

A Parallel, Distributed Algorithm for Relational Frequent Pattern Discovery from Very Large Data Sets

Annalisa Appice, Michelangelo Ceci, Antonio Turi, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{appice, ceci, turi, malerba}@di.uniba.it

Abstract. The amount of data produced by ubiquitous computing applications is quickly growing, due to the pervasive presence of small devices endowed with sensing, computing and communication capabilities. Heterogeneity and strong interdependence, which characterize ‘ubiquitous data’, require a (multi-)relational approach to their analysis. However, relational data mining algorithms do not scale well and very large data sets are hardly processable. In this paper we propose an extension of a relational algorithm for multi-level frequent pattern discovery, which resorts to data sampling and distributed computation in Grid environments, in order to overcome the computational limits of the original serial algorithm. The set of patterns discovered by the new algorithm approximates the set of exact solutions found by the serial algorithm. The quality of approximation depends on three parameters: the proportion of data in each sample, the minimum support thresholds and the number of samples in which a pattern has to be frequent in order to be considered globally frequent. Considering that the first two parameters are hardly controllable, we focus our investigation on the third one. Theoretically derived conclusions are also experimentally confirmed. Moreover, an additional application in the context of event log mining proves the viability of the proposed approach to relational frequent pattern mining from very large data sets.

1 Introduction

Recent advances in sensor technology and computing environments are moving toward mobile, finely distributed, interacting, dynamic environments. The pervasive presence of such environments in real life is driving the recent interest in the paradigm of *ubiquitous computing* [19], according to which computing is omnipresent and devices that do not look like computers are endowed with computing capabilities. All these devices are also capable of communicating and their synergetic activities potentially contribute to developing electronic environments that are sensitive and responsive to the presence of people.

The success of prospected technological advancements strongly depends on the actual capacity of newer ubiquitous computing applications to exploit knowledge which is hidden in the the huge amount of data produced by devices. An

aspect of ubiquitous scenarios is that they are usually described by many different data types [25]. In sensor networks, for instance, the snapshot of a given situation can be depicted by different sensors which collect multi-modal data (e.g, images, sounds, videos, temperature, light, and acceleration). Furthermore, data involved in ubiquitous scenarios (e.g., blog networks or cascaded sensors) have a complex inner structure with several explicit or implicit relationships. From a knowledge discovery perspective, the heterogeneity and the structure of ubiquitous data are a great source of complexity which demands sophisticated preprocessing and data integration techniques as well as advanced data mining algorithms.

Studies in (Multi-)Relational Data Mining (MRDM) [14] and Inductive Logic Programming (ILP) [26] have already addressed issues related to heterogeneous data coming from multiple sources [32], which can be of many different types and are typically stored in several database relations. Therefore, their application to knowledge discovery from data generated in an ubiquitous computing environment is, at least in principle, both appropriate and effective [13]. However, a common problem in the wide range of MRDM and ILP (or simply *relational*) solutions which can mine different types of patterns and models (e.g., classification rules, association rules, and clusters) remains their actual scalability. Despite some advances on this front, such as the declarative bias which limits the size of the search space [27], the query transformations [10] and the lazy evaluation of examples [5] which optimize the efficiency of testing each candidate hypothesis, much remains to be done in order to make current relational systems applicable to very large data sets such as those produced by ubiquitous computing environments.

In this paper we focus on the task of frequent pattern discovery, that is, the generation of those patterns which occur frequently, with respect to a given threshold, in a data set. Frequent patterns play an essential role in association rule mining, sequence mining, outlier detection, clustering and classification. They can be itemsets, sequences, subgraphs, and can be expressed in different languages. In this work we are interested in relational patterns, which can be expressed as sets of atomic formulae (or atoms). Our interest is governed by potential applications to data collected in ubiquitous computing environments.

Several systems allow relational frequent pattern discovery. Two representative examples of the state-of-the-art are WARMR [11] and SPADA [22]. They both represent relational data and domain (or background) knowledge *à la* Datalog [6], a logic programming language with no function symbols, specifically designed to implement deductive databases. Moreover, their design is based on the logical notions of generality order and downward/upward refinement operator, which are used to structure and search the space of relational patterns. The main difference is that WARMR is not able to properly and efficiently perform multi-level analysis, since it lacks mechanisms for dealing properly with concept hierarchies. By mapping concept hierarchies into different levels of granularity it is possible to extract patterns at different granularity (or abstraction) levels. More abstract patterns tend to be more frequent, but less informative, hence

the need for an inter-level exploration of the pattern space, in order to find the most interesting patterns. In SPADA, pattern construction proceeds from more general to more specific granularity levels, so that it is possible to profitably exploit information collected at higher levels, in order to prevent the generation and evaluation of weak patterns at lower levels.

The problem with both relational frequent pattern discovery systems is their actual applicability to very large data sets. Indeed, frequent pattern discovery is a computationally demanding task, because of the exponential size of the search space. In addition, the discovery of relational patterns is complicated by the inefficiency of the test of a single pattern against a data set. To develop a scalable, high performance system it is possible to resort to a parallel, distributed approach.

Strategies to parallelize ILP systems and to speed up the learning time are presented in [12, 16]. However, all proposed solutions work in a shared-memory architecture and do not permit a real advantage in terms of space complexity. Almost all methods proposed for distributed memory architectures face classification tasks [21, 18, 17], hence they are not appropriate for relational frequent pattern discovery.

In this paper, we propose a parallel, distributed algorithm in order to mine relational frequent patterns from very large databases. The algorithm wraps SPADA, although it is general enough to be applicable to any other relational frequent pattern discovery system, such as WARMR. It is three-stepped: 1) multiple samples from the original data are initially extracted; 2) locally frequent relational patterns are discovered by running SPADA on computational nodes of a GRID; 3) the set of globally frequent relational patterns is approximated by analyzing locally discovered patterns. The quality of approximation depends on three parameters: the proportion of data in each sample, the minimum support thresholds, and the number of samples in which a pattern has to be frequent in order to be considered globally frequent. Considering that the first two parameters are hardly controllable, since they depend on main memory capability and application requirements respectively, we focus our investigation on the third parameter, the number of samples. We characterize the probability distribution that a pattern is frequent in at least k out of n samples and we derive theoretical considerations on the expected quality of approximation. These considerations are also empirically supported by experiments on two real databases.

The contributions of this paper are three-fold. First, we develop a general parallel, distributed algorithm for relational frequent pattern mining. Second, we characterize the probability distribution of the patterns belonging to the set of frequent patterns. Third, we empirically evaluate our parallel, distributed algorithm on two real databases.

The paper is organized as follows. In the next section, some related works are revised. In Section 3, we discuss the theoretical background on the relational frequent pattern discovery in SPADA. The distributed, parallel algorithm, which makes feasible the application of SPADA to large data sets, is presented in

Section 4. Experimental results on two real databases are reported and discussed in Section 5. Finally, some conclusions are drawn.

2 Related Work

In recent years several parallel, distributed extensions of serial algorithms for frequent pattern discovery have been proposed. For instance, the CD [2], FDM [7], and DDM [34] algorithms parallelize Apriori [1], and PDM [30] parallelizes DHP [29]. In all these algorithms, frequent patterns are discovered by partitioning the tuples of a single large database relation among a set of computational nodes and then processing the task in a distributed manner.

Schuster *et al.* [35] propose a distributed frequent pattern mining algorithm, called D-sampling, which is inspired by the Sampling framework originally proposed by Toivonen [38]. The idea behind Sampling is that a random sample of the database is used to discover all frequent patterns, which are then validated in a single database scan. D-sampling parallelizes the computation of frequent patterns by sampling the original database at several nodes and by locally computing frequencies associated to patterns that are then globally validated.

Silvestri and Orlando [36] propose an algorithm for approximate mining of frequent patterns, called AP_{Interp} , which is a distributed version of the frequent pattern mining algorithm DCI [28]. AP_{Interp} computes local patterns independently for each node and then merges them. During a second phase, an approximate support inference heuristic is used to merge results.

Singh *et al.* [37] propose a strategy to mine frequent patterns in very large databases by exploiting both a Grid-platform and an efficient implementation of the standard Apriori algorithm. Random samples of the original database are generated and then patterns which are locally frequent in each random sample are efficiently mined. Approximate globally frequent patterns are generated from locally frequent ones.

Although all these approaches lead to interesting achievements in terms of scalability, they work in the classical propositional data mining setting and are not able to deal with structured and heterogeneous data stored in several relations of a database. To the best of our knowledge, the only distributed system for relational frequent pattern discovery is PolyFARM [8], a distributed version of WARMR. There are three types of components to PolyFARM: Farmer, Worker and Merger. The Farmer is responsible of generating candidate patterns according to a level-wise search strategy [24]. The generation is constrained by both a language bias and the frequent patterns from the previous level. Once the set of candidate patterns at a given level is generated, the database is partitioned so that each partition fits in main memory. Each Worker reads in all the candidates, its own database partition and the common background knowledge, and then it counts the local frequency of the patterns on the database partition. The Merger collects results of completed Worker jobs and sums local frequency counts up for each candidate pattern. This way, the exact frequency count is available for each candidate pattern. These data are reused by the Farmer to

prune infrequent patterns and to generate the next level of candidates. This iterative process stops when the set of candidate patterns generated by the Farmer is empty. The Workers are run on a Beowulf cluster (<http://www.beowulf.org/>).

PolyFARM presents several drawbacks. First, each iteration requires synchronization of all Workers, i.e., it is not possible to move to the next iteration unless all Worker jobs have been completed. The number of iterations depends on the size of the most specific frequent pattern, while the number of Worker jobs depends on the main-memory capabilities. Second, the overhead paid at each iteration is considerably large, since Workers are necessarily stateless and they need to read a database partition at each iteration. Third, the number of candidate patterns held in main-memory can grow impractically large [9].

Our contribution aims to overcome these limits through a different approach based on independent multi-sample mining [33]. In particular, several samples of the original database are generated such that each of them can fit in main memory. The number of samples is independent of the main-memory capability. Samples are shipped to the computation nodes of a Grid together with the common background knowledge and sets of locally frequent patterns are independently generated for each sample. Finally, these sets are post-processed by a combining procedure to produce the final set of approximate global frequent patterns.

This approach presents several advantages. It is not iterative and it does not require repeated process synchronization. Jobs activated on different Grid nodes are independent and do not require repeated data loading to generate (locally) frequent patterns. It is scalable to very large databases. The minimization of communication overhead makes it more suitable for loosely coupled systems, such as computational Grids, which transparently tolerate network topology changes and nodes failure. Only locally frequent patterns are loaded in main memory by the combining procedure. Finally, independent multi-sampling allows us to characterize the probability distribution of approximate global frequent patterns. The price paid for these advantages is that the returned set of frequent patterns is not guaranteed to be complete and correct.

3 Relational Frequent Pattern Discovery

SPADA is the only ILP system which addresses the task of relational frequent pattern discovery by dealing properly with concept hierarchies. In the following subsections, details on the representation formalisms for data and background knowledge used by SPADA, as well as details on the search strategy adopted by SPADA to discover multi-level frequent patterns, are reported. We assume the reader is familiar with the concepts of logic programming [23] or deductive databases [6].

3.1 Representing Data and Models

Data stored in distinct tables of a relational database describe distinct objects involved in the phenomenon under investigation. These objects play different

roles and it is necessary to distinguish between the set S of *reference* (or target) *objects*, which is the main subject of analysis, and the sets R_k , $1 \leq k \leq M$, of *task-relevant* (or non-target) objects, which are related to the former and can contribute to accounting for the variation. For each R_k , a generalization hierarchy H_k ($k = 1, \dots, M$) is defined. A function ψ_k maps objects in H_k into a set of granularity levels $\{1, \dots, L\}$.

In the logic framework adopted by SPADA, a relational database is converted to a deductive database D . Properties of both reference and task-relevant objects are represented in the extensional part D_E , while the background knowledge (BK) is expressed as a normal logic program which defines the intensional part D_I . Example 1 shows how the normal logic program in D_I allows deductions to be made (i.e. concluding additional atoms) from data stored in D_E .

Example 1. Let D be a deductive database which contains the event log of executed process instances. The constants $c1$ and $c2$ denote two distinct process instances (reference objects), while the constants $a1$, $a2$, $a3$, and $a4$ identify four activities and the constants $u1$ and $u2$ identify two actors (task-relevant objects). D_E includes the ground atoms:

process(c1). process(c2).
activity(c1,a1). activity(c1,a2). activity(c2,a3). activity(c2,a4).
is_a(a1,namemaker). is_a(a2,workflow). is_a(a3,workflow). is_a(a4,delete).
time(a1,10). time(a2,25). time(a3,22). time(a4,23).
actor(a1,paul). actor(a2,paul). actor(a3,paul). actor(a4,mary).
is_a(paul,user). is_a(mary ,admin).

while D_I is the normal logic program:

before(A1, A2) ← activity(C, A1), activity(C, A2), A1 ≠ A2,
time(A1, T1), time(A2, T2), T1 < T2,
not(activity(C, A), A ≠ A1, A ≠ A2, time(A, T), T1 < T, T < T2)

which entails the following temporal information: *before(a1, a2), before(a3, a4).*

The set of predicates can be categorized into four classes. The *key predicate* identifies the reference objects in S (e.g., *process* is the key predicate in Example 1). The *property predicates* are binary predicates which define the value taken by an attribute of an object (e.g., *time*). The *structural predicates* are binary predicates which relate task-relevant objects (e.g., *actor*) as well as reference objects with task-relevant objects (e.g., *activity*). The *is_a* predicate is a binary *taxonomic* predicate which associates a task-relevant object with a value of some H_k . In Example 1, ground atoms of *is_a* predicate define the two hierarchies shown in Figure 1. Each hierarchy has three levels which are naturally mapped into three granularity levels.

The *units of analysis* $D[s]$, one for each reference object $s \in S$, are subsets of ground facts in D_E , defined as follows:

$$D[s] = is_a(R(s)) \cup D[s|R(s)] \cup \bigcup_{r_i \in R(s)} D[r_i|R(s)], \quad (1)$$

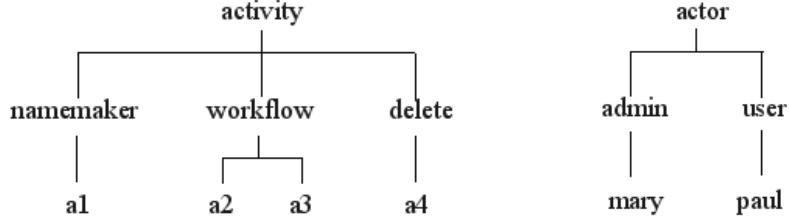


Fig. 1. Three-level hierarchies on activity and actor.

where:

- $R(s)$ is the set of task-relevant objects directly or indirectly related to s ;
- $is_a(R(s))$ is the set of is_a atoms which define the types of $r_i \in R(s)$;
- $D[s|R(s)]$ contains properties of s and relations between s and some $r_i \in R(s)$;
- $D[r_i|R(s)]$ contains properties of r_i and relations between r_i and some $r_j \in R(s)$.

This notion of unit of analysis is coherent with the individual-centered representation [3], which has both theoretical (PAC-learnability) and computational advantages (smaller hypothesis space and more efficient search). The set of units of analysis is a partitioning of D_E into a number of subsets $D[s]$, each of which includes ground atoms concerning the task-relevant objects (transitively) related to the reference object s (see Example 2).

Example 2. The unit of analysis $D[c1]$ is the set of ground atoms concerning the activities and actors involved in a specific process execution $c1$:

$is_a(a1,namemaker)$. $is_a(a2,workflow)$. $is_a(paul,user)$.
 $process(c1)$. $activity(c1,a1)$. $activity(c1,a2)$.
 $time(a1,10)$. $time(a2,25)$. $actor(a1,paul)$. $actor(a2,paul)$.

In this example $R(c1) = \{a1, a2, paul\}$.

Relational patterns are conjunctions of Datalog atoms, which can be expressed by means of a set notation. For this reason they are also called *atomsets*, [11] by analogy with itemsets introduced for classical association rules. A formal definition of relational pattern is reported in the following.

Definition 1. A relational pattern P is a set of atoms $p_0(t_0^1), p_1(t_1^1, t_1^2), p_2(t_2^1, t_2^2), \dots, p_r(t_r^1, t_r^2)$, where p_0 is the key predicate, while $p_i, i = 1, \dots, r$, is either a structural predicate or a property predicate or an is_a predicate.

Terms t_i^j are either constants, which correspond to values of property predicates, or variables, which identify reference objects either in S or in some R_k . Each p_i is either extensionally or intensionally defined. Patterns in the search space explored by SPADA satisfy the linkedness [20] property, which means that

each task-relevant object in a relational pattern P defined as in Def. 1 must be transitively linked to the reference object t_0^1 by means of structural predicates.

Each pattern P is associated with a granularity level l . This means that all taxonomic (*is_a*) atoms in P refer to task-relevant objects, which are mapped by some ψ_k into the same granularity level l . In multi-level association rule mining, it is possible to define an *ancestor* relation between two patterns P and P' at different granularity levels.

Definition 2. A pattern P at granularity level l is an ancestor of the pattern P' at granularity level l' , $l < l'$, if P' can be obtained from P by replacing each task-relevant object $h \in H_k$ at granularity level l ($l = \psi_k(h)$) with a task-relevant object h' , which is more specific than h in H_k and is mapped into the granularity level l' ($l' = \psi_k(h')$).

By assigning a pattern P with an existentially quantified conjunctive formula $eqc(P)$ obtained by transforming P into a Datalog query, we can now provide a formal definition of the support of P on D .

Definition 3. A pattern P covers $D[s]$ if $D[s] \cup BK \models eqc(P)$, i.e., $D[s] \cup BK$ logically entails $eqc(P)$.

Each relational pattern P is associated with a parameter $sup(P, D)$, which is the percentage of units of analysis in D covered by P (*support*). The minimum support for *frequent* relational patterns depends on the granularity level l ($1 \leq l \leq M$) of task-relevant objects. It is denoted as $minsup[l]$.

Definition 4. A pattern P [$sup(P, D)$] at level l is frequent if $sup(P, D) \geq minsup[l]$ and all ancestors of P are frequent at their corresponding levels.

Example 3. Let us consider the deductive database in Example 1 and suppose that $minsup[1] = 80\%$ and $minsup[2] = 40\%$. Then the following patterns:

$process(A), activity(A,B), is_a(B,activity), before(B,C), is_a(C,activity)$ [100%].
 $process(A), activity(A,B), is_a(B,namemaker), before(B,C), is_a(C,workflow)$ [50%].

are frequent at levels 1 and 2 respectively.

3.2 Ordering Patterns in the Search Space

Frequent pattern discovery in SPADA is performed according to both an intra-level search and inter-level search. The intra-level search explores the space of patterns at the same level of granularity. It is based on the level-wise method [24], which performs a breadth-first search of the space, from the most general to the most specific patterns, and prunes portions of the search space which contain only infrequent patterns. The application of the level-wise method requires a generality ordering, which is monotonic with respect to pattern support. The generality ordering considered in this work is based on the notion of θ -subsumption [31].

Definition 5. P_1 is more general than P_2 under θ -subsumption ($P_1 \succeq_\theta P_2$) if and only if P_1 θ -subsumes P_2 , that is, a substitution θ exists, such that $P_1\theta \subseteq P_2$.

Example 4. Let us consider the following relational patterns:

$$P_1 \equiv is_a(B, namemaker)$$

$$P_2 \equiv is_a(B, namemaker), before(B, C)$$

$$P_3 \equiv is_a(B, namemaker), before(B, C), is_a(C, workflow)$$

whose variables are implicitly existentially quantified. Then P_1 θ -subsumes P_2 ($P_1 \succeq_\theta P_2$) and P_2 θ -subsumes P_3 ($P_2 \succeq_\theta P_3$) with substitutions $\theta_1 = \theta_2 = \emptyset$.

The relation \succeq_θ is a quasi-ordering (or preorder), since it is reflexive and transitive but not antisymmetric. Moreover, it is monotonic with respect to support [22].

Proposition 1. Let P_1 and P_2 be two relational patterns at the same level l , defined as in Def. 1. If $P_1 \succeq_\theta P_2$, then $sup(P_1, D) \geq sup(P_2, D)$.

It is noteworthy that, if P_1 and P_2 are two relational patterns, such that $P_1 \succeq_\theta P_2$ and P_1 is not frequent ($sup(P_1, D) < minsup[l]$), then also P_2 is not frequent ($sup(P_2, D) < minsup[l]$). Therefore, the monotonicity of \succeq_θ with respect to support allows for pruning the search space without losing frequent patterns.

In the inter-level search, SPADA refines patterns discovered at level l by descending the generalization hierarchies by one level. Indeed, by the definition of a frequent pattern, a necessary condition for pattern P to be frequent at level $l + 1$ is that an ancestor pattern P' exists at level l , such that P' is frequent. The inter-level search takes advantage of statistics computed at level l to prune the search space at level $l + 1$.

In real-world applications, a large number of frequent patterns can be generated, most of which are useless. To prevent this, in SPADA it is possible to specify which atoms should occur in the discovered patterns (language bias). Pattern constraints can also be used to specify exactly both the minimum and the maximum number of occurrences of an atom in a pattern.

4 Parallel, Distributed Relational Pattern Discovery

Despite the general pruning mechanisms described above and the domain-specific constraints expressed in the language bias, the application of SPADA to a very large database is still hindered by both the high computational cost of the search and the usage of an in-memory deductive database. Computational limits of SPADA in processing large databases are overcome by distributing and (possibly) parallelizing the discovery of sets of *locally* frequent patterns and then deriving an approximation of the exact set of frequent patterns, which would be discovered on the entire database.

4.1 Data Sampling

Sampling can speed up the mining process by more than an order of magnitude, by reducing I/O costs and drastically shrinking the number of transactions to be considered [38, 40]. Moreover, when data are kept in main memory, as in SPADA, sampling is the only way to make their analysis feasible. The sampling procedure considered in this work is similar to that used in bootstrap estimation of a parameter (e.g., predictive accuracy of a classifier) [15], as well as in some ensemble data mining methods, such as bagging [4], which combine multiple models to achieve better prediction accuracy than any of the individual models.

More precisely, n sample extensional databases D_E^j , $j = 1, \dots, n$, are formed by randomly sampling, with replacement, the N ($N = |S|$) units of analysis in the original extensional database D_E . Each D_E^j includes m units of analysis, hence, m reference objects are used to compute the support of a local pattern. The proportion of units of analysis in each D_E^j is $p = m/N$.

It is noteworthy that the n sample extensional databases D_E^j are neither mutually exclusive nor exhaustive, i.e., they do not partition the original data set, so, even 10 samples with $p = 0.1$ do not generally cover the entire database. The probability that a particular unit of analysis is not in $\bigcup_j D_E^j$ is the following:

$$(1 - 1/N)^{nm}. \quad (2)$$

When $n = N/m$, i.e., $n = 1/p$, the above probability approximates e^{-1} for large N , where e is Euler's number (≈ 2.7183). Since $e^{-1} \approx 0.368$, this means that the expected number of reference objects in $\bigcup_j D_E^j$ is 62.3% of the those in S .

Differently from data partitioning, which is affected by only one parameter n (the number of partitions), the data sampling procedure used in this work is controlled by two parameters: p and n . The former is set on the basis of the actual storage capability of nodes of the distributed architecture. The latter affects the amount of computational resources required to generate the globally frequent patterns, as well as the accuracy of the set of global patterns discovered. Indeed, by increasing n , the probability (2) decreases, thus it is more likely that some D_E^j includes at least $p\%$ of reference objects in D_E , which support a pattern P at level l . If P is globally frequent ($sup(P, D_E) \geq minsup[l]$), it is more likely that P is a locally frequent pattern for some D_E^j . At the same time, if P is not frequent in D_E ($sup(P, D_E) < minsup[l]$), then it is unlikely that P is locally frequent in many of the samples D_E^j ($sup(P, D_E^j) \geq minsup[l]$). By properly choosing the number k of samples D_E^j in which P has to be locally frequent in order to be considered globally frequent, it is possible to find the right trade-off between precision and recall.

Theoretically, the sampling procedure used in this work is different from the sequential random sampling without replacement used by Zaki *et al.* [40] to mine conventional association rules from very large databases. However, it is practically the same when p is small ($p < 0.05$), since there is a very small probability that any unit of analysis will be chosen more than once. The advantages of our procedure are: (i) it is computationally simpler, since it is not necessary to remove the drawn unit of analysis from further consideration, (ii) it is potentially

more accurate, since the selection of frequent patterns is based on multiple samples and not a single one, and (iii) the estimate of the support of a pattern is based on several independent observations (samples) rather than on only one.

4.2 Distributing Computation on Grid

A sample extensional database D_E^j and the intensional database D_I may be shipped along with SPADA to several computation nodes of a Grid using gLite middleware. gLite¹ is a middleware for Grid computing which provides a framework for building Grid applications, utilizing the power of distributed computation and storage resources across the Internet. Distributed computation is done by submitting parametric jobs, described in JDL (Job Description Language), through the command line interface. The submission of jobs on Grid is performed in several steps, namely:

1. authentication,
2. preparation of the jobs,
3. uploading the sample databases (stage-in),
4. submission of a relative parametric job,
5. checking/waiting for the results,
6. getting the results (stage-out).

4.3 Computing Approximate Global Frequent Patterns

Once relational patterns have been discovered for each sample database, they are pairwise compared in order to compute the number of sample databases where they are locally frequent. The comparison of two patterns P and Q discovered in two distinct database samples is based on an *equivalence test* under θ -subsumption, which is defined as follows:

$$P \equiv Q \text{ iff } (P \succeq_{\theta} Q) \wedge (Q \succeq_{\theta} P). \quad (3)$$

Those patterns which are (locally) frequent in at least k sample databases, with $k \leq n$, are selected as globally frequent patterns and are added to a set $GFP(k)$.

By varying the parameter k from 1 to n it is possible to generate a series of sets $GFP(k)$, such that $GFP(1) \supseteq GFP(2) \supseteq \dots \supseteq GFP(n)$ (see Figure 2). The following proposition characterizes the probability distribution of frequent patterns in $GFP(k)$.

Proposition 2. *Let P be a relational pattern at level l with support $sup(P, D_E)$. Then:*

$$Pr(P \in GFP(k)) = \sum_{i=k}^n \binom{n}{i} q^i (1-q)^{n-i} \quad (4)$$

where:

$$q = \sum_{v=[m \cdot \text{minsup}[l]]}^m \binom{m}{v} \cdot sup(P, D_E)^v \cdot (1 - sup(P, D_E))^{m-v}$$

is the probability that P is frequent in a sample extensional database.

¹ <http://glite.web.cern.ch/glite/>

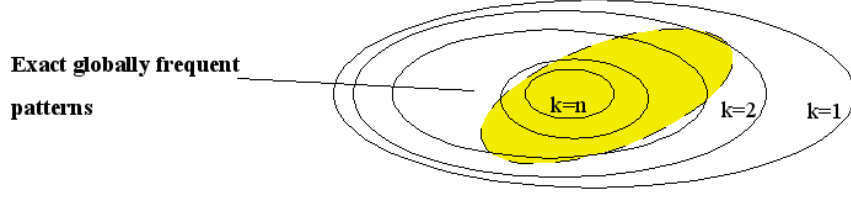


Fig. 2. The set of globally frequent patterns (GFP(k)) monotonically decreases with increasing k values. The set of frequent patterns in D_E (exact globally frequent patterns) includes most probably the set GFP(n) and is most likely included in the set GFP(1).

Proof. Let $S(P, D_E)$ be the support set of the relational pattern P in D_E , i.e., the set of units of analysis in D_E that are covered by P . Then

$$\text{sup}(P, D_E) = \frac{|\{s \in S \mid D[s] \in S(P, D_E)\}|}{|S|}. \quad (5)$$

Let D_E^j be a sample extensional database, $D_E^j \subseteq D_E$, with m units of analysis. The random selections of the m units of analysis to be added to D_E^j are independent trials, since units are sampled with replacement. By checking that each randomly selected unit of analysis belongs to the support set of P , we have a sequence of Bernoulli trials, where the probability of success is $\text{sup}(P, D_E)$. Therefore, the following random variable:

$X \stackrel{\text{def}}{=} \text{“}D_E^j \text{ includes exactly } v \text{ units of analysis which belong to } S(P, D_E)\text{”}$, has a binomial distribution:

$$\text{Pr}(X = v) = \binom{m}{v} \cdot \text{sup}(P, D_E)^v \cdot (1 - \text{sup}(P, D_E))^{m-v} \quad (6)$$

The probability that P is frequent in D_E^j can be defined as follows:

$$\text{Pr}(\text{sup}(P, D_E^j) \geq \text{minsup}[l]) = \text{Pr}(X \geq \lceil m \cdot \text{minsup}[l] \rceil) \quad (7)$$

From (6) it follows that:

$$\text{Pr}(X \geq \lceil m \cdot \text{minsup}[l] \rceil) = \sum_{v=\lceil m \cdot \text{minsup}[l] \rceil}^m \binom{m}{v} \cdot \text{sup}(P, D_E)^v \cdot (1 - \text{sup}(P, D_E))^{m-v} \quad (8)$$

Thus the probability that P is frequent in one sample extensional database is:

$$q = \sum_{v=\lceil m \cdot \text{minsup}[l] \rceil}^m \binom{m}{v} \cdot \text{sup}(P, D_E)^v \cdot (1 - \text{sup}(P, D_E))^{m-v} \quad (9)$$

In the case of n sample extensional databases, which are drawn independently, the probability that P is frequent in *exactly* k of them again follows a binomial distribution with probability q of success. Therefore, the probability that P is frequent in *at least* k sample extensional databases is:

$$Pr(P \in GFP(k)) = \sum_{i=k}^n \binom{n}{i} q^i (1-q)^{n-i} \quad (10)$$

□

Proposition 2 implies that frequent patterns in D_E are more likely to be in $GFP(k)$, for some fixed k . Indeed, if P is a frequent pattern in D_E at level l , then $sup(P, D_E)$ is relatively high (certainly it is greater than $minsup[l]$), therefore both q and $Pr(P \in GFP(k))$ tend to be relatively high. Moreover, proposition 2 also implies that for larger n , $Pr(P \in GFP(k))$ is higher, independently of whether P is frequent in D_E or not. This means that the recall of frequent patterns increases with n , at the expense of reduced precision (see next section for an empirical confirmation).

It is noteworthy that $Pr(P \notin GFP(1)) = (1-q)^n$, therefore, it is very unlikely that frequent patterns in D_E (i.e., patterns with high q value) are not in $GFP(1)$. Moreover, $Pr(P \in GFP(n)) = q^n$, therefore it is very unlikely that patterns locally frequent in all sample extensional databases are not frequent in D_E (i.e., they have a small q value).

The support of a globally frequent pattern P discovered at level l is approximated by averaging the support values computed on those samples where P is frequent. Formally:

$$approximateGlobalSup(P) = \frac{\sum_{j=1}^n sup(P, D_E^j) \cdot G(P, D_E^j)}{\sum_{j=1}^n G(P, D_E^j)} \quad (11)$$

where $G(P, D_E^j)$ is the indicator function defined as follows:

$$G(P, D_E^j) = \begin{cases} 1 & \text{if } sup(P, D_E^j) \geq minsup[l] \\ 0 & \text{otherwise.} \end{cases}$$

5 Experimental Results

In order to evaluate the proposed distributed frequent pattern discovery algorithm, we performed experiments on data derived from event logs. In ubiquitous computing applications event logs generated by systems in charge of polling sensors and registering which events happen in the environment when executing a given process instance. In this case, frequent pattern discovery is a mature technology to reconstruct an explanatory model of the underlying phenomenon from fragments of temporal sequences of actions performed by various actors. This model can then be used to explain (temporal) relationships between events,

to simulate/predict future behavior of any process as well as to understand and optimize the process itself. Since events may include activities and their properties, actors and their properties, relationships between activities and actors (who does what), temporal relationships between activities and additional relationships between actors, the natural choice for discovering frequent patterns is to resort to the relational data mining setting. In addition, multi-level analysis is a desirable characteristic, since activities and actors are often organized in hierarchies of classes. For example, an activity can be a ‘create’ task, a ‘delete’ task or an ‘execute’ task. An actor can be a user or an administrator. Multi-level analysis allows the mapping of these hierarchies into different levels of granularity and the extraction of fragments of process models (patterns) at various abstraction levels.

Experiments are performed by processing both an event log publicly available on ProM web site² and an event log provided by THINK3 Inc³. The goal of the experiments on the ProM data is to compare approximate global patterns, which are discovered in a distributed manner, with the exact global patterns, which are discovered by SPADA on the whole dataset. Statistics collected refer to both precision and recall of the parallel, distributed algorithm, as well as to the accuracy of the support estimation procedure. Statistics on time required for database sampling, local pattern discovery and globally frequent patterns approximation are also collected. The goal of experiments on the THINK3 data is to prove the actual scalability of the proposed method and its applicability to very large databases.

5.1 ProM Data

ProM database collects 374 executions of processes which handle the complaints (namely *Afschriften*) in a municipality in The Netherlands. The period under analysis is from May 4th 2005 to November 8th 2005. For each process instance, the database collects 24.5 activities on average. The total number of activities is 9,174, while the number of distinct actors is 29. Activities are classified as complete (1,343), schedule (6,673), resume (178), start (809), suspend (166) and unknown (5). Taxonomic knowledge on activities is encoded in a three level hierarchy (see Figure 3). Process instances play the role of reference objects, while activities and actors play the role of task-relevant objects.

For each activity, a textual description is registered in the event log. This description corresponds to the workflow name. In this experiment, we deal with 14 distinct workflows. The extensional database D_E includes 37,070 ground atoms. The intensional database D_I includes the definition of the temporal relations *simultaneous* and *before* which take into account the temporal autocorrelation of activities.

$$simultaneous(A1, A2) \leftarrow activity(C, A1), activity(C, A2), A1 \neq A2,$$

² <http://is.tm.tue.nl/~cgunther/dev/prom/>

³ <http://www.think3.com/en/default.aspx>

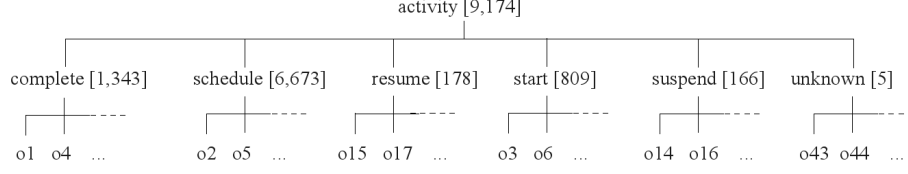


Fig. 3. ProM three-level hierarchy on activity.

$time(A1, T1), time(A2, T2), T1 = T2$.

$before(A1, A2) \leftarrow activity(C, A1), activity(C, A2), A1 \neq A2,$

$time(A1, T1), time(A2, T2), T1 < T2,$

$not(activity(C, A), A \neq A1, A \neq A2, time(A, T), T1 < T, T < T2).$

Different minimum support thresholds are defined for each level, so that the higher the level (i.e. the more abstract the task-relevant objects involved in the model), the higher the support (i.e. the more selective the discovery process). We set the following parameters $minsup[1] = 0.25$, $minsup[2] = 0.1$ and $maxLenPat = 9$. The latter defines the maximum number of atoms in the frequent relational patterns, which are evaluated during the search.

With the thresholds defined above, a set of 2,460 relational frequent patterns (FP) is discovered by mining the entire database. Two examples of frequent patterns discovered at level $l = 2$ are reported below.

P1: $process(A), activity(A, B), is_a(B, suspend), before(B, C), C \neq B,$
 $is_a(C, resume), before(C, D), D \neq B, D \neq C, is_a(D, schedule),$
 $simultaneous(D, E), E \neq B, E \neq C, E \neq D, is_a(E, complete).$

[#ro=61, sup=16.31%].

P1 describes the execution order between three activities. P1 is supported by 61 out of 374 executions (i.e., reference objects).

P2: $process(A), activity(A, B), is_a(B, start), actor(B, C), C \neq B, is_a(C, actor),$
 $before(B, D), D \neq B, D \neq C, is_a(D, schedule), workflow(B, ag08 GBA afnemer),$
 $workflow(D, ar01 Wacht Archief).$

[#ro=39, sup=10.42%].

P2 involves both activities and actors and is supported by 39 executions.

The frequent pattern discovery is then distributed on n sample databases, which contain $p\%$ units of analysis stored in the original database. Multi-level frequent patterns are locally discovered at each computation unit and global patterns are approximated from the local ones, by varying k from 1 to n .

To evaluate the quality of the approximation of the set of frequent patterns in D_E , we consider three statistics, namely *recall*, *precision*, and *F-score*, which are defined as follows [39]:

$$recall = \frac{|GFP(k) \cap FP|}{|FP|} \quad (12)$$

$$precision = \frac{|GFP(k) \cap FP|}{|GFP(k)|} \quad (13)$$

$$F - score = \frac{2 \times precision \times recall}{precision + recall}. \quad (14)$$

All these measures range between 0 and 1: the higher their values, the better the parallel, distributed algorithm performs. The recall estimates the probability $Pr(P \in GFP(k) | P \in FP)$, while the precision estimates the probability $Pr(P \in FP | P \in GFP(k))$. The F-score is the weighted harmonic mean of precision and recall, with equal weight for both recall and precision.

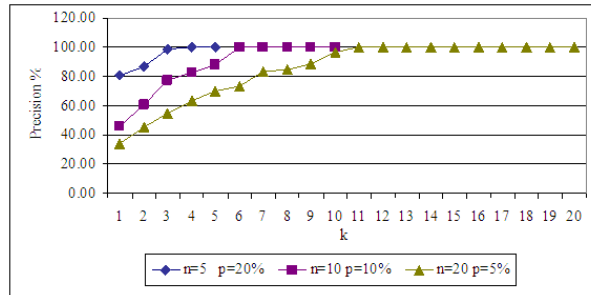
Some experiments are performed to empirically prove that recall decreases (conversely, precision increases) when k increases. We keep constant the exponent in formula 2, by setting $n = 1/p$. This way we ensure that about 62.3% of process instances are considered in the analysis. In particular, experiments are performed with the following parameter settings: $n=5$ and $p=20\%$, $n=10$ and $p=10\%$, $n=20$ and $p=5\%$.

The values of recall, precision and F-score obtained by varying the parameter k are reported in Figure 4. As expected, when $k=1$ recall is high (100%), although precision is low (between 40% and 60%). Conversely, recall decreases for increasing k values. The F-score increases with k up to a maximum and then decreases. The maximum is obtained when $k \approx \frac{n}{2}$, which represent the best trade-off between precision and recall. For example, when $n=10$ and $p=10\%$ the distributed version of SPADA discovers 2,732 patterns with $k=5$. A closer analysis of these patterns reveals that they cover 99.67% of the patterns that are frequent on the entire database, at the expense of some false positives (about 10.24%).

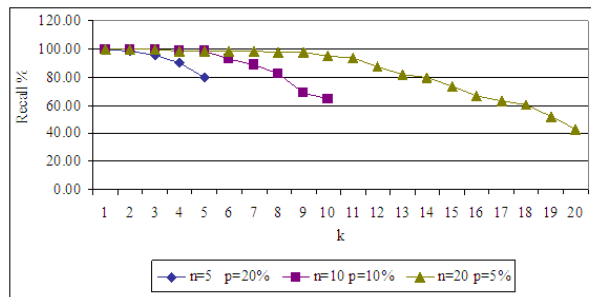
An additional set of experiments is performed by doubling the number of database samples, i.e., by setting $n = 1/p$. The F-score for various values of k is reported in in Figure 5. Also in this set of experiments, the maximum of the F-score is observed for values of k approximately equal to $\frac{n}{2}$.

The last observation raises the question of the real advantage of increasing the number of database samples, once p is fixed. To answer this question, the difference between the approximated support, computed according to formula 11, and the exact support, computed on D_E , is analyzed. The difference is computed for each exact pattern discovered by the distributed version of SPADA when $p=10\%$ and n is set to either $1/p$ or $1/2p$. Box plots (see Figure 6) are drawn by varying $k \in \{1, n/2, n\}$. We observe that:

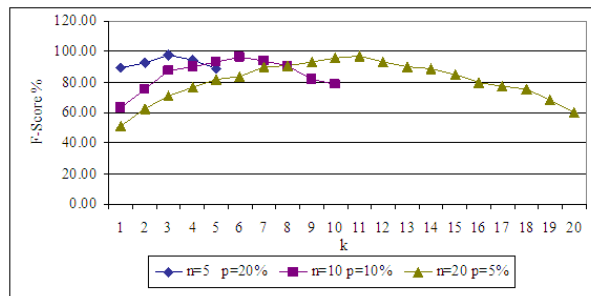
- when k approximates n , although the recall decreases, the estimated support of globally frequent patterns becomes closer to the exact support value;



(a)



(b)



(c)

Fig. 4. Precision (a), recall (b), and F-score (c) obtained with $n=5, p=20\%$, $n=10, p=10\%$ and $n=20, p=5\%$, by varying k in $[1,20]$.

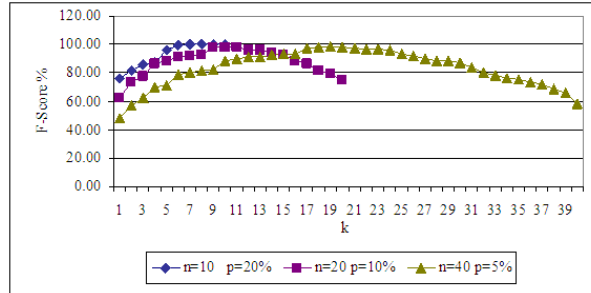


Fig. 5. F-score obtained with $n=10$, $p=20\%$, $n=20$, $p=10\%$ and $n=40$, $p=5\%$, by varying k in $[1,40]$.

- by doubling the number of samples, the support of globally frequent patterns is more accurate, independently of the chosen k .

Therefore, the real advantage of choosing larger values of n is the increased accuracy of the estimated support, i.e., the increased maximum value of the F-score. The disadvantage is the higher computational cost.

Running SPADA on the entire ProM database takes 1,350 secs. Statistics on time required by the distributed version of SPADA to generate database samples, to discover locally frequent patterns and to select those in $GFP(k)$ are reported in Tables 1, 2 and 3 respectively. Time of locally frequent pattern discovery is averaged over the database samples, while time of pattern selection is averaged over the k values.

We observe that time required for database sampling and pattern selection is negligible with respect to the average time required by the locally frequent pattern discovery processes. Therefore, if $f(N, l, minsup[l])$ is the time required by SPADA when it searches for frequent patterns of level l in the entire database D_E of N units of analysis, the amount of work done by the parallel, distributed relational pattern algorithm is approximately $n \cdot f(p \cdot N, l, minsup[l])$. Table 2 shows that for ProM database, $f(p \cdot N, l, minsup[l])$ is of the same order of magnitude of $f(N, l, minsup[l])$. Therefore, for small databases, like ProM, which fit in main memory, the generation of an approximate set of globally frequent patterns is not even beneficial from a computational viewpoint. The parallel, distributed

n	p		
	5%	10%	20%
5	-	-	8
10	-	11	14
20	20	26	-
40	56	-	-

Table 1. Time (in secs) of database sampling.

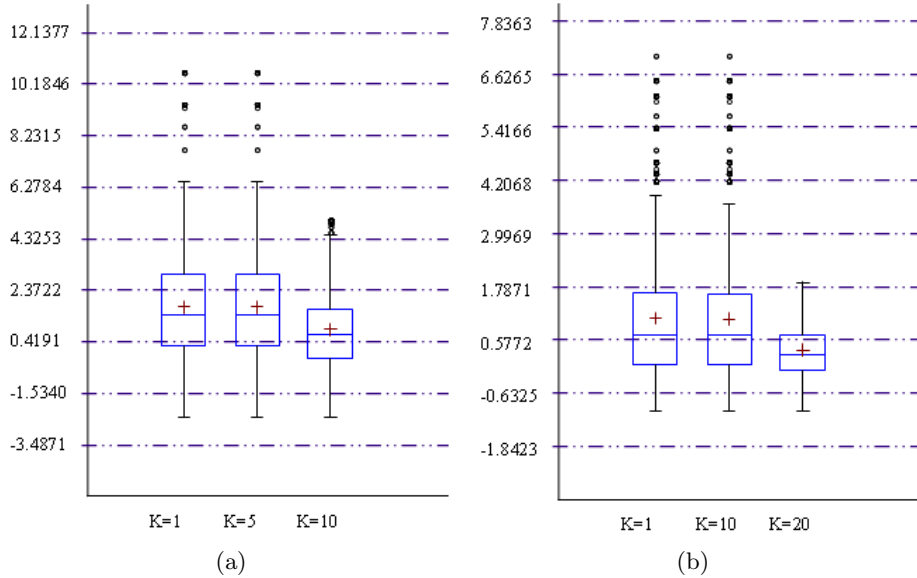


Fig. 6. Box plots of the differences between the approximate support of each pattern and the exact support, computed on the entire database. In both cases $p = 10\%$. In (a) $n = 1/p$ while in (b) $n = 1/2p$.

n	p		
	5%	10%	20%
5	-	-	1053
10	-	1286	1047
20	1480	1189	-
40	1493	-	-

Table 2. Average time (in secs) of locally frequent pattern discovery.

algorithm is suitable when the database does not fit in main memory (see next section) or, more in general, when $f(N, l, \text{minsup}[l]) \gg f(p \cdot N, l, \text{minsup}[l])$.⁴

Table 2 also shows the behavior of $f(p \cdot N, l, \text{minsup}[l])$. Indeed, the average time decreases from 1,493 to 1,053 when p increases from 5% to 20%. Moreover, it starts increasing for higher values of p (it is 1,350 when $p = 100\%$). This is justified by the following observation. For small datasets (small p values), the cost of evaluating a single pattern is low, but the number of false frequent patterns is high (see the difference in precision of $GFP(1)$ for various values of p in Figure 4a) and the pattern space is not pruned enough. On the contrary, for large datasets (large p values), the cost of evaluating a single pattern is higher, but there are few false frequent patterns and the search space is properly pruned.

⁴ When the database does not fit in main memory, we assume $f(N, l, \text{minsup}[l]) = \infty$.

n	P		
	5%	10%	20%
5	-	-	4
10	-	6	7
20	11	40	-
40	17	-	-

Table 3. Time (in secs) of pattern selection for GFP(k).

5.2 THINK3 Data

THINK3 event log describes 353,490 executions of business processes in a company. The period under analysis is from April 7th 2005 to January 10th 2007 for a total of 1,035,119 activities and 103 actors. Activities are classified as administrator tools (131), workflow (919052), namemaker (106839), delete (2767), deleteEnt (2354), prpDelete (471), prpSmartDelete (53), prpModify (34) and cast (1430). Actors are classified as user (103), viewer (3) or administrator (2). Taxonomic knowledge on activities and actors is encoded in two distinct hierarchies (see Figure 7).

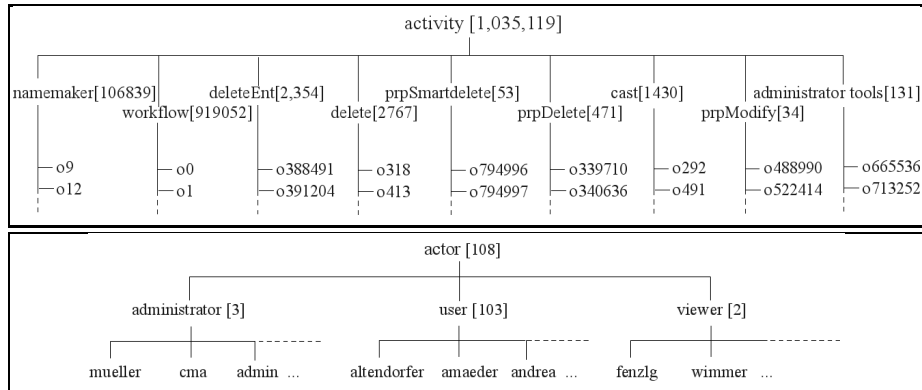


Fig. 7. THINK3 three-level hierarchies on activity and actor.

For each activity a textual description is registered in the event log, while for each actor a working group is defined. In this experiment, we have thirteen distinct descriptions of the activities and thirty-three distinct groups of actors. The extensional database D_E includes 4,374,840 ground facts, while the intensional part D_I includes the definition of the relation *before*. Additional predicates are intensionally defined to group together similar activities. For example, the following clauses:

$$release(X) \leftarrow description(X, freigabe).$$

Table 4. Number of global frequent patterns discovered by varying k in [1,100]

k	1	10	20	30	40	50	60	70	80	90	100
#P	1244	1043	820	747	669	619	574	539	498	468	369

$release(X) \leftarrow description(X, freigabe_h).$
 $release(X) \leftarrow description(X, freigabe_j).$
 $release(X) \leftarrow description(X, freigabe_m).$

define the predicate “release”, which describes the releasing activity (“freigabe” in German), independently of the release type (H, J or M). Similarly, other clauses in D_I provide a definition of new predicates, such as, *pruefung*, *techaend*, *cancelled*, *construction*, *ktgprocess*, *musterbau*, *nullserie*, *techniche*, *tiffprocess*, *undermodify* and *workinprogress*, which describe different kinds of activities. Process instances play the role of reference objects, while activities and actors play the role of task-relevant objects.

Data are sampled with $n=100$ and $p=1\%$, and the discovery of the local patterns is parallelized on 100 nodes. The size of the original database prevents the original algorithm SPADA from processing units of analysis all at once. We set the following parameters: $minsup[1] = 0.25$, $minsup[2] = 0.1$, $minsup[3] = 0.01$ and $maxLen_path = 14$. Actually, with the thresholds defined above, there are no frequent patterns with more than twelve atoms, and very few with more than fourteen atoms.

The exact set of global patterns is approximated from the sets of patterns which are locally discovered on each sample, by varying k from 1 to 100. The number is reported in Table 4 and, as expected, it decreases when k increases. The average number of local patterns (at any level) discovered on each sample is 673.11, while the standard deviation is relatively small (53.47). As reported in the last column of Table 4, 369 local patterns (about 54% on average) are common to all samples.

A pattern which expresses a fragment of a process model at level $l = 2$ is as follows:

P3: $process(A)$, $activity(A,B)$, $is_a(B,workflow)$, $before(B,C)$, $C \neq B$,
 $is_a(C,workflow)$, $before(C,D)$, $D \neq B$, $D \neq C$, $is_a(D,workflow)$, $actor(B,E)$,
 $is_a(E,user)$, $workinprogress(B)$, $release(D)$, $construction(C)$.
 $[k=90, \#ro=19789 \text{ approximateGlobalSup} \approx 20.78]$.

This pattern reports the execution order between three activities (B , C and D), within a process instance (A). Both B , C and D are workflow activities, but B is described as *work in progress*, C as *release* and D as *construction*. The actor of B is a simple user (E). The pattern P3 provides an explanatory model for at least 19,789 process executions traced in the entire event log and it is found to be frequent in at least 90 of the original 100 samples ($k = 90$). Its approximate support is 20.78% ($> minsup[2]$).

P3 is an ancestor of the following globally frequent pattern:

P4: $process(A), activity(A,B), is_a(B,workflow), before(B,C), C \neq B,$
 $is_a(C,workflow), before(C,D), D \neq B, D \neq C, is_a(D,workflow), actor(B,E),$
 $is_a(E, andrea), workinprogress(B), release(D), construction(C).$
[$k=90, \#ro=1139$ approximateGlobalSup ≈ 1.26].

which provides us with a deeper insight into the actor of the work-in-progress, who is identified as *andrea*. This pattern, however, covers only 1,139 process executions and its approximate global support is 1.26% ($> minsup[3]$).

As reported before, for this data set it was not possible to apply the original algorithm SPADA to the entire database, as done for ProM data. Therefore, statistics on precision, recall and F-score could not be collected. The main goal of this experiment is to prove the scalability of the proposed distributed method, even in the presence of very large databases. However, a careful setting of the two parameters p and n is likely to lead to good approximations of the global set of patterns, as in the previous experiments.

6 Conclusions

Data generated by ubiquitous computing applications present several distinctive characteristics which complicate the knowledge discovery process [25]. First, they are produced asynchronously in a highly decentralized way. Second, they emerge from a very high number of partially overlapping, loosely connected sources. Third, they are produced in large quantities. Fourth, they are usually described by many different data types. Fifth, they have a complex inner structure with several explicit or implicit relationships. This paper faces the last three issues when mining frequent patterns. In particular, we advocate a relational approach, in order to face the last two issues (heterogeneity and complex interactions) and a distributed, parallel approach, in order to face the scalability issue.

The algorithm proposed for relational frequent pattern discovery is based on random sampling. In particular, n samples are extracted from the original data set, such that each of them fits in main memory. Data are shipped to computational nodes of a Grid and relational patterns, which are locally frequent, are mined by means of the system SPADA. The sets of locally frequent patterns are then used to generate the set of globally frequent patterns, $GFP(k)$, by selecting only those relational patterns which are locally frequent in at least k ($k \leq n$) samples. A characterization of the probability distribution of frequent patterns in $GFP(k)$ allows us to draw some conclusions on precision and recall of $GFP(k)$, with respect to the set of frequent patterns (FP) computed on the whole data set. These conclusions have been empirically proved by evaluating the proposed algorithm on a data set of logged events (ProM data), which could be wholly processed in main memory. Statistics on the differences between estimated and actual support of globally frequent patterns show that when k approximates n , although the recall decreases, the estimated support of globally frequent patterns

becomes closer to the exact support value. Scalability of the proposed method has been proved on a very large data set of logged events (THINK3 data), which could not be directly processed by the original serial version of SPADA.

The choice of k is crucial in our proposal. Experimental results on ProM data showed that the best trade-off in maximizing both precision and recall is found when $k \approx \frac{n}{2}$. However, the dependence of k on both n and minimum support thresholds has to be more deeply investigated, both theoretically, by characterizing and maximizing the conditional probabilities $Pr(P \in GFP(k)|P \in FP)$ and $Pr(P \in FP|P \in GFP(k))$, and empirically, by means of additional experiments under controlled conditions. We also observe that SPADA is able to generate frequent patterns at different levels of granularity. Generally, the higher the level, the lower the support of discovered patterns. Therefore, we expect that our future investigations on the selection of the best k value will also allow us to define a suitable choice of this parameter for the various granularity levels.

We also intend to extend our analysis to alternative sampling-based approaches proposed in the literature for conventional association rule mining [38, 40], which are based on a single sample whose size is theoretically derived either on the basis of the Chernoff bound or on the basis of (an approximation of) the binomial distribution.

This work is limited to frequent pattern discovery, while SPADA is actually able to generate association rules from frequent patterns. The efficient evaluation/estimation of the confidence of association rules generated from $GFP(k)$ requires further investigation.

Finally, we intend to study an alternative approach to the scalability issues, which is based on the transformation of relational data into a propositional form. For this purpose, globally frequent patterns can be used to define the new boolean features of the propositional data set, while efficient algorithms for (propositional) frequent itemset mining can be applied to the whole transformed data set.

Acknowledgments

This work is in partial fulfillment of the research goals of the projects DIPIS (Distributed Production as Innovative System) and TOCAL.it (Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet). The authors wish to thank THINK3 Inc. for having provided the process data used in the experiments. The authors gratefully acknowledge the anonymous reviewers for their constructive suggestions and Lynn Rudd for reading the final version of this paper.

References

1. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *International Conference on Management of Data*, pages 207–216. ACM, 1993.

2. R. Agrawal and J. C. Shafer. Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):962–969, 1996.
3. H. Blockeel and M. Sebag. Scalability and efficiency in multi-relational data mining. *SIGKDD Explorations*, 5(1):17–30, 2003.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. R. Camacho. Improving the efficiency of ILP systems. In F. Moura-Pires and S. Abreu, editors, *Progress in Artificial Intelligence, 11th Portuguese Conference on Artificial Intelligence, EPIA 2003*, volume 2902 of *LNAI*, pages 224–228. Springer, 2003.
6. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.
7. D. W. Cheung, J. Han, V. T. Ng, A. W. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *the 4th International Conference on Parallel and Distributed Information Systems, DIS 1996*, pages 31–43, Washington, DC, USA, 1996. IEEE Computer Society.
8. A. Clare and R. D. King. Data mining the yeast genome in a lazy functional language. In *the 5th International Symposium on Practical Aspects of Declarative Languages, PADL 2005*, pages 19–36, London, UK, 2003. Springer-Verlag.
9. A. Clare, H. E. Williams, and N. Lester. Scalable multi-relational association mining. *Data Mining, IEEE International Conference on*, 0:355–358, 2004.
10. V. S. Costa, A. Srinivasan, R. Camacho, H. Blockeel, B. Demoen, G. Janssens, J. Struyf, H. Vandecasteele, and W. V. Laer. Query transformations for improving the efficiency of ILP systems. *Journal of Machine Learning Research*, 4:465–491, 2003.
11. L. Dehaspe and L. De Raedt. Mining association rules in multiple relations. In *the 7th International Workshop on Inductive Logic Programming, ILP 1997*, volume 1297, pages 125–132. Springer-Verlag, 1997.
12. L. Dehaspe and L. D. Raedt. Parallel inductive logic programming. In *Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995.
13. P. Domingos. Toward knowledge-rich data mining. *Data Mining and Knowledge Discovery*, 15(1):21–28, 2007.
14. S. Džeroski and N. Lavrač. *Relational Data Mining*. Springer-Verlag, 2001.
15. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, London, 1994.
16. N. A. Fonseca, F. M. A. Silva, and R. Camacho. Strategies to parallelize ILP systems. In *the 15th International Conference on Inductive Logic Programming, ILP 2005*, volume 3625 of *Lecture Notes in Computer Science*, pages 136–153. Springer-Verlag, 2005.
17. N. A. Fonseca, F. M. A. Silva, V. S. Costa, and R. Camacho. A pipelined data-parallel algorithm for ILP. In *IEEE International Conference on Cluster Computing, CLUSTER 2005*, pages 1–10. IEEE Computer Society, 2005.
18. J. Graham, C. D. Page, and A. Kamal. Accelerating the drug design process through parallel inductive logic programming data mining. In *IEEE Computer Society Conference on Bioinformatics, CSB 2003*, page 400, Washington, DC, USA, 2003. IEEE Computer Society.
19. A. Greenfield. *Everyware: The Dawning Age of Ubiquitous Computing*. New Riders Publishing, 2006.
20. N. Helft. *Progress in Machine Learning*, chapter Inductive generalization: a logical framework, pages 149–157. Sigma Press, 1987.

21. S. Konstantopoulos. A data-parallel version of ALEPH. *Proceedings of the Workshop on Parallel and Distributed Computing for Machine Learning*, abs/0708.1527, 2007.
22. F. A. Lisi and D. Malerba. Inducing multi-level association rules from multiple relations. *Machine Learning*, 55(2):175–210, 2004.
23. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, second edition, 1987.
24. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
25. M. May and L. Saitta. Blueprint in ubiquitous knowledge discovery. Technical report, Deliverable of the European project IST-6FP-021321 “KDubiq - Knowledge Discovery in Ubiquitous Environments”, 2008.
26. S. Muggleton. *Inductive Logic Programmings*. Academic Press, London, 1992.
27. C. Nédellec, C. Rouveirol, H. Adé, F. Bergadano, and B. Tausend. Declarative bias in ILP. In *Advances in Inductive Logic Programming*, pages 82–1038. IOS Press, Amsterdam, 1996.
28. S. Orlando, P. Palmerini, R. Perego, and F. Silvestri. Adaptive and resource-aware mining of frequent sets. In *the 2002 IEEE International Conference on Data Mining, ICDM 2002*, page 338, Washington, DC, USA, 2002. IEEE Computer Society.
29. J. S. Park, M.-S. Chen, and P. S. Yu. An effective hash-based algorithm for mining association rules. *SIGMOD Rec.*, 24(2):175–186, 1995.
30. J. S. Park, M.-S. Chen, and P. S. Yu. Efficient parallel data mining for association rules. In *the 4th International Conference on Information and Knowledge Management, CIKM 1995*, pages 31–36, New York, NY, USA, 1995. ACM.
31. G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
32. S. Price and P. Flach. Querying and merging heterogeneous data on approximate joins on higher-order terms. In *the 18th International Conference on Inductive Logic Programming, ILP 2008*, volume 5194 of *LNAI*, pages 226–243. Springer-Verlag, 2008.
33. F. Provost. Distributed data mining: Scaling up and beyond. In H. Kargupta and P. Chan, editors, *In Advances in Distributed and Parallel Knowledge Discovery*, pages 3–27. AAAI/MIT Press, 1999.
34. A. Schuster and R. Wolff. Communication-efficient distributed mining of association rules. In *the 2001 ACM SIGMOD International Conference on Management of Data*, pages 473–484, New York, NY, USA, 2001. ACM.
35. A. Schuster, R. Wolff, and D. Trock. A high-performance distributed algorithm for mining association rules. *Knowledge and Information Systems*, 7(4):458–475, 2005.
36. C. Silvestri and S. Orlando. Distributed approximate mining of frequent patterns. In *the 2005 ACM Symposium on Applied Computing, SAC 2005*, pages 529–536, New York, NY, USA, 2005. ACM.
37. R. P. Singh, A. Turi, and D. Malerba. Grid-based data mining for market-basket analysis in retail sector. In *the 8th International Conference on Data, Text and Web Mining and their Business Applications including Information Engineering, Data Mining 2007*, Southampton, UK., 2007. WIT Press.
38. H. Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *22th International Conference on Very Large Data Bases, VLDB 1996*, pages 134–145. Morgan Kaufmann, 1996.

39. C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.
40. M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *the 7th International Workshop on Research Issues in Data Engineering, RIDE 1997*, page 42, Washington, DC, USA, 1997. IEEE Computer Society.