Pattern Mining using Constraint Programming

Tias Guns

BUTO, VUB Brussel
http://homepages.vub.ac.be/~tiasguns/

CPAIOR Masterclass, June 2018
Data Mining

“Extracting useful information from data”

- Machine learning: extracting predictive models
- Clustering: extracting meaningful groups
- **Pattern mining:** extracting regularities
- Recommender systems: extracting preferences
Itemset mining

Transactions:

1) Doritos, Fritos, Lay's, Pringles
2) Oreo, KitKat
3) Apple, Coca-Cola
4) Doritos, Fritos, Lay's, Pringles
5) Oreo, KitKat
6) Doritos, Fritos, Lay's, Pringles
7) Pampers, Tide, Crest, Oral-B
8) Corona, Budweiser
9) Doritos, Fritos, Lay's, Pringles
10) Oreo, KitKat
11) Apple, Coca-Cola
12) Pampers, Tide, Crest, Oral-B
Biological sequence mining

Sequence: 1

Sequence: 10

Sequence: 11

Sequence: 12

Sequence: 13

Sequence: 14

Sequence: 15
(Discrete) Data mining: methods

Usually specific algorithms for specific problems

Highly scalable, but;

- New problems rarely fit existing methods well
- Tedious programming & hacks
- Refining solution methods is hard, even though typical in the knowledge discovery cycle
Constraint Solving

“Solving constraint satisfaction/optimization problems”

- Scheduling
- Routing
- Configuration
- Graph problems
Constraint solving: methods

“Combinatorial problem = Model + Solve”

**Model** = specification of constraints over variables

**Solve** = search for satisfying/optimal solutions

Many generic and efficient solvers available
Constraint solving: why data mining?

→ many DM problems are combinatorial problems

Modeling

+ Adding/removing/combining constraints
+ Complex constraints
  - Modeling choices matter

Solving

+ Reusing solving technology
+ Exhaustive, optimal
  - Scalability towards large datasets
Active research directions

- **Pattern Mining** B. Creilleux, L. De Raedt, T. Guyet, S. Jabbour, M. Jarvisalo, A. Kemmar, S. Loudni, S. Nijssen, B. O'Sullivan, P. Schaus, ...

- **Clustering** I. Davidson, T.B.H. Dao, K.C. Duong, S. Gilpin, P. Stuckey, P. Hansen, O. du Merle, A. Monreale, S. Nijssen, C. Vrain, ...

- **Structure learning** C. Bessiere, J. Cussens, O. Grinchtein, M. Heule, T. Jaakkola, M. Meila, B. O'Sullivan, D. Sontag, P. Van Beeck, S. Verwer, ...
In this talk

**Modeling:** generality

I. Itemset mining and constraints
II. A modeling language for constraint-based mining?

**Solving:** efficiency

III. Scalability of generic itemset solving
IV. Sequence mining and constraints
Itemset mining

Transactions:

1) {Doritos} (42%)
2) {Doritos, Oreo} (58%)
3) {Doritos, Oreo} (50%)
4) {Pampers} (33%)
5) {Doritos, Oreo} (50%)
6) {Doritos, Oreo, Pampers} (42%)
7) {Pampers} (83%)
8) {Doritos} (33%)
9) {Budweiser} (33%)
10) {Pampers} (33%)
11) {Doritos, Oreo, Pampers} (42%)
12) {Budweiser, Pampers} (33%)
Constraint-based Itemset Mining

- Solution to pattern exploitation
- Fundamental enumeration problem
- Many constraints
- Many applications
“Interesting” patterns:

- which patterns appear frequently in a dataset?
- which patterns have a certain structure?
- which patterns have a high cost or profit margin?
- which patterns summarize a dataset?
- which patterns are frequent on one dataset and infrequent on another?
- which patterns are significant w.r.t. a background model?

→ specified by constraints

“Constraint-based mining”
Frequent Itemset Mining

**Find:** all sets of *items* appearing frequently

\[ \text{cover}(\{\text{book1, book2}\}) = \{\text{man}, \text{woman}\} \]

\[ \text{frequency}(\{\text{book1, book2}\}) = |\{\text{man}, \text{woman}\}| = 2 \]
CP for Itemset Mining

coverage: \( \forall T_t: T_t = 1 \iff set(I_1, \ldots, I_n) \subseteq set(row_t) \)

frequency: \( \sum_t T_t \geq Freq \)

One solution = one frequent itemset: enumerate all
CP for Itemset Mining

coverage: \[ \forall T_t: T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0 \]

frequency: \[ \forall I_i: I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq Freq \]

[L. De Raedt, T. Guns, S. Nijssen, KDD 2008]
## Generality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints on data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum frequency</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Maximum frequency</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Emerging patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Condensed Representations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximal</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Closed</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\delta-$Closed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Constraints on syntax</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max/Min total cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Minimum average cost</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max/Min size</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Constraints on labelled data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum correlation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Maximum correlation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

[L. De Raedt, T. Guns, S. Nijssen, AAAI 2010]
Few constraints

Many constraints

Coverage + frequency

Specialized systems

Minimum support

MaxAvgCost

Runtime (s)
Take away message 1.

Constraint Programming for Itemset Mining framework:

- Mathematical and reasonably compact encoding
- Generic: many constraints can be expressed
- Effective in case of tight constraints

Many extensions (not in this talk):

- **Pattern set mining**  
  [T. Guns, S. Nijssen, L. De Raedt, TKDE 2013]  
  [A. Ouali, S. Loudni, Y. Lebbah, P. Boizumault, A. Zimmermann, L. Loukil, IJCAI 2016]

- **Skypatterns / multi-objective**  
  [W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, ECAI 2014]  
  [W. Ugarte, P. Boizumault, B. Crémilleux, A. Lepailleur, S. Loudni, M. Plantevit, C. Raïssi, A. Soulet, AIJ 2017]

- **SAT, BDD, ASP solvers**  
  [S. Jabbour, L. Sais, Y. Salhi. AIJ 2017]  
  [H. Cambazard, T. Hadzi, B. O'Sullivan, ECAI 2010]  
  [M. Jarvisalo, LPNMR 2011]
In this talk

Modeling
   I. Itemset mining and constraints
   II. A modeling language for constraint-based mining?

Solving
   III. Scalability of generic itemset solving
   IV. Sequence mining and constraints
A modeling language for pattern mining?

A long standing dream...

Many have roots in \textbf{Inductive Databases'} idea of Heikki Mannila → a database where \textit{data} and \textit{patterns} are both easily queried

Projects integrating mining in SQL:

- MINE RULE [Meo et al, 1996]
- MSQL [Imlieinksi & Virmani, 1999]
- Mining Views [Blockeel et al, 2012]

Mostly: mining algorithm parameters ↔ query parameters
A modeling language for pattern mining?

A long standing dream...

Others looked at constraint-based languages

- Levelwise [Mannila & Toivonen 1997]
- MusicDFS [Soulet & Cremilleux 2005]
- ConQueSt [Bonchi & Lucchese 2007]

Mostly: based on (anti-)monotonicity of constraints

Little support for other constraints (closed, maximal, discriminative) or combinations.
Modeling languages in CP

Constraint Programming has long tradition of modeling languages

- ECLiPSe and B-prolog (Constraint Logic Programming)
- OPL [Van Hentenryck, 1999]
- COMET [Van Hentenryck and Michel, 2005]
- MiniZinc [Nethercote et al, 2007]
- Essence [Frisch et al, 2008]

→ CP languages as starting point for pattern mining language
A modeling language for pattern mining?

MiningZinc

- Based on the established **MiniZinc language**
  - **High-level** mathematical-like notation
  - **User-defined** constraints and functions
  - **Solver independent** (10+ CP solvers & SAT & MIP)

- Modeling: pattern mining specific constrains and functions

- Solving: generic **AND specialised methods** (transparently)

Example: constrained itemset mining

```
include "lib_itemsetmining.mzn"

int: Nrl; int: NrT; int: MinFreq; int: MaxFreq;
array[1..NrT] of set of int: TDB;

var set of 1..Nrl: Items;

constraint card(cover(Items, TDB)) >= MinFreq;

constraint card(cover(Items, TDB)) <= MaxFreq;

array [1..Nrl] of int: Cost;
int: MinCost;

constraint sum(i in Items) (Cost[i]) >= MinCost

solve satisfy;
```
Toolchain for decomposition

1) Normalize to FlatZinc
2) Apply rewrite rules to:
   - add redundant constraints
   - detect (partial) applicability of specialized algorithms
   - tailor to constraint solvers
3) **Search** over all feasible rewrite combinations = execution plans
4) Heuristically rank + execute a plan

Experiments, hybrid solving

frequent itemset mining, with minimum size and closure constraint

Take away message 2.

Modeling:
- Can build on existing high-level CP languages
- Solver independence:
  - Automatic model rewriting
  - Automatic *chaining* of CP/DM algorithms: decomposition / hybridization
- But: not all constraint formulations compatible (closed-frequent vs closed-constr)

Open questions
- Multiple execution strategies: algorithm selection? parallelism?
- Problems not fitting standard CP
  - Skyline patterns / multi-objective
  - Dominance / preference over solutions / closed-constr

In this talk

Modeling

I. Itemset mining and constraints
II. A modeling language for constraint-based mining?

Solving

III. Scalability of generic itemset solving
IV. Sequence mining and constraints
Improving scalability?
IM search vs. CP search

- Highly efficient IM algorithms do depth-first search
- CP solver is at its core a principled depth-first search framework

**Algorithm 2** Constraint-Search($D$)

1: $D := \text{propagate}(D)$
2: if $D$ is a false domain then
3:     return
4: end if
5: if $\exists v \in V : |D(v)| > 1$ then
6:     $v := \arg \min_{v \in V, D(v) > 1} f(v)$
7:     $D_p := \text{split}(D(v))$
8:     Constraint-Search($D \cup \{ v \mapsto D_p \}$)
9:     Constraint-Search($D \cup \{ v \mapsto D - D_p \}$)
10: else
11:     Output solution
12: end if

What makes the difference?
Differences IM search / CP search

In pure CP model:

for each transaction a separate constraint

➔ data is split into many individual constraints (up to millions)
➔ CP has to do bookkeeping of constraints and its variables

in IM:

constraints are checked on entire data at once

➔ can use advanced data structures like vertical tidlists/bitsets
➔ can cache computations (e.g. frequency of each item)

→ keep data together in CP? ... global constraint
CP scalability

Wrote minimalistic CP solver that:

- implements standard generic CP search
- implements `BoolVector` variable type (bitwise computations)
- supports the generic global constraint $X \square (D \ A \approx B)$
  
  (cover, closed, maximal, minfreq, maxfreq rewrite to the above)

Within global constraint:

- can use (core of) same algorithms as specialised methods
- can do efficient bitwise computations and caching
Integrated solver

CP (original)

Splice (Closed)

CP (new solver)

T10I4D100K (Frequent)

[S. Nijssen, T. Guns, ECMLPKDD 2010]
In standard solver?

No bitvector variables, so no global with optimised datastructures

→ 1 million data rows = BoolVar array of size 1 million

Approach 1: hide all the $T_t$ transaction variables
- global constraint that does both 'cover' and 'minfreq'
- FrequentItemset(ItemVars, data, threshold)
  - only allows constraints on items (no maxfreq, closed, correlated, ...)

[ N. Lazaar, Y. Lebbah, S. Loudni, M. Maamar, V. Lemière, C. Bessiere, P. Boizumault:, CP 2016 ]

Approach 2: expose the frequency as a variable
- enforce bounds consistency on frequency variable
- CoverSize(ItemVars, data, FrequencyVar)
  - can add frequency constraints (maxfreq, correlated, ...) 

[P. Schaus, J. Aoga, T. Guns, CP 2017]
Within standard CP solver

CoverSize(ItemVars, data, FrequencyVar) global constraint

- Can use data structure and algorithms like specialised methods (Bitsets)

→ similarities to Compact-Table: Efficiently Filtering Table Constraints with Reversible Sparse Bit-Sets. [CP 2016]

- Can use backtrack-aware datastructure like state-of-the-art table propagator!

![Table and constraints]

[P. Schaus, J. Aoga, T. Guns, CP 2017]
Take away message 3.

Difference IM search / CP search

- Both use depth-first search
- IM is one big complicated algorithm
- CP decomposes problem in separate constraints

Increasing efficiency:

- Keep data together in a *global* constraint
- Can combine best of DM&CP in such constraints
- Efficiency versus generality trade-off! (*no closed/maximal...*)
In this talk

Modeling

I. Itemset mining and constraints
II. A modeling language for constraint-based mining?

Solving

III. Scalability of generic itemset solving
IV. Other pattern mining tasks
Sequential data

Example:

<Home, Work, Restaurant, Work, Home>
<Home, Work, Shops, Restaurant, Home>
...

Many applications:

- User mobility mining
- Web usage mining
- Event monitoring
- Biological sequence mining (DNA, Amino acids)
- ...
Sequence mining

T1 = <Home, Work, Restaurant, Gym, Work, Home>
T2 = <Home, Work, Home, Gym>
...

Example pattern P=<Home, Home>

- can have arbitrary symbols before/between/after
- ex: <Home, Work, Restaurant, Gym, Work, Home>

- Formally: \( T = 1 \iff \exists (e_1 \ldots e_n): e_1 < \ldots < e_n \land \forall j \ P[j] = X[e_j] \)

In CP... ?
in CP: add global constraint

\[
\sum_t T_t \geq \text{Freq}
\]

\[
\forall T_t: \quad T_t = 1 \iff \text{exist} - \text{embedding}(S, X_t)
\]

Global constraint with filtering algorithm \textit{like specialised methods}:

- \( T_t = 1 \iff \exists (e_1 \ldots e_n): \quad e_1 < \ldots < e_n \land \forall j \quad S[j] = X_t[e_j] \)

- Incremental: keep one pointer to \textit{last assigned match} \( e_j \)

[B. Negrevergne, T. Guns, CPAIOR 2015]
Keeping data together

One global constraint for **all** sequences (hide transaction variables):

- algorithmic improvements: last position map, last position list → precomputed and cached, speedups
- from CP community: use **backtracking-aware datastructure** → stores remaining transactions and their pointer in reversible vector

![Diagram showing sequence data and index positions](image)
Efficiency: outperforms specialised!
Other pattern mining settings

From itemsets to sequences to graphs?
  - enumerating frequent (constrained) subgraphs
  - cover of a subgraph: subgraph isomorphism checks (NP-complete)

Can use CP framework?
  - Representation: few CP solvers with 'graph' variables
  - needs cover and symmetry breaking globals and custom search strategy → few existing CP constraints relevant
  - unclear if beneficial from solving side

[T. Guns, S. Paramonov, B. Negrevergne. DLBP@AAAI 2016]
Current trends in pattern mining

Constrained pattern mining: one way to avoid **pattern explosion**

Increasingly, formulate as **set selection** problem to optimize:

- information theoretic measure: MDL, Maximum Entropy, ...
  → highly non-linear: predominantly greedy or heuristic techniques
- sampling (small set of diverse patterns)

Role of CP? **Yes!** as core component:

- Directly optimising MaxEnt over multi-relation itemsets in CP
  (iterative: propagation & branch-and-bound to find best addition)
  **[T. Guns, A. Aknin, J. Lijffijt, T. De Bie. ICDM 2016]**

- XOR-sampling with CP solver, also additional constraints:
  (iterative: use CP to handle XOR and other constraints)
  **[V. Dzyuba, M. van Leeuwen, L. De Raedt. DMKD 2017]**
Wrap-up

Finding the right level of abstraction:

• Modeling:
  – not query but set of constraints
  – can reuse existing constraint languages
  – decompositions at constraint level

• Solving:
  – CP as depth-first search framework with constraints
  – vectorize constraints to increase efficiency: global constraints with 'indexing' on the data, data structures
  – great for constrained enumeration
    → also for set selection optimisation, though understudied!
Questions?

Thanks to collaborators:
- L. De Raedt
- S. Nijssen
- A. Dries
- B. Negrevergne
- J. Aoga
- P. Schaus
- T. Le Van
- S. Paramonov
- B. Babaki
- A. Zimmermann
- G. Tack
- K. Marchal
- H. Sun
- A. Jiminez

For more pointers, see:
AIJ Special Issue March 2017: Combining Constraint Solving with Mining and Learning
IJCAI 2017 tutorial: Data Mining and Machine Learning using Constraint Programming Languages