Direct Mining of Subjectively Interesting Relational Patterns

Tias Guns
DTAI, KU Leuven
Leuven, Belgium
tias.guns@cs.kuleuven.be

Achille Aknin
ENS Ulm
Paris, France
achille.aknin@ens.fr

Jefrey Lijffijt, Tijl De Bie
IDLab, Ghent University – iMinds
Ghent, Belgium
{jefrey.lijffijt,tijl.debie}@ugent.be

***Abstract*—Data is typically complex and relational. Therefore, the development of relational data mining methods is an increasingly active topic of research. Recent work has resulted in new formalisations of patterns in relational data and in a way to quantify their interestingness in a subjective manner, taking into account the data analyst's prior beliefs about the data. Yet, a scalable algorithm to find such most interesting patterns is lacking. We introduce a new algorithm based on two notions: (1) the use of Constraint Programming, which results in a notably shorter development time, faster runtimes, and more flexibility for extensions such as branch-and-bound search; and (2), the direct search for the most interesting patterns only, instead of exhaustive enumeration of patterns before ranking them. Through empirical evaluation, we find that our novel bounds yield speedups up to several orders of magnitude, especially on dense data with a simple schema. This makes it possible to mine the most subjectively-interesting relational patterns present in databases where this was previously impractical or impossible.**

***Index Terms*—Data mining, Relational databases**

I. Introduction

Many data mining methods require the data to be in a particular simple form, often a tabular one. As data rarely occurs in tabular form, the data mining process often starts with data selection, preprocessing, and transformation steps to make existing methods applicable. Yet, in this process valuable information may be irretrievably lost [1].

Relational pattern mining Research into relational pattern mining aims to make these steps unnecessary by developing methods that find patterns directly in relational data in its full complexity. This poses a conceptual challenge: how can relational patterns be defined in a generic and yet intuitive manner, and how do we quantify how interesting they are to the data analyst? The work in [1] provides a first answer to these questions, in proposing a novel relational pattern syntax called Complete Connected Subsets, and in developing a measure of subjective interestingness for such patterns.

In that work, relational data is modelled in terms of a set E of *entities* of different *types* and a relational schema R that determines what types of entities may be related. Additionally, a set of *relationship instances* $\mathcal{R}$ is given, which are pairs of entities that are actually related. Fig. 1 shows a toy database that has the above relational schema. A Complete Connected Subset (CCS) in this data is a pattern characterized by a set of entities, where all entities whose type share a relation have to be connected. For example, in Fig. 1, $\{U1, M1, M2, G1\}$ is a CCS. Fig. 2 gives an example of a CCS found in a real database with 5 users, 13 action movies, and 1 genre.

The challenge An important open challenge is how the most interesting patterns can be found efficiently. The dedicated RMiner algorithm [1] exhaustively enumerates all (maximal) CCS patterns present in the data, before ranking them according to interestingness. However, as commonly the case in pattern mining, the number of patterns quickly blows up for large or dense data. This makes an enumerate-and-rank approach infeasible for many datasets.

Contributions In this paper we propose a new algorithmic strategy based on two high-level innovations. The first innovation is the reformulation of the CCS mining problem within the *Constraint Programming (CP)* paradigm. CP is a powerful search paradigm for combinatorial optimisation or constraint satisfaction problems, and highly optimised CP frameworks are readily available. We propose a decomposition of the CSS problem formulation in terms of constraints.

Moreover, the use of CP makes our second innovation readily achievable: *search directly only for the most interesting CCS*. To find the global optimum, CP algorithms make use of a *branch-and-bound* strategy. Thus their efficiency depends on the availability of tight yet efficiently computable bounds on the objective function for all solutions in a certain subtree of the search space. We developed a range of such bounds with varying trade-off between computational cost and tightness.

Results The best of these bounds often results in a speed-up of several orders of magnitude, and allows us to find the most interesting CCSs on data well beyond the capabilities of RMiner. We also compared the direct approach on traditional

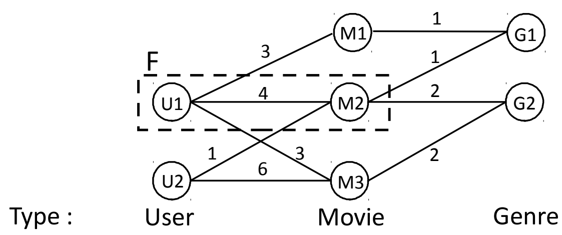

Fig. 1. An example relational database with 3 entity types ('User', 'Movie', and 'Genre') and 2 relations (users may like movies, and movies may belong to a particular genre). The edges represent the relationship instances, and these are annotated with their information content.

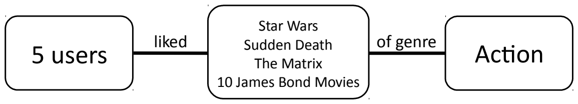

Fig. 2. Best pattern found for the Rotten Tomatoes dataset.

single-relation datasets from the itemset mining community. In that case, a maximal CSS corresponds to a closed itemset. Our direct approach is able to find the most interesting patterns on datasets with too many patterns to be enumerated even with the best closed itemset mining algorithms, like LCM [2].

II. Preliminaries

In this section we recall the main definitions and prior results required to understand the contributions of this paper.

A. Relational data and Complete Connected Subsets

Relational data We use the formalisation of relational data from [1]. To summarise, a *relational database* is defined as a tuple $\mathcal{D} = (E, t, \mathcal{R}, R)$. E denotes the set of *entities*, and $t : E \rightarrow \{1, \ldots, k\}$ is a function that maps an entity onto its *type* (assuming k types). For example, for the toy database in Fig. 1, $k = 3$ with $1 =$ User, $2 =$ Movies and $3 =$ Genre and $t(U1) = t(U2) = 1$ (User). $\mathcal{R}$ denotes the set of all *relationship instances*, which are unordered pairs of entities that are *related*. In a relational database[1], a pair of entities may be related only if allowed by the *relational schema* $R \subseteq \{1, \ldots, k\} \times \{1, \ldots, k\}$. For our toy dataset, $R = \{(1,2),(2,3)\}$ that is, user is connected to movies and movies to genre and $(M1, G1) \in \mathcal{R}$ with $(t(M1), t(G1)) = (2,3) \in R$. Note that since the pairs in $\mathcal{R}$ and R are unordered, $(e, f) \in \mathcal{R} \equiv (f, e) \in \mathcal{R}$ and $(t_1, t_2) \in R \equiv (t_2, t_1) \in R$. We write $\mathcal{R}_r$ to denote all relationship instances $(e, f) \in \mathcal{R}$ of type $r = (t(e), t(f))$.

Complete Connected Subsets (CCSs) A CCS is defined as a subset $F \subseteq E$ that is (1) *complete*: for all $e, e' \in F$ with $(t(e), t(e')) \in R$, it holds that $(e, e') \in \mathcal{R}$; that is, all entities whose types are related in the schema must have an edge between them in the data; and (2) *connected*: for all $e, e' \in F$, there exists a sequence $(e = e_1, e_2, \ldots, e_l = e')$ such that $(e_i, e_{i+1}) \in \mathcal{R}$ for all $i \in \{1, \ldots, l-1\}$.

Furthermore a CCS is *maximal* (i.e., an MCCS) if it is not a strict subset of another CCS. Intuitively, an MCCS is a CCS to which no other entity can be added without violating either completeness or connectedness. In Figure 1, $\{U1, M1, M2\}$ is a CCS but not an MCCS, while $\{U1, M1, M2, G1\}$ is an MCCS. $\{U1, M1, M2, G2\}$ is not a CCS because it is not complete, and $\{U1, G1\}$ is not connected.

An entity e is said to be *compatible* with a CCS F if it is related in $\mathcal{R}$ to every entity in F allowed by the schema:

$$\forall e' \in F,\ (t(e), t(e')) \in R \Rightarrow (e, e') \in \mathcal{R}.$$

Intuitively, e is compatible with F if $F \cup \{e\}$ is a CSS.

[1]Actually the database need not be a relational DB, i.e., a NoSQL DB is also fine, but we assume that the schema of the data is provided.

The subjective interestingness of a CCS Formalising the interestingness[2] of patterns in data is a conceptually challenging problem. In [3] it was argued that interestingness is ideally quantified in a subjective manner, by contrasting the pattern's presence with these prior beliefs. This strategy requires the user to state their expectations about the data. Then, the prior beliefs are formalised as a distribution over the set of possible data sets, which is estimated as the maximum entropy distribution given the prior belief constraints. This distribution is referred to as the *background distribution*.

Self-information The *self-information* of a pattern is then defined as minus the logarithm of the probability that the pattern is present, computed w.r.t. this background distribution. In the particular context of relational data, the prior beliefs could include an expectation about the total number of relationship instances, about the number relationship instances per relationship type, or about the number of relationship instances of each particular entity within each relationship type it participates in. Conveniently, the maximum entropy distribution takes the same parametric form for all these prior belief types, albeit with different values for the parameters. In particular, it is a product of independent Bernoulli distributions for each pair (e, e') for which $(t(e), t(e')) \in R$. Denoting the probability of an edge (e, e') to be present in $\mathcal{R}$ as $p_{e,e'}$, this means that the self-information of a CCS pattern over the set $F \subseteq E$ is given as:

$$\mathrm{SI}(F) \quad = \sum_{\substack{e,e' \in F, t(e) < t(e'),\\ (t(e), t(e')) \in R}} -\log(p_{e,e'}).$$

Each $-log(p_{e,e'})$ is called the *contribution* of the edge (e, e').

Description length The larger SI(F), the more informative (or surprising) the CCS pattern is to the user. Yet, it also becomes harder for the user to assimilate it as its description grows larger. In [4] it is proposed to model this description length as an affine function of the cardinality of F:

$$\mathrm{DL}(F) \quad = \quad a + \gamma \times |F|$$

with $a = |E| \log\left(\frac{1}{1-p}\right)$, $\gamma = \log\left(\frac{1-p}{p}\right)$ and $p \in (0,1)$ a parameter that is typically set equal to the edge density of the database (see [4] for details).

The subjective interestingness The final interestingness measure is the *Information Ratio*, formalised as the density of information compressed within the pattern [4]:

$$\mathrm{InformationRatio}(F) \quad = \quad \frac{\mathrm{SI}(F)}{\mathrm{DL}(F)}.$$

Iterative data mining Our algorithm directly extracts the most interesting MCCS in the data. Yet, a single MCCS pattern may not be sufficient for the data analyst: they may also want to see other MCCSs to further enhance their understanding of the data. An elegant way to achieve this is to adjust the background distribution, accounting for the data analyst's

[2]In some areas of data mining better known as *quality*, *objective*, or conversely, the *cost* function.

knowledge of the first MCCS. With F the previous MCCS, this can be done by setting $p_{e,e'} = 1$ for all $e, e' \in F$, to reflect the user now *knows with certainty* that e and e' are related. Then the most interesting MCCS can be sought again, which will provide the maximum amount of non-redundant information. The process can of course be iterated.

B. Constraint Programming

A Constraint Satisfaction Problem (CSP) $P = (V, D, C)$ consists of a set of variables V, a domain D that specifies the set of allowed values $D(v)$ for each variable $v \in V$, and a set of constraints C over (a subset of) the variables V. A *solution* to a CSP is an assignment S to the variables of V such that $\forall v \in V, S(v) \in D(V)$ and all constraints are satisfied.

Constraint solving is a generic methodology for solving CSPs. It reasons over partial solutions, where a partial solution is a domain D' such that $\forall v \in V : D'(v) \subseteq D(V)$. If the domain of a variable consists of just one value, $|D'(v)| = 1$, we say that variable is assigned. A key principle is that during solving, values can only be removed from the domain; which is executed repeatedly until all variables are assigned.

At their core, most CP solvers are generic depth-first search frameworks, with a *branch-and-propagate* mechanism. *Branching* consists of choosing one unassigned variable and either removing one value from its domain, or removing all but one value (i.e., assigning it to that value). *Propagation* is the use of a constraint to infer that certain values in the domain are infeasible for this constraint, and hence infeasible in the entire CSP. For example, let X, Y, Z be three variables with domain $D(X) = D(Y) = \{0, 1\}$ and $D(Z) = \{1\}$. Then, constraint $X = Y \vee Z$ can be used to infer that $D(X) = \{1\}$.

In a CP framework, every constraint is implemented through a propagator that does this inference. At each node in the search tree, the solver will call each of the propagators until fixpoint, that is, until the domain no longer changes. Constraint solvers can also be used to solve constraint optimisation problems (V, D, C, f) where f is a function that needs to be minimized or maximized. For this, solvers use *branch-and-bound* (see Section IV).

III. A CP approach to enumerating MCCSs

We first explain how the MCCS conditions can be formulated through constraints. We show that the completeness and maximality conditions are related and can be expressed as a constraint over every individual entity.

A. Reformulating the MCCS conditions as constraints

Completeness for a single relation For now, let $r = (t, t') \in R$ be the only relation in the relational schema. Let F be an MCCS, and let $e \in F$ be any entity with $t(e) = t$.[3]

The *completeness requirement* of an MCCS states that e should be connected to all entities in F that are of type t'—the other type in r. To reformulate this requirement in a form usable in a CP framework, we define $N_r(e)$ as the set of entities of type t' that are connected to e in relation r:

$$N_r(e) \equiv \{e' \in E \mid (e, e') \in \mathcal{R}_r\}.$$

Additionally, we define $O_r(t)$ as the set of all entities of the *other* type t' in r:

$$O_r(t) \equiv \{e' \in E | (t, t(e')) = r\}.$$

and hence $(F \cap O_r(t))$ is the set of all entities in F that are of that other type.

With these two definitions, the *completeness requirement* can now be reformulated as follows for a given $r = (t, t')$:

$$e \in F \Rightarrow (F \cap O_r(t(e))) \subseteq N_r(e).$$

That is, if an entity e is in the CCS, then all entities in the CCS of a type t' that are adjacent to $t(e)$ in the relational schema must be related to e in $\mathcal{R}_r$.

Maximality and completeness for a single relation *Maximality* states that if an entity can be added to the CCS while respecting connectedness and completeness, it must be added. For simplicity, we again study this first for a schema $R = \{r\}$ with a single relation $r = (t, t')$. In this special case, the maximality requirement can be reformulated as:

$$(F \cap O_r(t(e))) \subseteq N_r(e) \Rightarrow e \in F.$$

Combining maximality and completeness, we get:

$$e \in F \Leftrightarrow (F \cap O_r(t(e))) \subseteq N_r(e).$$

This is precisely the *coverage* requirement in frequent itemset mining, which computes the *transaction* entities e that are covered by a given set of *item* entities $O_r(t(e))$. Enforcing coverage on both items and transactions corresponds to finding all *closed* itemsets, which are a special case of CCSs [1].

Maximality and completeness in the general case First, denote with $Q(t) \equiv \{r \in R | \exists t', (t, t') = r\}$ the set of all relations involving entity type t. The joint completeness and maximality requirement can be stated for each entity e over all its relations $r \in Q(t(e))$:

$$e \in F \Leftrightarrow \forall r \in Q(t(e)), (F \cap O_r(t(e))) \subseteq N_r(e). \quad (1)$$

In other words, each entity must be completely connected to the entities in F for all relations that its type participates in, and all such entities must be part of the CCS.

Connectedness While the above ensures that all entities that share a relation are connected, it does not require that entities whose types do not share a relation are connected; for example, $\{D1, G1\}$ in Fig. 1. We wish to formulate this requirement as a constraint in a CSP, but not over every possible pair of entities as the number of entities can be huge. Instead, we formulate it over every pair of entity types: let $h(t, F) \equiv (\exists e \in F, t(e) = t)$ be a function that checks whether a CCS F contains an entity of type t. Furthermore, let $M_R(t, t')$ be a function that returns the multi-set of all sets of types that are on a path from t to t' in R. Connectivity is enforced by requiring that if two different entity types have at

[3]Note we overload t to signify a function that returns the type of an entity, or a constant type—which of these is meant should be clear from the context.

least one entity in F, there must exists a path in the relational schema R between the two types, such that each entity type on the path has an entity in F:

$$\forall t, t' : h(t, F) \wedge h(t', F) \rightarrow \\ \exists\, S \in M_R(t, t') \text{ such that } \forall t'' \in S,\, h(t'', F). \quad (2)$$

Together with the completeness constraint above, this formulation will ensure that there exists a path in $\mathcal{R}$ from each entity to each other entity.

B. MCCS in a constraint solver

Our model of the problem is inspired by how itemset and multi-relational mining [5] are modelled in the CP4IM framework [6]. We introduce an array of Boolean variables for each entity type, with one Boolean variable for each entity. Each Boolean variable indicates whether the entity is part of the MCCS or not, and hence one solution to the CSP will correspond to one maximal CCS. Within the CP4IM framework, it was shown that the coverage relation $e \in F \Leftrightarrow (F \cap O_r(t(e))) \subseteq N_r(e)$ for a single relation $r = (t, t')$ can be modelled as a *reified* linear sum over Boolean variables X, one variable for each entity of type t, and Boolean variables Y for each entity of type t' as such:

$$coverage(X, Y, r) \equiv \forall x, \\ X[x] \Leftrightarrow \sum_y : (Y[y] * (y \notin N_r(x))) = 0,$$

that is, x is in the CCS iff the sum of Boolean variables corresponding to entities y that are not connected to x is zero.

The multi-relation case (Eq. 1) can be transformed to the single-relation case by defining $O(t(e)) = \cup_{r \in Q(t(e))} O_r(t(e))$ and $N(t(e)) = \cup_{r \in Q(t(e))} N_r(t(e))$. Because each entity belongs to only one type, the sets are non-overlapping and hence the joint completeness and maximality requirement simplifies to: $e \in F \Leftrightarrow (F \cap O(t(e))) \subseteq N(e)$ which can be modelled in the same way as the single-relation case.

The connectivity requirement can be decomposed into an auxiliary Boolean variable for every $h(t, F) \equiv (\exists e \in F, t(e) = t)$ relation and the formula in Equation 2 where the multi sets $M_R(t, t')$ are pre-computed.

Using the above formulation of variables and constraints, all solutions to this CSP will correspond to all MCCSs.

IV. Bounds on InformationRatio to directly mine the most interesting MCCSs

To do *branch-and-bound*, we need a propagator that computes a bound on the InformationRatio given the current domain D that represents a partial solution. For conciseness, let F denote the entities assigned to 1 (i.e., the current CCS, and equivalently $\{e \in E | D(e) = 1\}$). Let $C = \{e \in E | D(e) = \{0, 1\}\}$ be the unassigned entities, and let $B = \{e \in E | D(e) = 0\}$ be the set of entities assigned to 0, e.g. because the constraints determined that they are incompatible with the current pattern or because they were assigned to 0 due to branching. We also use F_t to denote the set of entities in F of type t, and similarly for C_t and B_t.

We want to formulate an upper bound $U(F, C)$ on the maximal interestingness of any CCS reachable from this state. A *branch-and-bound* can then verify whether the best value achievable by the partial solution is worse than the best CCS F^* found so far. If so, this entire branch (and domain) can be pruned away from the search tree and backtracking occurs.

Recall that each $(e, e') \in \mathcal{R}$ has a contribution $-\log(p_{e,e'})$ as determined from the maximum entropy distribution. We denote by $\mathbf{v}$ the array of the contributions of all edges between entities of F and C, and C and C, sorted in decreasing order. If $r = (t_1, t_2) \in R$ is a relation, then $\mathbf{v}_r \equiv \mathbf{v}_{(t_1, t_2)}$ is the subset of $\mathbf{v}$ containing only contributions of edges between an entity of type t_1 and an entity of type t_2.

We will use as an example the dataset in Fig. 1 where each edge is labelled by its (fictional) contribution, and the current pattern F is highlighted in a dashed box.

A. Naive bound : estimating the number of edges

The most straightforward bound is the following: the maximum self-information is the self-information of $F \cup C$ and the minimum description length is the description length of F. Then the upper bound on the self-information is the self-information of F plus the sum of all remaining contributions, stored in $\mathbf{v}$. We can improve on this by observing that unless $F \cup C$ forms a clique, not all entities can be added into a single pattern, and hence not all edges in $\mathbf{v}$ can be added. If we can hence derive the maximum number of edges nb_r for each relation $r \in R$ that can be added to F, then we have a valid upper bound if we only count the nb_r biggest contributions (recall that we assume $\mathbf{v}$ is sorted):

$$U_{naive}(F, C) = \frac{\text{SI}(F) + \sum_{r \in R} \sum_{i=0}^{\text{nb}_r - 1} \mathbf{v}_r[i]}{\text{DL}(F)} \quad (3)$$

The maximum number of edges nb_r of relation $r = (t_1, t_2)$ that can be included in any extension of F can be estimated based on the *degrees* of the entities in the relation as follows. Let $d_{t_1 \rightarrow t_2}$ (resp. $d_{t_2 \rightarrow t_1}$) be the decreasingly ordered set of the number of edges from an entity of type t_1 (resp. t_2) in C to any entity of type t_2 (resp. t_1) in C, that is, the degrees of the nodes in C when counting only edges to other nodes in C. Let $0 \leq n_{t_1} \leq |C_{t_1}|$ be any number of entities of type t_1 that can possibly be added to F, and similarly for n_{t_2}, then we have the following inequalities when $n_{t_1} \neq 0$ and $n_{t_2} \neq 0$:

$$n_{t_1} \leq d_{t_2 \rightarrow t_1}[n_{t_2} - 1] \quad \text{and} \quad n_{t_2} \leq d_{t_1 \rightarrow t_2}[n_{t_1} - 1]. \quad (4)$$

Intuitively, for an n_{t_1} let the n_{t_1}-th largest degree of the entities in C be y, then the minimal degree of the n_{t_1} elements in C is less than y, so we can conclude that there can only be fewer than y entities of type t_2 added to this CCS, because the CCS has to be complete. An upper bound on nb_r for use in Eq. (3) is then obtained by taking the maximum value of

$$\text{nb}_{(t_1, t_2)}(n_1, n_2) = n_1 \times n_2 + n_1 \times |F_{t_2}| + |F_{t_1}| \times n_2$$

for all tuples (n_1, n_2) that satisfy the inequalities of Eq. (4).

B. Enumerating all denominators

The previous bound is easy to compute but loose because the nominator assumes as many entities as possible are included and the denominator assumes no additional entities are included. Instead, we will compute an upper bound for each possible combination $(n_1, \cdots, n_k)$ with n_t the number of entities of type t and take the maximum value of those as (tighter) upper bound.

While this may sound computationally expensive, namely $O(max_t(n_t)^k)$ combinations to check, we can use the inequalities of Eq. (4) to avoid many impossible combinations.

Consider the set $\mathcal{S}$ of all combination $(n_1, \cdots, n_k)$ such that the inequalities in Eq. (4) are verified for every relation;

$$U_{denom}(F, C) = \max_{(n_1,\cdots,n_k)\in\mathcal{S}} \frac{\text{SI}(F) + \sum_{\substack{(t_1,t_2)\in R,\\ t_1<t_2}} \text{SumContr}(t_1, t_2)}{\text{DL}(F) + \gamma \times (n_1 + \cdots + n_k)}$$

To compute SumContr, we first observe there are three different groups of possible contributions in every relation $r \in R$:

- $\mathbf{v}^{CC}_{(t_1,t_2)}$, the array of decreasingly ordered contributions of edges between entities in C_{t_1} and entities in C_{t_2}.
- $\mathbf{v}^{FC}_{(t_1,t_2)}$ between entities in F_{t_1} and entities in C_{t_2}.
- $\mathbf{v}^{CF}_{(t_1,t_2)}$ between entities in C_{t_1} and entities in F_{t_2}.

Leading to

$$\text{SumContr}(t_1, t_2) = \sum_{i=0}^{n_{t_1}\times n_{t_2}-1} \mathbf{v}^{CC}_{(t_1,t_2)}[i] + \sum_{i=0}^{|F_{t_1}|\times n_{t_2}-1} \mathbf{v}^{FC}_{(t_1,t_2)}[i] + \sum_{i=0}^{n_{t_1}\times |F_{t_2}|-1} \mathbf{v}^{CF}_{(t_1,t_2)}[i] \quad (5)$$

C. All denominators and pruning non-improving entities

Attempts to further improve this bound let to disproportionate computation costs. However, we can improve search by *pruning* entities for which we know they will not lead to a better MCCS than the currently best found. Let MaxInt be the interestingness of this current best one.

In the previous bound, we compute a value for each possible combination $\mathbf{n} = (n_1, \cdots, n_k)$. If we know, using these values, that there exists a type t and a size m such that for every combination $(n_1, \cdots, n_k)$ where $n_t < m$, the value obtained is less than MaxInt, then there is no CCS with strictly less than m entities of type t that has an InformationRatio greater than MaxInt. That can be used to prune those entities that, if they would be set to 1, would prevent having more than m additional entities of type t. These are the entities connected to less than m entities in C_t.

Given a specific entity type t and an integer x, we can, during the computation of the previous bound, compute bestval $[t]\,[x]$, the maximal interestingness reached with $n_t = x$. Then, for each entity type t, let minsize $[t] = min\{0 \le x \le |C_t|$ s.t. bestval $[t]\,[x] \ge$ MaxInt$\}$, with minsize $[t] = +\infty$ if there is no size x such that bestval $[t]\,[x] \ge$ MaxInt.

Theorem 1. *For each type t_1 and each entity e of type t_1, if there exists t_2 such that $(t_1, t_2) \in R$ and degree$_{t_1\to t_2}[x] <$ minsize $[t_2]$ then we can prune entity $e : D(e) = D(e) \setminus \{1\}$ (assigned to 0)*

If we have a relation $(t_1, t_2) \in R$ and an entity e of type t_1 such that e is related to strictly less than minsize $[t_2]$ entities of type t_2, then any MCCS containing F and e must contain less than minsize $[t_2]$ entities of type t_2. Due to the property of minsize, we can prune the entity e as there will be no MCCS containing e and F that are more interesting than the best MCCS we have so far.

V. Experiments

The experiments investigate how the CP algorithm compares to RMiner when enumerating all MCCSs, and the speed up achieved by directly searching for the most interesting MCCSs using the proposed bounds. We conducted experiments on a range of datasets: various sizes of the IMDB dataset (as in [1]) with 3 entity types (*Director*, *Movie*s, *Genre*s); rottentomatoes with 3 entity types (*User*, *Movie*s, *Genre*s); DBLP/chain with 4 entity types, (*Author 1*, related to *Paper1*s they authored, related to *Paper2*s it cites, related to those *Author2*s); and DBLP/star with 4 entities (*Paper*, *Venue*, *Author*s, *Year* of publication). Also datasets from the FIMI repository were used, as well as the FourSquare check-ins data [7], as used in [8]. Basic statistics of the datasets are given in Table I.

Fig. 2 gives an example of the type of pattern found. This is, according to the background distribution that takes the row and column marginals into account, the most interesting pattern in the rottentomatoes dataset, showing 5 users that all liked the same 10 James Bond movies along with 3 other movies of the action genre.

All experiments were run on quad-core Intel Xeon Ubuntu 14.04 servers with 32Gb RAM. We terminated any experiment after 5 hours. The Gecode CP solver was used (`http://gecode.org`) and our software is available on `https://bitbucket.org/ghentdatascience/cp`

Results The runtimes of the exhaustive enumeration algorithms and the direct search CP variants are presented in Table I. The number of solutions for exhaustive search is shown as well; solutions with a + indicate how many CP-closed found before it timed out.

Exhaustive CP versus RMiner On all but three datasets, RMiner timed out, while the CP approach completed on almost half the datasets included. For the dataset (imdb/1year) on which RMiner did not time out, CP was three orders of magnitude faster than RMiner, and for three of the four other datasets on which RMiner timed out, CP was two (imdb/5years, fimi/mushroom) and one (imdb/10years) orders of magnitude faster than the timeout limit. For the dblp-star dataset involving 3 relations, RMiner and CP perform similarly, while for the more complex dblp-chain dataset RMiner was significantly faster. This may indicate that the current CP decomposition of the completeness constraint for every relation separately is less efficient than RMiner's dedicated methods for enforcing completeness and connectivity across multiple relations.

TABLE I

Runtimes in seconds, with a timeout of 5 hrs. The first three columns give basic statistics of the datasets (number of relations; total number of entities; and overall density). The second three columns show runtimes for the exhaustive enumeration algorithms (LCM, which is applicable only to single-relational data; RMiner; and the proposed CP algorithm). The final three columns gives runtimes for directly searching for the most interesting pattern using the three proposed bounds.

	#Rels	#Ent.	Dens.	#Sols.	LCM	RMiner	CP-closed	CP-naive	CP-denom	CP-prune
fimi/mushroom	1	8243	19.33%	238 709	4.55	timeout	110.5	10.3	4.89	2.33
fimi/chess	1	3271	49.33%	411 000 000+	timeout	timeout	timeout	16.4	16.2	14.9
fimi/T10I4D100K	1	100870	1.16%	359 000+	10.9	timeout	timeout	3915	468	87
fimi/T40I10D100K	1	100942	4.20%	1 000 000+	timeout	timeout	timeout	timeout	7643	224
fimi/connect	1	67686	33.33%	13 000 000+	timeout	timeout	timeout	2754	2097	1640
fimi/retail	1	104632	0.06%	3 000+	11.5	timeout	timeout	timeout	timeout	1421
foursquare/checkins	1	224947	2.11%	232 747	0.6	timeout	7296	255	35.5	16.8
imdb/1year	2	3291	0.97%	583	-	2128	1.05	1.26	0.4	0.1
imdb/5years	2	30131	0.24%	3 887	-	timeout	208	242	26.2	5.06
imdb/10years	2	51203	0.12%	8 704	-	timeout	1764	1618	127.9	8.36
imdb/40years	2	111320	0.03%	15 900+	-	timeout	timeout	timeout	990	9.18
imdb/100years	2	514323	0.002%	15 900+	-	timeout	timeout	timeout	timeout	290
rottentomatos	2	12263	0.44%	13 000 000+	-	timeout	timeout	timeout	2986	279
dblp-star	3	8279	0.10%	7 699	-	207	269	305	19	3
dblp-chain	3	13862	0.09%	30 629	-	76	5423	9141	939	107

Exhaustive CP versus LCM MCCSs are equivalent with closed itemsets with a minimum frequency of 1 in single-relational data. Thus, a computational comparison of the CP approach with LCM, the state-of-the-art method for closed itemset mining on single-relational data, can give an idea of the optimality of the approach. Columns 'LCM' and 'CP-closed' in Tab. I show that the price paid for the generality of CP and multi-relational mining as compared to LCM is one to four orders of magnitude. However, on three of the datasets even LCM times out, underscoring that the generate-and-rank approach is simply impractical for such datasets.

Direct search versus exhaustive enumeration The last three columns in Tab. I show that, with the exception of the dblp-star and dblp-schema datasets, direct search using any bound is always faster then enumerating all patterns first, even when the search space consists of millions of patterns. Datasets, such as dense FIMI datasets, for which no generate-and-rank approach is feasible can now be solved. Furthermore, stronger bounds always lead to faster runtimes. Indeed, the difference is especially dramatic for CP-prune, which prunes away entities that can not lead to a better MCCS. On most datasets, using this bound is two orders of magnitude faster then enumerating all patterns and can be an order of magnitude faster or more than the CP-denom bound. We do observe that for all FIMI datasets on which LCM terminates, a post-processing of its output would be a faster approach. For the dblp-chain dataset, using RMiner and its post-processing mechanism would be faster too. For the other datasets, clearly the direct approach with the pruning bound is recommended.

Finally, experiments (not shown) demonstrate that the running time grows only very slowly over subsequent iterations, such that the proposed approach is computationally attractive even if more than just the most interesting pattern is required.

VI. Conclusion

We have addressed the challenging problems of (1) *exhaustively enumerating* all so-called MCCS patterns in a relational database using a novel CP formulation, and (2) directly searching for just the *most interesting* MCCS pattern through the use of a range of novel interestingness upper bounds in a branch-and-bound variant of this CP formulation. This results in speed-ups of several orders of magnitudes on a range of real-life datasets. These contributions make it possible to search for the most interesting relational patterns in relational databases beyond the capabilities of previous methods.

Looking forward, we see a number of possible innovations. The first is additional user-specified constraints on the MCCS patterns, limiting the search space, potentially at the cost of losing the most informative patterns. This strategy is used succesfully in RMiner to achieve dramatic speed-ups. Secondly, for larger number of relations, a propagator that can reason over multiple relations may be needed. Finally, it may be possible to speed-up subsequent iterations in the iterative mining process by exploiting intermediate results.

Acknowledgements: ERC Consolidator Grant FORSIED (ref. 615517) and Research Foundation, Flanders (postdoc).

References

[1] E. Spyropoulou, T. De Bie, and M. Boley, "Interesting pattern mining in multi-relational data," *DMKD*, vol. 28, no. 3, pp. 808–849, 2014.

[2] T. Uno, T. Asai, Y. Uchida, and H. Arimura, "An efficient algorithm for enumerating closed patterns in transaction databases," in *Proc. of DS*, 2004, pp. 16–31.

[3] T. De Bie, "An information-theoretic framework for data mining," in *Proc. of KDD*, 2011, pp. 564–572.

[4] E. Spyropoulou, T. De Bie, and M. Boley, "Mining interesting patterns in multi-relational data with N-ary relationships," in *Proc. of DS*, 2013, pp. 217–232.

[5] S. Nijssen, A. Jimenez, and T. Guns, "Constraint-based pattern mining in multi-relational databases," in *Proc. of ICDM Workshops*, 2011, pp. 1120–1127.

[6] L. De Raedt, T. Guns, and S. Nijssen, "Constraint programming for itemset mining," in *Proc. of KDD*, 2008, pp. 204–212.

[7] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui, "Exploring millions of footprints in location sharing services," in *Proc. of ICWSM*, 2011, pp. 81–88.

[8] J. Lijffijt, E. Spyropoulou, B. Kang, and T. De Bie, "P-N-RMiner: A generic framework for mining interesting structured relational patterns," *IJDSA*, vol. 1, no. 1, pp. 61–76, 2016.