Constraint Solving meets Data Mining and Machine Learning

Tias Guns
DTAI, KU Leuven, Belgium

Guest Lecture @ AI lab, VUB, Belgium
22 Feb. 2013
Constraint Solving

One of the success stories of A.I.

model: declarative specification of constraints

+ search: generic handling of variables and constraints & efficient propagation of individual constraints

Used in scheduling, planning, bio-informatics, game playing, verification, logical reasoning, ...
Data mining & Machine Learning

“Using historical data to improve decisions”

- Data Mining: discovering knowledge in data
  for example purchasing behaviour, biological data, ...

- Software we can't program by hand
  for example self-driving cars, speech recognition, …

- Self-customizing programs
  for example spam filters, recommender systems, …

Also hyped as data analytics and big data

[T. Mitchell, Machine Learning, 1997]
This lecture

How can the two fields benefit from each-other?

- Part A: Introduction
- Part B: Solving in ML & DM
- Part C: Learning in constraint solvers
Part A: introduction
Declarative vs Imperative

Two approaches to problem solving.
I want a carpet that is:

- 5m x 2m
- blue
- with fish patterns
- and little ropes on the side
Imperative

Now,

- put a blue wire,
- tighten it,
- add two layers of white,
- make a nod,
- ...
Example: graph coloring
Graph coloring, imperative

Naive:

- depth-first search over countries, do not assign neighbors the same color
- $O(k^n)$  \(k=\text{colors}, n=\text{nodes/countries}\)

Smart:

- $O(n^2^n)$

Development time vs execution time?
Graph coloring, declarative

Neighboring countries should have different colors
Graph coloring, declarative

Constraint programming

- Variables
  - ex. countries

- Domains
  - ex. colors

- Constraints
  - ex. neighbor colors
Imperative vs Declarative

Imperative:
- Low-level control
- Typically very fast and efficient
- Very specific for one problem

Declarative:
- high-level modeling
- less fast and scalable than hand-made algorithms
- general and reusable
From imperative to declarative

Example, evolution in databases:

- Data in files, access with IO calls (syscall level)
- Data in *records*, access by following pointers (data definition language + data manipulation language)
- Data in *tables/relations* (schema + query language)

Abstraction, reuse, query optimisation, indexing, continued progress and improvements

**Warning:** generality/efficiency trade-off remains

- NoSQL (key/value) for specialised data (images, graphs) or specific settings (high-speed low-consistency)
AI Research in 2012

Recent conferences and journals have:

- Search and Planning
- SAT and Constraints
- Probabilistic Planning
- Probabilistic Reasoning
- Inference in First-Order Logic
- Machine Learning & Data Mining
- Natural Language
- Vision and Robotics
- Multi-agent systems

Trend from imperative to declarative:

- SAT solvers
- CP solvers
- BDDs
- MIP & QP
- Bayesian Networks
- Markov Logic
- POMDPs
- ...

[H. Geffner, AI: From Programs to Solvers, Turing Session, ECAI-2012]
An incomplete categorisation

Constraint Solving

Symbolic
- SAT
  - pseudo-boolean
  - SMT
  - ASP
  - FO(.)
- Know. Comp.
  - BDD
  - ADD
- CP
  - weighted CP
  - Constr. based Local Search
  - LP
  - Convex opt.
  - MIP

Numeric

Other
Declarative Constraint Solving

Mantra:
Constraint Solving = Model + Search
by the user
by a solver
SAT solving

- Propositional Satisfiability
- Example: the 'frietkot' problem
SAT solving

- Propositional Satisfiability
- First proven NP-complete problem (Cook, 1971)
- Input: clauses
  - $X \lor Y \lor -Z$
  - $-X \lor Z$
- Output: UNSATISFIABLE, or, assignment to variables that satisfies all clauses
SAT solving

Advantages/disadvantages:

- Extremely optimised solvers
  - standard input format (making comparison easy)
  - yearly competitions
  - sustained scientific progress
SAT solver research

Number of problems solved

Time (max 1200 sec)
SAT solving

Advantages/disadvantages:

✔ Extremely optimised solvers
  - yearly competitions
  - standard input format (making comparison easy)
  - sustained yearly progress

✗ No support for modelling high-level problems
Constraint Programming, example

- **variables**
  
  \[E_{11} \ldots E_{99}\]

- **domains**
  
  \[E_{xy} = \{1 \ldots 9\}\]

- **constraints**
  
  `all_different([E_{11}])`, ...
  
  `all_different([E_{x1}])`, ...
  
  `all_different([E_{11} \ldots E_{33}])`, ...
Constraint Programming

- CSP or COP (satisfaction / optimisation)
- Solving combinatorial problems (typically in NP)
  *scheduling, routing, planning, ...*

- Input: Variables, Domains, Constraints

- High level modeling languages (Zinc, Essence, OPL)
- 'global constraints'
CP Search

Two key principles:

- **Propagation** of constraints
  
eg. \text{alldiff}(X, Y, Z) \ X=\{1\}, Y=\{1,2\}, Z=\{1,2,3,4\} → Y=\{2\}, Z=\{3,4\}

  Every constraint is implemented by a propagator.

- **Branch** over values of variables
  
eg. Propagation at fixpoint → branch over Z=\{3\}

Search is recursive and complete
Declarative constraint solving

Advantages

- general approach
- reuse of solvers, modeling primitives

Disadvantage

- need expertise (good model/bad model)
- search heuristics huge impact on performance

Use historical data to improve decisions?
→ Machine Learning
The use of historical data:

- Learning/improving models (constraints)
- Learning/improving search strategies, solver selection, heuristics, etc
What CS offers to DM & ML

- Declarative: model + search
- Decomposability and reuse
- General: many tasks, variations
- Rapid prototyping, iterative process
This lecture

Constraint Solving

Data Mining & Machine Learning

How can the two fields benefit from each-other?

- Part A: Introduction
- **Part B: Solving in ML & DM**
- Part C: Learning in constraint solvers
Part B: Solving in ML & DM
# Data Mining and Machine Learning

## Symbolic
- Rule learning
- Decision trees
- Clustering
- Pattern Mining
- ...

Mostly using hand-craft algorithms: Ripper, C4.5, k-means, Apriori, ...

## Numeric
- Regression
- SVMs
- Matrix factorisation
- ...

Mostly using numeric optimisation: least squares, gradient decent, convex optimisation, ...
Linear Regression

Notations:

- Datapoints $X = \{x_1, x_2, \ldots, x_n\}$ $x_i \in \mathbb{R}^d$
- Labels $y = \{y_1, y_2, \ldots, y_n\}$ $y \in \mathbb{R}$
- Linear decision function $f(\cdot): \mathbb{R}^d \rightarrow \mathbb{R}$

$$f(x) = w^T x$$

- Parameter vector $w$
Linear Regression

- Goal: find a linear function $\mathbf{Xw}$ that approximates the labels $\mathbf{y}$.
- For a new test point $\mathbf{x}$ the label $\mathbf{y}$ can be estimated as $\mathbf{w}^T\mathbf{x}$.

Sum Squared Error

$$E = \sum_{i=1}^{n} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 = \|\mathbf{y} - \mathbf{Xw}\|^2$$
Least Squares Regression

• Solve:

$$\min_w \|y - Xw\|^2 \quad \Rightarrow$$

$$\min_w w^T X^T X w + y^T y - 2w^T X^T y$$

(gradient equal to 0):

$$X^T X w - 2X^T y = 0 \quad \Rightarrow \quad w = 2 (X^T X)^{-1} X^T y$$

• This is Least Squares Regression.

• In alternative we can solve the linear system:

$$A w = b \quad \Rightarrow \quad A = X^T X, \quad b = 2X^T y$$
Ridge Regression

- LSR tend to overfit for noisy and high dimensional data.
- Solution: minimize the loss and at the same time restrict the capacity!

\[
\min_w \| y - Xw \|^2 \\
\text{s.t. } C(w) \leq C_{MAX}
\]

- Solve:

\[
\min_w \| y - Xw \|^2 \\
\text{s.t. } \| w \|^2 \leq 1
\]

- This is known as Regularized LSR or Ridge Regression.
Classifcation

Notations:

- Datapoints \( X = \{x_1, x_2, \ldots, x_n\} \quad x_i \in \mathbb{R}^d \)
- Labels \( y = \{y_1, y_2, \ldots, y_n\} \quad y_i \in \{-1, 1\} \)
- Linear decision function \( f(\cdot): \mathbb{R}^d \rightarrow \mathbb{R} \)
  \[
  f(x) = w^T x + b = \hat{w}^T \hat{x}
  \]
- Parameters \( \hat{W} = [w \ b] \) and data \( \hat{X} = [x \ 1] \).
Classification

• Binary classification:

\[ w^T x + b > 0 \]

\[ w^T x + b = 0 \]

\[ w^T x + b < 0 \]

• Which is the optimal separating hyperplane?

\[ f(x) = \text{sign}(w^T x + b) \]
Support Vector Machines

- Given a training set:
  \[ T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\} \]

- Goal:
  - Correctly classify all training data
    \[ y_i (w^T x_i + b) \geq 1 \quad \forall i \]
  - Maximize the Margin
    \[ \rho = \frac{2}{\|w\|} \]

- This is equivalent to the following

  \[
  \min_{w, b} \quad \frac{1}{2} \|w\|^2 \\
  \text{s.t.} \quad y_i (w^T x_i + b) \geq 1 \quad \forall i
  \]

- This is a Quadratic Programming (QP) problem.
Support Vector Machines

- What if the training set is not linearly separable?

- *Slack variables* $\xi_i$ can be added to allow misclassification of difficult or noisy examples.

- This is usually referred as *soft-margin SVM.*
Support Vector Machines

- Solve:

\[
\min_{w, b, \xi_i} \quad \frac{1}{2}\|w\|^2 + C \sum_i \xi_i \\
\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i
\]

- The problem is still convex and efficiently solvable.

- \(C\) is a \textit{regularization} parameter. It controls overfitting.
- It allows a trade-off between maximizing the margin and fitting the training data.
- A large \(C\) corresponds to assign a higher penalty to the errors.
Declarative methods in ML

Often problem expressible as linear/convex opt. problem → can solve with standard ILP/QP solvers

In practice, specialised solvers are often used:

- faster, *lighter* implementations
- special-purpose decompositions
- dealing with large and sparse data
Data Mining and Machine Learning

Symbolic
- Rule learning
- Decision trees
- Clustering
- Pattern Mining
- ...

Numeric
- Regression
- SVMs
- Matrix factorisation
- ...

Mostly using hand-craft algorithms:
Ripper, C4.5, k-means, Apriori, ...

Mostly using numeric optimisation:
least squares, gradient decent, convex optimisation, ...

Can declarative methods be used here too?
Basic mining task

Analysing a dataset to find patterns of interest

For example:

Analysing purchases (e.g. books)

Here, patterns are sets of 'items'

(e.g. + )
Patterns of interest:

- which patterns are frequent?
- which patterns have a high average price?
- which patterns are frequent on one dataset and infrequent on the other?
- which patterns are significant w.r.t a background model?
- ...

→ specified by constraints
Constraint-based Pattern Mining

- Numerous constraints proposed
- Numerous algorithms developed

Yet,

- new constraints mostly require new implementations
- very hard to combine different constraints

Surprisingly, CP had not been applied to Pattern Mining
Pattern Mining
Depends on type of data

Text Mining
Well, there’s egg and spam; egg sausage and spam; spam and bacon; egg bacon and spam; egg bacon sausage and spam; spam bacon sausage and spam; spam egg spam spam bacon and spam; spam sausage spam bacon spam tomato and spam; spam spam spam egg and spam; spam spam spam baked beans spam spam spam spam; or Lobster Thermidor, a Crevette with a mornay sauce served in a Provencale manner with shallots and aubergines garnished with truffle pate, brandy and spam.
Pattern Mining
Depends on type of data

Well, there's egg and spam; egg sausage and spam; egg bacon; egg bacon sausage and spam; spam bacon sausage and spam; spam bacon sausage bacon and spam; spam sausage spam bacon spam tomato and spam; spam spam spam tomato and spam; or Lobster Thermidor, a Crevette with a mornay sauce served in a Provencale manner with shallots and aubergines garnished with truffle pate, brandy and spam.
sets of items
Itemset Mining

**Find**: set of *items* appearing frequently

**Example:**

\{ , \} : frequency = 2
Declarative approach

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

coverage(, ) = \{, \}
frequency(, ) = 2
### CP for Itemset Mining

**Coverage:**

\[ \forall T_t : T_t = 1 \iff (\{ I_1, \ldots, I_n \} \subseteq row_t) \]

**Frequency:**

\[ \sum_t T_t \geq \text{Freq} \]
CP for Itemset Mining

coverage:

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

frequency:

\[ \forall I_i: \quad I_i = 1 \implies \sum_{t, D_{ti} = 1} T_t \geq \text{Freq} \]
Algorithm 1 Fim\_cp’s frequent itemset mining model, in Essence

1: \textbf{given} NrT, NrI : int
2: \textbf{given} TDB : matrix indexed by [int(1..NrT),int(1..NrI)] of int
3: \textbf{given} Freq : int
4: \textbf{find} Items : matrix indexed by [int(1..NrI)] of bool
5: \textbf{find} Trans : matrix indexed by [int(1..NrT)] of bool
6: \textbf{such that}
7: \$\text{encode TDB: every Trans its complement has no supported Items}\$
8: \textbf{forall} t: \text{int}(1..NrT).
9: \hspace{1em} Trans[t] \iff (\sum i: \text{int}(1..NrI). \text{Items}[i]*(1-TDB[t,i])) = 0),
10: \$\text{frequency: every Item is supported by sufficiently many Trans}\$
11: \textbf{forall} i: \text{int}(1..NrI).
12: \hspace{1em} Items[i] \Rightarrow (\sum t: \text{int}(1..NrT). \text{Trans}[t]*\text{TDB}[t,i]) \geq \text{Freq}$
Traditional search: proj. database
CP for Itemset Mining

coverage: $\forall T_t: \quad T_t = 1 \iff \bigwedge_{i, D_{ti}=0} \neg I_i$

freq $\geq 2$: $\forall I_i: \quad I_i = 1 \Rightarrow \sum_{t, D_{ti}=1} T_t \geq Freq$

\[
\begin{array}{cccccc}
& i1 & i2 & i3 & i4 \\
0/1 & 1 & 0 & 1 & 1 \\
t1 & 1 & 1 & 0 & 1 \\
t2 & 0/1 & 1 & 0 & 1 \\
t3 & 0/1 & 0 & 0 & 1 & 1
\end{array}
\]
CP for Itemset Mining

coverage: \[ \forall T_t: T_t = 1 \iff \land_{i, D_{ti} = 0} \neg I_i \]

freq >= 2:

\[ \forall I_i: I_i = 1 \Rightarrow \sum_{t, D_{ti} = 1} T_t \geq \text{Freq} \]

- propagate i2

Intuition: infrequent

i2 can never be part of freq. superset
CP for Itemset Mining

coverage: \( \forall T_t: \; T_t = 1 \iff \land_{i, D_t = 0} \neg I_i \)

freq \( \geq 2 \): \( \forall I_i: \; I_i = 1 \Rightarrow \sum_{t, D_t = 1} T_t \geq \text{Freq} \)

- propagate i2
- propagate t1

Intuition: unavoidable
\( t1 \) will always be covered
CP for Itemset Mining

coverage: \[ \forall T_t: \ T_t=1 \Leftrightarrow \bigwedge_{i, D_{ti}=0} \neg I_i \]

freq >= 2: \[ \forall I_i: \ I_i=1 \Rightarrow \sum_{t, D_{ti}=1} T_t \geq \text{Freq} \]

- propagate i2
- propagate t1
- branch i1=1
CP for Itemset Mining

coverage: \( \forall T_t: T_t = 1 \iff \bigwedge_{i, D_{ti} = 0} \neg I_i \)

freq \( \geq 2 \): \( \forall I_i: I_i = 1 \Rightarrow \sum_{t, D_{ti} = 1} T_t \geq Freq \)

- propagate i2
- propagate t1
- branch i1=1
- ...

\[
\begin{array}{cccccc}
& i1 & i2 & i3 & i4 \\
\hline
t1 & 1 & 0 & 1 & 1 & 1 \\
t2 & 0/1 & 1 & 1 & 0 & 1 \\
t3 & 0 & 0 & 1 & 1 \\
\end{array}
\]
More constraints

- Coverage (required)
  \[ T_t = 1 \iff \sum_i I_i(1 - D_{ti}) = 0 \]
- Frequent
  \[ I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq \text{Freq} \]
- Maximal
  \[ I_i = 1 \iff \sum_t T_t D_{ti} \geq \text{Freq} \]
- Closed
  \[ I_i = 1 \iff \sum_t T_t (1 - D_{ti}) = 0 \]
- Delta-closed
  \[ I_i = 1 \iff \sum_t T_t (1 - \delta - D_{ti}) = 0 \]

+ combinations
## Generality

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Constraints on data</strong></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>X</td>
</tr>
<tr>
<td>Maximum frequency</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Emerging patterns</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Condensed Representations</strong></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>X</td>
</tr>
<tr>
<td>δ−Closed</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Constraints on syntax</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max/Min total cost</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Minimum average cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>
<td>X</td>
</tr>
<tr>
<td><strong>Constraints on labelled data</strong></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></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>
Simple Itemset Mining

coverage+frequency

Minimum support

Run time (s)

Specialised systems

Simple Itemset Mining

CP (Gecode)

Specialised systems
Constraint-based mining

Specialised systems

CP (Gecode)
Correlated itemset mining

Also known as: discriminative itemset mining, contrast set mining, emerging itemsets, subgroup discovery, ...

- Given: labelled transactions

- Find: the itemset that best correlates with the class label

  : \{\textcolor{red}{+}, \textcolor{red}{-}\}
  \textcolor{red}{-}, \textcolor{red}{-}, \textcolor{red}{-}, \textcolor{red}{-} : \{\textcolor{green}{+}, \textcolor{red}{+}\}
Correlation constraint

\[ f(\sum_{t \in P} T_t, \sum_{t \in N} T_t) \geq \text{Bound} \]

- Existing pruning technique:
  
  only uses upper-bound of \( \sum T \)

- Our CP-based propagator:
  
  uses upper- and lower-bound of \( \sum T \)
  
  and look-ahead formulation \( I_i = 1 \Rightarrow \ldots \)

much stronger propagation!
## Correlated itemset mining

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CP</th>
<th>(Cheng et al. 2008)</th>
<th>(Morishita and Sese 2000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>anneal</td>
<td>0.22</td>
<td>22.46</td>
<td>24.09</td>
</tr>
<tr>
<td>australian-credit</td>
<td>0.30</td>
<td>3.40</td>
<td>0.30</td>
</tr>
<tr>
<td>breast-wisconsin</td>
<td>0.28</td>
<td>96.75</td>
<td>0.28</td>
</tr>
<tr>
<td>diabetes</td>
<td>2.45</td>
<td>-</td>
<td>128.04</td>
</tr>
<tr>
<td>heart-cleveland</td>
<td>0.19</td>
<td>9.49</td>
<td>2.15</td>
</tr>
<tr>
<td>hypothyroid</td>
<td>0.71</td>
<td>-</td>
<td>10.91</td>
</tr>
<tr>
<td>ionosphere</td>
<td>1.44</td>
<td>-</td>
<td>&gt;</td>
</tr>
<tr>
<td>kr-vs-kp</td>
<td>0.92</td>
<td>125.60</td>
<td>46.20</td>
</tr>
<tr>
<td>letter</td>
<td>52.66</td>
<td>-</td>
<td>&gt;</td>
</tr>
<tr>
<td>mushroom</td>
<td>14.11</td>
<td>0.09</td>
<td>13.48</td>
</tr>
<tr>
<td>primary-tumor</td>
<td>0.03</td>
<td>0.26</td>
<td>0.13</td>
</tr>
<tr>
<td>segment</td>
<td>1.45</td>
<td>-</td>
<td>&gt;</td>
</tr>
<tr>
<td>soybean</td>
<td>0.05</td>
<td>0.05</td>
<td>0.07</td>
</tr>
<tr>
<td>splice-1</td>
<td>30.41</td>
<td>1.86</td>
<td>31.11</td>
</tr>
<tr>
<td>vehicle</td>
<td>0.85</td>
<td>-</td>
<td>&gt;</td>
</tr>
<tr>
<td>yeast</td>
<td>5.67</td>
<td>-</td>
<td>781.63</td>
</tr>
</tbody>
</table>

Runtime in seconds
Decreasing the gap

An integrated CP solver would:

- use principles of both IM and CP
- focus on constraints for itemset mining

Hypothesis: unnecessary overhead in CP solver
Itemset Mining principles

- Search strategy: *Level-wise, BFS, DFS*

- Representation of data:

  \[
  \begin{array}{cccc}
  1 & 0 & 1 & 1 \\
  1 & 1 & 0 & 1 \\
  0 & 0 & 1 & 1 \\
  \end{array}
  \]

- Representation of sets:

  \[
  \begin{array}{cccc}
  1 & 0 & 1 & 1 \\
  0 & 1 & 0 & 0 \\
  \end{array}
  \]

\{1, 3, 4\} \quad \{2\}
### Integration 1/3

<table>
<thead>
<tr>
<th></th>
<th>Eclat Miner</th>
<th>Gecode CP Solver</th>
<th>DMCP CP Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Search strategy</strong></td>
<td>DFS</td>
<td>DFS (binary)</td>
<td>DFS (binary)</td>
</tr>
<tr>
<td><strong>Repres. of data</strong></td>
<td>Shared, vertical</td>
<td>In constraints (up to 4 copies)</td>
<td>Shared matrix (default: vertical)</td>
</tr>
</tbody>
</table>

- Data shared (read-only) by constraints
- Horizontal, positive and negative *views* available
### Integration 2/3

<table>
<thead>
<tr>
<th></th>
<th>Eclat Miner</th>
<th>Gecode CP Solver</th>
<th>Our <strong>DMCP</strong> CP Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repres. of sets</td>
<td>Sparse or Dense</td>
<td>Sparse</td>
<td>Sparse or Dense</td>
</tr>
<tr>
<td>Types of vars.</td>
<td>Boolean vector (set)</td>
<td>Bool, Int, Set, ...</td>
<td>Boolean vector (set)</td>
</tr>
</tbody>
</table>

- Represented by lower and upper bound:
  - Min: \{0, 0, 0, 1\}
  - Max: \{0, 1, 1, 1\}
<table>
<thead>
<tr>
<th>Constraints</th>
<th>Eclat Miner</th>
<th>Gecode CP Solver</th>
<th>Our <strong>DMCP CP Solver</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraints</td>
<td>Few, hard to combine</td>
<td>Many, easy to add/combine</td>
<td>Some, easy add/combine</td>
</tr>
<tr>
<td>Constraint activ.</td>
<td>Strict order (in algorithm)</td>
<td>On domain change</td>
<td>Change of lower/upper bound</td>
</tr>
</tbody>
</table>

- General matrix constraint:

\[ X \geq_1 1 \geq_2 \theta (\mathcal{A} \cdot Y) ; \]

Data representation (matrix)

Boolean vectors
Frequent Itemset Mining

Mushroom (Frequent)

Minimum support

Runtime (s)

Old, Gecode

New, DMCP

New, DMCP
Frequent Itemset Mining, scaling

T1014D100K (Frequent)

Old, Gecode

New, DMCP
Closed Itemset Mining

Old, Gecode

Splice (Closed)

New, DMCP

Minimum support

Runtime (s)

FIMCP
PATTERNIST
LCM5.3
LCM2.5
MAFIA
B_ECLAT
B_FPGROWTH
B_APRIORI
DMCP
CP for Itemset Mining

Advantages of CP modelling:

- Easily add new constraints
- Freely combine constraints

Advantage of IM/CP solver integration:

- Theoretical: polynomial delay analysis
- Practical: remove efficiency/scalability gap
Data Mining and Machine Learning

Symbolic

- Rule learning
- Decision trees
- Clustering
- Pattern Mining
- ...

Correlated Itemset Mining [Nijssen et al. KDD 09]
using CP [Bessiere et al. CP 09]
SAT & clustering [Davidson et al. SDM 10]
Itemset Mining: [Guns et al. KDD08, AIJ 12] [many more]
Sequence mining: [Coquery et al. ECAI 12], ...

Mostly using hand-craft algorithms: Ripper, C4.5, k-means, Apriori, ...

Can declarative methods be used here too?
Open Questions

- More tasks, different types of problems
- Structured data (sequences, trees, graphs)
- Efficiency/generality trade-off
- Scalability and specialised solvers
- High-level modeling language for DM?
This lecture

Constraint Solving

Data Mining & Machine Learning

How can the two fields benefit from each-other?

- Part A: Introduction
- Part B: Solving in ML & DM
- Part C: Learning in constraint solvers
Part C: Learning in constraint solvers
Declarative Constraint Solving

Mantra:

Constraint Solving = Model + Search

by the user

by a solver
Declarative Constraint Solving

Advantages

• general approach
• reuse of solvers, modeling primitives

Disadvantage

• need expertise (good model/bad model)
• search heuristics huge impact on performance

Use historical data to improve decisions? → Machine Learning
No Free Lunch theorem

No single algorithm is best on all problem instances → Can we characterize/learn which algorithm is best on which problem instance?

In many competitions (SAT, CP, rostering, …), big gap between 'single best solver' and 'oracle solver'

*Empirical hardness models*

Hardness of a problem vs design choices in an algo.
• given a number of solvers, which solver to choose?

→ algorithm selection
  (also known as: portfolio's, meta-learning, ...)

• given an algorithm with a number of parameters, which parameter values to set?

→ algorithm configuration
  (= alg. selection when small parameter space)
Algorithm selection

General approach:

1. Collect solvers
2. Collect problem instances/data
3. Calculate *features* on instance/data
4. Build predictive model

[Kotthof, Survey, 2012]
Algorithm selection

1. Collect solvers

- Availability of solvers?
- Too many solvers?
- Diversity of solvers?
Algorithm selection

2. Collect problem instances/data
   - Similar problems? (SAT vs CP)
   - Representable set of data?
   - Diversity of instances/data?
Algorithm selection

3. Calculate *features* on instance/data

- What features to use?
  → Domain specific!

- Number of features to use?

- Normalisation? Features selection? Stacking?

Features are arguably the MOST IMPORTANT choice
(in ML in general)
Algorithm selection

4. Build predictive model

- What model?
  - per solver, e.g. regression?
  - per pair-of-solver, classification?
  - per portfolio, classification or ranking?
- Sensitivity to features? (e.g. noise, redundancy)
- Clustering / hierarchical models?
SATZilla 2007: winning the SAT competition

1. Solvers:
   - From previous competitions (~20)
   - Subset selection to select ~10 diverse ones (as measured by repeatedly building portfolio's)
   - Two pre-solvers (limited amount of time)
   - One backup solver (if all else fails)

2. Problem instances/data
   - From previous competitions
SATZilla 2007: winning the SAT competition

3. Calculate features on instance/data
   - 64 SAT-specific features
   - feature selection
   - speed to compute vs gain in prediction performance

4. Build predictive model
   - logistic regression of $\log(\text{runtime})$
   - censored data/timeouts
   - hierarchical: predict sat/unsat
SATZilla 2007: winning the SAT competition
SATzilla 2011: more success

- More features (138)
- Learn best pre-solvers
- Feature computation prediction (2 levels of features)
- Backup solver: not best overall, but best on feature-timeout instances

Learning method:
- Replace regression by pairwise classification,
- Take cost of mis-classification (runtime) into account
General lessons

- Good classifier alone not enough to win a competition (pre-solvers, backup solver, time of feat. calculation)

- Need good features (and feature selection)

- Hierarchical models: clustering problem instances and having separate portfolio's offers gain

- The best classifiers take the actual runtime into account (new: and cost of misclassification?)
Why does it work?

Machine Learning perspective:

• Similar to *ensembles*: combining predictions = minimising the variance

• Many more interesting (underexplored?) connections to boosting, bagging, and ensembles.

In SAT community: even single solver has large variance on runtime for a single input file (heuristic choices)
Open Questions

- Bounds on improvement? Learning theory?
- Limited use of probabilistic techniques?

In wider context of improving constraint solving:
- Learning parameters [iRace] or entire search strategies [grammar approach IRIDIA]?
- Learning model reformulations? [ModRef]
- Learning constraints/models? [ConAcq, ModelSeeker]
This lecture

Constraint Solving

Data Mining & Machine Learning

How can the two fields benefit from each-other?

• Part A: Introduction
• Part B: Solving in ML & DM
• Part C: Learning in constraint solvers
What CS offers to DM & ML

- Declarative: model + search
- Decomposability and reuse
- General: many tasks, variations
- Rapid prototyping, iterative process
What DM & ML offers to CS

The use of historical data:

• Learning/improving models (constraints)
• Learning/improving search strategies, solver selection, heuristics, parameters, etc
Thank you for listening

Constraint Solving

Data Mining & Machine Learning

Questions?
Possible task:

SatZilla data:
http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/

Do solver selection with ML techniques:

- Features matter!
- Imperative or declarative learning methods?
- Ease of modification/improvement of learning technique?
- Additional improvements tuning with, e.g., iRace?