. trans. length in T	Num. of equiv. trans. $\|TE\|$	Size increase ratio $\frac{\|TE\|}{\|T\|}$	Num. of FP-tree nodes increase ratio
IBMT10D100KC0 Uniform	98,352	30	945,249	9.6	3.8
IBMT10D100KC1 Uniform	98,331	30	943,614	9.6	3.8
IBMT10D100KC0.25 Uniform	98,346	29	944,780	9.6	3.8
IBMT10D100KC0.25 Poisson	98,346	29	817,106	8.3	3.4

Table 8 reports, for four representative synthetic data sets with different characteristics, the original data set size $|T|$ (Column 1), the longest transaction length in T (Column 2), the equivalent data set size $|TE|$ (Column 3), and the size increase ratio $|TE|/|T|$ (Column 4). In particular, three weighted data sets with size 100,000, average transaction length 10, uniformly distributed weights, and different item correlation values (i.e., minimum (0), standard (0.25), maximum (1)) are considered. Furthermore, one data set with size 100,000, transaction length 10, Poisson distribution, and standard item correlation value is also analyzed. Similar statistics for the real avrova data sets are given in Table 6.

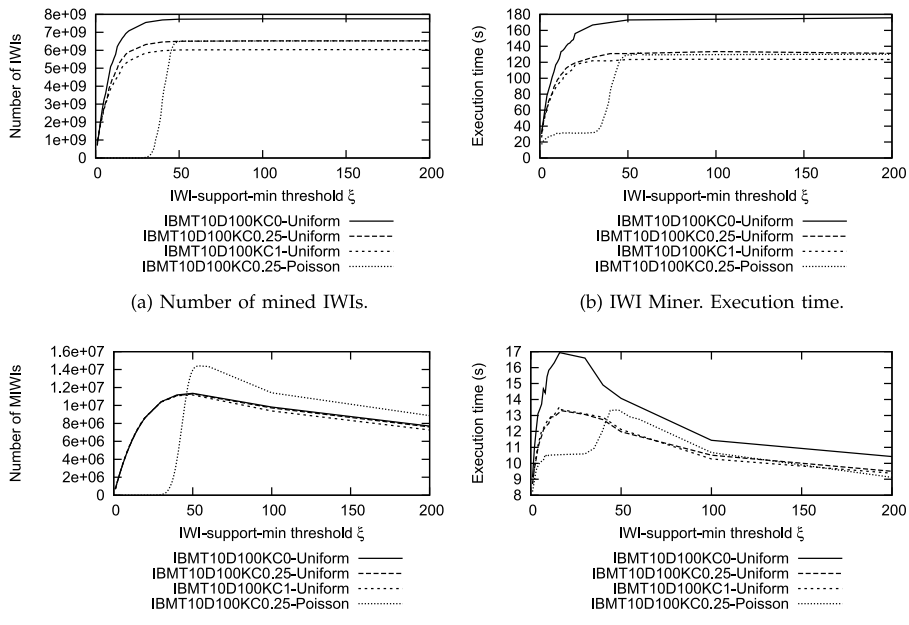
As discussed in Section 4.1, a more realistic estimate of the number of generated equivalent transactions should consider the average transaction length instead of the maximal one. The results reported in Table 6, achieved on synthetic data with average transaction length 10, confirm the expected trend. However, in practice, since many transactions are shorter than the longest one and many items have the same weight in each transaction, the actual size is significantly lower (e.g., for IBMT10D100KC0.25 the size increase ratio is 9.6 instead of 29). When using the Poisson distribution instead of the uniform one, weights are more correlated with each other and the size increase ratio further decreases (8.3 instead of 29). Similar results come out when coping with real data (e.g., based on the results in Table 6, D32cores achieves 6.92 against 32). As discussed in Section 4.1, a more precise estimate of the number of generated equivalent transactions is obtained by considering the product between the cardinality of T and its average transaction length. In the considered IBM data sets the average transaction length is equal to 10, and its use allows obtaining a good estimate of $|TE|$.

Since during the FP-Growth-like mining process many transactions may be collapsed into a single FP-tree path, we also analyzed the impact of the equivalent procedure on the FP-tree size. In particular, in Table 6 we reported the ratio between the number of nodes contained in the FP-tree generated from TE and the one relative to the FP-tree that would be generated from T (see Column 6). For all the evaluated data sets, the increase ratio, in terms of nodes, is lower than the one achieved in terms of number of transactions because, subsets of equivalent transactions generated from one original transaction are commonly compacted in the same FP-tree path.

5.3.2 Impact of the Maximum IWI-Support Threshold

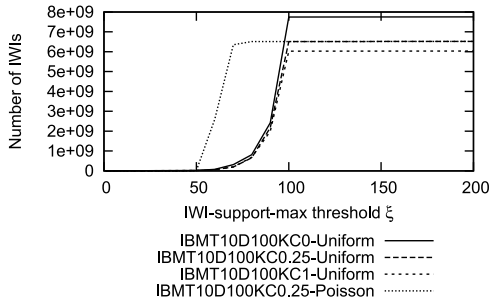
Since IWI-support threshold enforcement may affect the result of the (M)IWI mining process significantly, we analyzed its impact on IWI Miner and MIWI Miner performance on synthetic data. In particular, we performed different mining sessions, for many combinations of algorithms and data sets with different characteristics, by varying the maximum IWI-support-min and IWI-support-max thresholds.

Figs. 3a, 3c and Figs. 4a, 4b report the number of patterns extracted by IWI Miner and MIWI Miner from data

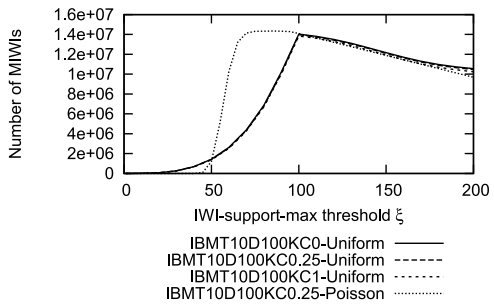


(a) Number of mined IWIs.
(b) IWI Miner. Execution time.
(c) Number of mined MIWIs.
(d) MIWI Miner. Execution time.

Fig. 3. Impact of the maximum IWI-support-min threshold on IWI Miner and MIWI Miner performance. IBM synthetic data sets with different data and weight distributions.



(a) Number of mined IWIs.



(b) Number of mined MIWIs.

Fig. 4. Impact of the maximum IWI-support-max threshold on IWI Miner and MIWI Miner performance. IBM synthetic data sets with different data and weight distributions.

sets with different characteristics by varying the IWI-support-min and IWI-support-max constraints, respectively. For the IWI Miner algorithm, the combinatorial growth of the number of possible infrequent item combinations makes the number of mined IWIs grow superlinearly with the IWI-support threshold until a steady state is reached, because all the possible IWIs have been extracted. Enforcing the IWI-support-max constraint instead of the IWI-support-min one yields the curves converging to the steady state condition more slowly, because IWIs have, on average, higher IWI-support values. Instead, for the MIWI Miner algorithm the combinatorial increase of the number of candidate MIWIs is counteracted by the discarding of some IWIs that become not minimal, i.e., some of their subsets satisfy the IWI-support constraint, from a certain point on. Hence, the total number of MIWIs becomes maximal at medium IWI-support thresholds (i.e., when ξ is around 25) while it decreases for higher IWI-support values.

The average length of the extracted MIWIs also reflects the selectivity of the enforced IWI-support thresholds. In particular, at lower IWI-support thresholds longer MIWIs are selected on average, while the average MIWI length decreases when increasing the maximum IWI-support threshold. As an extreme case, when very high IWI-support thresholds are enforced, only single weighted items get selected as minimal IWIs.

Synthetic data sets with different item correlation factors (i.e., the lowest (0), the highest (1), and the standard one (0.25)) have been generated and tested. Roughly speaking, the correlation factor is an approximation of the data set density, i.e., the more the items are correlated with each other, the more dense the analyzed data distribution is. As

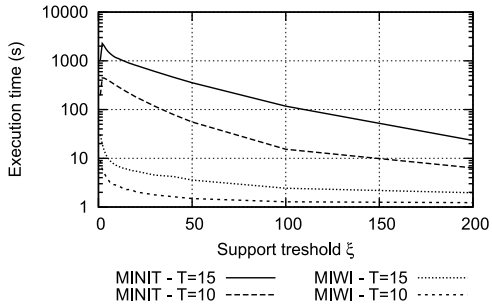


Fig. 5. Comparison between MIWI Miner and MINIT in terms of execution time. Synthetic data sets.

expected, the correlation factor turns out to be inversely correlated with the number of mined (M)IWIs. In fact, denser data sets contain on average a higher number of frequent patterns and, thus, a lower number of infrequent ones.

Weighted data sets with two different weight distributions, i.e., the Poisson distribution with a mean value equal to 50 and the uniform distribution, have also been analyzed. Using the Poisson distribution instead of the Uniform one produces, on average, fewer (M)IWIs with low IWI-support values. In fact, when using Poisson, the distribution of the IWI-supports of the extracted MIWIs is thickened around the mean value 50, while the IWI-supports of the extracted MIWIs are spread across the whole value range when using the uniform distribution.

Since the algorithm execution time and mined set cardinality are strongly correlated each other, the corresponding curves show a similar trend. Hence, due to the lack of space, we report detailed results only for the IWI-support-min measure (Figs. 3b and 3d). Similar results have been obtained by enforcing the IWI-support-max measure.

5.3.3 *Comparison with Traditional Non-Weighted Infrequent Itemset Mining*

This paper is, to the best of our knowledge, the first attempt to perform infrequent itemset mining from weighted data. However, other algorithms (e.g., [7], [8], [9], [17]) are able to mine infrequent itemsets from unweighted data. Hence, to also analyze the efficiency of the proposed approach when tackling the infrequent itemset mining from unweighted data, we compared MIWI Miner execution time with that of a benchmark algorithm, namely MINIT [9]. MINIT is, to the best of our knowledge, the latest algorithm that performs both minimal and non-minimal (unweighted) infrequent itemset mining from unweighted data. For MINIT, we exploited the C++ algorithm implementation available at http://mavdisk.mnsu.edu/haglin. For MIWI Miner, we set all item weights to 1 in order to mine traditional (unweighted) infrequent itemsets.

We compared MIWI Miner and MINIT performance, in terms of execution time, on synthetic and benchmark data sets with different characteristics. Fig. 5 reports the execution times achieved by varying the maximum support threshold in the range [0, 200] on two IBM synthetic data sets with 100,000 transactions and two representative average transaction length values (i.e., 10 and 15). The synthetic data sets are characterized by a fairly

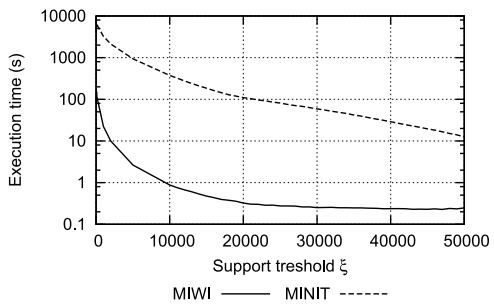


Fig. 6. Comparison between MIWI Miner and MINIT in terms of execution time. Connect UCI data set.

sparse data distribution. Similarly, Fig. 6 reports the results achieved on the real-life Connect data set downloaded from the UCI repository [20]. Connect is an averagely dense data set, characterized by 67,557 transactions and 42 categorical attributes, which has already been used for comparing infrequent itemset mining algorithm performance [9]. Since MINIT always takes more than 10 hours in mining non-minimal infrequent itemsets, while IWI Miner execution is orders of magnitude faster, the corresponding plots have been omitted.

Thanks to its FP-growth-like implementation and the applied pruning strategy, MIWI Miner is always at least one order of magnitude faster than MINIT in all the performed comparisons. In particular, when coping with denser data sets (e.g., connect) MIWI Miner becomes at least two orders of magnitude faster than MINIT for almost all the considered maximum support threshold values. Similar results have been obtained for the other real and synthetic data sets.

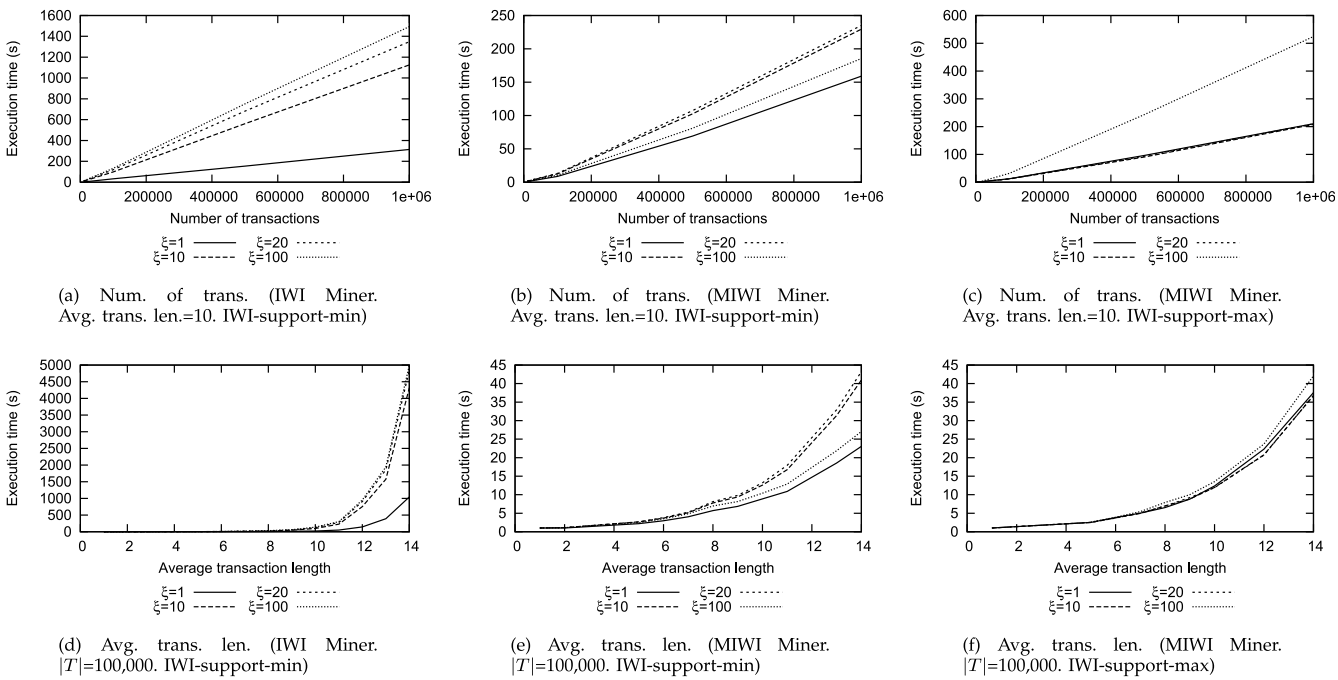
For the sake of completeness, we also considered another minimal infrequent mining algorithm, called IFP_min [17], beyond MINIT. IFP_min is, to the best of our knowledge, the latest minimal (unweighted) infrequent itemset miner. According to the results reported in [17], IFP_min is faster than MINIT only when relatively high maximum support threshold values are enforced. Unlike IFP_min, MIWI Miner performs better than MINIT *for every* support threshold value. Hence, when the mining task becomes more time consuming (i.e., lower support thresholds are enforced) it performs significantly better than both IFP_min and MINIT.

In summary, when dealing with unweighted data (i) IWI Miner is shown to be orders of magnitude faster than state-of-the-art algorithms for all considered parameter settings and data sets, and (ii) MIWI Miner is faster or competitive with state-of-the-art approaches, especially when setting lower maximum support thresholds or coping with denser data sets.

5.4 Scalability Analysis

We also analyzed the algorithm scalability, in terms of execution time, on synthetic data sets. To test the algorithm scalability with the number of data set transactions (i.e., the data set cardinality), we generated data sets of size ranging from 0 to 1,000,000 transactions by following the procedure described in Section 5.1.

Figs. 7a and 7b respectively report IWI Miner and MIWI Miner execution times by varying the data set cardinality and by setting three representative IWI-support-min threshold values. The mining computational complexity appears to be strongly correlated with the cardinality of the extracted patterns. In general, IWI Miner execution time significantly increases for higher IWI-support-min threshold values due to the combinatorial growth of the number of extracted patterns. Instead, MIWI Miner takes a higher execution time when medium IWI-support-min thresholds are enforced, because the cardinality of the mined MIWIs becomes maximal (see Section 5.32). However, both IWI



(a) Num. of trans. (IWI Miner. Avg. trans. len.=10. IWI-support-min)

(b) Num. of trans. (MIWI Miner. Avg. trans. len.=10. IWI-support-min)

(c) Num. of trans. (MIWI Miner. Avg. trans. len.=10. IWI-support-max)

(d) Avg. trans. len. (IWI Miner. $|T|$=100,000. IWI-support-min)

(e) Avg. trans. len. (MIWI Miner. $|T|$=100,000. IWI-support-min)

(f) Avg. trans. len. (MIWI Miner. $|T|$=100,000. IWI-support-max)

Fig. 7. IWI Miner and MIWI Miner scalability with different parameters. Correlation factor = 0.25.

Miner and MIWI Miner execution time scale roughly linearly with the data set size for all the tested settings.

We also analyzed the algorithm scalability, in terms of execution time, with the average transaction length. Figs. 7d and 7e report the IWI Miner and MIWI Miner execution times by varying the transaction length and by setting three representative IWI-support-min threshold values. When increasing the average transaction length the algorithm execution time increases because of the non-linear increase of the number of possible item combinations. To also compare the impact of the IWI-support-min and IWI-support-max constraints on algorithm scalability in Figs. 7c and 7f we report the MIWI Miner execution time achieved by enforcing three representative IWI-support-max constraint values and by varying the data set cardinality and the average transaction length, respectively. From the comparison between Figs. 7b and 7c it comes out that the effectiveness of the early item pruning strategy described in Section 4 is lower when pushing the IWI-support-max constraint into the mining process instead of IWI-support-min. Similar results, omitted due to space constraints, were achieved for IWI Miner (i.e., when non-minimal IWIs are mined). However, enforcing both constraints the proposed algorithms scale well even when coping with fairly complex data sets (e.g., with data set cardinality = 1,000,000, correlation factor = 0.25, and average transaction length = 10, MIWI Miner takes around 180 s with IWI-support-min threshold $\xi = 100$, whereas 520 s with IWI-support-max threshold $\xi = 100$).

6 Conclusions and Future Work

This paper faces the issue of discovering infrequent itemsets by using weights for differentiating between relevant items and not within each transaction. Two FP-Growth-like algorithms that accomplish IWI and MIWI mining efficiently are also proposed. The usefulness of the discovered patterns has been validated on data coming from a real-life context with the help of a domain expert. As future work, we plan to integrate the proposed approach in an advanced decision-making system that supports domain expert's targeted actions based on the characteristics of the discovered IWIs. Furthermore, the application of different aggregation functions besides minimum and maximum will be studied.

References

[1] R. Agrawal, T. Imielinski, and Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '93)*, pp. 207-216, 1993.

[2] M.L. Antonie, O.R. Zaiane, and A. Coman, "Application of Data Mining Techniques for Medical Image Classification," *Proc. Second Intl. Workshop Multimedia Data Mining in Conjunction with seventh ACM SIGKDD (MDM/KDD '01)*, 2001.

[3] G. Cong, A.K.H. Tung, X. Xu, F. Pan, and J. Yang, "Farmer: Finding Interesting Rule Groups in Microarray Datasets," *Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '04)*, 2004.

[4] W. Wang, J. Yang, and P.S. Yu, "Efficient Mining of Weighted Association Rules (WAR)," *Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and data Mining (KDD '00)*, pp. 270-274, 2000.

[5] F. Tao, F. Murtagh, and M. Farid, "Weighted Association Rule Mining Using Weighted Support and Significance Framework," *Proc. nineth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '03)*, pp. 661-666, 2003.

[6] K. Sun and F. Bai, "Mining Weighted Association Rules Without Preassigned Weights," *IEEE Trans. Knowledge and Data Eng.*, vol. 20, no. 4, pp. 489-495, Apr. 2008.

[7] A. Manning and D. Haglin, "A New Algorithm for Finding Minimal Sample Uniques for Use in Statistical Disclosure Assessment," *Proc. IEEE Fifth Int'l Conf. Data Mining (ICDM '05)*, pp. 290-297, 2005.

[8] A.M. Manning, D.J. Haglin, and J.A. Keane, "A Recursive Search Algorithm for Statistical Disclosure Assessment," *Data Mining and Knowledge Discovery*, vol. 16, no. 2, pp. 165-196, http://mavdisk.mnsu.edu/haglin, 2008.

[9] D.J. Haglin and A.M. Manning, "On Minimal Infrequent Itemset Mining," *Proc. Int'l Conf. Data Mining (DMIN '07)*, pp. 141-147, 2007.

[10] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, pp. 1-12, 2000.

[11] J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *J. ACM*, vol. 46, no. 5, pp. 604-632, 1999.

[12] C.-K. Chui, B. Kao, and E. Hung, "Mining Frequent Itemsets from Uncertain Data," *Proc. 11th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD '07)*, pp. 47-58, 2007.

[13] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Zuefle, "Probabilistic Frequent Itemset Mining in Uncertain Databases," *Proc. 15th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '09)*, pp. 119-128, 2009.

[14] C.K.-S. Leung, C.L. Carmichael, and B. Hao, "Efficient Mining of Frequent Patterns from Uncertain Data," *Proc. Seventh IEEE Int'l Conf. Data Mining Workshops (ICDMW '07)*, pp. 489-494, 2007.

[15] X. Wu, C. Zhang, and S. Zhang, "Efficient Mining of Both Positive and Negative Association Rules," *ACM Trans. Information Systems*, vol. 22, no. 3, pp. 381-405, 2004.

[16] X. Dong, Z. Zheng, Z. Niu, and Q. Jia, "Mining Infrequent Itemsets Based on Multiple Level Minimum Supports," *Proc. Second Int'l Conf. Innovative Computing, Information and Control (ICICIC '07)*, pp. 528-531, 2007.

[17] A. Gupta, A. Mittal, and A. Bhattacharya, "Minimally Infrequent Itemset Mining Using Pattern-Growth Paradigm and Residual Trees," *Proc. Int'l Conf. Management of Data (COMAD)*, pp. 57-68, 2011.

[18] M.J. Elliot, A.M. Manning, and R.W. Ford, "A Computational Algorithm for Handling the Special Uniques Problem," *Int'l J. Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 493-509, 2002.

[19] IBM, "IBM Quest Synthetic Data Generation Code," http://www.almaden.ibm.com/, 2009.

[20] A. Frank and A. Asuncion, "UCI Machine Learning Repository," http://archive.ics.uci.edu/ml, 2010.

Luca Cagliero received the master's degree in computer and communication networks and the PhD degree in computer engineering from Politecnico di Torino. He has been a post-doc fellow in the Dipartimento di Automatica e Informatica at the Politecnico di Torino since March 2012. His current research interests are in the fields of data mining and database systems. In particular, he has worked on data mining by means of itemset and association rule mining algorithms.

Paolo Garza received the master's and PhD degrees in computer engineering from the Politecnico di Torino. He has been an assistant professor (with non-tenure track position) at the Dipartimento di Automatica e Informatica, Politecnico di Torino, since December 2010. His current research interests include data mining and database systems. He has worked on the classification of structured and unstructured data, clustering, and itemset mining algorithms.