Noname manuscript No.
(will be inserted by the editor)

Interesting Pattern Mining in Multi-Relational Data

Eirini Spyropoulou · Tijl De Bie · Mario Boley

Received: date / Accepted: date

Abstract Mining patterns from multi-relational data is a problem attracting increasing interest within the data mining community. Traditional data mining approaches are typically developed for single-table databases, and are not directly applicable to multi-relational data. Nevertheless, multi-relational data is a more truthful and therefore often also a more powerful representation of reality. Mining patterns of a suitably expressive syntax directly from this representation, is thus a research problem of great importance.

In this paper we introduce a novel approach to mining patterns in multi-relational data. We propose a new syntax for multi-relational patterns as complete connected subsets of database entities. We show how this pattern syntax is generally applicable to multi-relational data, while it reduces to well-known tiles (Geerts et al 2004) when the data is a simple binary or attribute-value table. We propose RMiner, a simple yet practically efficient divide and conquer algorithm to mine such patterns which is an instantiation of an algorithmic framework for efficiently enumerating all fixed points of a suitable closure operator (Boley et al 2010). We show how the interestingness of patterns of the proposed syntax can conveniently be quantified using a general

E. Spyropoulou
Intelligent Systems Lab
University of Bristol
Woodland Road
Bristol, UK
E-mail: eirini.spyropoulou@bristol.ac.uk

T. De Bie
Intelligent Systems Lab
University of Bristol
Woodland Road
Bristol, UK

M. Boley
Fraunhofer IAIS
Schloss Birlinghoven
D-53754 Sankt Augustin, Germany

framework for quantifying subjective interestingness of patterns (De Bie 2011b). Finally, we illustrate the usefulness and the general applicability of our approach by discussing results on real-world and synthetic databases.

Keywords Multi-relational data mining · Pattern Mining · Interestingness measures · Maximum Entropy modelling · K-partite graphs

1 Introduction

Since the formalization of frequent itemset mining and association rule mining (Agrawal and Srikant 1994), the focus of pattern mining research has mostly been on mining frequent patterns in single-table databases (Srikant and Agrawal 1996; Zaki 2000; Uno et al 2004a; Zaki and Hsiao 2005) or graphs (Yan and Han 2002; Kuramochi and Karypis 2001). However, many datasets are inherently multi-relational and the information systems that manage them rely on multi-relational databases (MRDs). This imposes the need for exploring more complex patterns and corresponding data mining techniques. Application examples for multi-relational data mining could be mining patterns relating transactions, products and characteristics of products, in a sales database, or in a social network context, patterns relating authors with papers (co-authorship) as well as papers between each other (citations).

The key challenge in multi-relational data mining is the definition of a pattern type that is adequately expressive to capture the structure in the data, while it is easy to interpret. While Inductive Logic Programming approaches for multi-relational data make use of a very expressive pattern syntax (Dehaspe and Toivonen 1999; Nijssen and Kok 2003; Koopman and Siebes 2009), methods that work directly on the data instances have focused on transporting ideas from frequent itemset mining to the relational setting. The most common strategy is to first take the full join of all the tables of the MRD, after which standard itemset mining methods can be applied (Ng et al 2002; Koopman and Siebes 2008; Goethals et al 2010). However, in flattening the MRD in this way important structural information is inevitably lost. Finally, all previous approaches rely on transferring the notions of *recurring pattern* and *support* in the multi-relational setting either by measuring the support with respect to the entries of the join table (Ng et al 2002; Koopman and Siebes 2008) or with respect to just one table or entity in the database (Goethals et al 2010; Dehaspe and Toivonen 1999; Nijssen and Kok 2003; Koopman and Siebes 2009). We argue that this complicates the interpretation of the results, as it is not clear what it means for a multi-relational pattern to be frequent with respect to the join table or just one table (See Sec. 5 for a more detailed discussion.)

On top of these conceptual problems, most existing methods for mining MRDs also suffer from usability problems: the returned set of patterns is often overwhelmingly large and redundant, or subjectively not very interesting. Fortunately, these problems have recently been addressed by the pattern mining research community, albeit in simpler settings (mostly itemsets in binary databases). This includes the definition of new objective interestingness measures with various properties (see Geng and Hamilton (2006); Kontonasios et al (2012) for an overview), as well as the definition of general schemes to formalize subjective interestingness (Gionis et al 2007;

Hanhijarvi et al 2009; De Bie et al 2010; De Bie 2011b,a). Another related development, mainly aimed at reducing redundancies, is the focus on evaluating interestingness of pattern sets, instead of individual patterns (Siebes et al 2006; De Raedt and Zimmermann 2007). To improve multi-relational data mining methods, some of these ideas should be adopted.

Here we contribute on both these fronts: the conceptualization and search for patterns in MRDs, and the quantification of their interestingness. In particular, in Sec. 2 we propose Complete Connected Subsets, a new type of pattern syntax in MRDs that captures the structural information of an MRD and does not rely on the concept of support, thus avoiding some of the pitfalls in earlier work on this topic. We show that this type of pattern is *easy to interpret*, it is *generally applicable* to MRDs, while in simple settings it *subsumes itemsets as a special case* (or more accurately, tiles (Geerts et al 2004)). We further propose RMiner, a *simple*, yet *practically efficient* algorithm to mine all maximal Complete Connected Subsets (Sec. 3). The algorithm is an instantiation of a general divide and conquer enumeration framework for mining closed sets of restricted pattern languages (Boley et al 2010). We also show that RMiner can easily integrate further constraints that preserve the structural properties of the search space. We exploit this by providing a particularly useful minimum-coverage constraint that substantially improves the computation time. In Sec. 4 we show that the proposed pattern syntax lends itself well to *formalizing the subjective interestingness of patterns*, subject to certain prior knowledge on the data. In a similar way as De Bie (2011b) has done for itemsets in binary databases, this approach guarantees the interestingness of the returned patterns in a well-defined setting. We discuss related work in Sec. 5. In Sec. 6 we show results on real-world data and qualitatively compare our pattern syntax to other multi-relational pattern syntaxes. Finally, in Sec. 7 we show an evaluation of the interestingness measure, as well as a computational evaluation of the mining algorithm on synthetic datasets.

2 Multi-relational data and patterns

We first formalize multi-relational databases as considered in this paper. In an abstract manner this formalization is reminiscent of the Entity-Relationship (ER) model as explained in Elmasri and Navathe (2006). Then we show how an MRD as we formalise it, is uniquely represented as a K-partite graph. Finally, we move on to defining the proposed pattern syntax.

Multi-relational database (MRD) We formalize a **relational database** as a tuple $\mathbb{D} = (E, t, \mathcal{R}, R)$ where E is a finite set of *entities* that is partitioned into k *entity types* by a mapping $t : E \to \{1, \ldots, k\}$, i.e., $E = E_1 \dot\cup \ldots \dot\cup E_k$ with $E_i = \{e \in E \mid t(e) = i\}$. Moreover, $R \subseteq \{\{i, j\} \mid i, j \in \{1, \ldots, k\}, i \neq j\}$ is a set of *relationship types* such that for each $\{i, j\} \in R$ there is a binary relationship $\mathcal{R}_{\{i,j\}} \subseteq \{\{e_i, e_j\} \mid e_i \in E_i, e_j \in E_j\}$. The set $\mathcal{R}$ then is the union of all these relationships, i.e., $\mathcal{R} = \bigcup_{\{i,j\}\in R} \mathcal{R}_{\{i,j\}}$. Relationship types can be many-to-many, one-to-many, or one-to-one, depending on how many relationships the entities of either entity types can participate in. Note here that the fact that we do not allow relationship types

between an entity type and itself is not restrictive as we can model this as having two copies of the same entity type and a relationship type between them.

Example 1 Let us consider a toy example of a movie database, with 'year', 'title', and 'genre' as entity types, shown at the left hand side of Fig. 1. There is a relationship type between 'year' and 'title', specifying the year of release of the movie title, and between 'title' and 'genre', specifying the genres of a movie title. The first of these relationship types is a one-to-many relationship type, while the second is a many-to-many relationship type.

Remark 1 (What about attributes?) In an ER model, an entity can have attributes associated to it. Our formalism is different in that each attribute is treated as an entity type of its own. Associating attribute values with the entity they correspond to is done by making use of a one-to-many relationship type between the entity type of the attribute and the entity (Elmasri and Navathe 2006). E.g., in the toy example considered before, 'year' and 'title' would typically be modelled as attributes of the 'movie' entity. However, we model them as separate entities, with a relationship type between them. Note also that the 'title' is used to represent the entity 'movie' as they are one-to-one related. While this approach sacrifices some data modelling freedom, it allows a unified treatment of attributes and entities. This is desirable, as in the ER model the distinction between attributes and entities is often ambiguous, while we wish our methods to be independent of such modelling choices.

A graph representation of an MRD The MRDs resulting from our definition, can be represented as K-partite graphs. A graph is called K-partite if its nodes can be partitioned into k *blocks* such that there are no edges between the nodes of the same block. In the representation of an MRD as a K-partite graph, there is a node for each entity $e \in E$ in the MRD, and an edge between two entities e_k and e_l if $\{e_k, e_l\} \in \mathcal{R}$. We say that nodes representing entities of the same type are of the same node type, and similarly we say that edges representing relationships of the same type are of the same edge type. Clearly, the resulting graph is K-partite, each block in the graph containing nodes of the same node type. The graph representation of the toy MRD described in Ex.1 is shown at the right hand side of Fig. 1.

The pattern syntax The pattern type we introduce in this paper is called Complete Connected Subset (CCS). In what follows we are going to formally define the notion of *completeness* and *connectedness* for a subset of entities, and thus define a CCS.

Definition 1 (Completeness) A set $F \subseteq E$ is complete if for all $e, e' \in F$ with $\{t(e), t(e')\} \in R$ it holds that $\{e, e'\} \in \mathcal{R}_{\{t(e),t(e')\}}$.

Of course completeness alone does not suffice to have a meaningful pattern definition as it allows for completely unrelated (not connected) entities. In the MRD of Fig. 1, for example, the set $\{$Action,2010$\}$ is complete but not connected.

Definition 2 (Connectedness) A set $F \subseteq E$ is connected if for all $e, e' \in F$ there is a sequence $e = e_1, \ldots, e_l = e'$ with $\{e_1, \ldots, e_l\} \subseteq F$ such that for $i \in \{1, \ldots, l-1\}$ it holds that $\{e_i, e_{i+1}\} \in \mathcal{R}$.

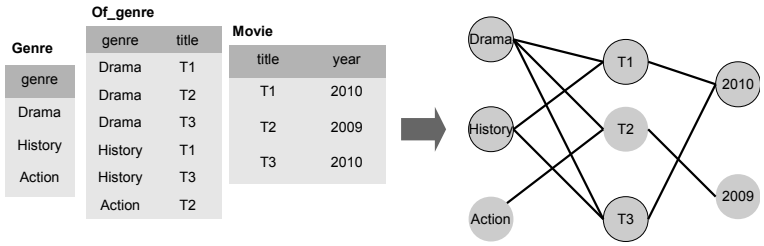


Fig. 1 Example of an MRD in table form (left) and in graph form (right). The entity types 'genre', 'title', 'year' correspond to different blocks in the graph and the entities of each entity type correspond to different nodes. The join table 'Of_genre' defines a many-to-many relationship between the entity types 'genre' and 'title' and the table 'Movie' defines an one-to-many relationship between entities 'title' and 'year'. Two entities are linked with an edge if they co-occur in a same tuple.

Note that a subset of size larger than one can be connected only if it contains entities of at least two different types.

A set $F \subseteq E$ is a **Complete Connected Subset** (CCS) if it satisfies both connectedness and completeness. Intuitively a CCS captures the co-occurrence of entities, within and between different relationship types. In this sense, it is a generalisation of tiles (itemsets and their supporting transactions) (Geerts et al 2004) for the case of MRDs. In the graph representation of the MRD, this pattern type corresponds to a K-partite clique.

As in other pattern mining tasks, the number of CCSs is typically massive even for moderately sized databases (exponential in the number of entities). Therefore enumerating all CCSs is impractical. To reduce the computational burden, we therefore opted to focus on only maximal CCSs which typically form a small subset of all CCSs. A **maximal Complete Connected Subset** (MCCS) is a CCS to which no element can be added without violating connectedness or completeness. Since each non-maximal CCS is (by definition) a subset of an MCCS, the set of MCCSs is a lossless representation of the set of CCSs. Additionally, we would argue that larger CCSs are more likely to be of interest than smaller ones, as they carry more information than their subset CCSs.

Example 2 In the MRD of Fig. 1 the set of entities $\{$T1, T3, Drama, History, 2010$\}$ represents an MCCS pattern. It is maximal as none of the remaining entities can be added without violating completeness. Looking at the graph representation one can see that this set of entities corresponds to a maximal K-partite clique. This pattern provides the information that titles T1 and T3 are both produced in 2010 and that are both of genre Drama and History.

MCCSs in special cases of MRDs Conceptually, MCCSs are easy to grasp, and the empirical results will further demonstate that this pattern syntax is a sensible and intuitive one. An additional argument in support of MCCSs is that they reduce to well-known pattern syntaxes of well-studied forms of data.

Consider a market-basket database, containing two entity types: items and transactions. There is one relationship type representing the fact that an item was bought in a transaction. An MCCS is a maximal tile in this database (Geerts et al 2004) or

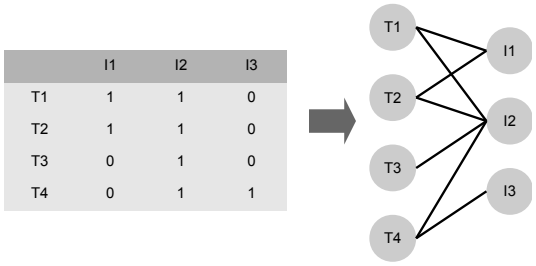


Fig. 2 Transcription database as a bipartite graph. Transactions and items represent different blocks of the graph and are linked with edges according to the '1's of the binary matrix.

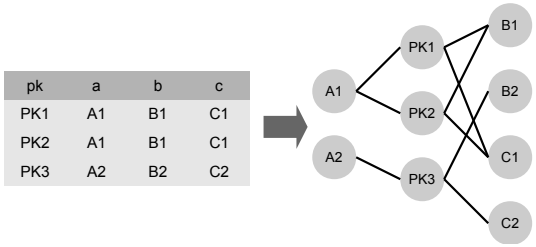


Fig. 3 Attribute-value database as a K-partite graph. Attributes represent different blocks of the graph and attribute values represent the nodes. Key and non-key attribute values are linked if they co-occur in the same tuple of the attribute-value table.

a closed itemset with its supporting transactions (Yahia et al 2006). It is well-known that a binary item-transaction database can be represented by means of a bipartite graph (Zaki and Ogihara 1998), and indeed this graph is exactly the graph representation of this special case of an MRD. An MCCS in this bipartite graph is a maximal biclique. This is depicted in Fig. 2, showing a database of three items and four transactions and the corresponding bipartite graph. The set of nodes $\{$T1, T2, I1, I2$\}$ is an example of a maximal tile or a maximal biclique in the graph representation.

Similarly, for a single attribute-value data table the entity types in our formalization consist of the entity type that uniquely identifies the rows of the table (typically identified by a primary key attribute), along with an entity type for each of the (non-key) attributes. Hence, for an attribute-value table with $K-1$ (non-key) attributes, we would have K entity types and $K-1$ relationship types between every non-key entity type and the primary key entity type. Traditionally these tables were binarized into a table that contained primary keys as transactions and attribute values as items, in order to apply itemset mining (Srikant and Agrawal 1996). An MCCS in this type of table contains a set of entities representing attribute-values and necessarily also a set of entities corresponding to the primary keys of the table. This is equivalent to a maximal set of attribute-values along with the supporting set of transactions (tiles) in the binarized version of the table. Figure 3 shows an attribute-value data table with three attributes and three transactions and the equivalent K-partite graph representation according to our formalization. Here, the set of entities $\{$A1, PK1, PK2, B1, C1$\}$

is an MCCS or a maximal tile or a closed itemset with its supporting transactions in the binarized version of this table.

We therefore showed that MCCSs correspond to closed itemsets in the reduced case of just two entity types. This equivalence is an additional important argument in favour of focusing on only mining MCCSs, as the set of closed frequent itemsets have been used extensively in the past as a lossless compression of the set of all frequent itemsets (Uno et al 2004a; Zaki and Hsiao 2002; Uno et al 2004b; Yahia et al 2006). As a cautionary note, we wish to point out that the notion of a maximal CCS is related to the notion of a closed itemset but not directly related to the notion of a maximal frequent itemset (Burdick et al 2005).

3 RMiner: An algorithm to search for all MCCSs

In the previous section we defined the pattern syntax of CCSs and we argued why it is sensible to focus on mining MCCSs only. Of course, for this choice to result in a true efficiency gain, an algorithm to mine MCCSs should successfully avoid enumerating the majority of CCSs that are not maximal.

To achieve this, we develop an algorithm which is an instantiation of a general divide and conquer algorithmic framework for listing all fixpoints of an arbitrary closure operator in a constrained search space, introduced in (Boley et al 2010; Boley 2011). This framework is used to efficiently enumerate all CCSs that are fixpoints of a closure operator, to which we will refer as *closed* CCSs. The set of closed CCSs contains the set of all MCCSs, while it is typically a small subset of the set of all CCSs (see Sec. 3.1 for more details).

In Sec. 3.1, we show that the fixpoint listing algorithm is indeed applicable to the set of CCSs. We then give an overview of the algorithm and introduce the proposed closure operator in Sec. 3.2. Finally, in Sec. 3.3, we define additional minimum coverage constraints and show how they can be incorporated into the same algorithmic framework. These additional constraints can be used to further reduce the search space on specific areas of interest, considerably reducing the computation times.

3.1 The applicability of the fixpoint listing algorithm

The divide-and-conquer fixpoint listing algorithm enumerates all closed sets of a closure operator from a given set system[1], as long as this set system possesses a structural property called *strong accessibility* (Boley et al 2010). A **set system** is a family of subsets $\mathcal{F} \subseteq \mathcal{P}(A)$ over some ground set A, where $\mathcal{P}(A)$ is the power set of A. At the end of this subsection we will show that the set of CCSs indeed forms a strongly accessible set system, demonstrating that the fixpoint listing algorithm is applicable for enumerating all *closed* CCSs.

[1] In contrast to some traditional fixpoint enumeration algorithms, as they are for instance used in the context of Formal Concept Analysis, this divide and conquer approach does neither assume an underlying complete lattice nor that the fixpoint set is closed under intersection. This is important because the set system of CCSs is not necessarily closed under intersection (due to connectivity) and two MCCSs cannot be joined to a common supremum (due to completeness).

A set $F \in \mathcal{F}$ is called *closed* if it is a fixpoint of some closure operator $\rho\colon \mathcal{F} \to \mathcal{F}$, i.e., $\rho(F) = F$. An operator $\rho : \mathcal{F} \to \mathcal{F}$ is a **closure operator** if it fulfils the following three properties (Birkhoff 1967):

- Extensivity: $F \subseteq \rho(F)$ for all $F \in \mathcal{F}$;
- Monotonicity: $\rho(F) \subseteq \rho(F')$ for all $F, F' \in \mathcal{F}$ with $F \subseteq F'$;
- Idempotence: $\rho(\rho(F)) = \rho(F)$ for all $F \in \mathcal{F}$.

The fact that a maximal CCS $F \in \mathcal{F}$ is always closed, trivially follows from the extensivity of the closure operator and the definition of maximality. Indeed, if F cannot be extended by any other entity, it follows from extensivity and from the fact that $\rho(F) \in \mathcal{F}$, that $\rho(F) = F$. This means that F is fixed under the closure operator and thus a closed CCS.

In Sec. 3.2, we will return to the definition of the particular closure operator used in this paper. Before doing that, we will first define the set system of CCSs and show that it is strongly accessible. This is a property that is required for the applicability of the algorithmic framework from Boley et al (2010). For a database $\mathbb{D} = (E, t, \mathcal{R}, R)$ the **set system of CCSs**, is defined as

$$\mathcal{F}_{\mathbb{D}} = \{F \subseteq E \mid F \text{ connected } \wedge F \text{ complete}\}.$$

The property of strong accessibility intuitively means that for two CCSs $X, Y \in \mathcal{F}_{\mathbb{D}}$ with $X \subset Y$, it is possible to iteratively extend X by one element at a time, only passing via sets from the set system and ultimately resulting in Y. Formally, for a set system $\mathcal{F} \subseteq \mathcal{P}(A)$, where A is the ground set, and a set $F \in \mathcal{F}$, let us denote by $\mathrm{Aug}(F) = \{a \in A \mid F \cup \{a\} \in \mathcal{F}\}$ the set of valid **augmentation elements** of F. Then $\mathcal{F}$ is called **strongly accessible**[2] if for all $X \subset Y \subseteq A$ with $X, Y \in \mathcal{F}$ there is an element $e \in (\mathrm{Aug}(X) \setminus X) \cap Y$. We can now state the desired result.

Theorem 1 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$, the set system $\mathcal{F}_{\mathbb{D}}$ of CCSs is strongly accessible.*

Proof To prove this theorem, we additionally rely on the notion of an independence system: $\mathcal{F}$ is an **independence system** if for all $X \in \mathcal{F}$, for every $Y \subseteq X$, $Y \in \mathcal{F}$. The set system $\mathcal{F}_{\mathbb{D}}$ is the intersection of the set system of connected subsets and the set system of complete subsets. First we are going to prove that the set system of connected subsets, $\mathcal{F}$, is a strongly accessible set system. For $X, Y \in \mathcal{F}$, $X \subset Y$ assume that there is no $e \in (\mathrm{Aug}(X) \setminus X) \cap Y$. This means that there is no element e in $Y \setminus X$ such that $e \in \mathrm{Aug}(X)$ which means that Y and $Y \setminus X$ are disconnected. This is a contradiction since $Y \in \mathcal{F}$.

Next we prove that the set system of complete subsets, $\mathcal{F}'$ is an independence system. For a $X \in \mathcal{F}'$ assume there is a $Y \subseteq X$, $Y \notin \mathcal{F}'$. This means there exist $e, e' \in Y$ such that $\{t(e), t(e')\} \in R$ and $\{e, e'\} \notin \mathcal{R}$. However because $Y \subseteq X$, $e, e' \in X$ which is a contradiction because it means that $X \notin \mathcal{F}'$.

Thus, the set system of CCSs $\mathcal{F}_{\mathbb{D}}$ is an intersection of a strongly accessible set system $\mathcal{F}$ and an independence system $\mathcal{F}'$. It can be confirmed that this intersection is indeed strongly accessible: Let X, Y with $X \subset Y$ be in both set systems. Then

[2] Strongly accessible set systems generalize *greedoids* such as, e.g., poset ideals (see Boley (2011, Sec. 3.5.2) and Korte and Lovász (1985)).

there must be an augmentation element $y \in Y \setminus X$ with $X \cup \{y\} \in \mathcal{F}$, because $\mathcal{F}$ is strongly accessible. The same augmentation element can also be used in the intersection $\mathcal{F} \cap \mathcal{F}'$ because $\mathcal{F}'$ being an independence system implies $Y \supseteq (X \cup \{y\}) \in \mathcal{F}'$. □

3.2 The basic RMiner algorithm

As we have already established that the divide and conquer fixpoint listing algorithmic framework of Boley et al (2010) is applicable to the case of CCSs, we now give an overview of the algorithm and define a suitable closure operator.

The general structure of this divide-and-conquer algorithm is shown in Algorithm 1. The algorithm can be described in terms of the set F (an intermediate solution), the set of valid augmentation elements $Aug(F)$, the set B of elements already considered as extensions to F, and a closure operator g. In each recursive call, the algorithm selects an element e from the set of augmentation elements and splits the search space into two subtrees: one subtree in which all CCSs include the element e (line 7) and another subtree in which all CCSs exclude e, which is achieved by adding it to B (line 10). Adding nodes from $Aug(F)$ only, ensures that every set explored is a CCS. The fact that only closed patterns are sought is ensured in line 2, where the expanded set $F \cup \{e\}$ is potentially further expanded by applying the closure operator. The recursive call in line 7 is applied if this expansion does not include any elements from B (line 4) thus avoiding duplicate solutions, and if it doesn't correspond to a maximal solution (lines 4-5).

As the divide and conquer fixpoint listing algorithmic framework enumerates all closed sets, we added lines 4-5 to ensure that RMiner outputs MCCSs only, i.e. CCSs F for which there are no augmentation elements not yet in F. Formally, this is performed by checking whether $F = Aug(F)$.

Algorithm 1 RMiner: List all MCCSs

RMiner(F, B)
1: Select $e \in Aug(F) \setminus (F \cup B)$
2: $F' = g(F \cup \{e\})$
3: **if** $F' \cap B = \emptyset$ **then**
4: **if** $F' = Aug(F')$ **then**
5: Output F'
6: **else**
7: **RMiner**(F', B)
8: **end if**
9: **end if**
10: **RMiner**(F, $B \cup \{e\}$)

As defined in Sec. 3.1, the set of augmentation elements $Aug(F)$ of a set F from a set system is the set of all elements that can be individually added to F to yield another set from the same set system. Specifically for the set system $\mathcal{F}_{\mathbb{D}}$ of CCSs, and given a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$, the set $Aug(F)$ corresponds to the following set: $Aug(F) = \{e \in E \mid F \cup \{e\}$ is complete and connected$\}$.

Note that for the sake of efficiency $Aug(F)$ can be recursively updated after expanding F to F', as we explain in Sec. 3.5.

The closure operator In order to define a closure operator for the set system $\mathcal{F}_{\mathbb{D}}$ we make use of the set of compatible entities which is defined as follows:

Definition 3 (Compatible Entities) For a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$ the set of compatible entities of a set $F \in \mathcal{F}_{\mathbb{D}}$ is defined as $Comp(F) = \{e \in E \mid F \cup \{e\} \text{ is complete}\}$.

Clearly, $Comp(F) \supset Aug(F)$. We note that, as opposed to $Aug(F)$, the set $Comp(F)$ is an anti-monotone set, i.e., for $F' \supseteq F$, $Comp(F') \subseteq Comp(F)$. This follows from the observation that bigger sets have less compatible elements than their subsets. We now define the following operator.

Definition 4 (g operator) For a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$ we define the operator $g : \mathcal{F}_{\mathbb{D}} \to \mathcal{P}(E)$ as

$$g(F) = \{e \in Aug(F) \mid Comp(F \cup \{e\}) = Comp(F)\} \ .$$

In order to conclude in Corollary 1 below that g is a closure operator on $\mathcal{F}_{\mathbb{D}}$, we first need to prove that the codomain of g is $\mathcal{F}_{\mathbb{D}}$ and then prove that three properties of a closure operator, i.e., extensivity, monotonicity and idempotence hold.

Proposition 1 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$, the codomain of the g operator is the set system $\mathcal{F}_{\mathbb{D}}$ of CCSs.*

Proof We need to show that for every CCS $F \in \mathcal{F}_{\mathbb{D}}$, $g(F)$ is also complete and connected.

Connectedness follows trivially from the fact that only elements from $Aug(F)$ are added, i.e. only elements for which $F \cup \{e\}$ is connected.

To show completeness, let us assume that $g(F)$ is not complete. This means that there exists a pair of elements $e, e' \in g(F)$ such that $\{t(e), t(e')\} \in R$ and $\{e, e'\} \notin \mathcal{R}$. However, it holds trivially that $e \in Comp(F \cup \{e\})$ and $e' \in Comp(F \cup \{e'\})$. Exploiting the fact that $Comp(F \cup \{e\}) = Comp(F \cup \{e'\}) = Comp(F)$ (from the definition of g), this means that $e \in Comp(F \cup \{e'\})$. Thus by definition of compatibility also $F \cup \{e\} \cup \{e'\}$ is complete—a contradiction.□

It is trivial to see that the operator g is extensive as it does not remove any elements from the set it is applied to. Let us now prove that g is also monotone and idempotent.

Proposition 2 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$, the operator g is monotone.*

Proof Assume the operator is not monotone, i.e., there is an $F', F \subseteq F', F, F' \in \mathcal{F}_{\mathbb{D}}$, such that $g(F) \not\subseteq g(F')$. This means that $\exists e \in g(F)$ such that $e \notin g(F')$. By the definition this can happen if:

– $e \notin Aug(F')$. This cannot be because $F' \cup \{e\}$ is not connected, since $e \in Aug(F)$ and F' is a CCS with $F' \supseteq F$. Therefore it must be because $F' \cup \{e\}$ is not complete. Therefore $\exists f \in F'$ such that $\{t(e), t(f)\} \in R$ but $\{e, f\} \notin \mathcal{R}$. Therefore $f \notin Comp(F \cup \{e\})$. However since $F' \supseteq F$ and $f \in F'$, it must hold that $f \in Comp(F)$. Since we have assumed that $e \in g(F)$, this is a contradiction.

– $Comp(F' \cup \{e\}) \subset Comp(F')$ (because of the anti-monotonicity of the $Comp$ set). Therefore there is an f such that $f \in Comp(F')$ but $f \notin Comp(F' \cup \{e\})$. From the definition of *completeness* this means that $\{t(e), t(f)\} \in R$, but $\{e, f\} \notin \mathcal{R}$. Therefore $f \notin Comp(F \cup \{e\})$. On the other hand since $f \in Comp(F')$ and $F' \supseteq F$, it follows that $f \in Comp(F)$. This is a contradiction since we have assumed that $e \in g(F)$.

□

Proposition 3 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$ with the property that $\nexists e \in E$ such that $\{e\} \cup E_i$ is complete and connected for an $i \in t(E)$, the operator g is idempotent.*

Proof Assume that for a database $\mathbb{D} = (E, t, \mathcal{R}, R)$, such that $\nexists e \in E$ such that $\{e\} \cup E_i$ is complete and connected for an $i \in t(E)$, the operator g is not idempotent. This can only happen if $Aug(F) \not\supseteq Aug(g(F))$, because if $Aug(F) \supseteq Aug(g(F))$ there can be no additional element holding the properties of the closure. $Aug(F) \not\supseteq Aug(g(F))$ means that $\exists f \in g(F)$ such that $\exists i \in t(Aug(\{f\}))$ with $i \notin t(Aug(F))$. But since $f \in g(F)$, this can happen only if $Comp(F \cup \{f\}) \cap E_i = Comp(F) \cap E_i$. However, because $i \notin t(Aug(F))$, it follows that $Comp(F \cup \{f\}) \cap E_i = E_i$. Therefore the set $E_i \cup \{f\}$ is complete and connected. This is a contradiction.□

Corollary 1 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$, with the property that $\nexists e \in E$ such that $\{e\} \cup E_i$ is complete and connected for an $i \in t(E)$, the operator g is a closure operator.*

This Corollary 1 together with Theorem 1 finally shows correctness of Algorithm 1.

3.3 The RMiner algorithm with additional constraints

In Sec. 3.2 we presented an algorithm to mine the set of MCCSs by enumerating the set of closed CCSs. To increase the scalability of the algorithm even more, we define a pattern syntax which corresponds to CCSs that satisfy an additional constraint. The goal now is therefore to mine maximal, constrained CCSs by enumerating a smaller set than the closed CCSs.

More specifically we define a constraint c on the minimum number of entities per entity type, and we refer to it as *minimum coverage constraint*. When reduced to the binary transaction database case, this constraint corresponds to having a minimum number for items in a pattern as well as a minimum number of supporting transactions. We now formally define this constraint.

Definition 5 (Minimum coverage constraint)
For all $i \in t(E)$ of a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$, we fix a number $c_i \in \mathbb{N}$ and define

$$c(F) = \begin{cases} 1, \text{ if } \forall i \in t(E), |F \cap E_i| \geq c_i \\ 0, \text{ otherwise} \end{cases} .$$

In order to exploit the fact that we are interested in mining only a subset of maximal CCSs, i.e., the ones that satisfy the constraint, we need to use this constraint for pruning (in line 6 of Algorithm 1). However $c(F)$ cannot directly be used for pruning as the invalidity of the constraint on an enumerated CCS F does not imply invalidity of its complete and connected supersets. A correct way of pruning is to check the validity of the constraint on the set $(Comp(F) \setminus B)$. Recall here that B in Algorithm 1 is the set of elements already considered as extensions to F. Checking the validity of the constraint on the set $(Comp(F) \setminus B)$ means verifying if the constraint would end up being satisfied in the most optimistic case when all allowable compatible elements are added to F. If $(Comp(F) \setminus B)$ does not satisfy the constraint, no CCS that is a superset of F will ever satisfy the constraint as there exist insufficient elements that F could potentially be extended with in order to satisfy the constraint. Thus any set F for which the constraint is not satisfied on $(Comp(F) \setminus B)$ can be pruned.

In what follows we formalize this intuition by defining an upper bound $\bar{c}$ of the constraint c that is based on $(Comp(F) \setminus B)$, where B is the set of elements already considered as extensions to F. We then prove that $\bar{c}$ is anti-monotone which allows us to show that the set system of CCSs that satisfy $\bar{c}$ is strongly accessible, such that the divide and conquer fixpoint listing algorithmic framework remains applicable.

Definition 6 (Constraint upper bound)
For all $i \in t(E)$ of a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$, we fix a number $c_i \in \mathbb{N}$ and define

$$\bar{c}(F) = \begin{cases} 1, \text{ if } \forall i \in t(E), |(Comp(F) \setminus B) \cap E_i| \geq c_i \\ 0, \text{ otherwise} \end{cases} ,$$

where B is the set of elements already considered as extensions to F.

Clearly, $\bar{c}(F) \geq c(F)$ for all $F \in \mathcal{F}_{\mathbb{D}}$ because $F \subseteq (Comp(F) \setminus B)$ and thus $|(Comp(F) \setminus B) \cap E_i| \geq |F \cap E_i|$. This means that whenever the upper bound is 0 the constraint is 0 as well. However this not true for when the upper bound is 1, which means that the upper bound of he constraint can only be used for forward pruning.

Proposition 4 *The upper bound $\bar{c}(F)$ is an anti-monotone constraint,i.e, if $\bar{c}(F) = 0$ for a set F then for every $F' \supseteq F$, $\bar{c}(F') = 0$*

Proof Assume the contrary, i.e., for a set F such that $\bar{c}(F) = 0$ there is a set $F' \supseteq F$ such that $\bar{c}(F) = 1$. This means that $\exists i \in t(E)$ such that $|(Comp(F) \setminus B) \cap E_i| < c_i$ but $|(Comp(F') \setminus B') \cap E_i| \geq c_i$. From the anti-monotonicity of the set $Comp$ we have that $Comp(F') \subseteq Comp(F)$. Also $B' \supseteq B$. Therefore $(Comp(F') \setminus B') \subseteq (Comp(F) \setminus B)$ which means that the assumption we made leads to a contradiction. Therefore the $\bar{c}(F)$ is an anti-monotone constraint□

We are now going to show that the **set system of CCSs that satisfy $\bar{\mathbf{c}}$** in a relational database $\mathbb{D} = (E, t, \mathcal{R}, R)$ defined as

$$\mathcal{F}_C = \{F \subseteq E \mid F \text{ connected } \wedge F \text{ complete} \wedge \bar{c}(F) = 1\}$$

is strongly accessible such that the divide and conquer fixpoint listing algorithm can be used directly to enumerate the set of closed CCSs that satisfy the upper bound $\bar{c}$. Then we are going to show how Algorithm 1 is adapted using $\bar{c}$ and that it outputs exactly the set of maximal CCSs that satisfy the original constraint c.

Proposition 5 *For all relational databases $\mathbb{D} = (E, t, \mathcal{R}, R)$ the set system $\mathcal{F}_C$ is strongly accessible.*

Proof From the anti-monotonicity of $\bar{c}$ it follows that the set system of subsets satisfying $\bar{c}$ is an independence system. Thus, $\mathcal{F}_C$ is an intersection of a strongly accessible set system (namely $\mathcal{F}_{\mathbb{D}}$) and an independence system. From the proof of Theorem 1 we already know that the intersection of a strongly accessible set system and an independence system is strongly accessible.□

To adapt Algorithm 1 to enumerate only the closed CSSs that satisfy the upper bound $\bar{c}$, we only need to add the extra pruning condition to line 3:

$$\textbf{if } F' \cap B = \emptyset \wedge \bar{c}(F') = 1 \textbf{ then}$$

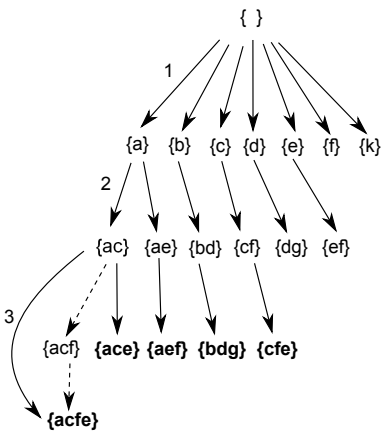
It remains to show that the adapted algorithm outputs exactly the set of maximal CCSs that satisfy the original constraint c. Let us denote as $\mathcal{C}_{\bar{c}}$ the set of closed CCSs, $\mathcal{C}$, that satisfy $\bar{c}$, as $\mathcal{M}_{\bar{c}}$ the set of maximal CCSs, $\mathcal{M}$, satisfying $\bar{c}$ and as $\mathcal{M}_c$ the set of maximal CCSs satisfying c. We already know that $\mathcal{M} \subseteq \mathcal{C}$. Therefore $\mathcal{M}_{\bar{c}} \subseteq \mathcal{C}_{\bar{c}}$. From the additional fact that maximal solutions cannot be extended and $\bar{c}$ is an upper bound of c, it follows that $\mathcal{M}_{\bar{c}} = \mathcal{M}_c$. Therefore $\mathcal{M}_c \subseteq \mathcal{C}_{\bar{c}}$, which means that the adapted algorithm enumerates a superset of the set of maximal CCSs that satisfy c and outputs exactly this set.

3.4 Illustrating Example

Before analysing the performance of RMiner we give an illustrating example of how it runs. Figure 4 shows the search space of RMiner on a toy dataset comprising of four entity types and two relationship types. It also shows in detail the values of all the relevant sets for the three running steps of RMiner that correspond to the leftmost branch of the tree.

3.5 Performance

In Secs. 3.1, 3.2 we showed the applicability of the fixpoint listing framework and described the algorithm at a high level. Here we show how Algorithm 1 is implemented, we discuss time and space complexity and give additional implementation details which make the algorithm practically efficient. To show the space and time complexity we follow a similar procedure as Boley et al (2010).



type 1	type 2
a	c
a	e
b	d

type 3	type 4
c	f
e	f
d	k

1. F={}
Adj({})={a,b,c,d,e,f,k}
select a
Comp({})={a,b,c,d,e,f,k}
Comp({a})={a,b,c,e,f,k}
g({a})={}

2. F={a}
select c
Adj({a})={a,c,e}
Comp({a})={a,b,c,e,f,k}
Comp({a,c})={a,c,f,e}
g({a,c})={}

3. F={a,c}
select f
Adj({a,c})={a,c,e,f}
Comp({a,c})={a,c,e,f}
Comp({a,c,f})={a,c,e,f}
g({a,c,f})={a,c,e,f}

Fig. 4 Illustrating example of running RMiner on a toy dataset comprising of four entity types and two relationship types. The left part of the figure shows the search space of RMiner for this toy dataset. The plain arrows represent the steps of RMiner, while the dashed arrows represent the search steps if the closure operator was not used. The right part of the figure shows in detail the values of all the sets for the running steps of RMiner that correspond to the leftmost branch of the search tree.

Algorithm 2 Implemented RMiner

Global:

1: $Comp\_list$ List of Compatible elements for every entity.
2: Rel_list List of related types for every entity type.

Main()

1: RMiner($\emptyset$, $\emptyset$, $Comp(\emptyset)$, $Aug(\emptyset)$, $types(\emptyset)$)

RMiner$(F, B, Comp(F), Aug(F), types(F))$

1: **for all** $e \in Aug(F) \setminus (F \cup B)$ **do**
2: $types((F \cup \{e\})) = types(F) \cup Rel\_list(t(e))$
3: $Comp(F \cup \{e\}) = Comp(F) \cap Comp\_list(e)$
4: $Aug(F \cup \{e\}) = construct\_aug(Comp((F \cup \{e\})), types((F \cup \{e\})))$
5: $F' = compute\_closure(Aug(F \cup \{e\}), Comp(F \cup \{e\}))$
6: **if** $F' \cap B = \emptyset$ **then**
7: **if** $F' = Aug(F')$ **then**
8: Output F'
9: **else**
10: **RMiner**(F', B, $Comp(F \cup \{e\})$, $Aug(F \cup \{e\})$, $types(F \cup \{e\})$)
11: **end if**
12: **end if**
13: $B = B \cup \{e\}$
14: **end for**

Implementation The implementation of RMiner is shown in Algorithm 2. We store two types of global information which we get directly from the data set. A structure named $Comp\_list$, which contains the set of compatible entities $Comp(e)$ for every $e \in E$ and a structure named Rel_list which contains all entity types an entity type is related through the relationship types, for every entity type.

In principle, the choice of the element $e \in Aug(F) \setminus (F \cup B)$ in line 1 of Algorithm 1 is free. We assume an arbitrary ordering of the elements in E and implement the second recursion of Algorithm 1 and the selection of an element with a for loop.

The implementation of RMiner is based on the fact that when adding an element e to a set F the we can compute $Comp(F \cup e)$ as $Comp(F \cup e) = Comp(F) \cap Comp(e)$, because adding an element e to a set F can only reduce the set of compatible elements due to the anti-monotonicity of this set (line 3). Also by keeping track of the entity types that are related to the entity types already in F (line 2) we can compute $Aug(F)$ from $Comp(F)$ by iterating through it and considering only entities of these types.

Space Complexity Let S_{types}, S_{Comp} and S_{Aug} the space required for storing $types$, $Comp$ and Aug respectively. The total space complexity for this implementation is $n \times (S_{types} + S_{Comp} + S_{Aug})$ where n is the number of entities in the input data, as we need to store these sets as many times as the depth of the search tree. Let m be the number of entity types in the input data. For the individual complexities we have:

- S_{types} is $O(m)$
- S_{Comp} is $O(n)$
- S_{Aug} is $O(n)$

Therefore the total space complexity is $O(n^2) + O(n * m)$ which finally is $O(n^2)$.

Although this space complexity is quadratic to the the input size, the scalability experiment we did in Sec. 7.3 shows that in practice it appears to be linear. This is because in practice the depth of the search tree is very small (in most cases the search space gets less than 2 times deeper for a two orders of magnitude increase of the input size). However the theoretical space complexity can become linear as well if one follows the implementation strategy of the modified algorithm proposed in (Boley et al 2010).

Time Complexity We study the time complexity of Algorithm 1 in terms of the delay between producing two closed CCSs. The total complexity is given by the number of closed CCSs times the delay between producing two closed CCSs. Let T_g be the complexity of computing the closure, T_{int} the complexity of intersecting two sorted sets which is the complexity of computing the set $Comp$, T_{Aug} the complexity of computing the set Aug, T_{types} the complexity of computing the set $types$ and T_{eq} the time to check the equality of two sorted sets. In the worst case the time between producing two closed CCSs (lines 8 and 10 of Algorithm 2) is $n \times (T_{types} + T_{Aug} + T_{int} + T_g)$, where n is the number of entities in the input size. This corresponds to the worst case time in which no closed CCS is produced because of the condition in line 6 being false. Let m be the number of entity types in the input. All the sets involved are ordered. For the individual time complexities we have:

- T_{types} is $O(m)$.
- T_{int} is $O(n)$.
- T_{Aug}: is $O(n)$.
- T_g: is $n \times T_{eq}$, which is $O(n^2)$.

Therefore, the delay between producing two closed sets is $O(n^3)$. Note here that in the case of RMiner with constraints the complexity of computing $\bar{c}(F)$ is $O(n)$ and does not change the delay.

Practical Performance Although the time delay between producing two closed CCSs is $O(n^3)$, practically it depends on the size of the set $Aug(F)\setminus(F\cup B)$ for the number of times the closure is computed without any closed CCS being produced and the set $Aug(F)$ for the number of elements checked if they are in the closure. $|Aug(F)|$ depends on the type of data and is smaller for sparser data sets. However since the next closed CCS is produced only when $F' \cap B = \emptyset$ we can incorporate this in the computation of the closure and stop checking if any more elements are in it as soon as one of them belongs to B. To further enhance the effectiveness of this approach, we additionally choose a specific ordering of the entities in E, in terms of increasing cardinality of the set $Aug(e)$. This way elements with small $Aug(e)$ which are going to need a small number of closure checks are considered first and elements which are going to need a larger number of closure checks are considered later. When elements that need more closure checks are considered, the set B has increased as well and therefore it's more likely that less closure checks are done.

In the case of RMiner with constraints the total time depends on the number of closed CCS that satisfy $\bar{c}$, i.e., the cardinality of the enumerated set of CCSs. This number depends on the order according to which elements of $Aug(F) \setminus (F \cup B)$ are considered and can be reduced by first considering entities of types not yet satisfying the constraint (if any). This helps pruning branches that have supersets not satisfying the constraint. In this case we therefore rearrange the order of the entities in every sub-call so that elements with smaller cardinality of the set $Aug(\{e\})$ that are also of a type not yet satisfying the constraint are considered first.

Finally improvement in the practical performance can be obtained if all sets are stored in composite structures of separate lists for different entity types. This way the set $Aug(F)$ does not need to be stored and computed explicitly as there could be a for loop iterating over the entity types in $Comp(F)$ and only considering entities of types in $types(F)$. Also the computation of $\bar{c}(F)$ in this case can be done in $O(m)$ which is an improvement since it holds that $m < n$.

4 Assessment of patterns

Although much smaller than the total number of CCSs, the number of MCCSs is usually still too large to be practical for an end user. This is similar to the fact that the set of closed itemsets typically needs further reduction to become useful. Typically this problem is addressed by selecting or ranking patterns using objective or subjective interestingness measures (Geng and Hamilton 2006). Here, we choose to define interestingness with respect to a specific type of prior information, by defining an interestingness measure which deems an MCCS to be more interesting if it is more unexpected given this prior information. More specifically, we consider as prior information the number of relationship instances each entity is involved in, in the different relationship types of the MRD. This corresponds to the degree of each

node in the different relationship types of the K-partite graph representation of the MRD. An MCCS is more interesting if it is harder to explain based on this prior information alone. For example in the setting of a movie MRD, an MCCS containing directors that have directed many movies would be deemed less interesting by our approach than an equally large MCCS containing less prolific directors, as the latter MCCS cannot as easily be attributed to randomness and is more unexpected.

To introduce the interestingness measure, we can closely follow the work presented in (De Bie 2011b; De Bie et al 2010), where it is argued that subjective interestingness can be formalized by contrasting patterns with a background model that is the Maximum Entropy model subject to the prior information. Thus we only need to detail the Maximum Entropy model for the case of MRDs (see Sec. 4.1), and the approach to contrast MCCS patterns with this model to arrive at an interestingness measure (see Sec. 4.2).

4.1 Maximum-Entropy model of the user's prior information

We consider as prior information the number of relationship instances each entity is involved in, for every relationship type in the MRD. Following De Bie (2011b), we formalize this prior information in a probability distribution P, fitting the Maximum Entropy distribution on the *MRD*, with constraints on the expected degree of the nodes for every relationship type being equal to their actual degree. This is the distribution of maximal uncertainty about the data with only the prior information as bias.

The nature of the constraints is such that they are defined for every relationship type $R_{\{i,j\}}$ of the *MRD* without imposing any dependence between the relationship types. Therefore, the Maximum Entropy distribution for the MRD subject to these constraints will be a product of independent Maximum Entropy distributions, one for each relationship type. Indeed, if there were dependencies between the relationship types, the Entropy of the joint distribution would be reduced by their mutual information (Cover and Thomas 2005), and would therefore not be maximal. Representing each relationship type as a binary database D_{ij} with $D_{ij}(k,l) = 1$ when $(e_i^k, e_j^l) \in \mathcal{R}_{\{i,j\}}$, the Maximum Entropy distribution for the MRD is thus:

$$P(\cup_{ij} D_{ij}) = \prod_{ij} P_{ij}(D_{ij}).$$

Maximizing the Entropy for every relationship type $R_{\{i,j\}}$ of the *MRD* represented by a binary matrix D_{ij} subject to constraints on the expected number of relationship instances for every entity, is equivalent to maximising the Entropy of a distribution for a binary database subject to constraints on the expected row and column sums. The solution of this problem was shown to be a product of independent Bernoulli distributions (De Bie 2011b):

$$P_{ij}(D_{ij}) = \prod_{k,l} P_{ij}^{kl}(D_{ij}(k,l)),$$
$$\text{with } P_{ij}^{kl}(D_{ij}(k,l)) = \frac{\exp\left(D_{ij}(k,l)(-\lambda_{ij}^k - \mu_{ij}^l)\right)}{1 + \exp(-\lambda_{ij}^k - \mu_{ij}^l)},$$

where λ_{ij}^k, μ_{ij}^l are parameters that can be computed efficiently. Indeed, as shown in De Bie (2011b), they can be found by solving the Lagrange dual of the maximum entropy optimization problem. This is a convex optimization problem of which the Hessian and the gradient can be computed efficiently, such that it can be solved efficiently using e.g. a few Newton iterations, or alternatively the conjugate gradient method for particularly large problems. For sparse datasets significant further optimizations can be made (See De Bie (2011b) for full details).

4.2 Contrasting MCCSs with the Maximum Entropy model

An interesting pattern conveys as much information as possible when contrasted with the user's prior information, as concisely as possible. Following earlier work (De Bie 2011b), we can formalize this idea by quantifying the interestingness of an MCCS F as the ratio of the self information of the MCCS and its description length:

$$\text{Interestingness}(F) = \frac{\text{SelfInformation}(F)}{\text{DescriptionLength}(F)}.$$

Here, the self information of an MCCS is defined given the probability of its edges under the Maximum Entropy model, as:

$$\text{SelfInformation}(F) = -\sum_{\{i,j\}\in R} \sum_{\{k,l\}:k\in F\cap E_i, l\in F\cap E_j} \log(P_{ij}^{kl}(1)).$$

An MCCS is described most naturally by the set of entities it contains. More specifically, we choose to describe MCCS patterns by specifying for each entity whether it does or does not belong to the pattern. To specify that an entity belongs to an MCCS, we will use $-\log(p)$ bits, and to specify it does not belong to the MCCS we will use $-\log(1-p)$ bits, where p is a probability parameter. Such a code satisfies Kraft's inequality exactly, and is thus optimal and asymptotically achievable (Cover and Thomas 2005). Using this approach, the description length of an MCCS pattern F with $n = |F|$ entities and given that the graph of the MRD has $N = |E|$ entities is given by:

$$\begin{aligned}\text{DescriptionLength}(F) &= -\sum_{i\notin F}\log(1-p) - \sum_{i\in F}\log(p),\\ &= n\log\left(\tfrac{1-p}{p}\right) + N\log\left(\tfrac{1}{1-p}\right).\end{aligned}$$

In De Bie (2011b) it was suggested to set p by default to the density of the database (ratio of the number of relationship instances to the number of entities), an approach we adopted in our empirical results as well. However, the parameter can be tuned so as to bias the search more toward larger in number of nodes MCCSs (larger p) or toward smaller in number of nodes MCCSs (smaller p), if desired.

5 Related Work

Mining multi-relational data is a research topic that has been concerning the data mining community for a long time (Srikant and Agrawal 1996). Most previous methods on this topic are frequent pattern mining methods either based on Inductive Logic Programming (ILP) (Dehaspe and Toivonen 1999; Nijssen and Kok 2003; Garriga et al 2007) or generalizing ideas from frequent itemset mining to the relational setting (Ng et al 2002; Goethals et al 2010; Koopman and Siebes 2008; Cerf et al 2009; Ji et al 2006). We discuss these methods in Sec. 5.1.

Although we consider our method as a pattern mining method, since our MRD formalisation allows for a graph representation of the data, graph mining methods are related as well. We discuss these methods in Sec. 5.2. In the same section we also discuss recent work on networks with multiple types of nodes and interactions, which is only broadly related since the mining tasks considered are very different to the one in this paper. Finally in Sec. 5.3 we discuss related work that does not fall into the categories mentioned above.

5.1 Mining multi-relational databases

Well known ideas and algorithms from frequent itemset mining can be used for MRDs unaltered if applied on the join of all tables. The syntax of this type of patterns is essentially that of itemsets, with items in this case being attribute values and transactions being the tuples of the join table (Ng et al 2002; Goethals et al 2010; Koopman and Siebes 2008). The characteristic of this pattern syntax is that a tuple always contains one attribute value per attribute and as a result it is impossible to have two values of the same attribute in the same pattern. An itemset of this type for instance would not be able to capture the fact that a director can be related to many films. This is something that an MCCS pattern naturally captures. However, itemsets on the join table can still capture co-occurrences of attribute values that belong to different attributes.

On the other hand, the support, measured as the ratio of the tuples of the join table that contain an itemset, does not have a clear meaning as attribute values are replicated due to the join operation. A different approach is taken by Smurfig (Goethals et al 2010) where the support is measured with respect to every table, as the relative number of keys that the items correspond to.

Some previous methods have extended the notion of formal concepts (or closed itemsets and their supporting transactions) by considering entity types that are 3-ary related (Ji et al 2006; Jäschke et al 2008; Trabelsi et al 2012) or n-ary related with $n \geq 2$ (Cerf et al 2009; Voutsadakis 2002). They define a pattern as valid if all the entities it contains are related in the data, which is similar to our notion of completeness. They also define a pattern as closed if no additional entity can be added to it, which is similar to our notion of maximality. However, the main difference between these methods and our method is that they are designed to work on one 3-ary or n-ary relationship while our method is designed to work on multi-relational data with many binary relationships. The methods that work on one n-ary relationship could be

used for the problem of mining MCCSs in multi-relational data if this problem was reduced to the problem mining closed patterns from one n-ary relationship. This can be done by considering the join of every combination of relationships and all single relationships in the database, resulting in a number of n-ary relationships that is exponential in the number of relationships in the database. Methods for mining closed patterns in one n-ary relationship can then be applied to each of these relationships resulting in the same set of patterns that would be the result of applying RMiner to the original problem. However, when taking this reduction, the original structure of the data is lost which means that the MCCSs involving different subsets of entity types are produced independently instead of being built up from a smaller MCCS containing less entity types as in the case of RMiner. This results in these methods quickly becoming prohibitively costly. We empirically illustrate this in Sect. 7.3, with an experiment comparing RMiner and the algorithm of Cerf et al (2009).

Departing from the pattern syntax of itemsets a group of research suggested mining association rules of simple conjunctive queries, which are simple forms of relational algebra queries,i.e, a selection succeeded by a projection (Jen et al 2010; Goethals and Le Page 2008). This is an interesting pattern syntax as relational queries are in general more expressive than itemsets. However the patterns are still linked to a support measure which is computed on the join table. Interestingly, the work of Jen et al (2010), uses the functional dependencies of the attributes to prove anti-monotonicity of the support for this pattern syntax which is a way to make use of the relational data structure rather than just the join table.

Warmr (Dehaspe and Toivonen 1999) and Farmer (Nijssen and Kok 2003) are methods based on ILP. The patterns have the form of logic rules which can be regarded as local models of the database. The goal of these methods is to mine for the most frequent rules. The support is defined as the relative number of key values of one target table that satisfy the rule. Therefore the more general the rule the higher its support will be. This type of pattern syntax is very expressive and can capture the relational structure well. However, the objective of these methods (frequent rules about the data) is different than ours (interesting patterns of co-occurring attributes). Finally the interestingness measure we propose in Sec. 4 cannot be applied on Warmr and Farmer patterns and evaluating the interestingness of this kind of patterns is a challenge.

Within the ILP framework the work in (Garriga et al 2007) defined closure operations for patterns of the syntax of Warmr and Farmer and proposed an extension of the LCM algorithm (Uno et al 2004a), originally proposed for frequent closed itemset mining, for mining frequent rules. Although the purpose of a closure operator in this context is the same as in the context of MCCSs, i.e., extending patterns with valid sets of elements to reduce the search space, the semantics are different. In the ILP case, rules are extended with atoms such that the extended rule is satisfied by the same set of terms in the data, whereas in our case a CCS is extended with an entity such that the extended CCS has the same set of compatible entities.

Warmr, Farmer, and Smurfig are all based on the notion of a recurring pattern, and they directly depend on a support notion. Measuring the support with respect to one or a set of target tables, makes the results difficult to interpret and therefore introduces usability issues. The potential user will have to understand what exactly

it means for a recurring pattern to be frequent with respect to a certain target table. Additionally, these techniques are likely to suffer from the same problems as other frequent pattern mining techniques, in particular the fact that support is usually only weakly related to interestingness.

RDB-Krimp (Koopman and Siebes 2009) is a method for mining relational databases which is related to ours in that it also uses information theoretic ideas for the assessment of patterns. It uses the pattern syntax of Farmer (Nijssen and Kok 2003) but considers just patterns of depth two (patterns of a target table and all the tables related to it with a foreign key). The most frequent patterns of this kind are mined for every table of the database as a target table and then RDB-Krimp finds the most characteristic patterns among them using the MDL principle. The focus of this method is on the total description length of the database joined with the patterns, and patterns are deemed more interesting if they are better at compressing this description length. We instead deem patterns more interesting if they describe surprising aspects of the database in a concise way, which we argue makes our results more relevant to an end-user. Finally RDB-Krimp relies on heuristic search to find the optimal set of patterns that best compress the database which is not the case for our method that searches exhaustively.

A recent approach which acknowledges the usability issues of the support as well, is presented in (Nijssen et al 2011). The task of mining multi-relational databases is formalised as a constraint programming problem where a conjunction of constraints is defined and a general constraint programming solver is used to find sets of entities[3], satisfying these constraints. The syntactic constraint used in this paper is defined on one relationship and enforces the corresponding entities to form a biclique. A conjunction of biclique constraints for all the relationships in the database corresponds to the syntax of CCSs. Size constraints are defined as well, for the minimum/maximum number of entities required, which are of the same nature as the minimum coverage constraint we define in Sect. 4. Finally, a maximality constraint on an entity type, with respect to the rest of the constraints is defined. Although a conjunction of such constraints with respect to every entity type could be used in addition to the biclique constraint to find MCCSs, a useful definition of closure that takes into account the relational setting to increase the efficiency of finding such patterns is not given. In fact it is mentioned that multi-relational closed pattern mining is possible by applying the maximality constraint to one or more of the entity types. However, while maximal patterns (MCCSs) correspond to closed ones when the problem is reduced to itemset mining (see Sect. 2), this relation is unclear for the case of more than one relationships (see Sect. 3).

Finally, our method might seem related to that in (Zaki et al 2007) on mining clusters of attributes in an attribute-value table. Indeed the proposed approach of this paper is based on modelling an attribute-value table as a K-partite graph and mining maximal cliques in this graph. However the modelling is different to ours as all attributes of the table can be connected to each other, whereas according to our modelling of a single table, there are relationship types only between every attribute

[3] Please note that by entities and entity types here, we actually refer to our notion of the terms. The same notions are defined as objects and entities respectively in (Nijssen et al 2011).

and the primary key attribute. Also from an algorithmic point of view, the proposed algorithm of this paper for mining maximal K-cliques is based on enumerating all K-cliques without taking advantage of any notion of closure.

5.2 Mining graphs and networks

Mining all maximal cliques of a graph is an old problem (Bron and Kerbosch 1973) for which many algorithms exist (Pardalos and Xue 1994). Recently, remarkably efficient methods for enumerating directly only the maximal cliques have been proposed. Notably Makino and Uno (2004) introduced a method for maximal clique enumeration with time and space complexities comparable RMiner. The time delay is $O(M(n))$ where $M(n)$ is the time complexity for multiplying two $n \times n$ matrices which can be done in $O(n^{2.376})$ time. The space complexity of this algorithm is $O(n^2)$ which corresponds to storing the two $n \times n$ matrices being multiplied, i.e. equal to the space complexity of RMiner.[4] Methods for (maximal) clique enumeration could therefore be worth investigating for use in mining MCCSs.

Clearly, MCCSs are not cliques in the graph representation of the database, given that some edges are forbidden (between entities of the same type, and between entities of types that are not related). However, one can try to reduce the problem of mining all MCCSs to the problem of mining of all (maximal) cliques by adding edges wherever these are forbidden. Let us call the resulting graph the auxiliary graph (note that this graph is typically going to be dense). It is straightforward to see that in the auxiliary graph each MCCS is indeed a clique, such that a clique enumeration method applied to the auxiliary graph would indeed also enumerate all MCCSs in the database.

However, this reduction is inefficient, as can be understood most by means of an example. Consider a multi-relational database with five entity types $E_1 \dots E_5$ and the following relationship types: $R_{12}, R_{23}, R_{34}, R_{45}$. Let us assume that there are n_1 MCCSs with entities from types E_1 and E_2 only, and n_2 MCCSs with entities from types E_4 and E_5 only. Now, note that in the auxiliary graph all entities of type E_1 and E_2 are connected to all entities of type E_4 and E_5. This means that the union of any MCCS over types E_1 and E_2 with an MCCS of type E_4 and E_5 will be a (maximal) clique in the auxiliary graph. Thus, these n_1+n_2 MCCSs give rise to $n_1 \times n_2$ maximal cliques, a quadratic blowup. In general the blowup is polynomial: depending on the number of entity types and the relationships between them, a maximal clique in the auxiliary graph may be the union of a larger number of MCCSs.

Another well known graph mining task which could be seen as related is frequent subgraph mining (Yan and Han 2002; Kuramochi and Karypis 2001). Given a database of many graphs and a support threshold, the goal of frequent subgraph mining is to mine all subgraphs that occur more often than the support threshold suggests.

[4] Note that practically, the quadratic space complexity of RMiner results from multiplying a linear space complexity with the maximal search tree depth, which, as we will show in Sect. 7.3, is practically a small constant. Also, as we discussed in Sect. 3.5, the practical time delay of RMiner depends on the density of the data set and can be optimised in practice by taking particular implementation choices. In Sect. 7.3 we show experiments where the total running time is linear in log scale with respect to the input size when constraints are used. Thus, even though the *theoretical* complexities of Makino and Uno (2004) and RMiner are comparable, RMiner probably scales better *in practice*.

A closed subgraph in this setting is a subgraph for which no proper supergraph with the same support exists in the database (Yan and Han 2003). The task of frequent subgraph mining is therefore very different to the one proposed in our paper where the goal is to mine maximal complete connected components in one K-partite graph.

Recent work from the networks community has focused on networks which contain different types of nodes or interactions. Although these networks are mostly referred to as heterogeneous networks (Sun et al 2009; Ji et al 2010, 2011; Sun et al 2012b,a; Tang et al 2012; Maruhashi et al 2011), this name does not signify a single structure. In fact, different structures under this name have been studied and different mining tasks on them considered.

A group of papers consider heterogeneous information networks that correspond to a star entity-relationship schema. The mining tasks that have been proposed for this type of networks are clustering, classification or link prediction. Two different clustering tasks have been proposed: clustering of the nodes that correspond to the middle entity using the nodes of the other entities as features (Sun et al 2009), or clustering of any entity (target entity) using the nodes of one another entity, connected through a path to the target entity, as features (Sun et al 2012b). For the classification task it is assumed that certain subgraphs of the network have a particular class tag. Then the goal is to find the confidence with which untagged nodes belong to a certain class using the number of edges of different relationship types that the node has with every class (Ji et al 2010). The same framework is used in combination with ranking, where nodes within a class are ranked based on their importance and this is used to give different weights to the different edges of unlabelled nodes with the class. As more and more nodes are assigned to classes the ranking is updated (Ji et al 2011). The link prediction task corresponds to predicting links of a target relationship type based on topological features of the network (Sun et al 2012a). Since our method is an unsupervised one and does not do any prediction, it is mostly related to clustering. However the clustering methods proposed do not employ the particular structure of the network but to define which is the entity to be clustered and what are the features used for clustering. After this, clustering algorithms for unstructured data are used. This approach also results in one-dimensional clusters. Instead, our method uses the K-partite structure of the network to produce multi-dimensional clusters of nodes.

A different type of heterogeneous network where there is one type of node and multiple types of edges/interactions, is considered in (Tang et al 2012). The proposed method finds cross-dimensional communities of nodes by integrating topological features from the different dimensions. The topological features correspond to the top eigenvectors of the modularity matrix and the cross-dimensional communities are found by maximising the summation of all the pair-wise correlations of the features. Although this task is very different to the one considered in our paper, it is an interesting approach as the communities found contain information from all the different dimensions of the data.

Finally, tensors are considered as heterogeneous networks in (Maruhashi et al 2011), where a tensor decomposition method is employed to mine patterns of entities of one dimension that share one entity from each of the other dimensions or patterns that correspond to bi-partite graphs in a two dimensional slice of the tensor.

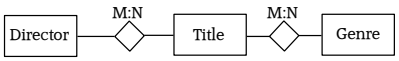


Fig. 5 Entity-Relationship diagram of the *imdb-3ent-1year* and *imdb-3ent-10years* datasets.

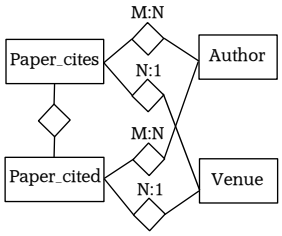


Fig. 6 Entity-Relationship diagram of the *dblp* dataset.

5.3 Other work

An approach for assessing the statistical significance of relational (SQL) queries based on randomisations of different tables is proposed in (Ojala et al 2010). Although this approach was not intended to propose a method to mine such patterns it provides an insight towards making relational patterns useful to the user.

Finally the idea of representing relational databases as graphs has also appeared in the Database research community with the Graph Database Models, but the focus was of course representation and querying which led to more complex structures (directed graphs often representing a hierarchical structure of entity types) (Angles and Gutierrez 2008).

6 Qualitative Evaluation

In this section we show and discuss the top ranked patterns of our method on different real world datasets in order to highlight how our method is useful in different real world scenarios. Moreover we qualitatively compare the patterns mined by RMiner with those of two previous methods (namely Farmer (Nijssen and Kok 2003) and Smurfig (Goethals et al 2010)) on the same dataset.

6.1 Real-world datasets

We did experiments on several real world datasets, the Entity-Relationship (E-R) diagrams of which are shown in Figures 5, 6 and 7 and their statistics are summarised in Table 1. We produced two different views of the IMDB data base [5], one containing the titles, genres and directors of the films produced in 2010 (*imdb-3ent-1year*) and one containing the titles, genres and directors of films produced in the years 2001-2010 (*imdb-3ent-10years*). To produce the *imdb-3ent-10years* dataset we neglected

[5] See `http://www.imdb.com/`

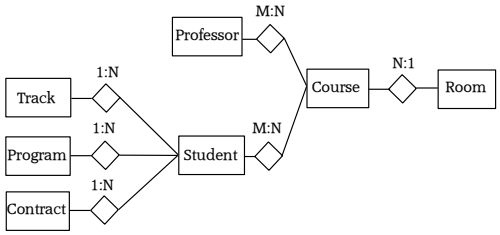


Fig. 7 Entity-Relationship diagram of the *studentdb* dataset.

Table 1 Database details. For every data set we show the total number of entities and the relationship density for every relationship type, which corresponds to the number of relationship instances divided by the number of entities involved in this relationship type.

	Num. of Entities	Rel. density Type 1	Rel. density Type 2	Rel. density Type 3	Rel. density Type 4	Rel. density Type 5	Rel. density Type 6
imdb-3ent-1year	30,130	7×10^{-5}	0.073				
imdb-3ent-10years	104,291	2.5×10^{-5}	0.058				
dblp	15,510	4×10^{-4}	4×10^{-4}	0.002	0.031	0.031	
studentdb	401	0.016	0.143	0.333	0.333	0.109	0.011

all "Short" films. From the Dblp bibliography database [6] we created the *dblp* dataset which only contains papers with citation information (4947 papers). Please note that not all papers contained their citations in the data we downloaded from Dblp. Finally we also used the student database of the Computer Science department of the University of Antwerp (Goethals et al 2010) (called *studentdb* in this paper).

6.2 Patterns of RMiner

Table 2 Output size and computation times of RMiner and RMiner_icdm.

	Constraints per entity type	Num. of Patterns	Time(sec) RMiner	Time(sec) RMiner_icdm
imdb-3ent-10years	(1,1,1)	54,672	577	2,280
dblp	(0,0,0,0)	26,377	3,609	39,614
studentdb	(1,1,1,1,1,1,1)	155	1	2

Table 2 summarizes all the different experiments we did, by showing the constraints we used, as well as the computation time and output size for each of them. A preliminary version of RMiner, which was not applying the divide and conquer

[6] See `http://www.informatik.uni-trier.de/~ley/db/`

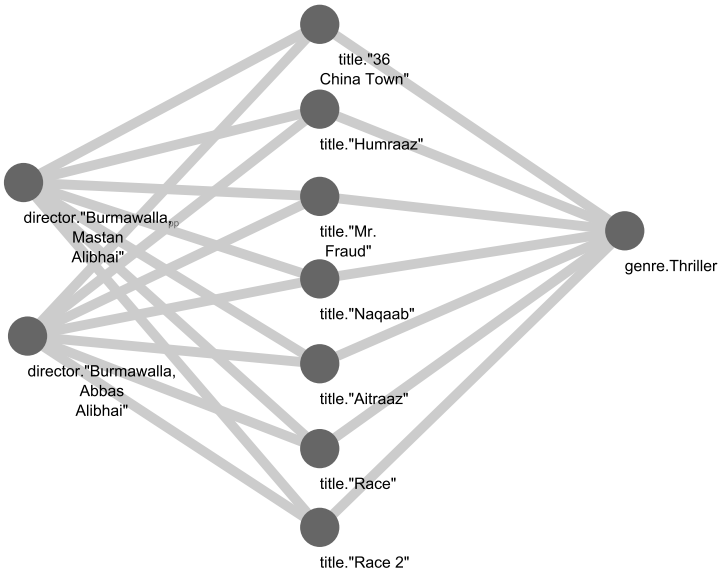


Fig. 8 The 1^{st} most interesting MCCS pattern in the *imdb-3ent-10years* dataset, as a graph. It represents a set of film titles that are all directed by the same two directors and are of genre "Thiller".

fixpoint listing framework but was rather an ad hoc algorithm not taking advantage of a full closure, was presented in (Spyropoulou and De Bie 2011). We extensively compare RMiner with this algorithm, which we refer to as RMiner_icdm, in Sect. 7. However, for completeness, we report computation times for both the RMiner and RMiner_icdm, here as well. We now analyse the top ranked patterns in every dataset.

IMDB Database We run RMiner on the *imdb-3ent-10years* dataset with constraints of at least one entity per entity type. The results are shown in Figs. 8, 9 and 10 which represent the 1^{st}, 2^{nd} and 3^{rd} most interesting patterns respectively, ranked by the interestingness measure presented in Section 4. They all have two directors (pairs of brothers) and six or seven films which makes them interesting as they carry a lot of information (number of edges), given that on average in this dataset a director directs 1.51 films and a film has 1.04 directors, and this information is conveyed in a concise way (number of nodes). The 1^{st} pattern (Fig. 8), contains the genre "Thiller" which is two times less probable to be connected to a film than that the genre "Comedy" and therefore it ranks higher than the other two because it contains more improbable edges given the prior information of the user. The 2^{nd} pattern (Fig. 9) ranks higher than the 3^{rd} (Fig. 10) because it contains more edges and therefore conveys more information. The fact that the directors involved in all three of these patterns have directed very few films in the dataset makes the edges of the patterns very improbable under the user's prior information and leaves the pattern involving the "Coen brothers" and all their "Comedy" films in the years 2001-2010 in the 22nd place.

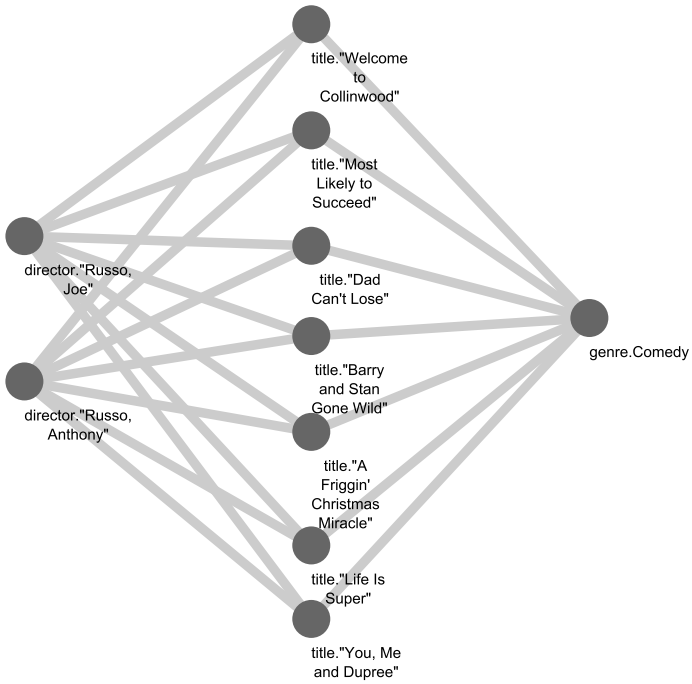


Fig. 9 The 2^{nd} most interesting MCCS pattern in the *imdb-3ent-10years* dataset, as a graph. It represents a set of film titles that are all directed by the same two directors and are of genre "Comedy".

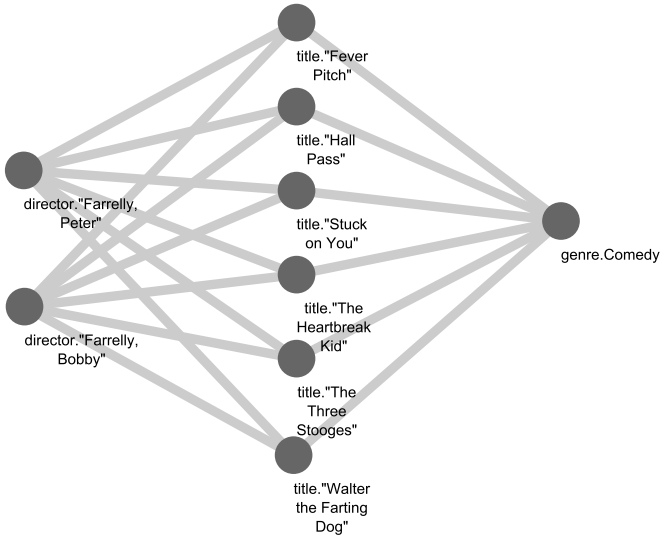


Fig. 10 The 3^{rd} most interesting MCCS pattern in the *imdb-3ent-10years* dataset, as a graph. It represents a set of film titles that are all directed by the same two directors and are of genre "Comedy".

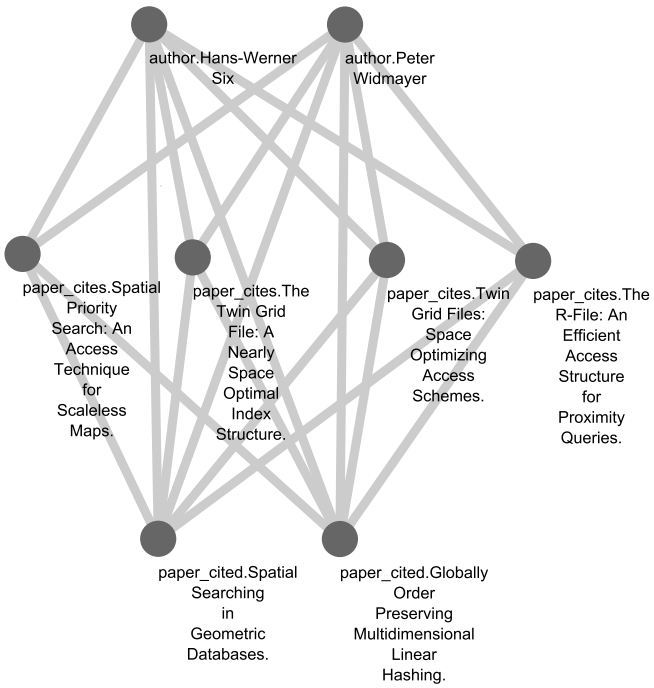


Fig. 11 The 1^{st} most interesting MCCS pattern in the *dblp* dataset, as a graph. It represents a group of authors, a group of papers they published and a group of papers that are self citations, indicating that these are papers these authors published on the same idea.

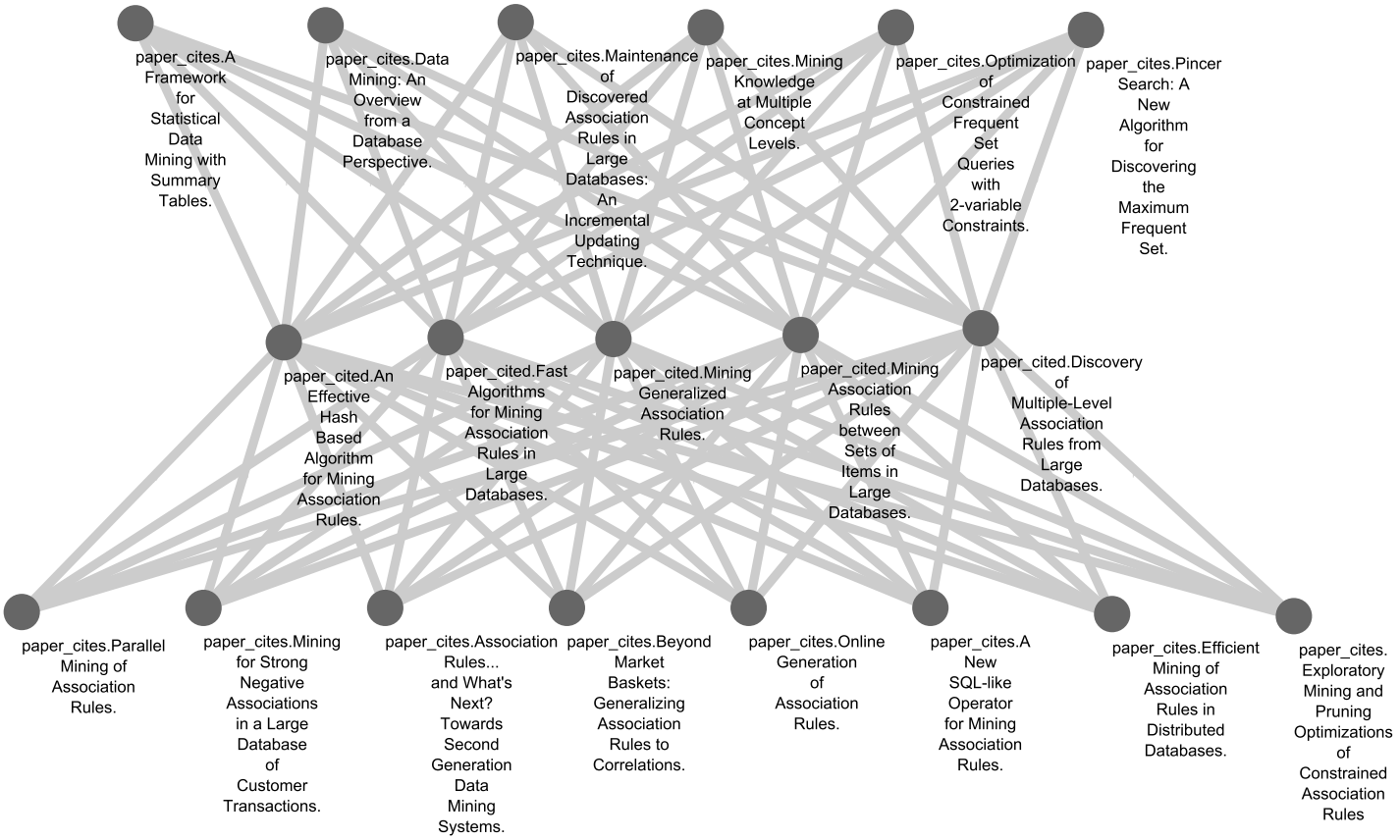


Fig. 12 The 6^{th} most interesting MCCS pattern in the *dblp* dataset, as a graph. It represents a group of papers all of them citing another group of papers, indicating that they are all papers on the same subject.

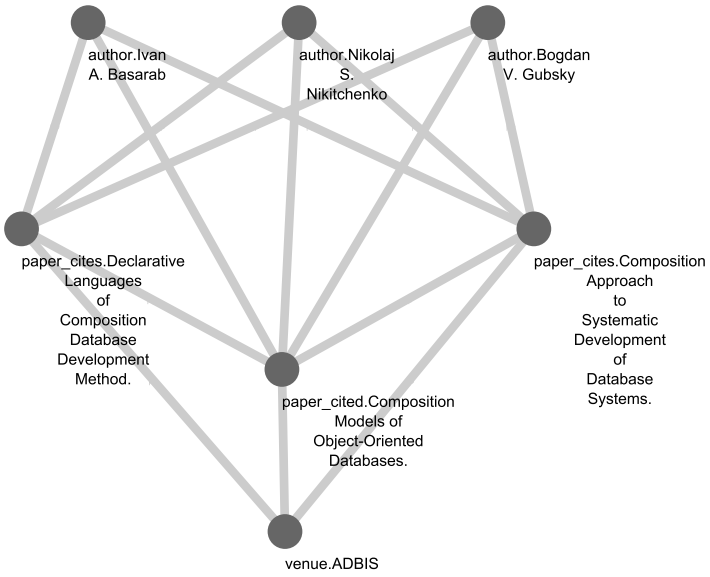


Fig. 13 The 8^{th} most interesting MCCS pattern in the *dblp* dataset, as a graph. It represents a group of authors, a group of papers they published and a group of papers that are self citations. All papers are published in the same conference. This pattern indicates that these are papers that these authors published on the same idea and at the same conference.

DBLP Database The results of this dataset are shown in Figs. 11, 12 and 13. To highlight the generality of our method we show the 1st, 6th and 8th most interesting patterns as they are quite diverse in the kind of information they represent. The 1^{st} pattern ranks high in terms of interestingness as it conveys a lot of information (number of edges which are improbable under the user's prior information) in a concise way (number of nodes). The edges involved in this pattern are improbable if one considers that the authors involved have written 7 and 11 papers respectively in this dataset and that the cited papers involved are both cited by only 7 citing papers in this dataset. The 6^{th} pattern, although it is very big in the number of nodes it ranks high in terms of interestingness as the amount of information it conveys makes up for it. More specifically the cited papers in the pattern get cited by 14 citing papers when the average in the dataset is 6.2. Finally the 8^{th} pattern contains a lot fewer edges than the other two patterns (for example it contains half the amount of edges of the 1^{st} one although it has only one node less) however, it still ranks high as the edges it contains are very improbable give the prior information of the user.

Student Database dataset The top-ranked MCCSs on the *studentdb* database are shown in Figs. 14 and 15. Since the first two patterns were structurally similar (although they convey non-redundant information), Figs. 14 and 15 show only the 1^{st} and the 3^{rd} most interesting patterns. The 1^{st} ranked pattern (Fig. 14) is interesting as it conveys a lot of information (number of edges) in a concise way (number of nodes). The 3^{rd} pattern (Fig. 15), is less interesting than the 1^{st} as it contains just 1

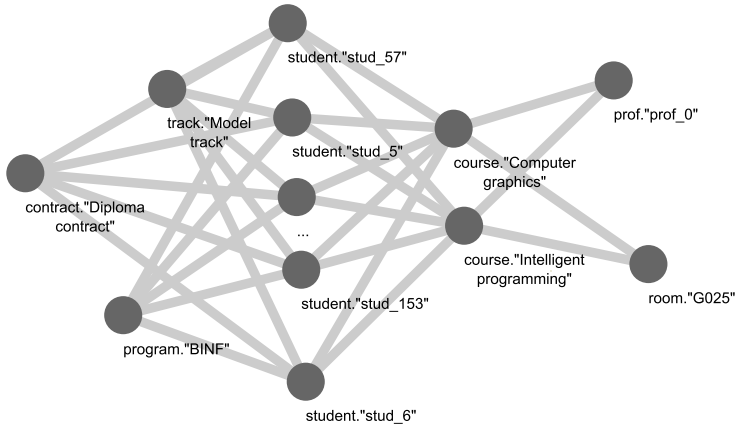


Fig. 14 The first most interesting MCCS pattern in the *studentdb* dataset, as a graph. It conveys information about a set of 67 students, the program, contract, the track they are following, two courses they attend, the professor teaching these courses and the lecture room they are taught in. Note that the number of student nodes is too large to show here, so we collapsed them onto one node labelled with "...".

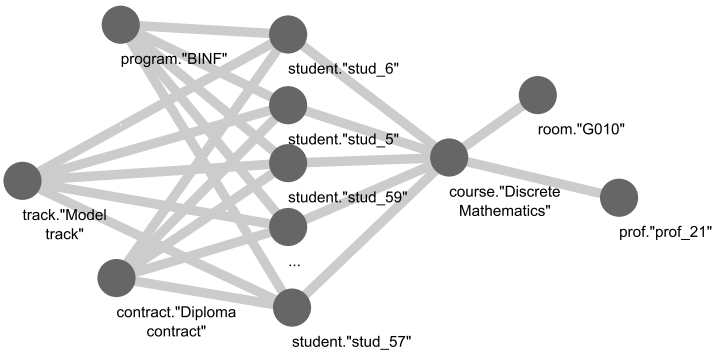


Fig. 15 The third most interesting MCCS patterns in the *studentdb* dataset, as a graph. It conveys information about a set of 67 students, their program, contract, and track, as well as one course, the professor teaching it and the lecture room. Note that the number of student nodes is too large to show here, so we collapsed them onto one node labelled with an "...".

node less while it explains 67 fewer edges and contains one more course room which appears more frequently in the database.

```
(genre=Action) (name="Appleford, Russell") (.title="40 Years")
(genre=Action) (name="Wagner, Ben") (title="6:00")
(genre=Action) (name="Gao, Fawn") (title="9 AM")
```

Fig. 16 Three of the most frequent patterns of Smurfig on *imdb-3ent-1years* database.

```
director(V0N0), directs(V0N0,V1N0), genre(V1N0,Drama)
director(V0N0), directs(V0N0,V1N0), genre(V1N0,Short)
director(V0N0), directs(V0N0,V1N0), genre(V1N0,Comedy)
```

Fig. 17 Top three most frequent patterns of Farmer on *imdb-3ent-1years* database.

6.3 Comparison with other methods

Here we qualitatively compare with the results of Smurfig (Goethals et al 2010) and Farmer (Nijssen and Kok 2003) on the *imdb-3ent-1year* dataset.

Smurfig patterns We ran Smurfig with a support threshold of 0.001 to be as inclusive as possible. To compare with the patterns of RMiner we selected the ones that contain items from all the three attributes. As pointed out in Sec. 5, each of these patterns can contain only one attribute value per attribute. Because of the nature of the *imdb-3ent-1year* dataset each of them has absolute support of 1. Figure 16 shows three of these patterns. Thus, Smurfig is clearly not suited to find relations in relational data of this kind.

Farmer patterns We ran Farmer with an absolute support threshold of 1. The pattern syntax we used had the following form: $director(X)$, $directs(X, Y)$, $genre(Y, g_1) \ldots genre(Y, g_n)$ and the key of the search is the atom $director(X)$. Figure 17 shows the top three most frequent of these patterns that contain all three predicates. None of these patterns contain more than one genre constants, which is to be expected as the most frequent rules are bound to be the more general rules. Note that if we found the directors and titles that satisfy these rules, these patterns would correspond to CCSs. The difference between Farmer patterns and CCSs is analogous to the difference between itemsets and tiles. Farmer patterns corresponding to MCCSs are expected to be less frequent as they are more specific.

7 Quantitative Evaluation

In this section we present a quantitative evaluation of our method. We first show that our interestingness measure indeed ranks high the most interesting patterns. Then we present how RMiner behaves in term of computation time on artificial data of different schemas. Finally we present a scalability study of RMiner on real-world data of increasing number of entities.

When studying the performance of RMiner we always compare it to its predecessor RMiner_icdm (Spyropoulou and De Bie 2011). As discussed in Sect. 6.3, the patterns mined by RMiner are qualitatively different than the ones of Farmer (Nijssen and Kok 2003). Therefore a direct comparison of the performance of the two algorithms would not be fair, since the two tasks are very different.

7.1 Evaluation of the Interestingness Measure

We investigated how different methods detect artificially embedded MCCS patterns of different sizes in the *imdb-3ent-1year* data. More specifically, we investigated how highly the embedded MCCS (or a larger MCCS containing it) is ranked by our method using the interestingness measure we propose. To compare with Farmer we checked the rank of the most frequent rule corresponding to this MCCS and, allowing Farmer an advantage due to the different pattern syntax, also the rank of any CCS containing a small *subset* of the embedded predicates.

To artificially embed a pattern, we added k genres, k directors, and k titles to the database, in such a way that each of these k genres and directors are connected to each of the k titles, forming a CCS. As this by itself would create an unrealistic disjoint part of the database, we additionally added random links preserving the overall connectivity and database statistics. E.g., we randomly added links between the existing genres and the newly added titles so as to ensure that, in expectation, the total fraction of titles each of the existing genres is linked with stays the same. This is done also between the existing titles and the newly added genres, and similarly for the directors and titles.

Table 3 shows the rank of the embedded MCCS pattern for increasing k. RMiner ranks the embedded pattern higher as the number of nodes per entity type increases and ranks it first when it contains more than just three nodes, showing that RMiner ranks high even relatively small patterns known to be present in the database.

For Farmer we used the same pattern syntax as in Sec. 6.3. Table 3 shows the rank of the highest ranked rule including all *genre* predicates in the embedded MCCS, as well as corresponding to a CCS containing a subset of just two or more of the embedded *genre* predicates. Unsurprisingly, Farmer ranks the CCS patterns more highly than the more specific and thus less frequent MCCS patterns. However, even the CCS patterns are ranked much lower than using RMiner.

Table 3 Rank of artificially embedded MCCS pattern in *imdb-3ent-1year* dataset with increasing number of nodes k per entity type.

k	2	3	4	6
RMiner Rank	103	6	1	1
Farmer Rank (MCCS)	121	502	1464	2141
Farmer Rank (CCS)	121	109	125	147

7.2 Computational Evaluation on different schemas

This subsection aims at showing how RMiner behaves on different Entity-Relationship (E-R) diagrams. We also compare RMiner with its predecessor RMiner_icdm (Spyropoulou and De Bie 2011). We produced random datasets based on the three E-R diagrams depicted in Figure 18 in the following way. For every entity type we fixed

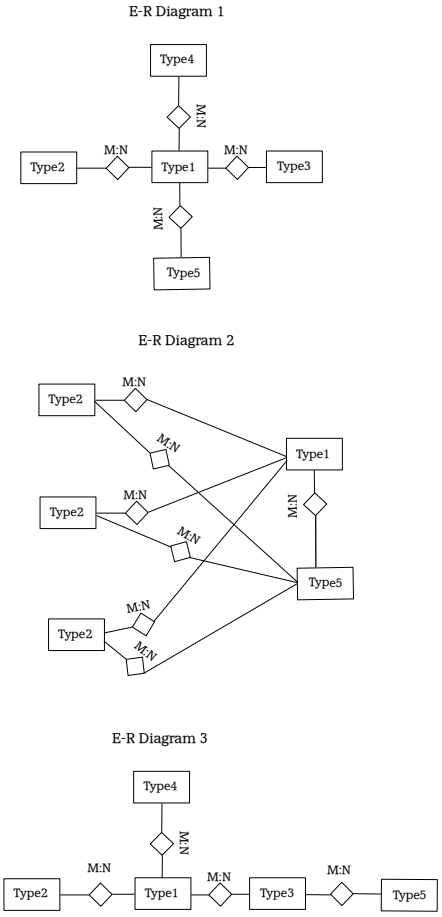


Fig. 18 Entity-Relationship diagrams of random datasets.

the number of entities to 500. Then for every relationship type we connect an entity of one entity type with an entity of the other with varying probability called *connection probability*.

More specifically we created 10 datasets for every E-R diagram by varying the *connection probability* in the interval $[0.001, 0.01]$ and computed the running time of RMiner and RMiner_icdm for each of the datasets without any constraints. Figure 19 shows a comparison of the running time for increasing *connection probability*. The running time of both algorithms increases exponentially as the data becomes more dense. Moreover RMiner is only marginally faster than RMiner_icdm in datasets with E-R 1 and 3 and as fast in datasets with E-R 2. We argue that this is due to the fact that random datasets are quite unstructured. This means that they are less likely to contain entities with shared connected entities which increases the number of closed patterns that are enumerated. As we show in Table 2 and we will see in the next section RMiner always outperforms RMiner_icdm on real world datasets.

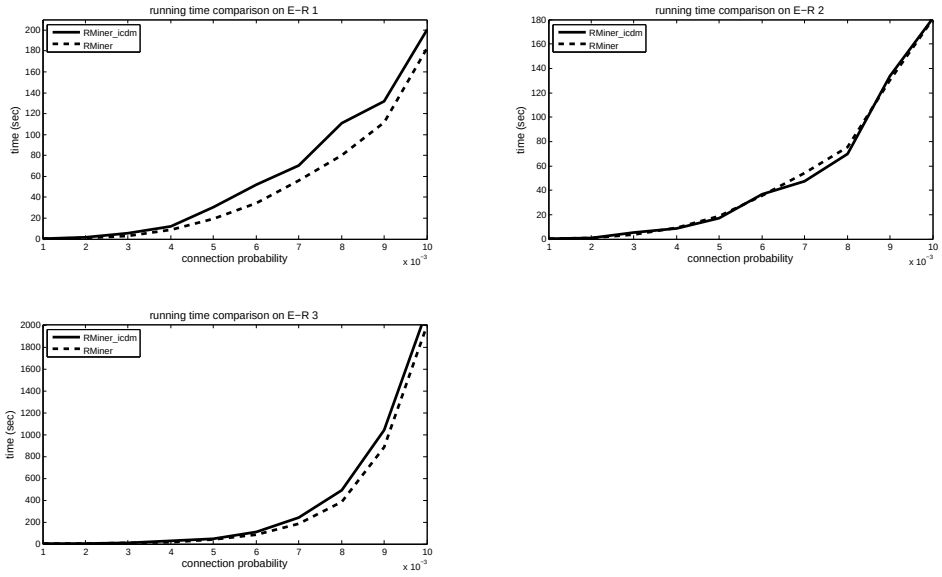


Fig. 19 Comparison of RMiner with RMiner_icdm on different E-R diagrams.

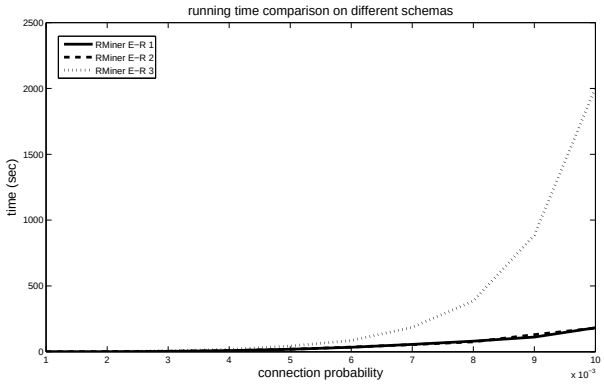


Fig. 20 Comparison of the running time of RMiner on different E-R diagrams.

Figure 20 shows a comparison of the running time of RMiner and Fig. 21 a comparison of the output size for increasing *connection probability* on datasets with different E-R diagrams. The output size as well as the computation time increases as the paths connecting the entity types become longer.

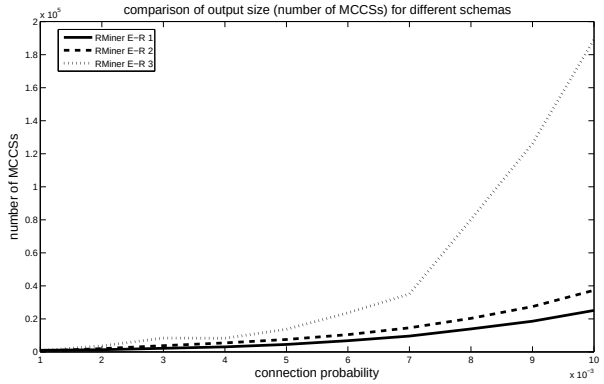


Fig. 21 Comparison of the output size of RMiner on different E-R diagrams.

Table 4 Scalability Experiment.

	RMiner						RMiner_icdm
Constraints	Num. of Entities	Num. of Patterns	Time (sec)	Space (Mb)	Depth	Size of max MCCS	Time (sec)
(0,0,0)	3,291	583	5,413	5	5	2823	-
	8,686	1,134	113,185	11	6	7872	-
	51,203	-	-	-	-	-	-
	111,320	-	-	-	-	-	-
	514,323	-	-	-	-	-	-
(1,1,1)	3,291	491	1.65	5	5	100	20
	8,686	980	12	10	6	174	215
	51,203	7621	610	51	6	360	5,471
	111,320	32,213	2,813	109	8	632	73,944
	514,323	253,148	34,758	477	8	632	96,738
(2,2,2)	3,291	3	0.18	5	4	10	549
	8,686	23	1.35	10	4	11	8,562
	51,203	125	30	50	6	80	-
	111,320	420	181	107	7	148	-
	514,323	1286	2598	461	7	179	-
(3,3,3)	3,291	0	0.14	4	2	0	1010
	8,686	0	1.02	10	2	0	21,938
	51,203	0	18	50	2	0	-
	111,320	1	121	107	4	11	-
	514,323	5	1506	461	7	11	-

7.3 Computational scalability evaluation

The purpose of this subsection is to show how RMiner scales with increasing number of entities in the dataset and using different constraints. In order to take full advantage of the closure operator we did this scalability study on real world datasets of increasing size corresponding to the schema of Fig. 5 of the IMDB Database. More specifically we took snapshots corresponding to 1 year, 2 years, 10 years, 40 years and 100 years of films starting from year 1910. Table 4 shows the number of entities

corresponding to each of these snapshots as well as number of patterns, the time, memory, maximum depth of the search tree and size of the maximum clique we get from running RMiner on each of these datasets. It also shows the running time of RMiner_icdm on each of these datasets. The ”-”s in the table mean that the respective algorithm did not finish running in 2 days.

From Table 4 we can see that when not using any constraints per entity type the running time of RMiner increases rapidly with the number of entities, while when using the constraints it appears to increase linearly in log scale, with a slope approximately equal to 2. When using constraints of even one entity per entity type we can run RMiner in almost the whole of IMDB Database (100 years of films) in a few hours. This time reduces to a few minutes when using constraints of at least two entities per entity type. This is a very useful feature of RMiner as truly relational patterns are the ones that involve more than two entity types. The fact that the running time scales poorly when not using any constraints is expected since the task is strictly harder than running frequent title mining on each of the relationships separately, with a support threshold of zero.

The space used at run time increases linearly with the number of entities irrespective of the constraints. This is due to the fact that the space complexity of the algorithm depends on the maximum depth of the search tree (see Sec. 3.5) which, in these experiments, is a small constant. Even when increasing the input size by two orders of magnitude the maximum search tree depth only increases by approximately factor of 3 in the worst case, i.e, when the constraints are (3,3,3). Also, the fact that the maximum depth of the search tree is never greater than seven, means that the number of closed CCSs is only up to a small factor larger than the number of MCCSs.

Table 4 also shows the size of the maximum MCCS which gets smaller as the constraints increase. However, the size of the maximum MCCS also corresponds to the maximum depth of the search tree if the closure operator was not used. Comparing this with the maximum depth of the search tree of RMiner, we can get an idea about the effectiveness of the closure operator or the compacity of the closed CCSs. As we can see in Table 4, the maximum depth of the search tree of RMiner is up to three orders of magnitude smaller than the size of the maximum MCCS.

Comparing RMiner with RMiner_icdm we see that RMiner always outperforms RMiner_icdm by up to three orders of magnitude, like in the case of constraints of three entities per entity type and the dataset of 8,686 entities.

Finally, in Sect. 5.1 we described how the problem of mining MCCSs could be reduced to the problem of mining closed patterns in n-ary relations by taking the joins of all allowed combinations of relationships and mining n-ary closed patterns on each one of them. Here we show empirically that this reduction doesn't scale for large datasets, by comparing RMiner with DataPeeler (Cerf et al 2009). More specifically, we compared the running time of RMiner for the case of (1,1,1) constraints with that of DataPeeler on the join of all relationships (not shown in the Table 4). DataPeeler run to completion for the first three datasets with running times 2.36s, 22.41s, 15,492.6s and after this point it did not run to completion within 2 days.

7.4 Practical guidance on using RMiner

In Sect. 3.5 we analysed the complexity of the delay between producing two CCSs and found it to be cubic to the input size. We also argued that in practice the delay is a product of the input size and a factor related to the density of the dataset. However, the total running time depends on the output size which can be very large in some cases. As an example, consider the simplest database containing two entity types, E_1 and E_2 and a binary relationship $\mathcal{R}_{12}$ between them. Let's assume that each of the entity types contains n entities, and that the binary relationship is as follows: $\mathcal{R}_{1,2} = \{\{e_i, e_j\} : i \neq j\}$. For this construction, each of the 2^n subsets of E_1 together with its neighbors in E_2 forms a closed CCS. This is the worst case for this particular database where the number of closed CCSs (and thus MCCSs) grows exponentially to the input size.

However, as with all local pattern mining methods, when dealing with data sets where the actual relationships can be arbitrary, the size of the output is not known until the algorithm runs. Nevertheless, based on our empirical scalability analysis (Sect. 7.2 and 7.3), we list a few factors that the output size and as a result the total running time, depend on:

- The input size (number of entities): the output size scales polynomially to the input size (Table 4).
- The density of the data: the output size of RMiner scales exponentially to the data density (Fig. 21).
- The database schema: The longer paths the schema contains the larger the output size (Fig. 21).

Given the uncertainty about the output size, we recommend increasing it progressively by making use of the constraints. More specifically, one could start running RMiner using high values for the constraints (i.e., producing a small output) and continue by reducing them until the point when RMiner still runs within an acceptable amount of time.

8 Discussion and Future Work

Multi-relational data mining is a very promising field as it suggests mining for more complex patterns than we were able to with frequent itemset mining or frequent subgraph mining. A main challenge in this field is the definition of an appropriate and intuitive pattern syntax in such complex data, and we feel that the notion of a Complete Connected Subset (CCS) as proposed in the current paper is promising due to its conceptual simplicity, while fully honouring the relational nature of the data. We further confirmed the promise of CCSs as a pattern syntax by developing an efficient algorithm for mining all maximal CCSs, and by validating the ideas on a number of artificial and real-life databases.

We see several opportunities for further research. On the algorithmic side, other approaches for exploring set systems can be worth investigating. As an example, the algorithm for enumerating all maximal independent sets of an independence system

proposed in (Lawler et al 1980) seems potentially relevant. If this algorithm can be adapted for the enumeration of all maximal sets in the intersection of an independence and a strongly accessible system, then it would be possible to directly enumerate all MCCSs. A different algorithmic direction could be taken if one considered a different compression of the CCS in the same lines as for example the non-derivable itemsets (Calders and Goethals 2007). For applying this idea to the case of CCSs, the semantics of this reduction should be investigated and a suitable algorithm should be developed.

In terms of pattern syntax, MCCSs as defined in this paper allow only for binary relationships between the entity types. However, in practical relational databases relationships can be of any arity. A new pattern syntax considering relationships of any arity, would add to the generality of our method. Furthermore the definition of itemsets and the respective mining algorithms have been extended in order to be fault-tolerant in the case of noisy data (Poernomo and Gopalkrishnan 2009; Gupta et al 2008). This is important when dealing with real world data, especially experimental, and would constitute a useful extension of our work.

Finally the maximum entropy framework used in this paper for modelling the subjective interestingness of patterns allows for incorporating this work into an iterative data mining framework where the interestingness of a pattern is quantified based on the patterns that the user has already seen (De Bie 2011a).

Acknowledgements We are grateful to Michael Mampaey for providing the Smurfig code and data and for his support in using Smurfig, Siegfried Nijssen for his assistance in using Farmer and Thomas Gärtner for discussions on this work. This work was partially funded by PASCAL 2 Network of Excellence. Eirini Spyropoulou and Tijl De Bie are supported by EPSRC grant EP/G056447/1. Mario Boley is partially funded by DFG (German National Research Foundation) under GA 1615/2-1.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), pp 487–499

Angles R, Gutierrez C (2008) Survey of graph database models. ACM Comput Surv 40(1):1:1–1:39

Birkhoff G (1967) Lattice Theory. American Mathematical Society

Boley M (2011) The efficient discovery of interesting closed pattern collections. PhD thesis

Boley M, Horvath T, Poigné A, Wrobel S (2010) Listing closed sets of strongly accessible set systems with applications to data mining. Theoretical Computer Science 411(3):691 – 700

Bron C, Kerbosch J (1973) Algorithm 457: finding all cliques of an undirected graph. Commun ACM 16(9):575–577

Burdick D, Calimlim M, Flannick J, Gehrke J, Yiu T (2005) Mafia: a maximal frequent itemset algorithm. Knowledge and Data Engineering, IEEE Transactions on 17(11):1490 – 1504

Calders T, Goethals B (2007) Non-derivable itemset mining. Data Mining and Knowledge Discovery 14(1):171–206

Cerf L, Besson J, Robardet C, Boulicaut JF (2009) Closed patterns meet n-ary relations. ACM Trans Knowl Discov Data 3(1):3:1–3:36

Cover TM, Thomas JA (2005) Elements of Information Theory. Wiley

De Bie T (2011a) An information theoretic framework for data mining. In: Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), pp 564–572

De Bie T (2011b) Maximum entropy models and subjective interestingness: an application to tiles in binary databases. Data Mining and Knowledge Discovery 23(3):407–446

De Bie T, Kontonasios KN, Spyropoulou E (2010) A framework for mining interesting pattern sets. SIGKDD Explorations pp 92 – 100

De Raedt L, Zimmermann A (2007) Constraint-based pattern set mining. In: Proceedings of the SIAM International Conference on Data Mining (SDM), pp 237–248

Dehaspe L, Toivonen H (1999) Discovery of frequent datalog patterns. Data Mining Knowledge Discovery 3:7–36

Elmasri R, Navathe SB (2006) Fundamentals of Database Systems. Addison Wesley

Garriga GC, Khardon R, De Raedt L (2007) On mining closed sets in multi-relational data. In: Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI), pp 804–809

Geerts F, Goethals B, Mielikainen T (2004) Tiling databases. In: Proceedings of Discovery Science, pp 278–289

Geng L, Hamilton HJ (2006) Interestingness measures for data mining: A survey. ACM Computing Surveys 38

Gionis A, Mannila H, Mielikinen T, Tsaparas P (2007) Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data 1(3)

Goethals B, Le Page W (2008) Mining association rules of simple conjunctive queries. In: Proceedings of the SIAM International Conference on Data Mining (SDM)

Goethals B, Page WL, Mampaey M (2010) Mining interesting sets and rules in relational databases. In: Proceedings of the ACM Symposium on Applied Computing (SAC), pp 997–1001

Gupta R, Fang G, Field B, Steinbach M, Kumar V (2008) Quantitative evaluation of approximate frequent pattern mining algorithms. In: Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining (KDD), pp 301–309

Hanhijarvi S, Ojala M, Vuokko N, Puolamaki K, Tatti N, Mannila H (2009) Tell me something i don't know: randomization strategies for iterative data mining. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM, pp 379–388

Jäschke R, Hotho A, Schmitz C, Ganter B, Stumme G (2008) Discovering shared conceptualizations in folksonomies. Web Semantics: Science, Services and Agents on the World Wide Web 6(1):38–53

Jen TY, Laurent D, Spyratos N (2010) Computing supports of conjunctive queries on relational tables with functional dependencies. Fundam Inform 99(3):263–292

Ji L, Tan KL, Tung AKH (2006) Mining frequent closed cubes in 3d datasets. In: Proceedings of the international conference on Very large data bases, VLDB Endowment, VLDB, pp 811–822

Ji M, Sun Y, Danilevsky M, Han J, Gao J (2010) Graph regularized transductive classification on heterogeneous information networks. In: ECML/PKDD (1), pp 570–586

Ji M, Han J, Danilevsky M (2011) Ranking-based classification of heterogeneous information networks. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 1298–1306

Kontonasios K, Spyropoulou E, De Bie T (2012) Knowledge discovery interestingness measures based on unexpectedness. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery pp n/a–n/a

Koopman A, Siebes A (2008) Discovering relational item sets efficiently. In: Proceedings of the SIAM Conference on Data Mining (SDM), pp 108–119

Koopman A, Siebes A (2009) Characteristic relational patterns. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 437–446

Korte B, Lovász L (1985) Relations between subclasses of greedoids. Mathematical Methods of Operations Research 29:249–267

Kuramochi M, Karypis G (2001) Frequent subgraph discovery. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp 313–320

Lawler EL, Lenstra JK, Kan AHGR (1980) Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. SIAM J Comput 9(3):558–565

Makino K, Uno T (2004) New algorithms for enumerating all maximal cliques. In: Scandinavia Workshop on Algorithm Theory (SWAT), pp 260–272

Maruhashi K, Guo F, Faloutsos C (2011) Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining, ASONAM '11, pp 203–210

Ng EKK, Ng K, Fu AWC, Wang K (2002) Mining association rules from stars. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp 322–329

Nijssen S, Kok J (2003) Efficient frequent query discovery in FARMER. In: Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), pp 350–362

Nijssen S, Jiménez A, Guns T (2011) Constraint-based pattern mining in multi-relational databases. In: ICDM Workshops, pp 1120–1127

Ojala M, Garriga GC, Gionis A, Mannila H (2010) Evaluating query result significance in databases via randomizations. In: Proceedings of the SIAM Conference on Data Mining (SDM), pp 906–917

Pardalos PM, Xue J (1994) The maximum clique problem. Journal of Global Optimization 4

Poernomo AK, Gopalkrishnan V (2009) Towards efficient mining of proportional fault-tolerant frequent itemsets. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 697–706

Siebes A, Vreeken J, van Leeuwen M (2006) Item sets that compress. In: Proceedings of the SIAM Conference on Data Mining (SDM), pp 393–404

Spyropoulou E, De Bie T (2011) Interesting multi-relational patterns. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp 675 –684

Srikant R, Agrawal R (1996) Mining quantitative association rules in large relational tables. In: Proceedings of the ACM SIGMOD international conference on Management of data, pp 1–12

Sun Y, Yu Y, Han J (2009) Ranking-based clustering of heterogeneous information networks with star network schema. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 797–806

Sun Y, Han J, Aggarwal CC, Chawla NV (2012a) When will it happen?: relationship prediction in heterogeneous information networks. In: Proceedings of the fifth ACM international conference on Web search and data mining, WSDM '12, pp 663–672

Sun Y, Norick B, Han J, Yan X, Yu PS, Yu X (2012b) Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In: KDD, pp 1348–1356

Tang L, Wang X, Liu H (2012) Community detection via heterogeneous interaction analysis. Data Mining and Knowledge Discovery 25(1):1–33

Trabelsi C, Jelassi N, Ben Yahia S (2012) Scalable mining of frequent tri-concepts from folksonomies. Advances in Knowledge Discovery and Data Mining pp 231–242

Uno T, Asai T, Uchida Y, Arimura H (2004a) An efficient algorithm for enumerating closed patterns in transaction databases. In: Discovery Science, pp 16–31

Uno T, Kiyomi M, Arimura H (2004b) Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI)

Voutsadakis G (2002) Polyadic concept analysis. Order 19(3):295–304

Yahia B, Hamrouni T, Nguifo EM (2006) Frequent closed itemset based algorithms: a thorough structural and analytical survey. SIGKDD Explor Newsl 8(1):93–104

Yan X, Han J (2002) gspan: Graph-based substructure pattern mining. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp 721–730

Yan X, Han J (2003) Closegraph: mining closed frequent graph patterns. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp 286–295

Zaki M, Hsiao CJ (2002) CHARM: An efficient algorithm for closed itemset mining. In: Proceedings of the SIAM International Conference on Data Mining (SDM), pp 457–473

Zaki M, Hsiao CJ (2005) Efficient algorithms for mining closed itemsets and their lattice structure. Knowledge and Data Engineering, IEEE Transactions on 17(4):462–478

Zaki M, Ogihara M (1998) Theoretical foundations of association rules. In: Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery

Zaki MJ (2000) Scalable algorithms for association mining. IEEE Trans on Knowl and Data Eng 12(3):372–390

Zaki MJ, Peters M, Assent I, Seidl T (2007) Clicks: An effective algorithm for mining subspace clusters in categorical datasets. Data and Knowledge Engineering 60(1):51 – 70