# A Grid-Based Multi-relational Approach to Process Mining

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

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

**Abstract.** Industrial, scientific, and commercial applications use information systems to trace the execution of a business process. Relevant events are registered in massive logs and process mining techniques are used to automatically discover knowledge that reveals the execution and organization of the process instances (cases). In this paper, we investigate the use of a multi-level relational frequent pattern discovery method as a means of process mining. In order to process such massive logs we resort to a Grid-based implementation of the knowledge discovery algorithm that distributes the computation on several nodes of a Grid platform. Experiments are performed on real event logs.

## 1 Introduction

Many information systems, such as Workflow Management Systems, ERP systems, Business-to-business systems and Firewall systems trace behavior of running processes by registering relevant events in massive logs. Events are described in a structured form that includes properties of cases and activities. A case represents the process instance which is being handled, while an activity represents the operation on the case. Information on timestamp and on the person executing the event (performer) is available in the logs. Both activities and performers may belong to different categories. Event logs are stored in multi-terabyte warehouses and sophisticated data mining techniques are required to process this huge amount of data and extract knowledge concerning the execution and organization of the recorded processes. This huge amount of data is the main concern of research in process mining whose aim is to discover a description or prediction of real process, control, organizational, and social structures [10].

Process mining poses several challenges to the traditional data mining tasks. In fact, data stored in event logs describe objects of different type (cases, activities and performers) which are naturally modeled as several relational data tables, one for each object type. Foreign key constraints express the relations between these objects. This (relational) data representation makes necessary distinguishing between the reference objects of analysis (cases) and other task-relevant objects (activities and performers), and to represent their interactions. Another challenge is represented by the temporal autocorrelation. Events are temporally related according to a timestamp. This means that the effect of a

S.S. Bhowmick, J. Küng, and R. Wagner (Eds.): DEXA 2008, LNCS 5181, pp. 701–709, 2008.

property at any event may not be limited to the specific event. Furthermore, activities and performers are generally organized in hierarchies of categories (e.g. the performer of operations on a text file can be a writer or a reader). By descending or ascending through a hierarchy, it is possible to view the same object at different levels of abstraction (or granularity). Finally, reasoning is the process by which information about objects and their relations (e.g. operator of indirect successor) are used to arrive at valid conclusions regarding the object relations [7]. This source of knowledge cannot be ignored in the search.

Currently, many algorithms [2,1,11,3] have dealt with several of these challenges and some of them are integrated into the ProM framework [12]. Anyway, to the best of our knowledge, methods of process mining neither support a multi-level analysis nor use inferential mechanisms defined within a reasoning theory. Conversely, the multi-relational data mining method SPADA [5] offers a sufficiently complete solution to all the challenges posed by the process mining tasks in descriptive case. However, SPADA is not applicable in practice. Indeed, frequent pattern discovery is a very complex task, particularly in the multi-relational case [5]. In addition SPADA, similarly to most of the multi-relational data mining algorithms, operates with data in main memory, hence it is not appropriate for processing massive logs. Advantages of the multi-relational approach in facing the challenges of the process mining justify the attempt of resorting to the computational power of distributed high-performance environments (e.g., computational Grids [4]) to mitigate the complexity of the relational frequent pattern discovery on massive event logs.

In this paper, we present G-SPADA, an extension of SPADA, which discovers approximate multi-level relational frequent patterns by distributing exact computation of locally frequent multi-level relational patterns on a computational Grid and then by post-processing local patterns in order to approximate the set of the globally frequent patterns as well as their supports. Distributing relational frequent pattern discovery on a Grid poses several issues. Firstly, relational data must be divided in data subsets and each subset has to be distributed on the Grid. Split must take into account relational structure of data, that is, each data split must includes a subset of reference objects and the task-relevant objects to reconstruct all interactions between them. Secondly, it is necessary a framework for building the Grid applications utilizing the power of distributed computation and storage resources across the Internet. Finally, processing local patterns to approximate global ones requires a way of combining distinct sets of patterns into a single one and obtaining an estimate of the global support.

## 2 Multi-level Relational Frequent Pattern Discovery

The multi-level relational pattern discovery task is formally defined as follows: *Given*: a set $S$ of reference objects, some sets $R_k$, $1 \leq k \leq m$ of task-relevant objects, a background knowledge $BK$ which includes hierarchies $H_k$ on the objects in $R_k$ and domain knowledge in form of rules, a deductive database $D$ that is formed by an extensional ($D_E$) part where properties and relations of reference

objects and task-relevant objects are expressed in derived ground predicates and an intensional part ($D_I$) where domain knowledge in $BK$ is expressed in form of rules, $M$ granularity levels in the descriptions (1 for the highest), a set of granularity $\psi_k$ which associate each object in $H_k$ with a granularity level to deal with several hierarchies at once, a threshold $minsup[l]$ for each granularity level $l$ ($1 \leq l \leq M$), *Find*, for each granularity level $l$, the *frequent*[1] relational patterns which involve properties and relations of task relevant-objects at level $l$ of $H_k$ .

The relational formalization of the task of frequent pattern discovery is based on the idea that each unit of analysis (or example) $D[s]$ includes a reference object $s \in S$ and all the task-relevant objects of $R_k$ which are (directly or indirectly) related to $s$ according to some foreign key path in $D$. The frequency (support) of a pattern is based on the number of units of analysis, i.e., reference objects, covered by the pattern. An example of relational pattern is:

*Example 1.* Let $D_E$ be the extensional database described in Example 1. A possible relational pattern P1 on $D$ is in the form:

P1: *case(A), activity(A,B), is_a(B,activity), before(B,C), is_a(C,activity), description(C,workinprogress), user(B, D), is_a(D, performer)* [*72.25%*]

P1 expresses the fact that a process $A$ is formed by two sequential activities, namely $B$ and $C$, the performer of $B$ is generic. The support is 72.5%.

By taking into account hierarchies on task-relevant objects, relational patterns can be discovered at multiple level of granularity.

*Example 2.* Let us consider two level hierarchies defined on performers and activities defined in the followings:

*administrator, user* $\rightarrow$ *performer*; *namemaker, delete, workflow* $\rightarrow$ *activity*

P2 is a finer-grained relational pattern than P1 obtained by descending one level in hierarchies. P2 is in the form:

P2: *case(A), activity(A,B), is_a(B,namemaker), before(B,C), is_a(C,workflow), description(C,workinprogress), is_a(D, administrator)* [*62.5%*]

P2 provides better insight than P1 on the nature of $B$, $C$ and $D$.

In SPADA [5], multi-level relational frequent patterns are discovered according to the *levelwise* method [6] that is based on a breadth-first search in the lattice of patterns spanned by $\theta$-subsumption [8] generality order ($\succeq_\theta$).

## 3 G-SPADA

Similarly to Partition [9], G-SPADA splits a dataset into several partitions to be processed independently. It approximates the multi-level relational frequent pattern discovery by means of a three stepped strategy. In the first step, the set of original $N$ reference objects is partitioned into $n$ approximately equally-sized subsets ($n << N$). Each partition includes a subset of the reference objects and the set of task-relevant objects. In the second step, the frequent pattern

[1] With support greater than $minsup[l]$.

computation is parallelized and distributed on $n$ nodes of a Grid platform, one node for each partition. In this way, G-SPADA generates $n$ parallel executions of SPADA at the same time and retrieves local patterns which are frequent in at least one of the data partition. In the third step, G-SPADA approximates the set of globally frequent patterns by merging patterns discovered at the nodes.

The basic idea in approximating the global patterns is that each globally frequent pattern must be locally frequent in at least $k$ partitions of the original dataset. In the case $k$ is set to 1, this guarantees that the union of all local solutions is a superset of the global solution. However, a merge step with $k = 1$ may generate several false positives, i.e. patterns that result locally frequent but globally infrequent. Hence, value of $k$ should be adequately tuned between 1 and $n$ in order to find the best trade-off between false positive and false negative frequent patterns. The merge step also attempts to approximate values of support for the global patterns starting from the local values of support.

### 3.1 Relational Data Partitioning

G-SPADA pre-processes the deductive database of logs and completes the description explicitly provided for each example ($D_E$) with the information that is implicit in the domain knowledge ($D_I$). An example of this saturation step is:

*Example 3.* Let us consider the deductive database:
*case(c1). case(c2). activity(c1,a1). activity(c1,a2). activity(c1,a3). activity(c2,a4). time(a1,10). time(a2,25). time(a3,29). time(a4,13). description(a1,create). ...*
*before(A,B):-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).*
By performing the saturation step, the following predicates are made explicit in the database: *before(a1,a2). before(a2,a3).*

Saturation precedes data partitioning. In this way, redundant inferences are prevented for properties and relations of task-relevant objects shared from two or more reference objects belonging to different data partitions.

Data partitioning is performed by randomly splitting the set of reference objects in $n$ approximately equal-sized partitions such that the union of the partitions is the entire set of reference objects. These data partitions are enriched by adding the ground predicates which describe properties and relations of the reference objects falling in the partition at hand. Subsequently, properties and relations of task-relevant objects related to reference objects according to some foreign key path are also added to the partition.

### 3.2 Distributing Computation on Grid

Each dataset partition is shipped along with the G-SPADA pattern discovery algorithm to computation nodes on Grid using gLite[2] middleware. This is done

[2] gLite (http://glite.web.cern.ch/glite/) is a next generation middleware for Grid computing which provides a framework for building Grid applications.

by submitting parametric jobs described in JDL (Job Description Language) through the CLI (command line interface). Submission of jobs on Grid are divided in several steps: (*i*) Authenticate on a UI (user interface) through PKI based authentication system with proxy credentials (GSI); (*ii*) Prepare the jobs (JDL, shell script to automate procedure, input file); (*iii*) Upload (Stage-in) a set of dataset; (*iv*) Submit a relative parametric job; (*v*) Check/wait results; (*vi*) Finally, once the job is executed on Grid, we get the output (Stage-out) files containing the frequent pattern sets along with their support for each sample.

### 3.3 Computing Approximate Global Frequent Patterns

The $n$ sets of local frequent patterns are collected from the computation nodes of the Grid platform and then merged to approximate the set of global patterns.

For each local pattern discovered in at least $k$ data partitions ($1 \leq k \leq n$), G-SPADA derives an approximate of the global support by averaging the support values collected on the partitions where the pattern is found to be frequent. The check that the same local pattern occurs in different partitions is based on an equivalence test between two patterns under $\theta$-subsumption, which corresponds to performing a double $\theta$-subsumption test ($P \succeq_\theta Q$ and $Q \succeq_\theta P$). Local patterns occurring in less than $k$ partitions are filtered out. The global frequent patterns obtained following this merge procedure approximate the original frequent patterns which can be possibly mined on the entire dataset.

An example of approximate global process pattern is:

*case(A), activity(A,B), is_a(B,namemaker), before(B,C), is_a(C,workflow), description(C,workinprogress)* [*7, 72.5%*]

which describes the order of execution between two activities, namely $B$ and $C$, in the process $A$. $B$ is a name-maker activity while $C$ is a workflow activity. In addition, $C$ is described as work in progress. 7 means that this pattern is found in 7 partitions (sample-level support), while 72.5% indicates the macro average support obtained by averaging the support values computed on the 7 samples.

## 4 Experimental Results

Experiments are performed by processing event logs provided by THINK3 Inc[3] in the context of the TOCAI.It project[4]. THINK3 is a global player in Cad and Plm market whose mission is to help manufacturers optimizing their entire product development processes. G-SPADA is run on the deductive database that is obtained by boiling down the event logs from January 1st to February 28th, 2006 and considering as domain knowledge the definition of the "before" predicate. In the experiments, each case (process instance) traced in the logs is considered as a whole and multi-level relational patterns are discovered from traced business processes. These patterns capture the possible relation between the order of activities and the properties of their performers.

[3] http://www.think3.com/en/default.aspx

[4] http://www.dis.uniroma1.it/∼tocai/index.php

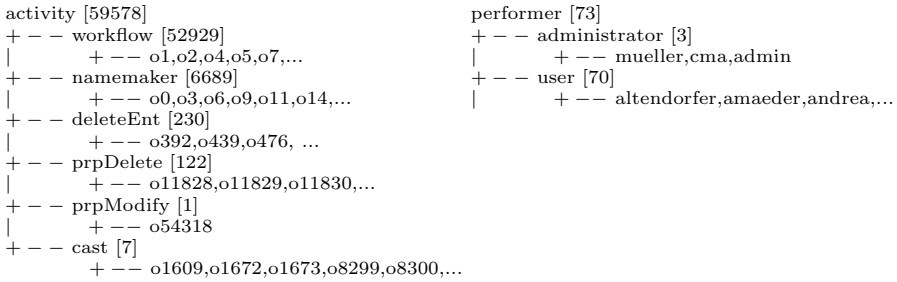


**Fig. 1.** Three-level hierarchies on activity and performer

### 4.1 Data Description

Data trace the behavior of 21,256 instances of a business process recorded in the period under analysis. A case is traced by registering its events. Each event describes the activity executed within a case and the activity performer. The activities correspond to six tasks (or classes of operations), that is, workflow, namemaker, deleteEnt, prpDelete, prpModify and cast while the performers are the category of person (or system) executing the activity, that is, user or administrator. This corresponds to model activities and performers by means of three-level hierarchies (see Figure 1). Each hierarchy is mapped into the three granularity levels thus allowing to deal uniformly with both hierarchies at once. By descending or ascending through the hierarchy, it is possible to view the same activity or performer at different levels of granularity.

For each activity, a text description of the operation is registered in the event logs. This text includes a left part and a right one ("left::right"). The right part is a characterization of the description of the operation provided in the left part. Some examples of descriptions registered in the event logs are: *create::workinprogress*, *t2f::freigabe*, *wip2f::freigabe*.

Thirty-six distinct descriptions are registered in the logs, but several of them share the same left or right part: *k2f::freigabe, m2f::freigabe* , *document::doccad, document::doccad3d*. By interpreting this structure, the activity descriptions is practically boiled down into two predicates, that is:

*leftDescription(activity, text). rightDescription(activity, text).*

Finally, each performer is described by the belonging group. Twenty two distinctive groups are registered in the event logs. Performers labeled as users and administrators can possibly belong to the same group.

### 4.2 Local and Global Multi-level Relational Patterns Discovery

G-SPADA is run on the event logs including 395,404 ground predicates. Reference objects are the cases, while task-relevant objects are the activities and performers. In this way, the description of an activity and of the group of its performer is not limited to the specific event.

**Table 1.** Number of global frequent patterns discovered by varying $k$ in [1,20]

| k | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #P | 428 | 424 | 412 | 412 | 400 | 372 | 372 | 364 | 356 | 356 | 344 | 344 | 314 | 312 | 312 | 311 | 263 | 259 | 259 | 235 |

Data of the event logs are split into twenty partitions. The discovery of the local multi-level frequent relational patterns is then distributed on twenty nodes. The Grid infrastructure is required to process the huge amount of data registered in the logs. Indeed, SPADA generates a memory exception when running on the entire dataset. Multi-level relational patterns are discovered at each node with $minsup[l] = 0.2$ $(l = 1, 2)$ and $max\_len\_path = 9$ [5]. Finally, for each level of granularity, global patterns are approximated from the local ones by varying $k$ between 1 and 20. The number of discovered global patters are reported in Table 1. Obviously, the number of global patterns decreases by increasing $k$.

Global patterns provide a compact description of the instances of process traced in the logs. They provide a multi-level insight of the order of execution and/or organization of the processes traced in the logs.

At level 1, G-SPADA discovers the global relational pattern P1:

**P1:** *case(A),activity(A,B),before(B,C),user(B,D),is_a(B,activity),is_a(C,activity), is_a(D,performer),descright(C,release).* [*k=20, avgSup=63.55%*]

P1 captures the execution order between two activities ($B$ and $C$) within a case ($A$). One activity ($B$) is performed by a generic performer $D$, while the other activity ($C$) is described as a *release* activity. By descending one level of the hierarchies, G-SPADA discovers the finer grained global relational pattern P2:

**P2:** *case(A),activity(A,B),before(B,C),user(B,D),is_a(B,workflow), is_a(C,workflow),is_a(D,user),descright(C,release).*[*k=20,avgSup=51.43%*]

P2 clarifies that the performer $C$ is a user, while $B$ and $D$ are workflow activities. Support of P2 is reconstructed from the support of P2 on the local partitions, that is, P2 covers at least 10935 cases registered in the original event logs.

The pattern P3:

**P3:** *case(A),activity(A,B),before(B,C),user(B,D),is_a(C,workflow),is_a(D,user), descright(B,construction),descright(C,release).* [*k=4, avgSup=29.06%*]

is a specialization of P2 under $\theta$-substitution which describes $B$ as a *construction* activity. Obviously, the support of P3 decreases with respect to the support of P2, due to the $\theta-$substitution antimonotonicity of support.

Finally, the relational pattern:

**P4:** *case(A), activity(A,B),before(B,C),before(C,D), is_a(B,namemaker), is_a(C,workflow), is_a(D,workflow), descleft(C,creation), descleft(D,wip2k)* [*k=16, avgSup=21.35%*]

describes the execution order among three sequential activities, namely $B$, $C$ and $D$. $B$ is a namemaker activity, while $C$ and $D$ are workflow activities. $C$ is described as a *creation* activity, while $D$ is described as *wip2k* operation.

[5] *max_len_path* is the maximum number of predicates to be included in a pattern.

## 5 Conclusions

In this paper, we present G-SPADA, an extension of the system SPADA, to discover approximate multi-lever relational frequent patterns in the context of process mining. G-SPADA exploits a multi-relational approach in order to deal with both multiple nature of data stored in event logs and temporal autocorrelation. G-SPADA faces the need of processing massive logs by resorting to a grid based architecture. Experiments on the real event logs allow us to discover interpretable patterns which capture regularities in the execution of activities and the characteristics of the performers of a business process. Such patterns can be used to deploy new systems supporting the execution of business processes or analyzing and improving already enacted business processes.

## Acknowledgments

This work is in partial fulfillment of the research objectives of "TOCAI.it" project "Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet". The authors wish to thank THINK3 Inc. for providing data.

## References

1. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
2. Cook, J.E., Wolf, A.L.: Software process validation: Quantitatively measuring the correspondence of a process to a model. ACM Trans. Softw. Eng. Methodol. 8(2), 147–176 (1999)
3. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. 18(8), 1010–1027 (2006)
4. Li, T., Bollinger, T.: Distributed and parallel data mining on the grid. In: ARCS Workshops. LNI, vol. 41, pp. 370–379. GI (2004)
5. Lisi, F.A., Malerba, D.: Inducing multi-level association rules from multiple relations. Machine Learning 55(2), 175–210 (2004)
6. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Mining and Knowledge Discovery 1(3), 241–258 (1997)
7. Michalski, R.S.: A theory and methodology of inductive learning, pp. 323–348 (1993)
8. Plotkin, G.D.: A note on inductive generalization 5, 153–163 (1970)
9. Savasere, A., Omiecinski, E., Navathe, S.B.: An efficient algorithm for mining association rules in large databases. In: VLDB, pp. 432–444 (1995)
10. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., de Medeiros, A.K.A., Song, M., Verbeek, H.M.W.: Business process mining: An industrial application. Inf. Syst. 32(5), 713–732 (2007)

11. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data Knowl. Eng. 47(2), 237–267 (2003)
12. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)