Découverte de motifs : Enumération, Programmation par Contraintes/SAT et Bases de données[1]

Tutoriel BDA 2011

Lhouari Nourine[1] Jean-Marc Petit[2] Lakhdar Saïs[3]

[1]Université Blaise Pascal, CNRS, LIMOS
[2]Université de Lyon, CNRS, INSA Lyon, LIRIS
[3]Université d'Artois, CNRS, CRIL

Oct 24, 2011 – Rabat, Marocco

[1]Work done in the DAG project, funded by the ANR DEFIS 2009 program

Pattern Mining Problems

A main theme in data mining

- Basket data analysis, seminal paper of Apriori [AS94]
- Plenty of such *problems*
- Even more *applications* and
- an overflow of *research papers* since 1994 !

Examples

- frequent itemsets (and variants), sequences, trees, graphs
- functional, inclusion, multivalued dependencies
- learning monotone function
- minimal transversals of hypergraph

➩ A wide class of problems, some being studied for years in combinatorics, artificial intelligence and databases

Practical Applications

Pattern mining problems ➫ hidden behind practical applications

For instance:

1. Basket data analysis (Agrawal et al, VLDB'93) [AS94]
2. Query rewriting in data integration (H. Jaudoin et al, DL'05) [JFPT09]
3. Discovering complex matchings across web query interfaces: a correlation mining approach (B. He et al, KDD'04) [HCH04]
4. and much more ...

⇒ data-centric steps of many practical applications

Main constat

Data mining research in this (sub-)area ?
➾ most of the time, ad-hoc solutions (with customized data structures)

- Can be seen as a competition to devise (low-level) code (to beat previous implementations)
- I/O routines sometimes as important as algorithmic strategies !

For one problem common to many applications, one solution per application !

- efficient low level code **very** difficult to reuse
- a slight change in the problem statement (data, pattern or predicate) often means to re-start development from scratch

Our Motivations

Elegant and concise solution should exist !

➾ Rapid prototyping of new problems should be easy
➾ Low-level details should be hidden to developers
➾ Efficient and scalable implementations

Long-term objective

➾ Pushing forward **declarative approaches** (SAT/CP, Databases) for pattern mining problems

➾ Towards a wider dissemination of data mining techniques

Related works

Main trends for declarative approaches in data mining

- C++ library (DMTL [CHSZ08], iZi [FDP09]) – *remains programmer-dependent, lack of declarative languages + optimization*
- Inductive logic programming (e.g. [Wro00, NR06]) – *highly expressive, not efficient enough*
- Inductive Databases (e.g. [IM96, LGZ10, RT11])
- Constraint programming (De Raedt group [RGN08], Caen, Lens, Lyon) – *new trends of research, relatively active*
- Databases and Data Mining (e.g. [HFW$^+$96, Cha98, STA98, IV99, CW01, BCC05, FL10, BCF$^+$11, OP11]) – *Many attempts, driven by the "elephants"*
- Theoretical frameworks for pattern mining (e.g. [MT97, GKM$^+$03, AU09, GMS11])

Requirements on Inductive Databases

Three dimensions [RT11]:

- The KDD as a process: closure principle[2], completeness, reusability
- The data source to explore and the patterns to discover: Expressiveness, meta-schema definition, extensibility
- The system architecture that supports the query language: support for efficient algorithm programming, flexibility, standardization (e.g. PMML)

[2] The closure principle is sometimes not required [TVS$^+$07].

Related works

Many attempts, not very successful yet

Compromise to be found between many opposite goals: genericity, efficiency, easy of use, seamless integration with SQL . . .

The elephants (Oracle, DB2, SQLServer) have their own data mining solutions

- built on top of existing DBMS, not fully integrated with SQL
- can be seen as syntactic sugar

Our feeling

- The scope of IDB should be narrowed, even for pattern mining problems themselves (without classifications, clustering ...)
- Lack of theoretical background for pattern mining $\Rightarrow$ Need to specify classes of problems on which declarative techniques may apply.
- No hope in the large !

Outline

Outline

Background
- Notations
- Isomorphism with a boolean lattice
- Complexity

CP/SAT and Pattern Mining
- Constraint Programming (CP) and Satisfiability (SAT): a brief overview
- CP for Frequent Itemset Mining
- CP/SAT for Sequence Mining

Concluding remarks

Outline

Notations

Mainly from (Mannila and Toivonen, DMKD, 1997) [MT97]

Consider the following framework:

1. Let $\mathcal{D}$ be a database
2. Let $\mathcal{L}$ be a set of patterns (or a finite language)
3. Let $\mathcal{P}$ be a predicate to qualify interesting patterns X in $\mathcal{D}$, noted $\mathcal{P}(X, \mathcal{D})$

Definition (Problem statement P)

Given $\mathcal{D}$, $\mathcal{L}$ and $\mathcal{P}$, enumerate all interesting patterns of $\mathcal{L}$ in $\mathcal{D}$

In other words, enumerate the set
$Th(\mathcal{D}, \mathcal{L} \mathcal{P}) = \{X \in \mathcal{L} \mid \mathcal{P}(X, \mathcal{D}) true\}$

Sometimes, $\mathcal{D}$ is made up of patterns of $\mathcal{L}$
Without any other knowledge, how to solve P?

Structuring the search space (1/2)

Specialization/generalization relation may exist among patterns

4 Let $\preceq$ be a partial order on $\mathcal{L}$

$X \preceq Y$: X generalizes Y and Y specializes X

Many possible partial orders specific to patterns, e.g. sets, sequences, trees, inclusion dependencies

Structuring the search space (2/2)

Influence of the partial order on the predicate ?

The most studied property in data mining: monotonic property

Definition

$\mathcal{P}$ is said to be monotone with respect to $\preceq$ if for all $X, Y \in \mathcal{L}$ such that $X \preceq Y, \mathcal{P}(Y, \mathcal{D}) \Rightarrow \mathcal{P}(X, \mathcal{D})$

Equivalent problem statements

Two (complementary) notions emerges: the positive and negative borders, i.e. the most specialized interesting patterns and the most generalized non interesting patterns

Definition (New problem statement P')

Given $\mathcal{D}$, $\mathcal{L}$ and $\mathcal{P}$, enumerate positive (or negative) border of interesting patterns of $\mathcal{L}$ in $\mathcal{D}$

In other words, enumerate the sets:
$bd^{+}(\mathcal{D}, \mathcal{L}, \mathcal{P}, \preceq) = \{X \in Th \mid \nexists Y \in \mathcal{L}\,(X \preceq Y \Rightarrow Y \in Th)\}$
$bd^{-}(\mathcal{D}, \mathcal{L}, \mathcal{P}, \preceq) = \{X \in \mathcal{L} \mid X \notin Th, \forall Y \in \mathcal{L}\,(Y \preceq X \Rightarrow Y \in Th)\}$

$\Rightarrow$ Characterize DAG problems

Example of frequent itemset mining (FIM)

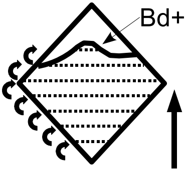
Let A be a set of items, ϵ a user-defined threshold, $\mathcal{D}$ a transactional database, $\mathcal{L} = 2^A$ and $\mathcal{P}(X, \mathcal{D})$ defined as:
$\mathcal{P}(X, \mathcal{D})$ true wrt ϵ iff $card(\{t \in \mathcal{D} | X \subseteq t\}) \geq \epsilon$
$\mathcal{P}(X, \mathcal{D})$ monotone wrt $\subseteq$

- 'Apriori' levelwise search with clever **candidate generation**
- Depth-first search
- **Relationship between borders**
- Specialized data structures to optimize the counting operation, to compress the database ...

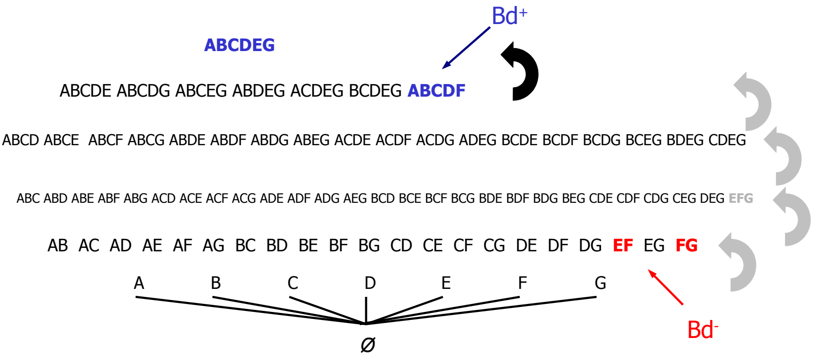
Many contributions with international competitions: FIMI 2003, FIMI 2004, OSDM 2005 workshops

Example (end)

Levelwise search



Pruning strategy: based on the monotonicity property



Outline

Isomorphism with a boolean lattice

Basic idea

Patterns encoded in the powerset of some set and inversely

- For some finite set E, a function f from $\mathcal{L}$ to 2^E has to exist such that:
  - f^{-1} is computable
  - f bijective
  - f preserves the partial order, i.e. $X \preceq Y \Leftrightarrow f(X) \subseteq f(Y)$

➡ Quite severe assumption

➡ Define the so-called representable as set pattern mining problems

Main interests of "representable as sets" problems

For any representable as set problem:

1. Clear separation between DB accesses for predicate evaluation and candidate enumerations on patterns
2. Set oriented algorithms can be used everywhere
   2.1 candidate generation in levelwise algorithms
   2.2 relationship between borders: notion of dualization (minimal transversal enumeration in an hypergraph)
3. Same algorithm principles can be applied to every problem

Main known class of pattern mining problems

- Formally defined, good candidate to apply declarative approaches
- Quite restrictive due to the surjectivity constraint
  ➩ The set of patterns has to have 2^n patterns

Outline

Complexity of enumeration algorithms

Main points to be studied:

1. Dualization problem (the heart of the many pattern mining problems)
2. Encoding/decoding of pattern mining problems (new classes of problems)
3. Relaxation of enumeration problems vs extended enumeration (new idea)

Enumeration problems

Definition (Enumeration Problem)

Input: A finite discrete structure S and a predicate P over S.
Output: The set $P(S)$ of elements of S which satisfy P.

Definition (Decision problem)

Input: A finite discrete structure S, a predicate P over S and a set $X \subseteq P(S)$.
Question: Does $X = P(S)$ holds?

Definition (Decision problem with counterexample)

Input: A finite discrete structure S, a predicate P over S and a set $X \subseteq P(S)$.
Question: Does $X = P(S)$ holds? Otherwise find $x \in P(S) \setminus X$.

Enumeration problems

Definition (Enumeration Problem)

Input: A finite discrete structure S and a predicate P over S.
Output: The set $P(S)$ of elements of S which satisfy P.

- $|P(S)|$ can be exponential in $|S|$.
- Polynomial complexity : $O((|S| + |P(S)|)^k)$.
- Quasi-Polynomial complexity : $n^{O(log(n))}$, where $n = |S| + |P(S)|$.

Dualization problem

Let V be a finite set of patterns, $\mathcal{C} \subseteq 2^V$ and $A \subseteq \mathcal{C}$.
We note: $A^+ = \{x \in \mathcal{C} \mid \exists a \in A,\ a \subseteq x, \}$
$A^- = \{x \in \mathcal{C} \mid \exists a \in A,\ x \subseteq a, \}$
The negative border of A can be written as:
$bd^-(A) := \max_{\subseteq}\{x \mid x \in \mathcal{C} \backslash A^+\}$

Dualisation (Enumeration)

Input: $\mathcal{C} \subseteq 2^V$ et $A \subseteq \mathcal{C}$
Question: Enumerate $bd^-(A)$.

Dualization (Decision)

Input: $\mathcal{C} \subseteq 2^V$, $A \subseteq \mathcal{C}$ et $X \subseteq bd^-(A)$
Question: Is $bd^-(A) = X$? Otherwise find $x \in bd^-(A) \setminus X$.

- Complexity depends on the structure and the encoding of $\mathcal{C}$
- For the boolean lattice, the encoding is implicite, i.e. $\mathcal{C} = 2^V$.

Some known results about dualization

- $\mathcal{C} = 2^V$ is a boolean lattice: Quasi-Polynomial [FK96].
- $(\mathcal{C}, \subseteq)$ Is a product of chains: Quasi-Polynomial [Elb09]
- *A* is the set of basis of a matroid: Polynomial [EMR09]
- $(\mathcal{C}, \subseteq)$ is a lattice: *coNP*-complet [BK11].
- $(\mathcal{C}, \subseteq)$ is a distributive lattice: OPEN.

Outline

Context

Example (Frequent Itemset Mining (Agrawal et al. [AIS93]))

- Let $\mathcal{I}$ a set of objects and λ the minimum support threshold
  - $\mathcal{D}$: a transaction database $\mathcal{T}$ ($t \in \mathcal{T}$, $t \subseteq \mathcal{I}$)
  - $\mathcal{L} = 2^{\mathcal{I}}$
  - $p(\Phi, \mathcal{D}) \Leftrightarrow |\{t \in \mathcal{T} \mid \Phi \in \mathcal{L}, \Phi \subseteq t\}| \geq \lambda$ (Frequency constraint)

Example

- $\mathcal{I} = \{pain, jus, fromage, yaourt\}$
- $\mathcal{T} = \{\{pain, fromage, yaourt, jus\}, \{yaourt, jus\}\}$
- for $\lambda = 2$, $\{\{yaourt\}, \{jus\}, \{yaourt, jus\}\}$ are **frequent** itemsets (patterns)
- $\{yaourt, jus\}$ is **maximal** (another constraint)

Motivations

Constraint-based data mining,

- A large number of constraints have been defined
- Several data mining systems have been designed

- difficulty to add new constraints (e.g. maximal and frequent, ...)
- often require new implementations

Challenge: Design of declarative, efficient and generic data mining systems

A constraint programming framework for DM [Luc De Raedt et al. [RGN08]]

A first declarative approach for data mining based on constraint programming

- Models and solves a wide variety of constraint based itemset mining tasks (frequent, maximal, closed, cost-based, discriminative...)
- CP4IM implementation (`http://dtai.cs.kuleuven.be/CP4IM/`) using one of the well known CP systems (Gecode library [Sch] `http://www.gecode.org/`)
- Demonstrates the feasibility of the approach with respect to specialized data mining systems

Declarative approaches for Data mining

New research issue initiated by Luc De Raedt group

- Several recents publications
- A Dagstuhl seminar "Constraint programming meets machine learning and data mining"
- An international workshop on "declarative pattern mining" (to be held in conjunction with ICDM'2011 conference)

Outline

Constraint programming (CP)

One of the most popular AI model for solving combinatorial problems (e.g. scheduling, planning, configuration)

- **Declarative**: the user specify how the problem is modeled and a general search engine is then used to find solutions
  - The problem is modeled as constraint system
  - The solver search for a solution, all solutions or optimal solutions
- **Generic**: general solving paradigm (search + propagation)
- **Efficient**: widely used for solving a variety of real world problems

Constraint programming

Definition (Constraint satisfaction problem (CSP))

Let,

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of **variables**, with their associated finite **domains** $D(x_1), \ldots, D(x_n)$
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a set of **constraints** defined on subsets of $\mathcal{X}$
  - $C_j(x_{k_1}, \ldots, x_{k_{n_j}}) : D(x_{k_1}) \times \cdots \times D(x_{k_{n_j}}) \to \{0, 1\}$

decide if there exists a valuation ρ s.t. $\rho(x_i) \in D(x_i)$ and $\rho \models C_1 \wedge \cdots \wedge C_n$.
We say that ρ is a *model or solution* of the CSP.

CP: modeling

Different kind of constraints:

- All tutorials must be scheduled at different time-slots (all different constraint)
- Number of students must be less than a given capacity limit (inequality constraint)
- ...

Example (Crypto-arithmetic example)

$SEND + MORE = MONEY$

- Variables: V = [S, E, N, D, M, O, R, Y]
- Domains: domain([E, N, D, M, O, R, Y], 0, 9), domain([S, M], 1, 9),
- Constraints:
  - $1000 \times S + 100 \times E + 10 \times N + D \quad +$
    $1000 \times M + 100 \times O + 10 \times R + E \quad =$
    $1000 \times M + 100 \times N + 10 \times E + Y$
  - $all\_different(Sol)$
- Search: $labbeling(Sol)$ $\qquad Sol = [9, 5, 6, 7, 1, 0, 8, 2]$

CP: Search

- **Propagation** (deterministic): eliminates values from the domains of the variables
  - $D_x = \{3, 4, 5\}$, $D_y = \{0, 1, 2, 3, 4\}$, $C_1 : x \leq y$
  - $D_x \rightarrow \{3, 4, \not{5}\}$, $D_y \rightarrow \{\not{0}, \not{1}, \not{2}, 3, 4\}$
  - *Propagator for* $x \leq y$:
    - if $D(x) = v$, and $v \geq max_{d \in D(y)}$ then delete v from $D(x)$
    - if $D(y) = v$, and $v \leq min_{d \in D(x)}$ then delete v from $D(y)$
- **Branching** (non-deterministic):
  - recursively select and instantiate a variable to a value (e.g. recursive call with $x = 3$ and with $x = 4$)

CP: Backtrack search algorithm

Algorithm 1 Constraint-Search(D)

```
D := propagate(D)
if D is a false domain then
    return
end if
if ∃x ∈ 𝒱 : |D(x)| > 1 then
    x := arg min_{x∈𝒱, D(x)>1} f(x)
    for all d ∈ D(x) do
        Constraint-Search(D ∪ {x ↦ {d}})
    end for
else
    Output solution
end if
```

Constraint programming

The constraint programming model includes several,

- kind of constraints and propagators (e.g. a catalogue of more than 2 hundreds of global constraints)
- enhancements of the backtrack search algorithm (e.g. search heuristics, non-chronological backtracking and nogoods recording)

For a survey see,

- Books:
  - Constraint Processing, by Rina Dechter (editor), Morgan Kaufmann, 450 pages, 2003
  - Handbook of Constraint Programming, by Francesca Rossi, Peter van Beek and Toby Walsh, Elsevier, 978 pages, 2006
- Links:
  - Association for Constraint Programming (ACP): `http://4c110.ucc.ie/acp/a4cp/`
  - Constraints archive: `http://4c.ucc.ie/web/archive/`
  - International conference on constraint programming (CP)

Boolean Satisfiability (SAT)

- Given a CNF formula $\mathcal{F}$

  $(a \vee b \vee c) \wedge (\neg a \vee b) \wedge (\neg b \vee c) \wedge (\neg c \vee a)$

- $\mathcal{F}$ admits a model?
  - $\mathcal{F}$ is satisfiable : $\{a = true, b = true, c = true\}$ is a model
  - $\mathcal{F} \cup \{(\neg a \vee \neg b \vee \neg c)\}$ is unsatisfiable

- Bad news: SAT is NP-Complete [Cook 71]
- Good news : Modern SAT solvers can solve instances with millions of variables and clauses in few seconds!
  $\Rightarrow$ Widely used in formal verification, planning, bioinformatics, cryptography, ...

An exemple : post-cbmc-zfcp-2.8-u2.cnf

p cnf **11 483 525** (vars) **32 697 150** (clauses)

1 -3 0
2 -3 0 ← $x_1 = \wedge(x_2, x_3)$
-1 -2 3 0
. . .
. . .
-11482897 -11483041 -11483523 0
11482897 11483041 -11483523 0
11482897 -11483041 11483523 0 ← $(x_3 \Leftrightarrow x_2 \Leftrightarrow x_3)$
-11482897 11483041 11483523 0
-11483518 -11483524 0
-11483519 -11483524 0
-11483520 -11483524 0
-11483521 -11483524 0 ← $x_6 = \wedge(x_7, x_8, x_9, x_{10}, x_{11}, x_{12})$
-11483522 -11483524 0
-11483523 -11483524 0
11483518 11483519 11483520 11483521 11483522 11483523 11483524 0
-8590303 -11483524 -11483525 0
8590303 11483524 -11483525 0
8590303 -11483524 11483525 0 ← $(x_{13} \Leftrightarrow x_{14} \Leftrightarrow x_{15})$
-8590303 11483524 11483525 0
-11483525 0

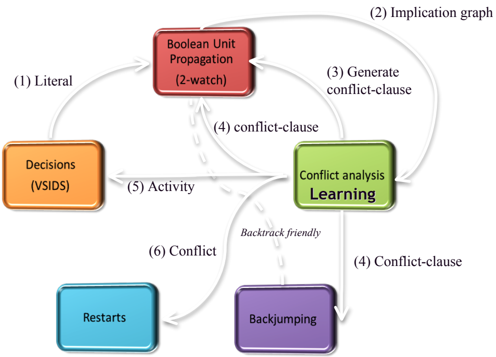
Solved in less than 1 minute [Talk by Carla Gomes]

Modern SAT solvers: four basic bricks

1. Heavy tailed phenomena: Gomes et al. [GSC97] $\rightarrow$ Restarts
2. Resolution based conflict analysis: Marques Silva et al. [MSS96] $\rightarrow$ Learning
3. Activity-based variable ordering: [Brisoux et al. [BGS99], Moskewicz et al. [MMZ$^+$01] $\rightarrow$ efficient heuristics
4. Watched literals: [H. Zhang el al. [Zha97], Moskewicz et al. [MMZ$^+$01] $\rightarrow$ Efficient BCP

- Four component proposed in Four years

Modern SAT solvers: architecture



[Source: Talk L. Bordeaux and Y. Hamadi]

Definitions and notations

- <u>CNF</u> : $\mathcal{F} = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_3)$
- <u>Partial interpretation</u> : $\rho : X \subseteq \underline{\mathcal{V}(\mathcal{F})} \rightarrow \{faux, vrai\}$
- <u>Simplification</u> : $\mathcal{F}|_\rho$ denotes the formula simplified by ρ
- <u>Implication</u> : $\overrightarrow{imp}(x_3) = (x_1 \wedge x_2 \rightarrow x_3)$, $\overrightarrow{exp}(x_3) = \{x_1, x_2\}$
- <u>Formula $\mathcal{F}$ closed by UP</u> : $\mathcal{F}^* = (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$
- <u>Resolvent</u> : $\eta[x_2, (\neg x_1 \vee x_2), (\neg x_2 \vee \neg x_3)] = (\neg x_1 \vee \neg x_3)$
- <u>Logical consequence</u> : $\mathcal{F} \models (\neg x_1 \vee \neg x_3)$

Conflict Driven Clause Learning (CDCL)

$\mathcal{F} \supseteq \{c_1, \ldots, c_9\}$

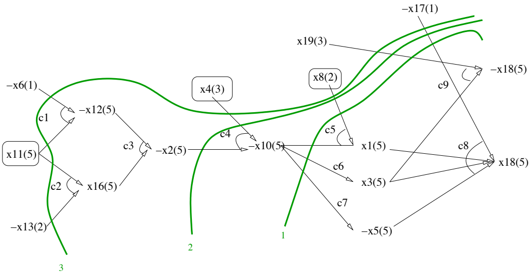
(c_1)	$x_6 \vee \neg x_{11} \vee \neg x_{12}$	(c_2)	$\neg x_{11} \vee x_{13} \vee x_{16}$
(c_3)	$x_{12} \vee \neg x_{16} \vee \neg x_2$	(c_4)	$\neg x_4 \vee x_2 \vee \neg x_{10}$
(c_5)	$\neg x_8 \vee x_{10} \vee x_1$	(c_6)	$x_{10} \vee x_3$
(c_7)	$x_{10} \vee \neg x_5$	(c_8)	$x_{17} \vee \neg x_1 \vee \neg x_3 \vee x_5 \vee x_{18}$
(c_9)	$\neg x_3 \vee \neg x_{19} \vee \neg x_{18}$		

Notations: x_i^j literal x_i assigned at level j.

$\rho = \langle \ldots \neg x_6^1 \ldots \neg x_{17}^1 \rangle \langle (x_8^2) \ldots \neg x_{13}^2 \ldots \rangle \langle (x_4^3) \ldots x_{19}^3 \ldots \rangle \ldots$

$\langle (x_{11}^5), \neg x_{12}^5, x_{16}^5, \neg x_2^5, \neg x_{10}^5, x_1^5, x_3^5, \neg x_5^5 \rangle$

Classical Learning



$$\Delta_1 = \eta[x_{18}, c_9, c_8] = (\neg x_{19}^3 \vee x_{17}^1 \vee x_1^5 \vee x_3^5 \vee x_5^5)$$

$$\Delta_2 = \eta[x_5, \Delta_1, c_7] = (\neg x_{19}^3 \vee x_{17}^1 \vee x_1^5 \vee x_3^5 \vee x_{10}^5)$$

$$\Delta_3 = \eta[x_3, \Delta_2, c_6] = (\neg x_{19}^3 \vee x_{17}^1 \vee x_1^5 \vee x_{10}^5)$$

$$\underline{\mathcal{A}_1} = \eta[x_1, \Delta_3, c_5] = (\neg x_{19}^3 \vee x_{17}^1 \vee \neg x_8^2 \vee x_{10}^5) \Leftarrow \text{Asserting Clause (AC in short)}$$

Modern SAT solver Vs resolution

- CDCL: Marques Silva et al. [MSS96], Moskewicz et al. [MMZ+01]
  is a fundamental component of Modern SAT solvers
  - **Modern SAT solvers**: $\approx$ **General resolution** , Knot et al. [PD09]
  - **DPLL-like solver**: $\approx$ **Tree-Like resolution**

Propositional Satisfiability

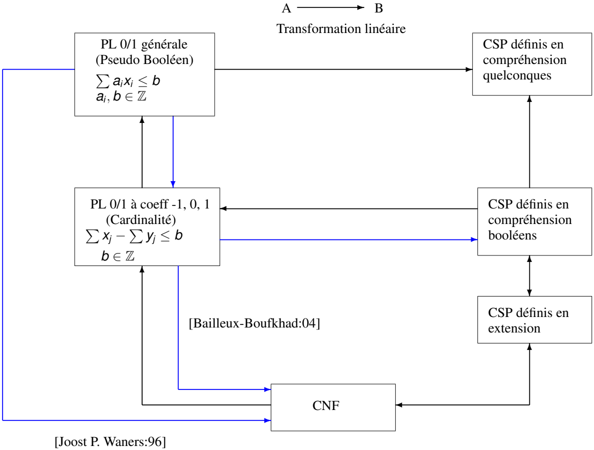
For a survey on propositional satisfiability see,

- Books:
  - Problème SAT : Progrès et Défis, by Lakhdar Sais (editor), Hermes Publishing Ltd, 352 pages, may 2008
  - Handbook of satisfiability, by Armin Biere et al. (editor), IOS Press, 980 pages, february 2009
- Links:
  - SatLive: `http://www.satlive.org/`
  - SAT competition: `http://www.satcompetition.org/`
  - International Conference on Theory and Application of Satisfiability Testing (SAT)

CSP, SAT and PL-(0/1): Summary

	SAT	CSP	PL 0/1
Var.	Bivaluées (0/1)	Multi-Valuées	Bivaluées (0/1)
Contr.	$(x_1 \vee \neg x2 \vee x_3)$	*T*able *P*(rédicats) *G*(lobales) ...	$\sum_{i=1}^{k} a_i x_i \leq b$ $a_i, b \in \mathbb{Z}$
Forme normale	Oui	Non	Oui
Extensions	MaxSAT, W-MaxSAT QBF, #SAT	Max-CSP, WCSP, QCSP,	PLNE

SAT, CP and PL-01: Summary



[Source Bahia Project, PRC IA, 1992]

Outline

PC - Pattern discovery modelisation

A naive approach for pattern discovery:

- 1 variable x_Φ with domain $\mathcal{L}$
- Constraints encoding the database $\mathcal{D}$ and the predicate p
  - how to achieve propagation
- the set of interesting patterns is derived thanks to an exhaustive enumeration of the CSP solutions.

Frequent Itemset Mining (FIM) [De Readt et al. KDD'2008]

Variables:

- the pattern Φ is represented by $|\mathcal{I}|$ boolean variables I_i ($D(I_i) = \{0, 1\}$).
  $\rightarrow$ $I_i = 1$ if the item i appears in the pattern Φ
- For each transaction $t \in \mathcal{T}$, we associates a boolean variable T_t ($D(T_t) = \{0, 1\}$).
  $\rightarrow$ $T_t = 1$ if the transaction t contains Φ

Frequent Itemset Mining (FIM) [De Readt et al. KDD'2008]

Constraints:

- Notation: $D_{ti} = 1$ iff the transaction t contains the item i
- Constraints
  - Exact covering: $\forall t \in \mathcal{T}, T_t = 1 \Leftrightarrow t \supseteq \Phi$
    - $\forall t \in \mathcal{T}, T_t = 1 \Leftrightarrow \sum_{i \in \mathcal{I}} I_i(1 - D_{ti}) = 0$
  - Frequency: $\sum_{t \in \mathcal{T}} T_t \geq s$
    - $\forall i \in \mathcal{I}, I_i = 1 \Rightarrow \sum_{t \in \mathcal{T}} T_t D_{ti} \geq s$

For more details see [Tutorial by De Readt]

Itemset Mining - other variations

Flexibility of the Constraint programming for encoding variations of the problem:

- Maximal:

$$\forall i \in \mathcal{I}, I_i = 1 \Leftrightarrow \sum_{t \in \mathcal{T}} T_t D_{ti} \geq s$$

- Closed: frequency +

$$\forall i \in \mathcal{I}, I_i = 1 \Leftrightarrow \sum_{t \in \mathcal{T}} T_t (1 - D_{ti}) = 0$$

- Maximal / Minimal cost:

$$\sum_{i \in \mathcal{I}} c_i I_i \leq cmax \qquad \sum_{i \in \mathcal{I}} c_i I_i \geq cmin$$

- Minimal average cost:

$$\sum_{i \in \mathcal{I}} (c_i - cmin) I_i \geq 0$$

Outline

CP/SAT for Sequence Mining

A first Constraint Programming Approach for Enumerating Motifs in a Sequence

Joint work between LIRIS (E. Coquery) and CRIL (S. Jabbour and L. Saïs)

International Workshop on Declarative Pattern Mining (held in conjunction with ICDM 2011) [CJS11]

Important remarks:

- Sequence patterns are not "representable as sets", i.e. a one-to-one mapping between the set of sequence patterns and a Boolean lattice does not exist
- Classical set-oriented algorithms (e.g. "Dualize and Advance") can not be applied

Preliminary definitions

Definition (Sequence)

Let Σ be an alphabet, st. $\circ \notin \Sigma$ ($\circ$ is called a wildcard). A sequence S is a string of Σ^* i.e. $S = S_1 S_2 \ldots S_n \in \Sigma^*$. The set of position is denoted by $O = \{1 \ldots n\}$.

Definition (Pattern)

A pattern is a string $M = M_1 M_2 \ldots M_m \in (\Sigma \cup \{\circ\})^*$ st. $m \leq n$ and $M_1 \neq \circ$ et $M_m \neq \circ$

Definition (Inclusion)

Let $S = S_1 S_2 \ldots S_n$ be a sequence and $M = M_1 M_2 \ldots M_m$ a pattern. We say that M appears in S at position $p \in O$ denoted $M \subseteq_p S$, if $\forall i \in O$, we have $M_i = S_{p+i-1}$ or $M_i = \circ$. We note $L_S(M) = \{p \in O | M \subseteq_p S\}$.
We say that $M \subseteq S$ iff $\exists p \in O$ st. $M \subseteq_p Q$.

Sequence Mining Problem

Definition (Sequence Mining Problem (SMP))

The sequence mining problem is defined as follows:
Input: A sequence S and a quorum λ
Output: All frequent patterns (motifs) M of S st. $|L_S(M)| \geq \lambda$

In the sequel, we limit (without loss of generality) to patterns of fixed maximal size m.

Property (Anti-monotonicity)

Let M_1 and M_2 be two patterns ofS with $M_1 \subseteq M_2$. If $|L_S(M_2)| \geq \lambda$ then $|L_S(M_1)| \geq \lambda$.

CP model of SMP : Variables

- M_i $(1 \leq i \leq m)$ represent the ith symbol of the candidate motif M. The domain of M_i is $\Sigma \cup \{\circ\}$.
- P_k $(1 \leq k \leq n)$ *true* $(= 1)$ if the motif M appears at position k in S; *false* otherwise.

An instantiation of $M_1 \ldots M_m$ to $a_1 \ldots a_m$ represents the motif $a_1 \ldots a_l$ s.t. $a_l \neq \circ$ and $\forall i$, if $l < i \leq m$ then $a_i = \circ$.

- l is the last position of a solid character (symbol different from $\circ$) in $a_1 \ldots a_m$.
- An instantiation of $M_1 \ldots M_6$ to $a \circ b \circ \circ\circ$ represents the motif $a \circ b$.
- We add $m - 1$ $\circ$ at the end of S.

The set of variables P_k for $1 \leq k \leq n$ represents the support of M.

CP model of SMP: Constraints

M appears in S at position k :

$$inc(k, M, S) = \bigwedge_{i=1}^{m} (M_i = \circ \vee S_{k+i-1} = M_i)$$

Inclusion of M at each position k in S :

$$support(M, S) = \bigwedge_{k=1}^{n} (P_k \Leftrightarrow inc(k, M, S))$$

The frequency constraint is then defined as follows:

$$freq(S) = \sum_{k=1}^{n} P_k \geq \lambda$$

We also add the unary constraint : $M_1 \neq \circ$.

The Constraint Satisfaction Problem (CSP)

The Sequence Mining Problem is defined by the following CSP $\mathcal{P} = (\mathcal{V}, \mathcal{C})$:

- $\mathcal{V} = \{M_i | 1 \leq i \leq m\} \cup \{P_k | 1 \leq k \leq n\}$
- $\mathcal{C} = support(M, S) \wedge freq(S) \wedge M_1 \neq \circ$

The set of solutions of $\mathcal{P}$ corresponds to the set of frequent patterns (motifs) of S with maximal size m.

Propositional Satisfiability (SAT) encoding

Encoding the problem as a Boolean formula to benefit from

- The clause learning component (anti-monotonic property)
- The recent progress in Satisfiability testing

Propositional Satisfiability (SAT) encoding

- **Boolean variables**
  - for each M_i we associate $|\Sigma| + 1$ boolean variables $\{M_i^c \mid c \in \Sigma \cup \{\circ\}\}$. These variables constitute a *strong backdoor set* .
  - The other variables P_k are Boolean.
- **Clauses** are obtained as follows:
  - *Domains encoding*: expresses that a given variable M_i must be assigned to exactly one value from $\Sigma \cup \{\circ\}$
  - *Constraints encoding*: the support constraint is a boolean formula. For the frequency constraint there exists efficient CNF encoding [Bailleux 06, 09, Warners 96]
    - encoded with a binary adder
    - linear in the size of the frequency constraint.
    - It is also possible to natively integrate the frequency constraint: pseudo boolean, SAT Modulo Theory

SAT: anti-monotonic property encoding

The integration of no-goods is natural in SAT (Learning component)

- The SAT solver generates its own no-goods (leant clauses) $\rightarrow$ express possible interesting properties ?

Anti-monotonic constraints

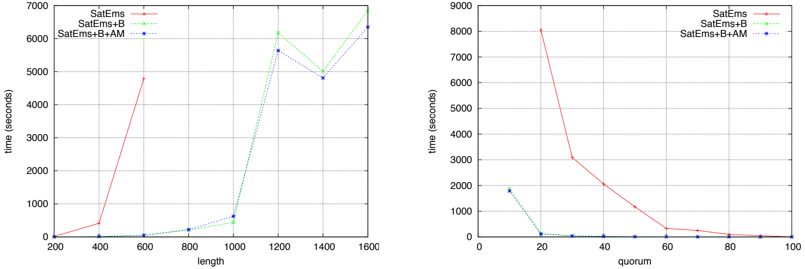
- M' proved non frequent (no-good) $\rightarrow$ Eliminates all futures motifs M s.t. $M' \subseteq M$.
- Let $M' = M'_1 M'_2 \dots M'_m$ and $\{i_1, \dots i_l\}$ the ordered set of positions of M' s.t. $\forall j \in \{1 \dots l\}, M'_{i_j} \neq \circ$.

$$antiMon(M', M) = \bigwedge_{x=1}^{m-i_l+1} \bigvee_{y=1}^{l} (M'_{i_y} \neq M_{i_y+x-1})$$

First experiments

- The CNF Boolean formula is generated using a Java platform, and solved with a modified modern SAT solver MiniSAT [ES05]:
  - Search for all solutions
  - generation of the anti-monotone no-goods
  - integration of the strong backdoor set
- Real world data
  - Bioinformatics (proteinic sequence of amino-acid)
  - computer security (command history of UNIX computer users)

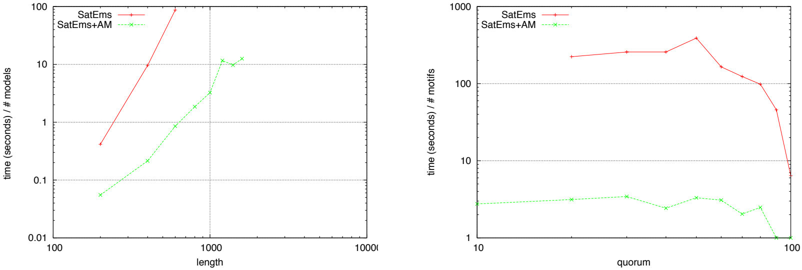
Impact of the strong backdoor and anti-monotone no-goods



Motifs extraction time Vs size and quorum

- the integration of strong backdoor is crucial
- limited impact of anti-monotone no-goods no-goods
  - huge number of no-goods ?
  - most of them are redundant % unit propagation?

Promising results



Extraction time *per motif* wrt. size and quorum

Several Perspectives

- Improve the efficiency CP/SAT model for mining itemsets and sequences
- Pseudo boolean and/or SAT modulo Theory models ?
- Define high declarative language (logic or algebraic) for Data mining
- How about other kind of complex patterns (graphs, trees, . . .)

Outline

Conclusion

- Declarative approaches in data mining
  - CP/SAT ++
    - easier to modify constraints than patching C++ code !
    - allows rapid prototyping of data mining algorithms
    - efficient for more constrained problems (e.g. top-k)
  - CP/SAT –
    - less efficient than specialized implementations,
    - What about the level of declarativity ?
  - DB++
    - driven by the "elephants" and the market
  - DB –:
    - not fully integrated with SQL [STA98]

Conclusion

Some tentatives, not fully successful yet

neither in academia (US gurus don't like it!) nor in industry (from a clean and theoretical point of view)

DAG website: `http://liris.cnrs.fr/dag/`

R. Agrawal, T. Imielinski, and A. Swami.
Mining associations between sets of items in massive databases.
In *Proceedings of the International Conference on Management of Data, ACM-SIGMOD, Washington D.C.*, pages 207–216, 1993.

R. Agrawal and R. Srikant.
Fast algorithms for mining association rules in large databases.
In *Proceedings of the 20th International Conference on Very Large Databases, Santiago de Chile, Chile*, pages 487–499, 1994.

Hiroki Arimura and Takeaki Uno.
Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems.
In *SDM*, pages 1087–1098, 2009.

Elena Baralis, Tania Cerquitelli, and Silvia Chiusano.
Index support for frequent itemset mining in a relational dbms.
In *ICDE*, pages 754–765, 2005.

Hendrik Blockeel, Toon Calders, Elisa Fromont, Bart Goethals, Adriana Prado, and Celine Robardet.
An inductive database system based on virtual mining views.
Data Mining and Knowledge Discovery, to appear, 2011.

Laure Brisoux, Éric Grégoire, and Lakhdar Sais.
Improving backtrack search for sat by means of redundancy.
In *ISMIS*, pages 301–309, 1999.

Mikhail A. Babin and Sergei O. Kuznetsov.
Enumerating minimal hypotheses and dualizing monotone boolean functions on lattices.
In *ICFCA*, pages 42–48, 2011.

Surajit Chaudhuri.
Data mining and database systems: Where is the intersection?
Data Engineering Bulletin, 21(1):4–8, 1998.

Vineet Chaoji, Mohammad Al Hasan, Saeed Salem, and Mohammed Javeed Zaki.

An integrated, generic approach to pattern mining: data mining template library.
Data Min. Knowl. Discov., 17(3):457–495, 2008.

Emmanuel Coquery, Said Jabbour, and Lakhdar Sais.
A constraint programming approach for enumerating motifs in a sequence.
In *International workshop on declarative pattern mining (in conjunction with IEEE- ICDM'2011 conference)*, Vancouver, Canada, december 2011.

Toon Calders and Jef Wijsen.
On monotone data mining languages.
In *DBPL*, pages 119–132, 2001.

Khaled M. Elbassioni.
Algorithms for dualization over products of partially ordered sets.
SIAM J. Discrete Math., 23(1):487–510, 2009.

Khaled M. Elbassioni, Kazuhisa Makino, and Imran Rauf.
Output-sensitive algorithms for enumerating minimal transversals for some geometric hypergraphs.
In *ESA*, pages 143–154, 2009.

N. Een and N. Sörensson.
Minisat - a sat solver with conflict-clause minimization.
In *SAT 2005*, 2005.

Frédéric Flouvat, Fabien De Marchi, and Jean-Marc Petit.
Advanced Techniques for Data Mining and Knowledge Discovery, chapter The iZi project: easy prototyping of interesting pattern mining algorithms, pages 1–15.
LNCS. Springer-Verlag, September 2009.

Michael L. Fredman and Leonid Khachiyan.
On the complexity of dualization of monotone disjunctive normal forms.
J. Algorithms, 21(3):618–628, 1996.

Lujun Fang and Kristen LeFevre.
Splash: ad-hoc querying of data and statistical models.
In *EDBT*, pages 275–286, 2010.

Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm.
Discovering all most specific sentences.
ACM Trans. Database Syst., 28(2):140–174, 2003.

Arnaud Giacometti, Patrick Marcel, and Arnaud Soulet.
A relational view of pattern discovery.
In *DASFAA*, pages 153–167, 2011.

Carla P. Gomes, Bart Selman, and Nuno Crato.
Heavy-tailed distributions in combinatorial search.
In *CP*, pages 121–135, 1997.

Bin He, Kevin Chen-Chuan Chang, and Jiawei Han.
Discovering complex matchings across web query interfaces: a correlation mining approach.
In *KDD*, pages 148–157, 2004.

Jiawei Han, Yongjian Fu, Wei Wang, Krzysztof Koperski, and Osmar Zaiane.
Dmql: A data mining query language for relational databases.
In *Workshop. on Research Issues on Data Mining and Knowledge Discovery (DMKD'96*, pages 27–33, 1996.

Tomasz Imielinski and Heikki Mannila.
A database perspective on knowledge discovery.
Commun. ACM, 39(11):58–64, 1996.

Tomasz Imielinski and Aashu Virmani.
Msql: A query language for database mining.
Data Min. Knowl. Discov., 3(4):373–408, 1999.

Hélène Jaudoin, Frédéric Flouvat, Jean-Marc Petit, and Farouk Toumani.
Towards a scalable query rewriting algorithm in presence of value constraints.
J. Data Semantics, 12:37–65, 2009.

Hong-Cheu Liu, Aditya Ghose, and John Zeleznikow.
Towards an algebraic framework for querying inductive databases.
In *DASFAA (2)*, pages 306–312, 2010.

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an efficient SAT solver.
In *Proceedings of the 38th Design Automation Conference (DAC'01)*, pages 530–535, 2001.

Joao P. Marques-Silva and Karem A. Sakallah.
GRASP - A New Search Algorithm for Satisfiability.
In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227, 1996.

H. Mannila and Hannu Toivonen.
Levelwise search and borders of theories in knowledge discovery.
Data Mining and Knowledge Discovery, 1(3):241–258, 1997.

Siegfried Nijssen and Luc De Raedt.
Iql: A proposal for an inductive query language.
In *KDID*, pages 189–207, 2006.

Carlos Ordonez and Sasi K. Pitchaimalai.
One-pass data mining algorithms in a dbms with udfs.
In *SIGMOD Conference*, pages 1217–1220, 2011.

Knot Pipatsrisawat and Adnan Darwiche.
On the power of clause-learning sat solvers with restarts.
In *CP*, pages 654–668, 2009.

Luc De Raedt, Tias Guns, and Siegfried Nijssen.
Constraint programming for itemset mining.
In *KDD*, pages 204–212, 2008.

Andrea Romei and Franco Turini.
Inductive database languages: requirements and examples.
Knowl. Inf. Syst., 26(3):351–384, 2011.

Christian Schulte.
Gecode: An open constraint solving library.
http://www.gecode.org/.

Sunita Sarawagi, Shiby Thomas, and Rakesh Agrawal.
Integrating mining with relational database systems: Alternatives and implications.
In *SIGMOD Conference*, pages 343–354, 1998.

Manolis Terrovitis, Panos Vassiliadis, Spiros Skiadopoulos, Elisa Bertino, Barbara Catania, Anna Maddalena, and Stefano Rizzi.
Modeling and language support for the management of pattern-bases.
Data Knowl. Eng., 62(2):368–397, 2007.

Stefan Wrobel.
Inductive logic programming for knowedge discovery in databases.
pages 74–99, 2000.

Hantao Zhang.
SATO: An efficient propositional prover.
In William McCune, editor, *Proceedings of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, 1997. Springer Verlag.