Sampling Very Large Databases for Parallel and Distributed Relational Frequent Pattern Discovery

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 stored in relational databases is quickly growing due to the pervasive presence of sensors. Processing these data in their original relational form requires the application of (Multi-)Relational Data Mining algorithms, but most of them are unable to process very large databases. In this paper we propose an extension of a multi-relational algorithm for frequent pattern discovery which resorts to a data sampling strategy and to distributed computation in grid environments in order to overcome the computational limits of the original serial algorithm. Discovered patterns are an approximation of the exact solution found by the serial algorithm. An application to event log mining proves the viability of the proposed extension to large databases without significantly affecting the quality of extracted patterns.

1 Introduction

The pervasive presence of sensors in real life is driving the recent interest in the area of ubiquitous computing. The success of prospected technological advancements, however, strongly depends on the actual capacity of newer applications to exploit all information which can be extracted from the huge amount of data produced by sensors. This requires improved knowledge discovery methods which can handle large collections of data which are heterogeneous and naturally modeled by means of several relations of a database.

Studies in (Multi-)Relational Data Mining (MRDM) [5] and Inductive Logic Programming (ILP) [13] offer a rather complete set of solutions to mine different forms of patterns and models (e.g., association rules, classification rules, clusters, ...) from relational data. A problem with these solutions remains the actual applicability to very large databases because of the intrinsic computational complexity of the solved problems. In the case of frequent pattern discovery, which is the main subject of this work, the high complexity also depends on the exponential size of search space [14].

The need of scalable and high performance systems motivates the resort to parallel and distributed data mining solutions. At the best of our knowledge, there is no attempt in the literature to parallelize the process of frequent patterns discovery in the multi-relational setting. Previous works propose solutions

to parallelize relational data mining algorithms for the predictive task of classification. Strategies to parallelize ILP systems and to speed up the learning time are presented in [3, 6]. However, all proposed solutions work in a shared-memory architecture and do not permit to have a real advantage in terms of space complexity. Methods proposed to work in a distributed memory architecture [10, 8, 7] have been developed only for the classification task.

In this paper, we tackle the problem of parallelizing and distributing the relational frequent pattern discovery in order to analyze very large databases. Transactional data are stored as Datalog atoms in the extensional part of a deductive database, while domain knowledge, if available, is expressed as a normal logical program which defines the intensional part of the deductive database. The proposed method, called sG-SPADA, extends the serial algorithm SPADA [11] developed for multi-relational association rule discovery, and enhances the distributed algorithm G-SPADA [16]. Both sG-SPADA and its predecessor G-SPADA discover approximate relational frequent patterns by means of a three stepped strategy which (i) extracts n subsets from the original database, (ii) distributes the discovery of locally frequent patterns on separate computation units and (iii) determine approximate globally frequent patterns from the local ones.

sG-SPADA enhances G-SPADA along three directions. First, sG-SPADA works on samples of any size of the original database, while G-SPADA works on partitions of initial data, with the consequence that the size of each partition is linked to the number of partitions. Second, sG-SPADA demands the computation of intensionally defined predicates (saturation step) to each node of the grid, while G-SPADA performs saturation on the whole database before partitioning, with the consequence that one node of the grid should be well-provided to perform this resource-consuming task. Third, sG-SPADA optimizes G-SPADA data transmission by providing the computation nodes with only data actually needed for local pattern mining.

The paper is organized as follows. In the next section the theoretical background for relational frequent pattern discovery is introduced. The distributed and parallel algorithm sG-SPADA is described in Section 3, while experimental results on two event log mining tasks are reported and discussed in Section 4.

2 Relational Frequent Pattern Discovery

The task addressed in this work concerns the discovery of (frequent) relational patterns which involve properties and relationships between objects represented by means of several tables of a relational database. WARMR [4] and SPADA[11] are two ILP systems which address this task. They both represent relational data *à la* Datalog [2], a logic programming language with no function symbols specifically designed to implement deductive databases. Moreover, they both take into account a domain (or background) knowledge (BK) expressed in Prolog. The main difference is that WARMR is not able to mine relational patterns at multiple levels of granularity since it lacks of mechanisms for dealing properly

with concept hierarchies. When these are available, it is important to take them into account since association rules involving more abstract objects are better supported but less precise, while association rules involving more specific objects present low support but higher confidence values. Hence by efficiently exploring the patterns space at different levels of abstraction (or granularity) it is possible to find the right trade-off between these two conflicting criteria.

In this work, we focus on SPADA because the application domain considered in the experiments, i.e. process mining, allows for the natural definition of hierarchies on the concepts of actors and activities, which are then used to find patterns at different levels of granularity. Nevertheless, the extensions reported in this paper are general enough to be applicable to WARMR as well. The representation formalism adopted in SPADA for both input data and discovered patterns is detailed in the following subsection.

2.1 Representing Relational Data and Patterns

Data stored in different tables of a relational database describe different objects involved in the phenomenon under study. These objects play different roles, and it is necessary to distinguish between the set S of *reference* (or target) *objects*, which are 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 account for the variation. A reference object is described not only by its own attributes (in the target table). It also has an identifier that maps into bags of related task-relevant objects from different background tables.

In the logic framework adopted by SPADA, a relational database is boiled down into a deductive database. Properties of both reference and task-relevant objects are represented in the extensional part D_E, while the domain knowledge is expressed as a normal logic program which defines the intensional part D_i.

Example 1. A deductive database D describing process executions, where constants $c1$ and $c2$ denote two distinct process executions (reference objects), while $a1$, $a2$, $a3$, and $a4$ denote four distinct actions and $u1$ and $u2$ two distinct promoters (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).
promoter(a1,paul). promoter(a2,paul). promoter(a3,paul). promoter(a4,mary).
is_a(paul,reader). is_a(mary ,admin).

while D_I is the normal logic program:

before(A1, A2) $\leftarrow$ *activity(C, A1),activity(C, A2),*
time(A1,T1), A1≠ A2, 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 refined into four classes. The *key predicate* identifies the reference objects (in the Example 1, *process* is the key predicate). 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., *promoter*) as well as reference objects with task-relevant objects (e.g., *activity*). The *is_a* predicate defines the types of objects. For each type, a generalization hierarchy H_k $(k = 1, .., M)$ is defined. In Example 1, the following hierarchies can be defined:

$[a1] \rightarrow$ *namemaker*; $[a2, a3] \rightarrow$ *workflow* ; $[a4] \rightarrow$ *delete*
$[$*namemaker, delete, workflow*$] \rightarrow$ *activity*
$[mary] \rightarrow$ *admin*; $[paul] \rightarrow$ *reader*
$[$*admin, reader*$] \rightarrow$ *promoter*

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:

$$D[s] = is\_a(R(s)) \cup D[s|R(s)] \cup \bigcup_{r_i \in R(s)} D[r_i|s] \text{ where:}$$

- $R(s)$ 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 r_i;
- $D[r_i|s]$ contains properties of r_i and relations (indirectly) relating r_i to s.

In the specific application to log analysis, the set of units of analysis is a partitioning of D_E.

Example 2. The unit of analysis $D[c1]$ includes the ground atoms:
is_a(a1,namemaker). is_a(a2,workflow). is_a(paul,reader).
process(c1). activity(c1,a1). activity(c1,a2).
time(a1,10). time(a2,25). promoter(a1,paul). promoter(a2,paul).

This notion of unit of analysis is coherent with the individual-centered representation [1] which has both theoretical (PAC-learnability) and computational advantages (smaller hypothesis space and more efficient search).

The definition of a relational pattern is formulated in the following.

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

Each p_i is either extensionally or intensionally defined. 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 formal definition of the support of P on D.

Definition 2. *A pattern P* covers *$D[s]$ if $D[s] \cup BK$ logically entails $eqc(P)$.*

Each relational pattern is associated with a parameter $s\%$, 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 3. *A pattern* P [$s\%$] *at level* l *is* frequent *if* $s \geq minsup[l]$ *and all ancestors of* P *with respect to* H_k *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 frequent patterns are discovered at level 1 and 2 respectively:
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%].

2.2 Ordering Patterns in the Search Space

SPADA performs both an intra-level search and inter-level search. The intra-level search is performed in the space of patterns where the *is_a* atoms refer to objects defined at the same level of generalization hierarchies. This space is ordered according to the θ-subsumption generality order between patterns.

Definition 4. 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 relational patterns:
$P_1 \equiv$ *is_a(B,namemaker)*
$P_2 \equiv$ *is_a(B,namemaker)* $\wedge$ *before(B,C)*
$P_3 \equiv$ *is_a(B,namemaker)* $\wedge$ *before(B,C)* $\wedge$ *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 $\theta_1 = \theta_2 = \oslash$.

The relation $\succeq_\theta$ is a quasi-ordering (or preorder) since it is reflexive and transitive but not antisymmetric. The monotonicity of $\succeq_\theta$ with respect to support allows for the application of the level-wise method described in [12].

In the inter-level search, SPADA refines patterns discovered at level l by descending the generalization hierarchies of one level. Indeed, by definition of frequent pattern, a necessary condition for pattern P being frequent at level $l+1$ is that an ancestor pattern P' at level l exists, such that P' is frequent. The ancestor pattern is obtained by considering more abstract task-relevant objects (with respect to some generalization hierarchy) than those involved in P. Therefore, the inter-level search takes advantage of statistics computed at a level l to prune the search space at level $l+1$.

Once frequent patterns are discovered, association rules can be generated. This step performed by SPADA is out of the scope of this paper.

3 Parallel and Distributed Relational Pattern Discovery

The application of SPADA to very large database is not straightforward [9] since the high computational cost of search and the usage of in-memory deductive database do not allow for mining large data. sG-SPADA overcomes computational limits of SPADA by distributing and (possibly) parallelizing the discovery

of local frequent relational patterns on random-multisampling of the original database and then deriving an approximation of the set of exact global frequent patterns (patterns to be discovered on the entire database) from the various sets of local patterns.

Random multi-sampling is a method to generate n samples with replacement each of which includes a percentage p of the data in the original database. The samples generated are not a data partitioning, so even 10 samples with $p = 10\%$ do not generally correspond to the entire database. The choice of multi-sampling, as an alternative to data partitioning, is motivated by results in [17, 15] which confirm the intuition that sampling speeds up the mining process by more than an order of magnitude by reducing I/O costs drastically shrinking the number of objects to be considered. Moreover, when training data are kept in main memory as in SPADA, sampling is the only way to make their analysis feasible. By working on sampled databases, it is possible to distribute the computation on a Grid and to parallelize the execution of the data mining algorithm. The problem of discovering spurious local patterns can be mitigated by increasing the number of samples, so that the set of "approximate" global patterns derived from local patterns is more likely representative of the set of "exact" global patterns.

3.1 Relational Data Sampling

In sG-SPADA sampling aims at transforming the original problem of frequent patterns discovery into several simpler problems analogous to the original. Each unit of analysis is added to a sample with probability $p = 1/m$ in order to generate a sample extensional database $D_{E_{Sample}}$ of fixed size m. The probability that a particular reference object is not picked in a set of N units of analysis can be estimated as $(1 - 1/N)^{nm} = (1 - 1/N)^{n/p}$. When $p = n/N$, i.e., $m = N/n$, then the probability estimate approximates e^{-1} for large N (e.g. the case of large samples) with $e = 2.7183$ the base of natural logarithms. Since $e^{-1} = 0.368$ this means that even in the case in which we have n samples of N/n transactions, only 63.2% of units of analysis will be considered. This gives us a criterion to set the parameter n once the size of a sample is defined, i.e. either m or p are specified. In particular, when $n = N/m = N \cdot p$ we are sure that approximately 63.2% of units of analysis are considered.

3.2 Distributing Computation on Grid

A sample extensional database $D_{E_{Sample}}$ and the intensional database D_I may be shipped along with sG-SPADA to computation nodes on Grid using gLite middleware. This is done by submitting parametric jobs described in JDL (Job Description Language) through the CLI (command line interface). Submission of jobs on Grid is performed in several steps, namely i) authentication, ii) preparation of the jobs, iii) uploading the sample databases (stage-in), iv) submission of a relative parametric job, v) checking/waiting results, vi) getting the results (stage-out).

3.3 Computing Approximate Global Frequent Patterns

The n sets of relational patterns which are (locally) frequent in sample databases are collected from the corresponding computation nodes of the Grid platform and then merged in order to approximate the set of patterns which are potentially frequent on the original database. Local frequent patterns that occur in less than k ($k \leq n$) sets of patterns are filtered out. The test that the same local pattern occurs in different sets is based on an *equivalence test* between two patterns P and Q under θ-subsumption, i.e. $P \succeq_\theta Q$ and $Q \succeq_\theta P$.

For each pattern that is locally frequent on at least k data samples (*k-locally frequent pattern*), sG-SPADA derives an approximation of the global support by averaging the support values of the local patterns. The approximate support of an n-locally frequent pattern is very close the true support of the same pattern on the entire dataset.

4 Experimental Results

Experiments are performed by processing both an event log publicly available on ProM web site[1] and an event log provided by THINK3 Inc[2].

4.1 ProM Data

ProM database collects 374 instances of processes related to handling of complaints *Afschriften* in a municipality in The Netherlands. The period under analysis is from May 4th, 2005 to November 8th, 2005. For each execution, the database collects 24.5 events on average. Each event describes an activity and its performer and it is associated with a timestamp. The total number of activities is 9,174, while the number of distinct promoters 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 and promoters is encoded in two distinct hierarchies, each of which is mapped into three granularity levels. For each activity, a textual description is registered in the event log. This description corresponds to the name of the workflow model. In this experiment, we deal with 14 distinct workflow models. The extensional database D_E includes 37,070 ground atoms. Reference objects are the process executions, while task-relevant objects are both the activities and the promoters. The intensional database D_i includes the definition of the predicates *simultaneous* and *before* which allow to take into account the temporal autocorrelation of events.

simultaneous(A1, A2) ← *activity(C, A1), activity(C, A2), A1≠ A2,*
time(A1,T1), time(A2,T2), T1=T2.
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).

[1] http://is.tm.tue.nl/∼cgunther/dev/prom/
[2] http://www.think3.com/en/default.aspx

In this experiment different minimum support thresholds are defined for each level, such that the higher the level (i.e. the more abstract the task-relevant objects involved in the pattern), the higher the support (i.e. the more selective is the discovery process). In particular, we set the following parameters $minsup[1] = 0.25$, $minsup[2] = 0.1$ and max_len_pat=9. The last parameter defines the maximum number of atoms in a frequent pattern considered by SPADA during its search. With the thresholds defined above, there are 2,460 relational patterns which are discovered by mining the entire database without resorting to any sampling procedure (Exp_{DB}). Some examples of relational patterns discovered at level $l = 2$ are the following:

P1: *process(A), activity(A,B), is_a(B,suspend), before(B,C), C≠B, is_a(C,resume), before(C,D), D≠B, D≠C, is_a(D,schedule), simultaneous(D,E), E≠B, E≠C, E≠D, is_a(E,complete).* [#ro=61, supp=16.31%]

P1 describes the execution order between three activities. This pattern is supported by 61 out of 374 executions (support= 16.31%).

P2: *process(A), activity(A,B), is_a(B,start), originator(B,C), C≠B, is_a(C,a21), before(B,D), D≠B, D≠C, is_a(D,schedule), workflow(B,ag08 GBA afnemer), workflow(D,ar01 Wacht Archief).* [#ro=39, supp=10.42%]

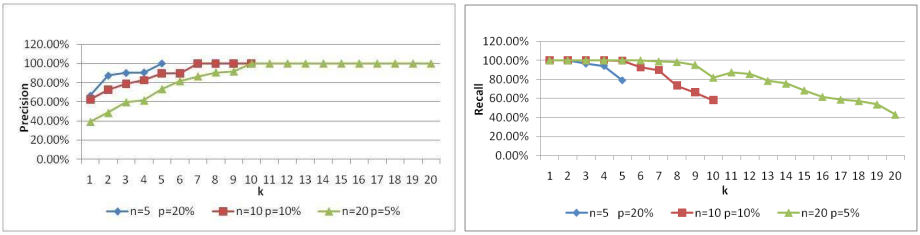
P2 is supported by 39 executions (support=10.42%).

sG-SPADA computation is then distributed on n sample databases which contain p% units of analysis stored in the original database. Different experiments are performed with n=5 and p=20%, n=10 and p=10%, n=20 and p=5%. Local multi-level patterns are discovered at each computation unit and approximate global patterns are identified from the local ones by varying k between 1 and n. To evaluate the quality of the approximate global patterns, we consider two evaluation measures, that is, recall and precision of the approximate global patterns discovered with respect to the exact patterns.

$$recall = \frac{\#(Exp_{n,p,k} \cap Exp_{DB})}{\#Exp_{DB}} \text{ and } precision = \frac{\#(Exp_{n,p,k} \cap Exp_{DB})}{\#Exp_{n,p,k}}.$$

$\#Exp_{n,p,k}$ is the number of approximate global patterns produced by sG-SPADA with n, p and k, while $\#Exp_{DB}$ is the number of exact patterns retrieved on the entire database. Recall and precision results by varying k between 1 and n are reported in Figure 1. As expected, when k=1 recall is equal to 100%, but precision is very low. Conversely, recall tends to be 0 when increasing k. We can observe that the best trade-off in maximizing precision and recall is found when $k \approx \frac{n}{2}$. For example, when n=10 and p=10% sG-SPADA discovers 2,732 relational patterns with k=5. A deeper analysis of these 2,732 patterns reveals that sG-SPADA correctly reconstructs 99.67% of frequent patterns directly discovered on entire database, while 10.24% of these approximate global patterns are not frequent on the entire database.

Finally, for each exact pattern discovered by sG-SPADA (n=10, p=10%), the differences between the approximated support computed by sG-SPADA (merge step) and the exact support computed on the entire database are plot in Figure 2. Box plots are drawn by varying $k \in \{1, 5, 10\}$. We observe that the minimum



(a) (b)

Fig. 1. Percentage of true frequent patterns (a) and false frequent patterns (b) discovered with n=5, p=20%, n=10, p=10% and n=20, p=5% by varying k in [1,20].

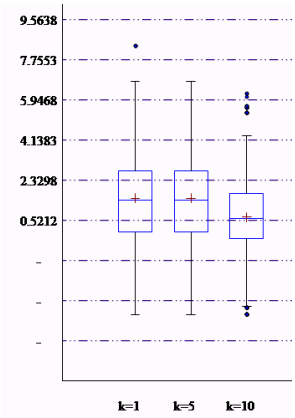


Fig. 2. Box plot of differences between the support approximated by sG-SPADA (merge step) and the real support of each true pattern. sG-SPADA is run with n=10, p=10% and k=1,5,10.

difference of support values is constant (-3.71; k=1,5,7), while the maximum difference of support values decreases from 8.37 (k=1) to 6.8 (k=5) and 6.23 (k=10). The mean of differences decreases from 1.52 (k=10) and 1.5 (k=5) to 0.69 (k=10). This confirms that when k approximates n, although the number of approximate global patterns decreases, approximate patterns are frequent on the entire database and the estimate of their support is close to the exact support value with a low approximation error of 0.69% in average.

4.2 THINK3 Data

THINK3 data traces 353,490 process executions of one customer of THINK3. The period under analysis is from April 7th 2005 to January 10th, 2007 for a total of 1,035,119 activities and 103 promoters. Activities are classified as administrator tools (131), workflow (919052), namemaker (106839), delete (2767),

deleteEnt (2354), prpDelete (471), prpSmartDelete (53), prpModify (34) and cast (1430). Promoters are classified as user (103), viewer (3) or administrator (2). Taxonomic knowledge on activities and promoters is encoded in two distinct hierarchies. For example, "muller" is-a "administrator", "altendorfer" is-a "user", "administrator" is-a "originator" and so on. For each activity a textual description is registered in the event log, while for each promoter a working group is defined. In this experiment, we have thirteen distinct descriptions of the activities and thirty-three distinct groups of promoters. The extensional database D_E includes 4,374,840 ground facts, while the intensional part D_I includes the definition of the predicate *before*. Additional predicates are intensionally defined to group similar activities. For example, the following clauses:

release(X) $\leftarrow$ *description(X,freigabe).*
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 describe the activity of release ("freigabe" in German) independently from the release type (H, J or M). Similarly, other clauses in D_I define the predicates *pruefung*, *techaend*, *cancelled*, *construction*, *ktgprocess*, *musterbau*, *nullserie*, *techniche*, *tiffprocess*, *undermodify* and *workinprogress* which describe an activity.

sG-SPADA is run by randomly extracting n=100 sample databases with p=1%. The discovery of the local multi-level frequent relational patterns is parallelized on 100 nodes. The size of the original database prevents SPADA from processing units of analysis altogether.

We set the following SPADA's parameters: $minsup[1] = 0.25$, $minsup[2] = 0.1$, $minsup[3] = 0.01$ and $max\_len\_path = 14$. With the thresholds defined above, there are no frequent patterns with more than twelve atoms, this means that in this case study, sGSPADA returns the set of all approximate global patterns.

Approximate global patterns are reconstructed from the local ones by varying k from 1 to 100. Their number is reported in Table 1 and, as expected, it decreases when k increases. The average number of local frequent 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 1, 369 local frequent patterns (about 54% on average) are common to all samples.

Table 1. 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**

For this dataset it was not possible to apply sG-SPADA to the entire database as done for ProM data. Therefore, statistics on precision and recall could not be collected. The main goal of this experiment is that of proving the scalability of

the proposed distribute 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 set of global frequent patterns, as in the previous experiments.

5 Conclusions

The inherent complexity of the problems generally solved by MRDM and ILP methods general prevents their applicability to very large databases, such as those generated by typical ubiquitous computing applications. In this paper we have proposed a distributed algorithm for relational frequent pattern discovery. Results of our experiments on two real databases are promising both in terms of approximation of the exact set of frequent relational patterns and in terms of scalability. A crucial point of our proposal is the right choice of the parameter k, which is the minimum number of local sets where a global patterns must be locally frequent. As future work, we intend to investigate some methods to automatically determine the optimal k value, since k can be linked to the minimum support of local patterns (small k values for high *minsup* values, and high k values for low *minsup* values).

Acknowledgments

This work is supported in partial fulfillment of the research objectives of both "FAR" project "Laboratorio di bioinformatica per la biodiversitá molecolare" and "TOCAI.it" project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet". The authors wish to thank THINK3 Inc. for having provided process data used in the experiments.

References

1. H. Blockeel and M. Sebag. Scalability and efficiency in multi-relational data mining. *SIGKDD Explorations*, 5(1):17–30, 2003.
2. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag New York, Inc., New York, NY, USA, 1990.
3. L. Dehaspe and L. De Raedt. Parallel inductive logic programming. In *Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, 1995.
4. L. Dehaspe and L. De Raedt. Mining association rules in multiple relations. In *ILP 1997*, volume 1297, pages 125–132. Springer-Verlag, 1997.
5. S. Džeroski and N. Lavrač. *Relational Data Mining.* Springer-Verlag, 2001.
6. N. A. Fonseca, F. M. A. Silva, and R. Camacho. Strategies to parallelize ILP systems. In S. Kramer and B. Pfahringer, editors, *ILP*, volume 3625 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
7. N. A. Fonseca, F. M. A. Silva, V. S. Costa, and R. Camacho. A pipelined data-parallel algorithm for ILP. In *CLUSTER*, pages 1–10. IEEE, 2005.

8. J. Graham, C. D. Page, and A. Kamal. Accelerating the drug design process through parallel inductive logic programming data mining. In *CSB '03: Proceedings of the IEEE Computer Society Conference on Bioinformatics*, page 400, Washington, DC, USA, 2003. IEEE Computer Society.
9. W. Klosgen and M. May. Spatial subgroup mining integrated in an object-relational spatial database. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 2431 of *LNAI*, pages 275–286. Springer-Verlag, 2002.
10. S. Konstantopoulos. A data-parallel version of aleph. *CoRR*, abs/0708.1527, 2007.
11. F. A. Lisi and D. Malerba. Inducing multi-level association rules from multiple relations. *Machine Learning*, 55(2):175–210, 2004.
12. H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
13. S. Muggleton. *Inductive Logic Programmings*. Academic Press, London, 1992.
14. C. Silvestri and S. Orlando. Distributed approximate mining of frequent patterns. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 529–536, New York, NY, USA, 2005. ACM.
15. H. Toivonen. Sampling large databases for association rules. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *International Conference on Very Large Data Bases*, pages 134–145. Morgan Kaufman, 1996.
16. A. Turi, A. Appice, M. Ceci, and D. Malerba. A grid-based multi-relational approach to process mining. In *DEXA*, 2008.
17. M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *RIDE 1997*, 1997.