Scalable Mining of Frequent Tri-concepts from *Folksonomies*

Chiraz Trabelsi[1], Nader Jelassi[1], and Sadok Ben Yahia[1,2]

[1] Faculty of Sciences of Tunis, University Tunis El-Manar, 2092 Tunis, Tunisia
[2] Institut TELECOM, TELECOM SudParis, UMR 5157 CNRS Samovar, 91011 Evry Cedex, France
{chiraz.trabelsi,sadok.benyahia}@fst.rnu.tn

Abstract. Mining frequent tri-concepts from *folksonomies* is an interesting problem with broad applications. Most of the previous tri-concepts mining based algorithms avoided a straightforward handling of the triadic contexts and paid attention to an unfruitful projection of the induced search space into dyadic contexts. As a such projection is very computationally expensive since several tri-concepts are computed redundantly, scalable mining of *folksonomies* remains a challenging problem. In this paper, we introduce a new algorithm, called TRICONS, that directly tackles the triadic form of *folksonomies* towards a scalable extraction of tri-concepts. The main thrust of the introduced algorithm stands in the application of an appropriate closure operator that splits the search space into equivalence classes for the the localization of tri-minimal generators. These tri-minimal generators make the computation of the tri-concepts less arduous than do the pioneering approches of the literature. The experimental results show that the TRICONS enables the scalable frequent tri-concepts mining over two real-life *folksonomies*.

Keywords: Folksonomies, Triadic Concept Analysis, Closure Operator, Equivalence Classes, Triadic Concepts.

1 Introduction and Motivations

Complementing the Semantic Web effort, a new breed of so-called Web 2.0 applications recently emerged on the Web. Indeed, social bookmarking systems, such as *e.g.,* DEL.ICIO.US[1], BIBSONOMY [2] or FLICKR[3] have become the predominant form of content categorization of the Web 2.0 age. The main thrust of these Web 2.0 systems is their easy use that relies on simple, straightforward structures by allowing their users to label diverse resources with freely chosen keywords *aka* tags. The resulting structures are called *folksonomies*[4], that is, "taxonomies" created by the "folks". Considered as a tripartite hyper-graph [9] of tags, users and resources, the new data of *folksonomy* systems

[1] http://www.delicious.com
[2] http://www.bibsonomy.org
[3] http://www.flickr.com
[4] http://wwww.vanderwal.net/folksonomy.html

P.-N. Tan et al. (Eds.): PAKDD 2012, Part II, LNAI 7302, pp. 231–242, 2012.

provides a rich resource for data analysis, information retrieval, and knowledge discovery applications. Recently, the discovery of shared conceptualizations opens a new research field which may prove interesting also outside the *folksonomy* domain: closed tri-sets (triadic concepts) mining in triadic data [6]. Actually, this line of Triadic Concept Analysis did not grasp a broad attention. However, with the rise of *folksonomies*, formally represented as triadic contexts, many researches advocate the extraction of lossless concise representations of interesting patterns from triadic data.

In this paper, we are mainly interested in the mining of frequent triadic concepts (tri-concepts for short) from 3-dimensional data, *i.e., folksonomy*. These patterns are among the recent research topics in Triadic Concept Analysis. In this respect, a determined algorithmic effort was furnished to get out this type of patterns. Worth of mention, the pioneering work of Stumme *et al.,* through the TRIAS algorithm [6], for tri-concepts mining. TRIAS inputs a *folksonomy*, formally represented as a triadic context, and computes all tri-concepts. However, the main moan that can be addressed to TRIAS, stands in its need to transform the triadic context into dyadic contexts in order to extract tri-concepts. Thus, the mining task becomes very computationally expensive and could be avoided by extending the basic notions of *FCA* (Formal Concept Analysis) for the triadic case. Ji et *al.,* in [7], have proposed un alternative algorithm, called CUBEMINER, which directly operates on the triadic context. It consists in using cubes called *cutters* generalizing the cutters introduced for constraint-based mining of formal concepts [1]. Yet, in a *folksonomy*, the number of cutters may be very large as far as the cardinality of at least one dimension of a *folksonomy* is high. Besides, the CUBEMINER algorithm operates in a depth-first manner, which has the risk of causing infinite trees. More recently, Cerf *et al.*, in [2], proposed the DATA-PEELER algorithm with the challenge of beating both later algorithms in terms of performance. The DATA-PEELER algorithm is able to extract all closed concepts from n-ary relations. DATA-PEELER enumerates all the n-dimensional closed patterns in a depth first manner using a binary tree enumeration strategy. However, similarly to CUBEMINER, the strategy of DATA-PEELER, involving a depth-first approach implies its depth's recursion, in the worst case, to the total number of elements (whatever the dimension). Moreover, DATA-PEELER is hampered by the large number of elements that may contain any of the *folksonomy's* dimensions and its strategy becomes ineffective and leads to a complex computation of tri-concepts.

In this respect, a compelling and thriving issue is to introduce a new scalable algorithm, that overcomes the flaws of the previous ones. Hence, in this work, the main contribution is to introduce a new algorithm for tri-concepts mining, called TRICONS, aiming at providing better scalabilty than do the pioneering approches of the literature, by applying an appropriate closure operator. In fact, the closure operator splits the search space into equivalence classes in order to find the tri-minimal generators. These tri-minimal generators, representative of the different equivalence classes, make the computation of the tri-concepts less arduous than do the aforementioned ones. Indeed, the tri-minimal generators are the smallest elements, *i.e.,* tri-sets, in an equivalence class, while their associated closure is the largest one within the corresponding equivalence class. Thus, the pairs - composed by Tri-MGs and their related closures - allow, *(i)* an easier localization (extraction) of each tri-concept since it is necessarily

encompassed by an Tri-MG and the related closures and; *(ii)* to straightforwardly handle the triadic form of a *folksonomy* towards an efficient extraction of tri-concepts.

The remainder of the paper is organized as follows. Section 2 recalls the key notions used throughout this paper. We scrutinize the related work of mining triadic concepts in section 3. In section 4, we introduce a new closure operator to the triadic context as well as the TRICONS algorithm dedicated to the extraction of frequent tri-concepts. The empirical evidences about the performance of our approach are provided in Section 5. Finally, we conclude the paper with a summary and we sketch ongoing research in section 6.

2 Key Notions

In this section, we briefly sketch the key notions that will be of use in the remainder of this paper. In the following, we start by presenting a formal definition of a *folksonomy* [6].

Definition 1. (FOLKSONOMY) *A folksonomy is a set of tuples* $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$, *where* $\mathcal{Y} \subseteq \mathcal{U} \times \mathcal{T} \times \mathcal{R}$ *is a triadic relation such as each* $y \subseteq \mathcal{Y}$ *can be represented by a triple:* $y = \{(u, t, r) \mid u \in \mathcal{U}, t \in \mathcal{T}, r \in \mathcal{R}\}$, *denoting that the user* u *annotated the resource* r *using the tag* t.

Example 1. An exemple of a *folksonomy* $\mathcal{F}$ is depicted by Table 1 with $\mathcal{U} = \{u_1, u_2, \ldots, u_7\}$, $\mathcal{T} = \{t_1, t_2, \ldots, t_5\}$ and $\mathcal{R} = \{r_1, r_2, r_3\}$. Each cross within the ternary relation indicates a tagging operation by a user from $\mathcal{U}$, a tag from $\mathcal{T}$ and a resource from $\mathcal{R}$, *i.e.*, a user has tagged a particular resource with a particular tag. For example, the user u_1 has assigned the tags t_2, t_3 and t_4, respectively, to the resources r_1, r_2 and r_3.

Table 1. A toy example of a *folksonomy* that would be of use throughout the paper

$\mathcal{U}/\mathcal{R}$-$\mathcal{T}$	r_1					r_2					r_3				
	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5	t_1	t_2	t_3	t_4	t_5
u_1		×	×	×			×	×	×			×	×	×	
u_2		×	×	×		×	×	×	×		×	×	×	×	
u_3		×	×	×		×	×	×	×		×	×	×	×	
u_4						×			×		×			×	
u_5		×	×	×	×		×	×	×	×		×	×	×	
u_6				×	×				×	×					
u_7	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×

The following definition presents the frequent tri-set [6].

Definition 2. (A (FREQUENT) TRI-SET) *Let* $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$ *be a folksonomy. A tri-set of* $\mathcal{F}$ *is a triple* (A, B, C) *with* $A \subseteq \mathcal{U}$, $B \subseteq \mathcal{T}$, $C \subseteq \mathcal{R}$ *such that* $A \times B \times C \subseteq \mathcal{Y}$. *A tri-set* (A, B, C) *of* $\mathcal{F}$ *is said* frequent *whenever* $|A| \geq minsupp_u$, $|B| \geq minsupp_t$ *and* $|C| \geq minsupp_r$, *where* $minsupp_u$, $minsupp_t$ *and* $minsupp_r$ *are user-defined thresholds.*

As the set of all frequent tri-sets is highly redundant, we will in particular consider a specific condensed representation, *i.e.,* a subset which contains the same information, namely the set of all frequent tri-concepts. The latter's definition is given in the following [6,8].

Definition 3. ((FREQUENT) TRIADIC CONCEPT) *A triadic concept (or a tri-concept for short) of a* folksonomy $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$ *is a triple* (U, T, R) *with* $U \subseteq \mathcal{U}$, $T \subseteq \mathcal{T}$, *and* $R \subseteq \mathcal{R}$ *with* $U \times T \times R \subseteq \mathcal{Y}$ *such that the triple* (U, T, R) *is maximal, i.e., for* $U_1 \subseteq U$, $T_1 \subseteq T$ *and* $R_1 \subseteq R$ *with* $U_1 \times T_1 \times R_1 \subseteq \mathcal{Y}$, *the containments* $U \subseteq U_1$, $T \subseteq T_1$, *and* $R \subseteq R_1$ *always imply* $(U, T, R) = (U_1, T_1, R_1)$. *A tri-concept is said to be frequent whenever it is a frequent tri-set. The set of all frequent tri-concepts of* $\mathcal{F}$ *is equal to* $\mathcal{TC} = \{TC \mid TC = (U, T, R) \in \mathcal{Y}$ *is a tri-concept*$\}$.

Given a tri-concept $\mathcal{TC} = (U, T, R)$, the U, R and T parts are respectively called ***Extent***, ***Intent***, and **Modus**.

Example 2. Consider the *folksonomy* depicted by table 1. We can denote that the tri-set S_1=$\{\{u_5, u_7\}, \{t_2, t_3, t_4\}, \{r_1, r_2\}\}$ is not a tri-concept of $\mathcal{F}$. Whereas, TC_1=$\{\{u_5, u_7\}, \{t_2, t_3, t_4\}, \{r_1, r_2, r_3\}\}$ is a tri-concept of $\mathcal{F}$: it includes all maximal tags and resources shared by the users u_5 and u_7.

3 Related Work

With the rise of *folksonomies*, formally represented as triadic contexts, many researches advocate the extraction of implicit shared conceptualizations formally sketched by tri-concepts. Indeed, Jäschke et *al.*, in [6], introduced the TRIAS algorithm to compute frequent tri-concepts from a *folksonomy*. Hence, tackling a *folksonomy* $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$, TRIAS first constructs a dyadic context $\mathcal{K}_1 = (\mathcal{U}, \mathcal{T} \times \mathcal{R}, \mathcal{Y}_1)$ whose columns correspond to couples of elements from $\mathcal{T}$ and $\mathcal{R}$ and then, via a projection, according to the $\mathcal{T}$ and $\mathcal{R}$ axis, extracts formal concepts. The second step of TRIAS consists, for each formal concept, in checking whether it is closed *w.r.t.* $\mathcal{U}$. Actually, the main feature of TRIAS is to exploit the subsets of tri-concepts already extracted in order to check whether they lead to new tri-concepts. However, several tri-concepts are computed redundantly inducing a number of unnecessary computations. This drawback occurs because of the particular order of extraction of tri-concepts which is strongly inspired by the way of doing of the NEXTCLOSURE algorithm [4], dedicated to building of a lattice of frequent closed itemsets. Nevertheless, Ji et *al.,* in [7], have introduced an alternative algorithm called CUBEMINER, which directly operates on the triadic context. It consists in using cubes called *cutters* generalizing the cutters introduced for constraint-based mining of formal concepts in [1]. These cutters are recursively processed to generate candidates at each level, thus, the number of levels of the execution equals that of cutters. For each cutter applied to a tri-set, three candidates are constructed accordingly to the three axis of the *folksonomy* as long as the tri-set contains all elements of the current cutter. When no more cutter is applicable on a tri-set, it becomes a tri-concept. Yet, in a *folksonomy*, the number of cutters may be very large as far as the cardinality of at least one set of $\mathcal{F}$ is high. Besides, the CUBEMINER algorithm operates in a depth-first manner, which has the risk of causing infinite trees. Moreover, at each level, several checks

are performed on each candidate to ensure its closeness and its uniqueness which is very computationally expensive. Indeed, each candidate must be compared twice to the elements of the cutters. More recently, Cerf *et al.*, in [2], proposed the DATA-PEELER algorithm with the challenge of outperforming both TRIAS and CUBEMINER algorithms in terms of performance. The DATA-PEELER algorithm is able to extract closed concepts from n-ary relations by enumerating all the n-dimensional closed patterns in a depth first manner using a binary tree enumeration strategy. At each level, the current node of the tree is split into two nodes after selecting the element to be enumerated. In addition, the DATA-PEELER algorithm does not store the previously computed patterns in main memory for duplicate detection and closure checking. However, similarly to CUBEMINER, the strategy of DATA-PEELER, involving a depth-first approach, may cause infinite trees. Aiming at palliating these hindrances in effectively extracting tri-concepts, we introduce the TRICONS algorithm dedicated to an efficient extraction of frequent triadic concepts from a *folksonomy*. Following the minimum description length principle, the set of frequent tri-concepts represents a concise representation of frequent tri-sets, by providing the shortest description of the whole set of these frequent patterns. The main thrust of the TRICONS algorithm stands in the localisation of the smallest elements, *i.e.,* tri-sets, called *tri-Minimal generators* (Tri-MGs), in an equivalence class. Indeed, these Tri-MGs are the first reachable elements of their respective equivalence classes, thanks to a breadth-first sweeping of the associated search space. Doing so, makes the computation of the tri-concepts less arduous than do the aforementioned ones.

4 The TRICONS Algorithm

In this section, we firstly, introduce a new closure operator for a triadic context as well as an extension of the notion of minimal generator. Thereafter, we describe the TRICONS algorithm.

4.1 Main Notions of the TRICONS Algorithm

Lehmann and Wille have introduced in [8] two closure operators for the construction of triadic concepts. However, these operators are only of use on dyadic contexts, *i.e.*, the *folksonomy* should be split into three dyadic contexts. Hence, we introduce, in what follows, a new closure operator for a triadic context.

Definition 4. *Let $S = (A, B, C)$ be a tri-set of $\mathcal{F}$. A mapping h is defined as follows :*

$$h(S) = h(A, B, C) = (U, T, R) \mid U = \{u_i \in \mathcal{U} \mid (u_i, t_i, r_i) \in \mathcal{Y} \forall\, t_i \in B, \forall\, r_i \in C\}$$
$$\wedge\, T = \{t_i \in \mathcal{T} \mid (u_i, t_i, r_i) \in \mathcal{Y} \forall\, u_i \in U, \forall\, r_i \in C\}$$
$$\wedge\, R = \{r_i \in \mathcal{R} \mid (u_i, t_i, r_i) \in \mathcal{Y} \forall\, u_i \in U, \forall\, t_i \in T\}$$

Roughly speaking, $h(S)$ computes the largest tri-set in the *folksonomy* which contains maximal sets of tags and resources shared by a group of users containing A. For example, considering the *folksonomy* $\mathcal{F}$ depicted by Table 1, we have $h\{u_1, \{t_2, t_3, t_4\}, r_1\} = \{\{u_1, u_2, u_3, u_5, u_7\}, \{t_2, t_3, t_4\}, \{r_1, r_2, r_3\}\}$.

Proposition 1. *h is a closure operator.*

Proof. To prove that *h* is a closure operator, we have to prove that this closure operator fulfills the three properties of **extensivity**, **idempotency** and **isotony** [3].

(1) ***Extensivity***
Let $T = (A, B, C)$ be a tri-set of $\mathcal{F} \Rightarrow h(T) = (U, T, R)$ such that :
$U = \{u_i \in \mathcal{U} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ t_i \in B, \forall\ r_i \in C\} \supseteq A$ since we have $(u_i, t_i, r_i) \in \mathcal{Y}$ $\forall\ u_i \in A, \forall\ t_i \in B, \forall\ r_i \in C$,

$T = \{t_i \in \mathcal{T} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ r_i \in C\} \supseteq B$ since $U \supseteq A$
and $R = \{r_i \in \mathcal{R} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ t_i \in T\} \supseteq C$ since $U \supseteq A$ and $T \supseteq B$.
Then, $(A, B, C) \subseteq (U, T, R) \Rightarrow T \subseteq h(T)$

(2) ***Idempotency***
Let $T = (A, B, C)$ be a tri-set of $\mathcal{F} \Rightarrow h(T) = (U, T, R) \Rightarrow h(U, T, R) = (U', T', R')$ such that :

$U' = \{u_i \in \mathcal{U} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ t_i \in T, \forall\ r_i \in R\} = U$,
$T' = \{t_i \in \mathcal{T} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ r_i \in C\} = T$,
and $R' = \{r_i \in \mathcal{R} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ t_i \in T'\} = R$.
Then, $(U', T', R') = (U, T, R) \Rightarrow h(h(T)) = h(T)$

(3) ***Isotony***
Let $T = (A, B, C)$ and $T' = (A', B', C')$ be tri-sets of $\mathcal{F}$ with $T \subseteq T' \Rightarrow h(T) = (U, T, R)$ and $h(T') = (U', T', R')$ such that :

On the one hand, $U' = \{u_i \in \mathcal{U} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ t_i \in B', \forall\ r_i \in C'\}$.
and $U = \{u_i \in \mathcal{U} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ t_i \in B, \forall\ r_i \in C\}$.
$\Rightarrow U' \supseteq U$ since $B \subseteq B'$ and $C \subseteq C'$ [8].
On the other hand, $T = \{t_i \in \mathcal{T} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ r_i \in C\}$, $R = \{r_i \in \mathcal{R} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ t_i \in T\}$, $T' = \{t_i \in \mathcal{T} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ r_i \in C\}$ and $R' = \{r_i \in \mathcal{R} \mid (u_i, t_i, r_i) \in \mathcal{Y}\ \forall\ u_i \in U, \forall\ t_i \in T'\}$
$\Rightarrow T \subseteq T'$ since $U \subseteq U'$ and $R \subseteq R'$ since $U \subseteq U'$ and $T \subseteq T'$ [8].
Then, $(U, T, R) \subseteq (U', T', R') \Rightarrow h(T) \subseteq h(T')$

According to (1), (2) and (3), *h* is a closure operator.

Like the dyadic case [10], the closure operator induces an equivalence relation on the power set of elements, *i.e.,* tri-sets in the *folksonomy*, portioning it into disjoint subsets called *equivalence classes* that we introduce in the following :

Definition 5. (EQUIVALENCE CLASS) *Let $S_1 = (A_1, B_1, C_1)$, $S_2 = (A_2, B_2, C_2)$ be two tri-sets of $\mathcal{F}$ and $TC \in \mathcal{TC}$. S_1 and S_2 belong to the same equivalence class represented by the tri-concept TC, i.e., $S_1 \equiv_{TC} S_2$ iff $h(S_1) = h(S_2) = TC$.*

The smallest tri-set (*w.r.t.* the number of items) in each equivalence class is called a tri-minimal generator and is defined as follows:

Definition 6. (TRI-MINIMAL GENERATOR) *Let $g = (A, B, C)$ be a tri-set such as $A \subseteq \mathcal{U}$, $B \subseteq \mathcal{T}$ and $C \subseteq \mathcal{R}$ and $TC \in \mathcal{TC}$. The triple g is a tri-minimal generator (tri-generator for short) of TC iff $h(g) = TC$ and $\nexists\, g_1 = (A_1, B_1, C_1)$ such as :*

1. $A = A_1$,
2. $(B_1 \subseteq B \wedge C_1 \subset C) \vee (B_1 \subset B \wedge C_1 \subseteq C)$, *and*
3. $h(g) = h(g_1) = TC$.

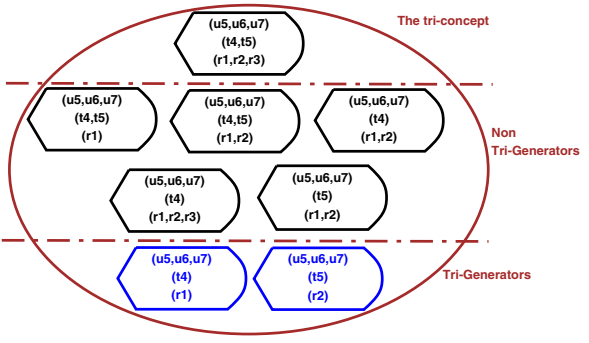

Fig. 1. Example of an equivalence class from $\mathcal{F}$

Figure 1 sketches a sample class of the induced equivalence relation from the *folksonomy* depicted by table 1. The largest unsubsumed tri-set $TC = \{\{u_5, u_6, u_7\}, \{t_4, t_5\}, \{r_1, r_2, r_3\}\}$, has three tri-generators g_1, g_2 and g_3. However, $g_4 = \{\{u_5, u_6, u_7\}, \{t_4, t_5\}, r_1\}$ is not a tri-generator of TC since it exists g_1 such as $g_1.extent$=$g_4.extent$, ($g_1.intent = \subseteq g_4.intent \wedge g_1.modus \subset g_4.modus$).

4.2 Description of the TRICONS Algorithm

TRICONS operates in three steps as follows:

1. The extraction of tri-generators;
2. The computation of the `modus` part of tri-concepts;
3. The computation of the `intent` part of tri-concepts.

The pseudo code of the TRICONS algorithm is sketched by Algorithm 1. TRICONS takes as input a *folksonomy* $\mathcal{F} = (\mathcal{U}, \mathcal{T}, \mathcal{R}, \mathcal{Y})$ as well as three user-defined thresholds : $minsupp_u$, $minsupp_t$ and $minsupp_r$. The TRICONS algorithm outputs the set of all frequent tri-concepts that fulfill these aforementioned thresholds. TRICONS operates as follows : it starts by invoking the **TRISORT** procedure (Line 2), that sorts the *folksonomy w.r.t.* the fields r, t and u, respectively. This sorting facilitates the handling of the *folksonomy* in order to extract the tri-generators. Then, TRICONS calls the FINDMINIMALGENERATORS procedure (Step 1), which pseudo-code is given by Algorithm 2, in order to extract the tri-generators which are stored in the set $\mathcal{MG}$ (Line 4) : for each triple (u, t, r), FINDMINIMALGENERATORS computes the set U_s which is the maximal set of users (including u) sharing the tag t and the resource r (Algorithm 2, Line 4).

ALGORITHM 1: TRICONS

Data :
1. $\mathcal{F}$: ($\mathcal{U}$, $\mathcal{T}$, $\mathcal{R}$, $\mathcal{Y}$) : A *Folksonomy*.
2. $minsupp_u$, $minsupp_t$, $minsupp_r$: User-defined thresholds.

Result : $\mathcal{TC}$: {Frequent tri-concepts}.

begin
  TRISORT($\mathcal{F}$);
  /*Step 1 : The extraction of tri-generators*/
  FINDMINIMALGENERATORS($\mathcal{F}$, $\mathcal{MG}$, $minsupp_u$);
  /*Step 2 : The computation of the modus part*/
  foreach *tri-gen* $g \in \mathcal{MG}$ **do**
    $Increase\_Set(\mathcal{MG}, minsupp_u, minsupp_t, g, \mathcal{TS}, true)$;
  end
  PRUNEINFREQUENTSETS($\mathcal{TS}$,$minsupp_t$);
  /*Step 3 : The computation of the intent part*/
  foreach *tri-set* $s \in \mathcal{TS}$ **do**
    $Increase\_Set(\mathcal{TS}, minsupp_u, minsupp_t, s, \mathcal{TC}, false)$;
  end
  PRUNEINFREQUENTSETS($\mathcal{TC}$,$minsupp_r$);
end
return $\mathcal{TC}$;

ALGORITHM 2: FINDMINIMALGENERATORS

Data :
1. $\mathcal{MG}$: The set of frequent tri-generators;
2. $\mathcal{F}$ ($\mathcal{U}$, $\mathcal{T}$, $\mathcal{R}$, $\mathcal{Y}$) : A *folksonomy*;
3. $minsupp_u$: User-defined threshold of user's support.

Result : $\mathcal{MG}$: {The set of frequent tri-generators}.

begin
  while *(u, t, r)* $\neq NULL$ **do**
    (u, t, r):=NEXTTRIPLE($\mathcal{F}$);
    U_s= $\{u_i \in \mathcal{U} \mid (u_i, t, r) \in \mathcal{Y}\}$;
    if $\mid U_s \mid \geq minsupp_u$ **then**
      $g.extent = U_s$; $g.intent = r$; $g.modus = t$;
      AddTri($\mathcal{MG}$, g)
    end
  end
end
return $\mathcal{MG}$;

Algorithm 2 invokes both **ADDTRI** and **NEXTTRIPLE** functions. The first one allows to add the tri-set Tri to the set $\mathcal{S}$, whereas the second one returns for each call the next triple (u, t, r) of the *folksonomy* $\mathcal{F}$.

ALGORITHM 3: $Increase\_Set$

Data :
1. $\mathcal{S}_{IN}$: The set of frequent tri-generators/tri-sets.
2. min_u, min_t : User-defined thresholds of extent and modus support.
3. tri : A tri-generator/tri-set.
4. *flag* : a boolean indicator.

Result : $\mathcal{S}_{OUT}$: {The set of frequent tri-sets/tri-concepts}.

```
1  begin
2      foreach tri-set tri' ∈ S_IN do
3          if flag and tri.intent = tri'.intent and tri.extent ⊆ tri'.extent then
4              s.intent = g.intent; s.extent = g.extent; s.modus = g.modus ∪
               g'.modus; AddTri(S_OUT, s);
5          end
6          else if flag and tri.intent = tri'.intent and tri and tri' are incomparables
           then
7              g''.extent = g.extent ∩ g'.extent; g''.modus = g.modus ∪ g'.modus;
               g''.intent = g.intent; If | g''.extent | ≥ min_u then AddTri(MG, g'');
8          end
9          else if not flag and tri.extent ⊆ tri'.extent and tri.modus ⊆ tri'.modus and
           tri.intent ≠ tri'.intent then
10             TC.extent = s.extent; TC.modus = s.modus; TC.intent = s.intent ∪
               s'.intent; AddTri(S_OUT, TC);
11         end
12         else if not flag and tri and tri' are incomparables then
13             s''.extent = s.extent ∩ s'.extent; s''.modus = s.modus ∩ s'.modus;
               s''.intent = s.intent ∪ s'.intent;
14             If | s''.extent | ≥ min_u and | s''.modus | ≥ min_t then AddTri(TS, s'');
15         end
16     end
17 end
18 return S_OUT ;
```

Afterwards, TRICONS invokes the $Increase\_Set$ procedure (Step 2) for each tri-generator of $\mathcal{MG}$ (Lines 6-8), which pseudo-code is given by Algorithm 3, in order to compute the modus part of the tri-concepts. The two first cases of Algorithm 3 (Lines 3 and 6) have to be considered by $Increase\_Set$ according to the extent of each tri-generator before returning the set $\mathcal{TS}$ of tri-sets. The boolean indicator *flag* marked by TRICONS shows whether the tri-set processed by the $Increase\_Set$ procedure is a tri-generator. Then, infrequent tri-sets, *i.e.*, whose the modus part cardinality does not fulfill the minimum threshold $minsupp_t$ are pruned (Line 9). In the third and final step, TRICONS invokes a second time the $Increase\_Set$ procedure for each tri-set of $\mathcal{TS}$ (Lines 11-13), in order to compute the intent part. $Increase\_Set$ looks for tri-sets s' of $\mathcal{TS}$ having a different intent part than a given tri-set s (Algorithm 3, Line 9). Before

returning the set $\mathcal{TC}$ of tri-concepts, TRICONS prunes the infrequent ones, *i.e.*, whose the intent cardinality does not fulfill the minimum threshold $minsupp_r$ by invoking the **PRUNEINFREQUENTSETS** procedure (Line 14). TRICONS comes to an end after invoking this procedure and returns the set of the frequent tri-concepts which fulfills the three thresholds $minsupp_u$, $minsupp_t$ and $minsupp_r$.

Example 3. Considering the *folksonomy* depicted by Table 1 (page 4) with $minsupp_u$ = 3, $minsupp_t$ = 3 and $minsupp_r$ = 2 yields the following track for the TRICONS algorithm. The first step of TRICONS consists in the extraction of (frequent) tri-generators from the context (step 1) thanks to the FINDMINIMALGENERATORS procedure. Then, invoking firstly the $Increase\_Set$ procedure, on these tri-generators, allows the reduction of the number of candidates. Hence, only five candidates, at step 2, are generated which directly lead to the frequent tri-concepts extracted by TRICONS. So, the set $\mathcal{TS}$ contains the tri-sets $\{\{u_1, u_2, u_3, u_5, u_7\}, \{t_2, t_3, t_4\}, r_1\}$, $\{\{u_1, u_2, u_3, u_5, u_7\}, \{t_2, t_3, t_4\}, r_2\}$, $\{\{u_1, u_2, u_3, u_5, u_7\}, \{t_2, t_3, t_4\}, r_3\}$, $\{\{u_2, u_3, u_7\}, \{t_1, t_2, t_3, t_4\}, r_2\}$ and $\{\{u_2, u_3, u_7\}, \{t_1, t_2, t_3, t_4\}, r_3\}$. At this level, TRICONS generates a number of candidates by far lower than its competitors, thanks to the generation of tri-generators. The third and final step, *i.e.,* the second call to the $Increase\_Set$ procedure, tends to increase the intent part of each tri-set belonging to $\mathcal{TS}$ in order to extract frequent tri-concepts. For example, the two latter tri-sets merge giving the tri-concept $\{\{u_2, u_3, u_7\}, \{t_1, t_2, t_3, t_4\}, \{r_2, r_3\}\}$ which is added to the set $\mathcal{TC}$. Contrariwise to both CUBEMINER and TRIAS, the tri-concepts are extracted only once. The final result set $\mathcal{TC}$ is then returned by TRICONS which comes to an end with frequent tri-concepts that fulfill the minimum thresholds mentioned above.

5 Experimental Results

In this section, we show through extensive carried out experiment the assessment of the TRICONS[5] performances *vs.* those of TRIAS and DATA-PEELER[6], respectively. We have applied our experiments on two real-world datasets. The first dataset, *i.e.,* DEL.ICIO.US, is considered to be dense, *i.e.,* containing many long frequent tri-concepts at various levels of minimum thresholds values, while the second is considered to be sparse, *i.e.,* containing a large number of tags but only a few of them frequently co-occur in tri-concepts (on average, no more than 2 tags).

• **DEL.ICIO.US: DENSE DATASET**: The DEL.ICIO.US dataset used for our experiments is around 10 MB in size (compressed) and it is freely downloadable[7]. The dense dataset contains 48000 triples : 6822 users, 671 tags and 13102 resources.

• **MOVIELENS: SPARSE DATASET**: The MOVIELENS dataset used is around 13 MB in size (compressed) and it is freely downloadable[8]. The sparse dataset contains 48000 triples : 33419 users, 18066 tags and 13397 resources.

[5] The TRICONS algorithm is implemented in *C++* (compiled with *GCC* 4.1.2) and we used an Intel Core *i*7 CPU system with 6 GB RAM. Tests were carried out on the Linux operating system UBUNTU 10.10.1.

[6] Unfortunately, the code of the CUBEMINER algorithm is not available.

[7] *http://data.dai-labor.de/corpus/delicious/*

[8] *http://www.grouplens.org*

Table 2. Performances TRICONS *vs.* those of TRIAS and DATA-PEELER (in **seconds**) above the DEL.ICIO.US and *MovieLens* datasets

# Triples	Dataset (Type)	TRICONS	TRIAS	DATA PEELER	Dataset (Type)	TRICONS	TRIAS	DATA PEELER
5000		**0,05**	0, 51	638, 22		**0,06**	0, 14	43, 64
15000		**0,55**	0, 91	1538, 15		**0,53**	1, 50	1271, 49
25000	*DEL.ICIO.US*	**3,31**	5, 68	1937, 23	*MovieLens*	**0,91**	3, 30	2010, 77
35000	*(Dense)*	**11,73**	17, 67	2318, 07	*(Sparse)*	**1,51**	6, 34	2909,91
48000		**13,73**	20, 67	2718, 07		**2,69**	11, 52	3851, 38

Performances of TRICONS vs. TRIAS and DATA-PEELER: For mining frequent tri-sets and frequent tri-concepts, we set minimum support values of $minsupp_u$ = 2, $minsupp_t$ = 2 and $minsupp_r$ = 1, *i.e.*, in a frequent tri-concept, at least, 2 users have assigned the same tags (2 at least) to a same resource at least. Table 2 compares the performances (in sec) of the three algorithms above for different values of the number of triples over the mentioned datasets. With respect to the aforementioned minimum support values, the number of the extracted tri-concepts from the DEL.ICIO.US dataset is around 3877. Whereas 1088 tri-conepts are extracted from *MovieLens* dataset.

• **TRICONS** vs. **TRIAS:** For both datasets, the different tests highlight that TRICONS always shows better performances than do TRIAS. For exemple, TRICONS reaches almost $13, 73$ sec when handling 48000 triples from DEL.ICIO.US, showing a drop in execution time of around $33, 57\%$, compared to TRIAS. Moreover, the obtained results, on the both datasets, confirm that this discrepancy between the two algorithms stills in favor of TRICONS as far as the number of triples grows. Interestingly enough, for the sparse dataset, *i.e.,* MOVIELENS, we note, for all values of the number of triples, an average reduction of TRICONS execution time reaching almost $69, 54\%$ compared to TRIAS. The performance differences between these mentioned algorithms can be explained by the fact that TRIAS starts by storing the entire *folksonomy* into main memory before extracting frequent tri-concepts. This memory greedy storage has the drawback to slow the algorithm and alters its execution time as far as the number of triples becomes significant. Contrarily to TRICONS that firstly invokes the FINDMINIMALGENERATORS procedure to extract the tri-generators which constitues the core of the tri-concepts. This specific treatment of TRICONS reduces the memory greediness. Indeed, the number of tri-generators are often by far below the total number of the triples in a *folksonomy*.

• **TRICONS** vs. **DATA-PEELER:** For both datasets and for all values of the number of triples, DATA-PEELER algorithm is far away from TRICONS performances. Indeed, the poor performance flagged out by DATA-PEELER, is explained by the strategy adopted by this later which starts by storing the entire *folksonomy* into a binary tree structure, which should facilitate its run and then the extraction of tri-concepts. Indeed, such structure is absolutely not adequate to support a so highly sized data, which is the case of the *folksonomies* considered in our evaluation. Furthermore, TRICONS is the only one algorithm that does not store the dataset in memory before proceeding the extraction of tri-concepts. In addition, TRICONS generates very few candidates thanks to the clever use of tri-generators that reduce the search space significantly. In contrast, TRIAS and

DATA-PEELER, in addition to store in memory the whole dataset, generate an impressive number of candidates, most of which are stored in memory uselessly given the small number of tri-extracted concepts.

• **TRIAS** vs. **DATA-PEELER:** Contrariwise to experimental results shown in [2], TRIAS outperforms DATA-PEELER since the considered datasets are far away larger. We used real-world datasets similar to those used in [6] which explains why TRIAS is better in terms of performance than its competitor.

6 Conclusion and Future Work

In this paper, we introduced an extension of the notion of closure operator and tri-generator in the *folksonomy* and we thoroughly studied their theoretical properties. Based on these notions, we introduced the TRICONS algorithm, for a scalable mining of tri-concepts, that heavily relies on the order ideal shape of the set of tri-minimal generators. In nearly all experiments we performed, the obtained results showed that TRICONS outperforms the pioneering algorithms of the literature; that is owe to the non-injectivity property of the closure operator. Other avenues for future work mainly address the extraction of other concise representations of frequent tri-sets. In this respect, we will try to expand the steady effort carried within the diadic case towards defining concise representations, *e.g.,* disjunction-free sets (closed) non-derivable sets, (closed) essential itemsets, to cite but a few. It is a thriving issue, since these concise representation have already shown interesting compactness rates [5].

References

1. Besson, J., Robardet, C., Boulicaut, J., Rome, S.: Constraint-based concept mining and its application to microarray data analysis. Intelligent Data Analysis 9, 59–82 (2005)
2. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.: Closed patterns meet n-ary relations. ACM Transactions on Knowledge Discovery from Data 3, 1–36 (2009)
3. Couch, A.L., Chiarini, M.: A Theory of Closure Operators. In: Hausheer, D., Schönwälder, J. (eds.) AIMS 2008. LNCS, vol. 5127, pp. 162–174. Springer, Heidelberg (2008)
4. Ganter, B., Wille, R.: Formal Concept Analysis. Springer (1999)
5. Hamrouni, T., Yahia, S.B., Nguifo, E.M.: Sweeping the disjunctive search space towards mining new exact concise representations of frequent itemsets. Data and Knowledge Engineering 68(10), 1091–1111 (2009)
6. Jäschke, R., Hotho, A., Schmitz, C., Ganter, B., Stumme, G.: Discovering shared conceptualisations in folksonomies. Web Semantics: Science, Services and Agents on the World Wide Web 6, 38–53 (2008)
7. Ji, L., Tan, K.L., Tung, A.K.H.: Mining frequent closed cubes in 3d datasets. In: Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, pp. 811–822 (2006)
8. Lehmann, F., Wille, R.: A Triadic Approach to Formal Concept Analysis. In: Ellis, G., Rich, W., Levinson, R., Sowa, J.F. (eds.) ICCS 1995. LNCS, vol. 954, pp. 32–43. Springer, Heidelberg (1995)
9. Mika, P.: Ontologies are us: A unified model of social networks and semantics. Web Semantics: Science, Services and Agents on the World Wide Web 5(1), 5–15 (2007)
10. Zaki, M.J.: Closed itemset mining and non-redundant association rule mining. In: Liu, L., Ozsu, M.T. (eds.) Encyclopedia of Database Systems. Springer (2009)