Declarative constraint-based pattern mining: from modeling to solving

Tias Guns

BUTO, VUB Brussel
Declarative Languages & AI lab, KU Leuven
http://homepages.vub.ac.be/~tiasguns/

DeLBP workshop @ IJCAI 2017
A.I.

- Data Mining
- Constraint Solving
Itemset mining

Transactions:

1) {Doritos, Tostitos, Fritos} (58%)
2) {Doritos} (33%)
3) {Fritos} (42%)
4) {Doritos, Tostitos} (42%)
5) {Doritos, Fritos} (42%)
6) {Doritos} (33%)
7) {Pampers, Heineken, Pringles} (83%)
8) {Pampers, Heineken, Dove} (33%)
9) {Pampers, Heineken} (42%)
10) {Heineken} (33%)
11) {Dove} (42%)
12) {Pampers} (50%)
Biological sequence mining
(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. Cremilleux, L. De Raedt, T. Guns, T. Guyet, S. Jabbour, M. Jarvisalo, A. Kemmar, S. Loudni, S. Nijssen, B. O'Sullivan, ...

- **Clustering**
  B. Babaki, I. Davidson, T.B.H. Dao, K.C. Duong, S. Gilpin, V. Grossi, 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
Constraint-based Itemset Mining

- Fundamental enumeration problem
- Well studied
- 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

\[
\text{cover}(\text{Item 1, Item 2}) = \{\text{Item 1, Item 2}\}
\]

\[
\text{frequency}(\text{Item 1, Item 2}) = |\{\text{Item 1, Item 2}\}| = 2
\]
CP for Itemset Mining

One solution = one frequent itemset: enumerate all

coverage: \( \forall T_t: T_t = 1 \iff \text{set}(I_1, \ldots, I_n) \subseteq \text{set}(\text{row}_t) \)

frequency: \( \sum_t T_t \geq \text{Freq} \)
CP for Itemset Mining

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

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

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

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

Run time (s)

Minimum support

DMCP 30% 25% 20% 15% 10% 5% 1%

MaxAvgCost

FIM_CP_1%

FIM_CP_5%

FIM_CP_10%

PATTER_1%

PATTER_5%

PATTER_10%

LCP_10%

CP

Many constraints

coverage+frequency

Specialized systems

CP
Take away message 1.

Constraint Programming for Itemset Mining:
- 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. Cremilleux, ECAI 2014]
  [W. Ugarte, P. Boizumault, B. Crémilleux, A. Lepailleur, S. Loudni, M. Plantevit, C. Raïssi, A. Soulet, AIJ 2017]
- **SAT, BDD, ASP solvers**
  [JP. Metivier, P. Boizumault, B. Cremilleux, M. Khiari, S. Loudni, IDA 2012]
  [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 *Inductive Databases*’ idea of Heikki Mannila → a database where *data* and *patterns* are both easily queried

Projects integrating mining in SQL:

- MINE RULE [Meo et al, 1996]
- MSQL [Imielinski & 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

library with itemset mining specific functions and predicates

```mzn
include "lib_itemsetmining.mzn"

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

var set of 1..NrI: Items;

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

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

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

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

solve satisfy;
```
Solver independence

\begin{verbatim}
var set of 1..Nrl: Items; array[int] of set of int: TDB;
constraint card(cover(Items, TDB)) >= 20;
constraint card(cover(Items, TDB)) <= 40;
solve satisfy;
\end{verbatim}

From text-based model to `execution plans``:

1) specialised solvers \textit{(if supported constraint combination)}

2) automatic post-processing:
   - use specialised solver on subset of constraints
   - post-process remaining constraints with generic solver

3) generic (CP) solvers
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: hybridization

Open questions

- Multiple execution strategies: algorithm selection? parallelism?
- Problems not fitting standard CP
  - Skyline patterns / multi-objective
  - Dominance / preference over solutions
- Text-based language vs. embedded language

[W. Ugarte, P. Boizumault, S. Loudni, B. Cremilleux, ECAI 2014]
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 := 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
```
Differences IM search / CP search

In pure CP model:
for each transaction a separate constraint
  ➔ data is split into many individual constraints
  ➔ 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
  ➔ can cache computations (e.g. frequency of each item)

→ keep data together in CP?
CP scalability

Wrote minimalistic CP solver that:

- implements standard generic CP search
- implements BoolVector variable type (bitwise computations)
- supports the generic global constraint $X \Box (D A \approx B)$
  (other constraints could be added as in any CP solver)

Within global constraint:

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

Minimum support

Runtime (s)

Splice (Closed)

CP (original)

CP (new solver)

[S. Nijssen, T. Guns, ECMLPKDD 2010]
Frequent Itemset Mining, scaling

Minimum support

Runtime (s)

CP (original)

CP (new solver)

T10I4D100K (Frequent)

[S. Nijssen, T. Guns, ECMLPKDD 2010]
Within standard CP solver

- Standard CP solver
- One global constraint for:
  - computing the cover and enforcing minimum frequency
  - no need to expose transaction variables 'T'

=> can use (and hide) same datastructures as specialised methods

[N. Lazaar, Y. Lebbah, S. Loudni, M. Maamar, V. Lemière, C. Bessiere, P. Boizumault, CP 2016]
[P. Chaus, 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
- Bonus: advanced data structures and indexing
- Efficiency versus generality trade-off!
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
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

Cover
= subsequence relation
= ordered matching

Pattern: \(<H, G, ?, ?, ?>\)

T1: \(<S,B,H,R,G,H,M>\)

T2: \(<S,G,H,W,L,W,M>\)

T3: \(<R,H,W,H,D,G,H>\)

multiple embeddings possible:
T3: \(<R,\mathbf{H},W,H,D,G,H>\)
Why CP?

Many constraints, we identify four categories:

- Constraints on syntax:
  - size, regular expr., ...
- Constraints on data:
  - min_freq, max_freq, discriminative, ...
- Preferences over the solution set
  - closed, maximal, relevant, multi-objective, ...
- new Constraints on inclusion relation:
  - max_gap, min_gap, max_span

Hard-coded in specialised algorithms...
Standard sequences

\[ X = \langle \text{Home, Work, Restaurant, Gym, Work, Home} \rangle \]

Example sequence \( P = \langle \text{Home, Home} \rangle \)

- can have arbitrary symbols before/between/after
- ex: \( \langle \text{Home, Work, Restaurant, Gym, Work, Home} \rangle \)

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

- In CP: One variable for each \( e_j \), multiple constraints?
What specialized algorithms do

P: <H, H, ?, ?, ?>
T1: <R,H,W,H,G,D,H>
T2: <S,B,H,R,G,H,M>

PrefixSpan:
- Linear scan of each transaction, keep only pointer to first match of last symbol
- When symbol added to P, continue from pointer (incremental)
- O(1) space, O(n) algorithm

In CP, O(n) e_j variables and multiple constraints over them?
in CP: add global constraint

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

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

Global constraint with filtering algorithm:

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

- incremental: keep one pointer to last assigned match \( e_j \)

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

One global constraint for **all** sequences:

- algorithmic improvements: last position map, last position list → precomputed and cached, speedups

- use **backtracking-aware datastructure** → stores cover and prefix point in reversible vector

---

[J. Aoga, T. Guns, P. Chaus, ECMLPKDD 2016]
Efficiency: outperforms specialised!
Improving generality

Constraints:

- Constraints on sequence: compatible
- Constraints on cover set: compatible
- Preferences over the solution set: compatible
- Constraints on inclusion relation: not compatible

$=>$ best known: min/max gap and span

gap: 1

<Home, Work, Restaurant, Gym, Work, Home, Bar>

span: 4

gap: 2

can modify the global cover constraint to also enforce gap/span

$\rightarrow$ improves state-of-the-art (with backtrack-aware datastructure)

[J. Aoga, T. Guns, P. Chaus, CPAIOR 2017]
Constraints
Take away message 4.

**Sequence mining**: more complex *coverage* relation

Global constraint:

- hides complexity of *coverage* relation
- fast (incremental, PrefixSpan-like)
- good way to hybridize with data mining techniques

=> necessary to be **efficient**

Even more efficient with backtracking-aware datastructures
Finding the right level of abstraction:

• Modeling:
  – not query but set of constraints
  – solver/algorithm independence
  – automatic rewrite rules

• Solving:
  – each constraint can be made independent
  – vectorize constraints to increase efficiency (internal data structures)
  – CP as framework for pattern mining
Questions?

Thanks to collaborators:
- L. De Raedt
- S. Nijssen
- A. Dries
- B. Negrevergne
- J. Aoga
- P. Chaus
- 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