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

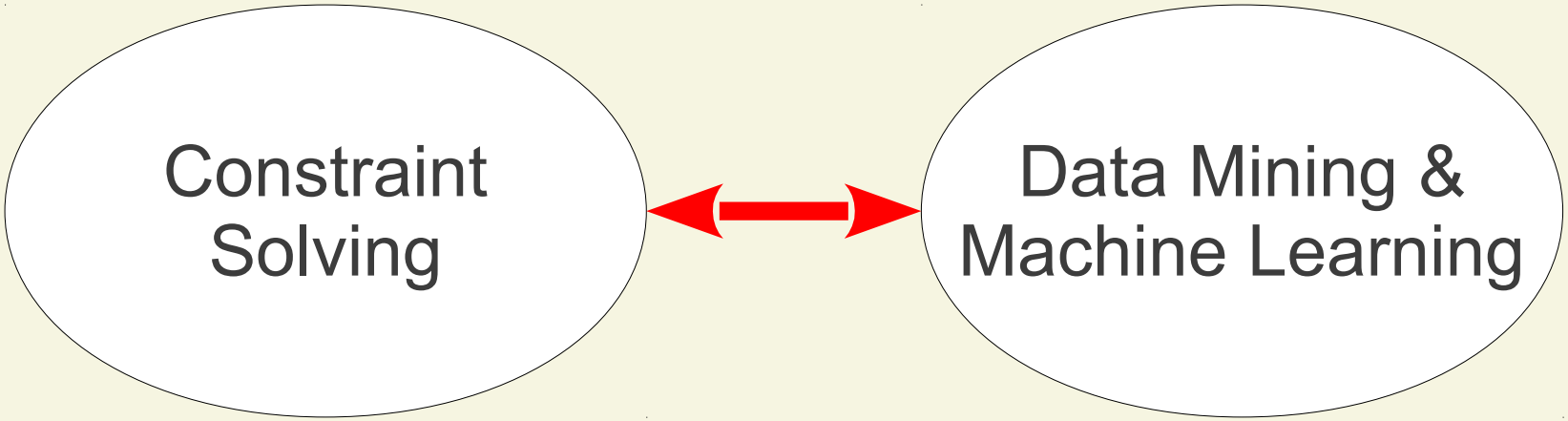
“Using historical data to improve decisions”

- 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.

Declarative

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

Naieve:

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

Smart:

- based on the principle of inclusion-exclusion and the zeta transform [Björklund, A.; Husfeldt, T.; Koivisto, M. (2009), "Set partitioning via inclusion–exclusion", SIAM Journal on Computing 39 (2): 546–563]
- $O(n*2^n)$

Development time vs execution time?

Graph coloring, declarative

Neighboring countries should have different colors

Graph coloring, declarative

<u>Constraint programming</u>

- 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:

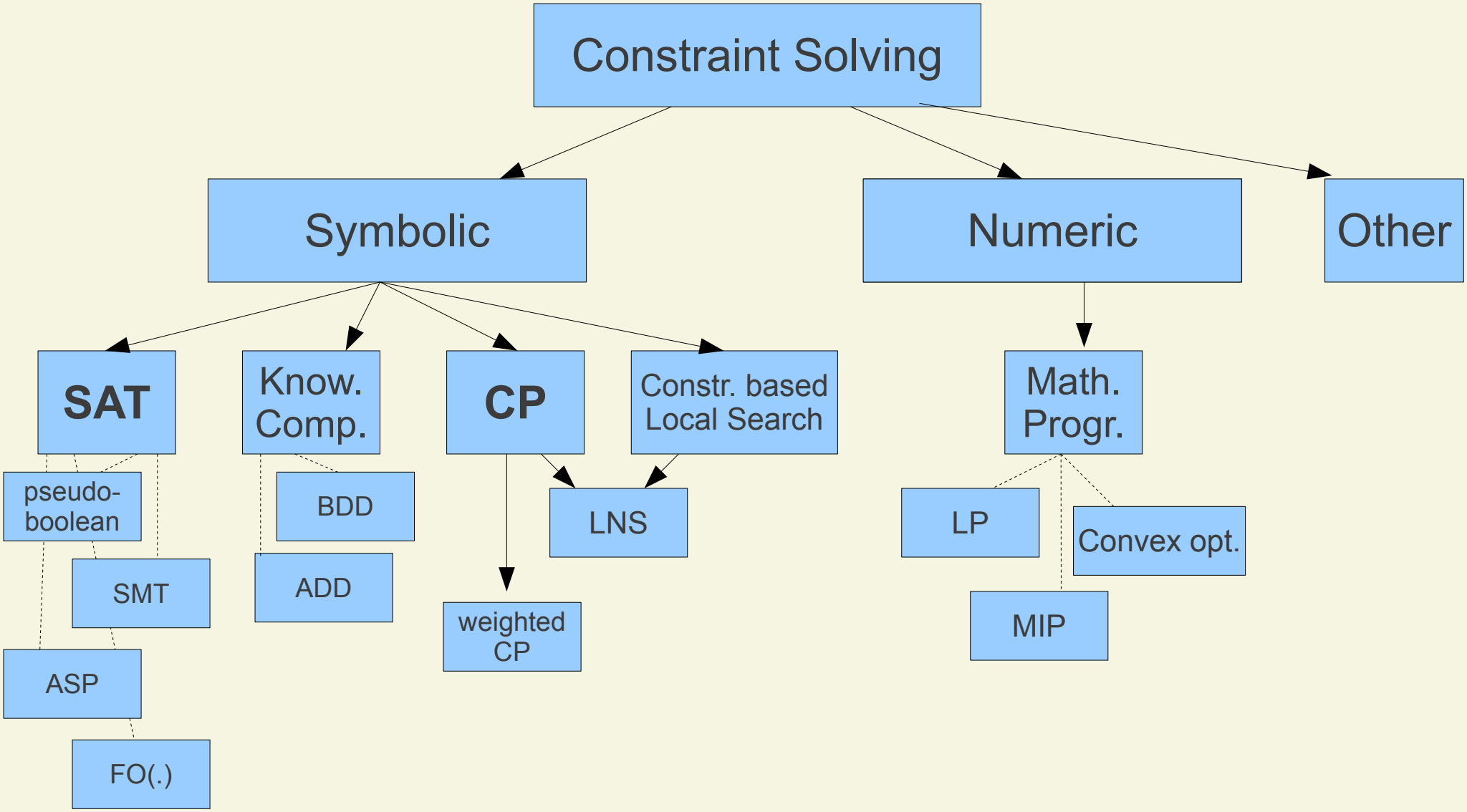
- Search and Planning
- SAT and Constraints
- Probabilistic Planning
- Probabilistic Reasoning
- Inference in First-Order Logic
- Machine Learning & Data Mining
- Natural Language
- Vision and Robotics
- Multi-agent systems

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



Declarative Constraint Solving

Mantra:

Constraint Solving = Model + Search

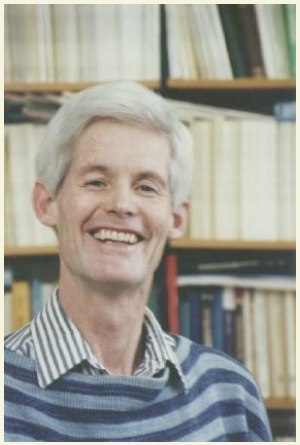
by the user

by a *solver*

SAT solving

- Propositional Satisfiability
- Example: the 'frietkot' problem

SAT solving



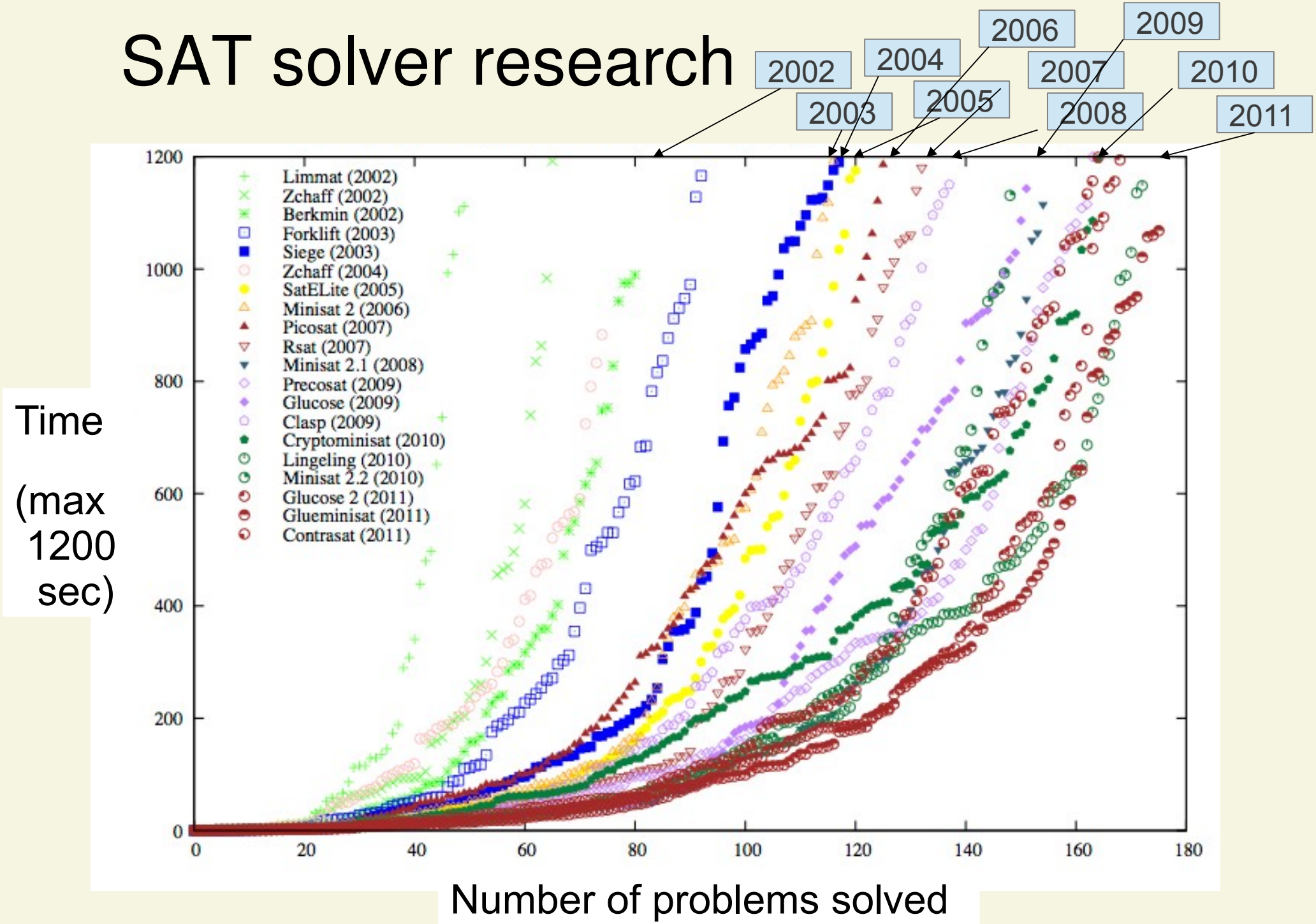
- Propositional Satisfiability
- First proven NP-complete problem (Cook, 1971)
- Input: clauses
  - X ∨ Y ∨ -Z
  - -X ∨ Z
- Output: UNSATISFIABLE, or,
  assignment to variables that satisfies all clauses

SAT solving

Advantages/disadvantages:

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

SAT solver research



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

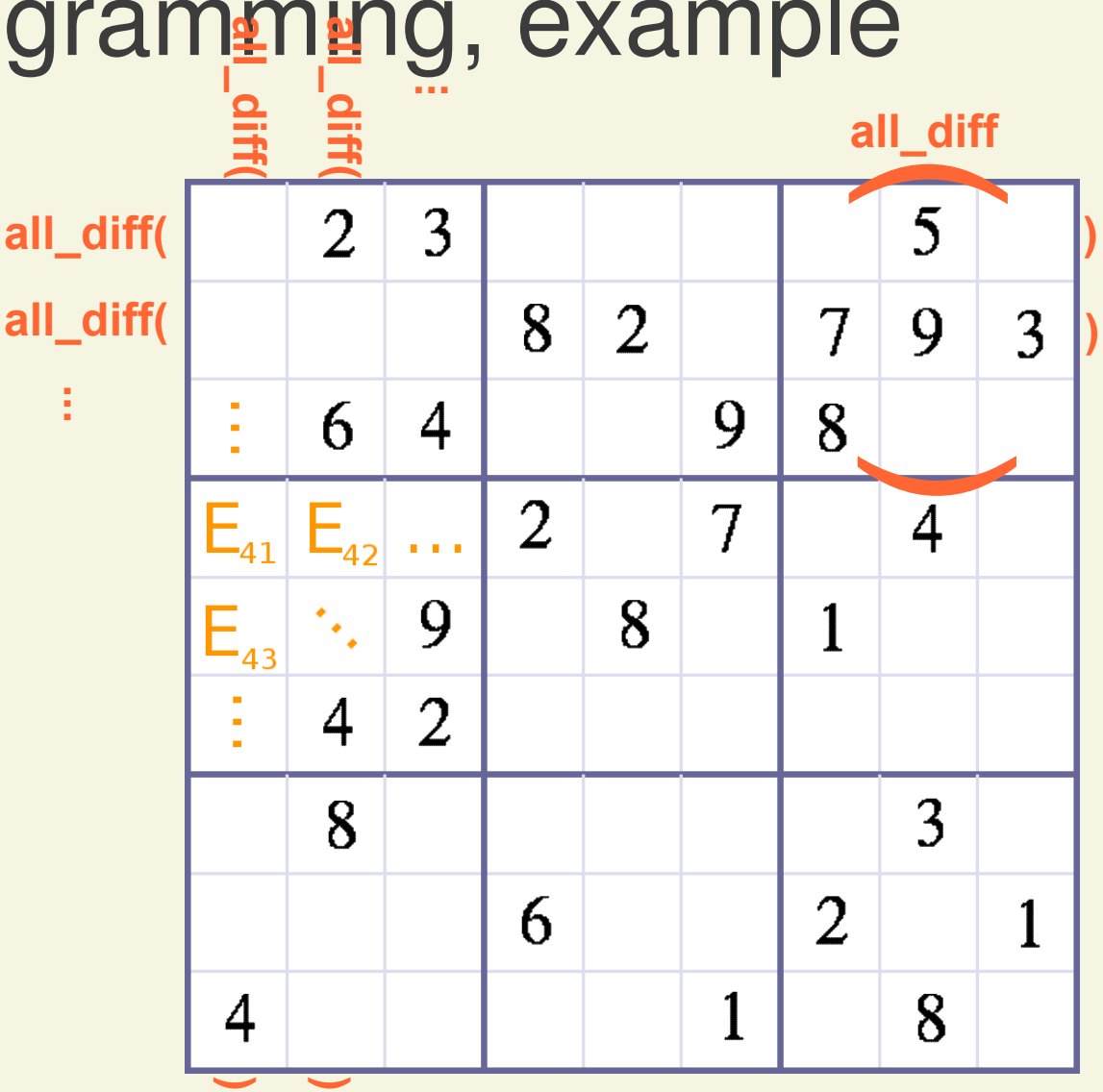
  $E_{xy} = \{1 \ldots 9\}$

- constraints

  all_different($[E_{1x}]$), ...

  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. alldiff(X,Y,Z) X={1},Y={1,2},Z={1,2,3,4} $\rightarrow$ Y={2},Z={3,4}

  Every constraint is implemented by a propagator.

- **Branch** over values of variables

  eg. Propagation at fixpoint $\rightarrow$ branch over Z={3}

  Search is recursive and complete

Declarative constraint solving

Advantages

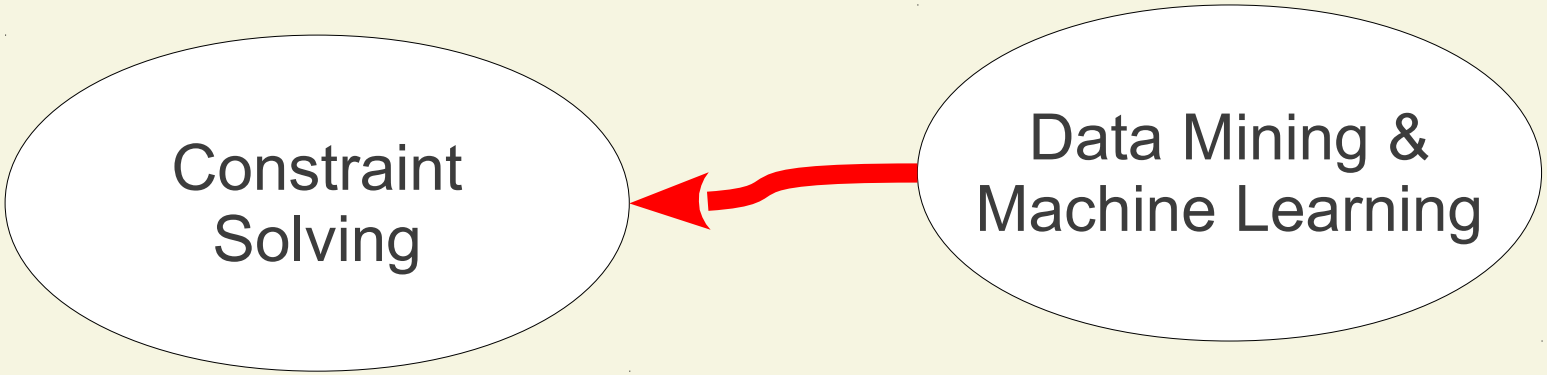
- general approach
- reuse of solvers, modeling primitives

Disadvantage

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

Use historical data to improve decisions?
$\rightarrow$ Machine Learning

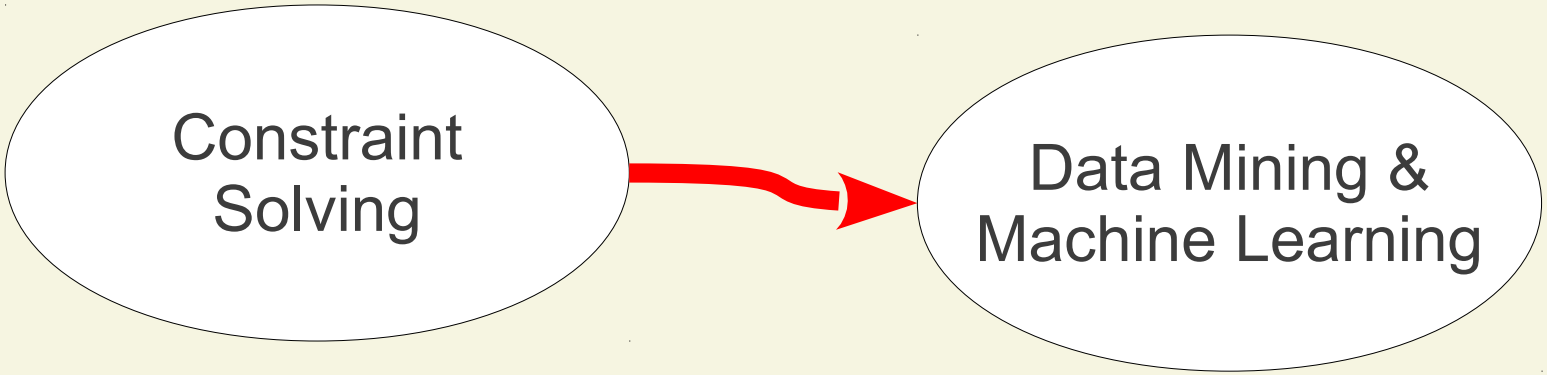
What DM & ML offers to CS



The use of historical data:

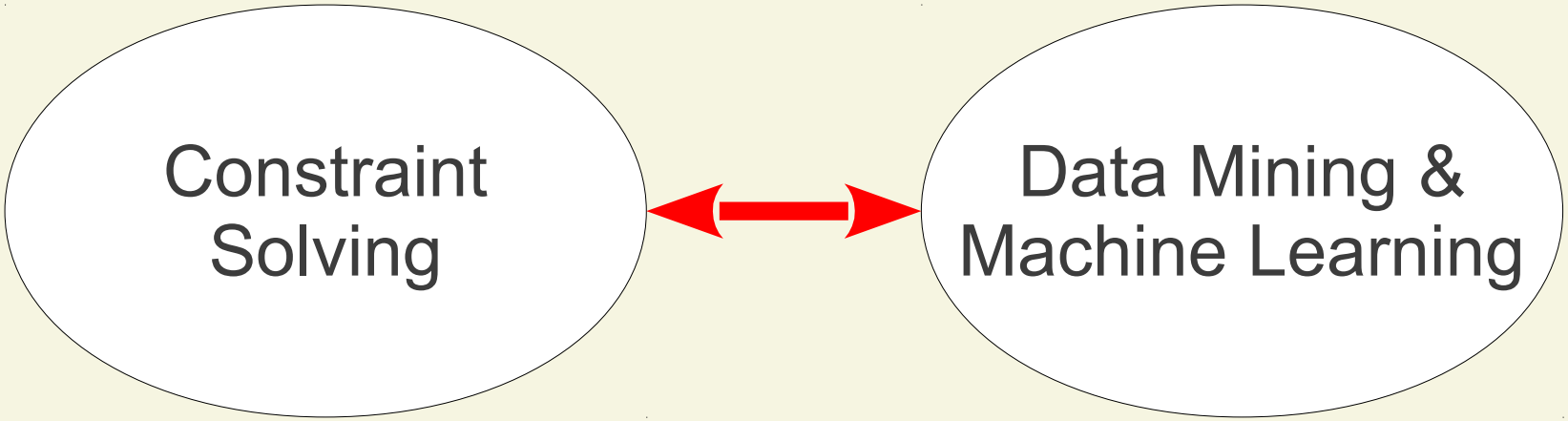
- Learning/improving models (constraints)
- Learning/improving search strategies, solver selection, heuristics, etc

What CS offers to DM & ML



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

This lecture



How can the two fields benefit from each-other?

- Part A: Introduction
- <u>Part B: Solving in ML & DM</u>
- 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, ...

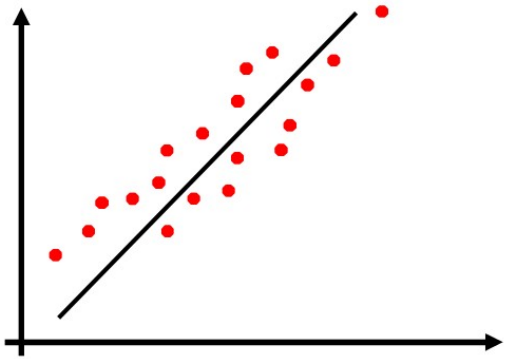
Linear Regression

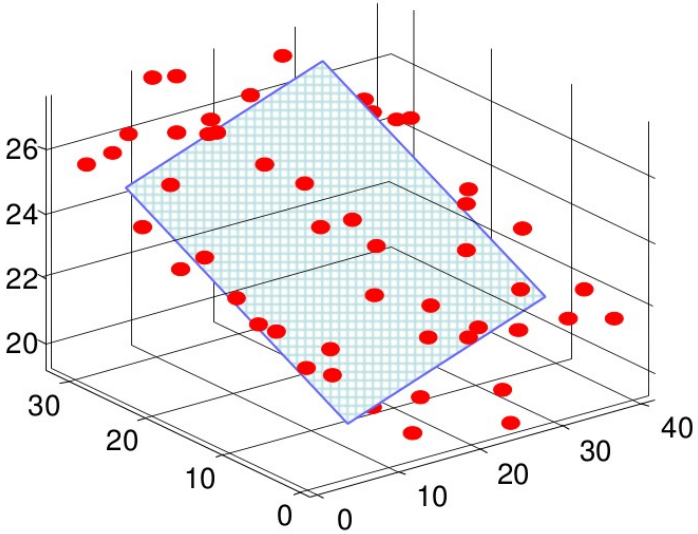
Notations:

- Datapoints $\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n\}$ $\quad \mathbf{x}_i \in R^d$
- Labels $\mathbf{y}=\{y_1, y_2, \ldots, y_n\}$ $\quad \mathbf{y} \in R$
- Linear decision function $f(\cdot): R^d \rightarrow R$

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

- Parameter vector $\mathbf{w}$

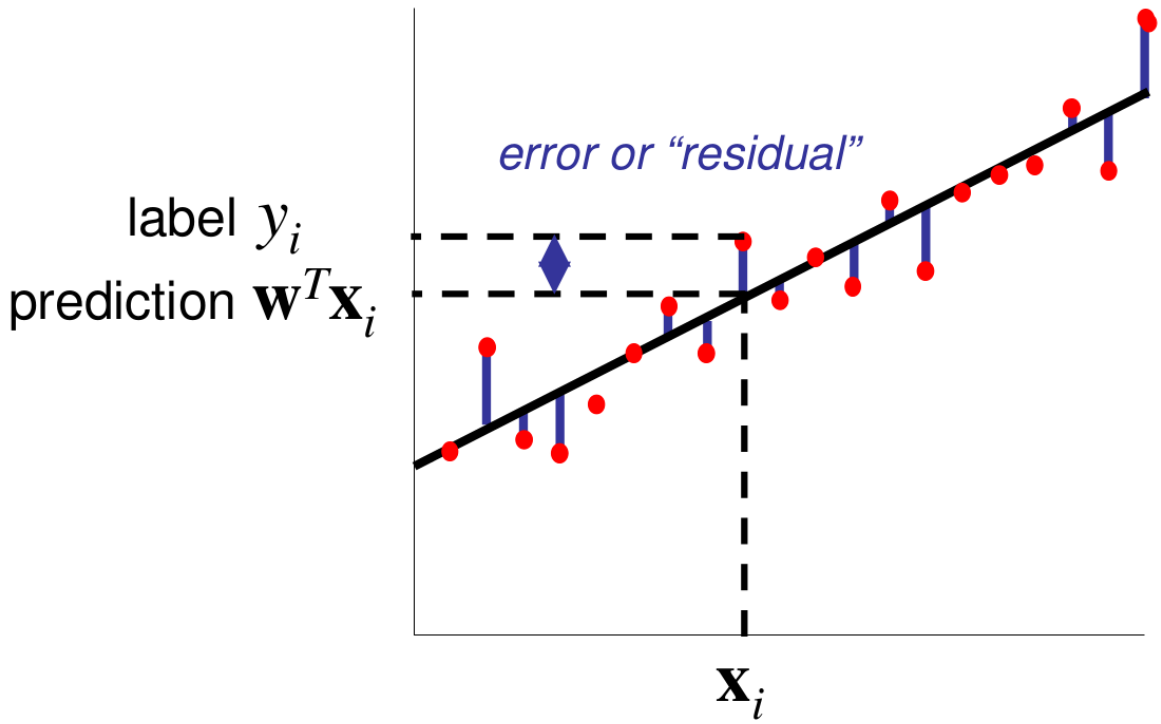




[Slide by E. Ricci, Analysis of Patterns, 2009]

Linear Regression

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



Sum Squared Error $\quad E = \sum_{i=1}^{n} \left(y_i - \mathbf{w}^T \mathbf{x}_i \right)^2 = \left\| \mathbf{y} - \mathbf{X}\mathbf{w} \right\|^2$

[Slide by E. Ricci, Analysis of Patterns, 2009]

Least Squares Regression

- Solve:

$$\min_{\mathbf{w}} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \qquad \Rightarrow$$

$$\min_{\mathbf{w}} \quad \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y} - 2\mathbf{w}^T\mathbf{X}^T\mathbf{y}$$

(gradient equal to 0):

$$\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y} = 0 \qquad \Rightarrow \qquad \mathbf{w} = 2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- This is *Least Squares Regression*.
- In alternative we can solve the linear system:

$$\mathbf{A}\mathbf{w} = \mathbf{b} \qquad \Rightarrow \qquad \mathbf{A} = \mathbf{X}^T\mathbf{X}, \quad \mathbf{b} = 2\mathbf{X}^T\mathbf{y}$$

[Slide by E. Ricci, Analysis of Patterns, 2009]

Ridge Regression

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

$$\min_{\mathbf{w}} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$
$$\text{s.t.} \quad C(\mathbf{w}) \leq C_{MAX}$$

- Solve:

$$\min_{\mathbf{w}} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$$
$$\text{s.t.} \quad \|\mathbf{w}\|^2 \leq 1$$

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

[Slide by E. Ricci, Analysis of Patterns, 2009]

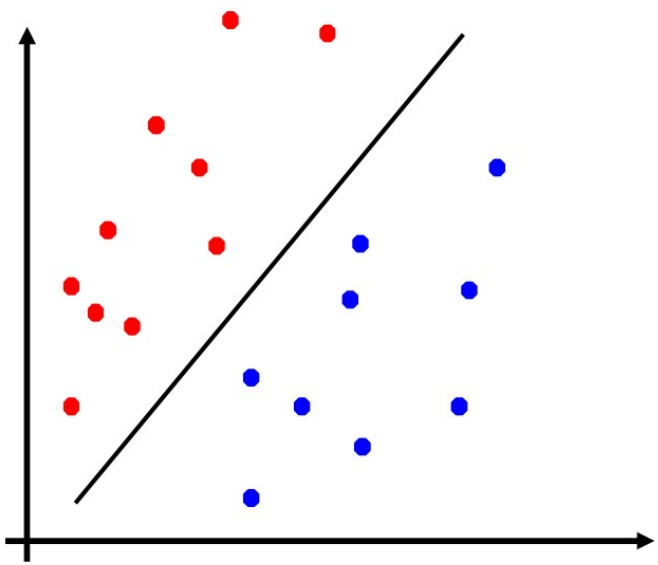
Classification

Notations:

- Datapoints $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ $\quad \mathbf{x}_i \in R^d$
- Labels $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ $\quad y_i \in \{-1, 1\}$
- Linear decision function $f(\cdot): R^d \rightarrow R$

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \hat{\mathbf{w}}^T \hat{\mathbf{x}}$$

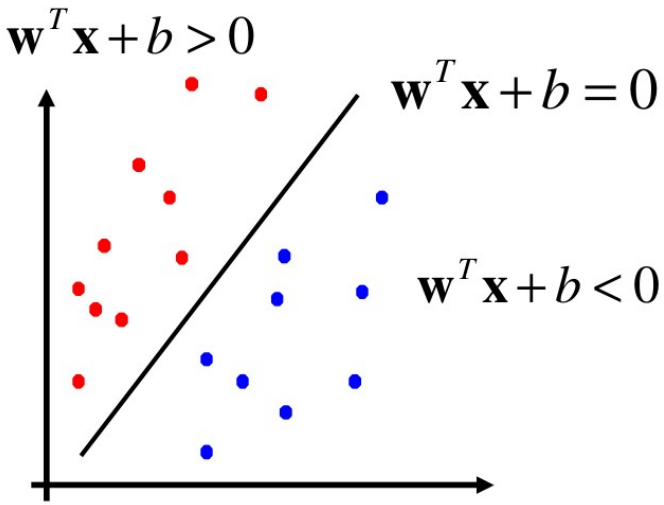
- Parameters $\hat{\mathbf{w}} = [\mathbf{w}\ b]$ and data $\hat{\mathbf{x}} = [\mathbf{x}\ 1]$.



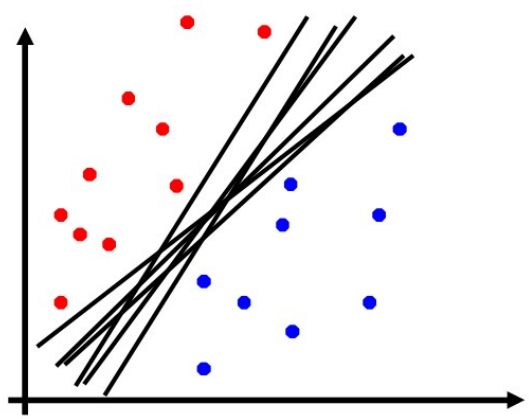
[Slide by E. Ricci, Analysis of Patterns, 2009]

Classification

- Binary classification:



- Which is the optimal separating hyperplane?



$$f(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x}+b\right)$$

[Slide by E. Ricci, Analysis of Patterns, 2009]

Support Vector Machines

- Given a training set:

$$T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \qquad \mathbf{x}_i \in R^d, \qquad y_i \in \{-1,1\}$$

- Goal:
  - Correctly classify all training data $\quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$
  - Maximize the Margin $\quad \rho = \dfrac{2}{\|\mathbf{w}\|}$
- This is equivalent to the following

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

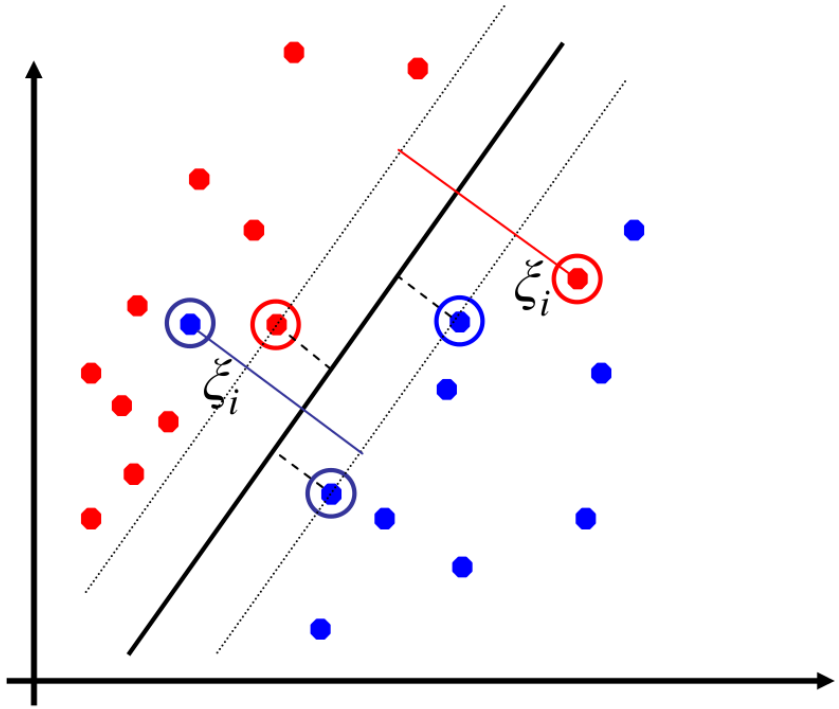
$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall i$$

- This is a Quadratic Programming (QP) problem.

[Slide by E. Ricci, Analysis of Patterns, 2009]

Support Vector Machines

- What if the training set is not linearly separable?
- *Slack variables* ξ_i can be added to allow misclassification of difficult or noisy examples.
- This is usually referred as s*oft-margin SVM*.



[Slide by E. Ricci, Analysis of Patterns, 2009]

Support Vector Machines

- Solve:

$$\min_{\mathbf{w},b,\xi_i} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$$

$$\text{s.t.} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0 \quad \forall i$$

- The problem is still convex and efficiently solvable.

- C is a *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.

[Slide by E. Ricci, Analysis of Patterns, 2009]

Declarative methods in ML

Often problem expressible as linear/convex opt. problem $\rightarrow$ 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?

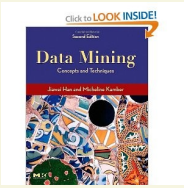
Basic mining task

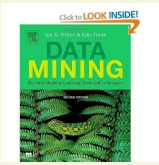
Analysing a dataset
to find patterns of interest

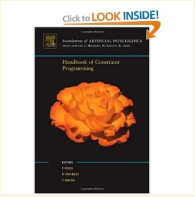
For example:

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 <u>new implementations</u>
- very <u>hard to combine</u> different constraints

Surprisingly, CP had not been applied to Pattern Mining

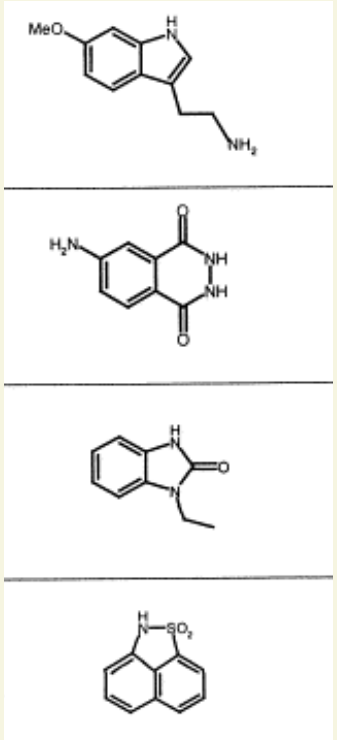
Pattern Mining

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 spam bacon spam tomato and spam; spam spam spam egg and spam; 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.

Graph Mining



Itemset Mining

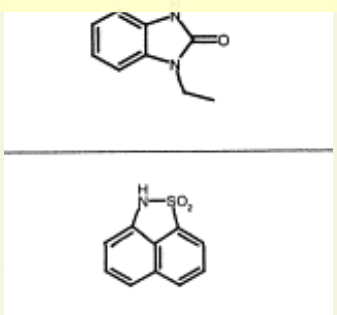
Pattern Mining

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 spam bacon and spam; spam sausage spam spam bacon spam tomato and spam; spam spam spam egg and spam; 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.

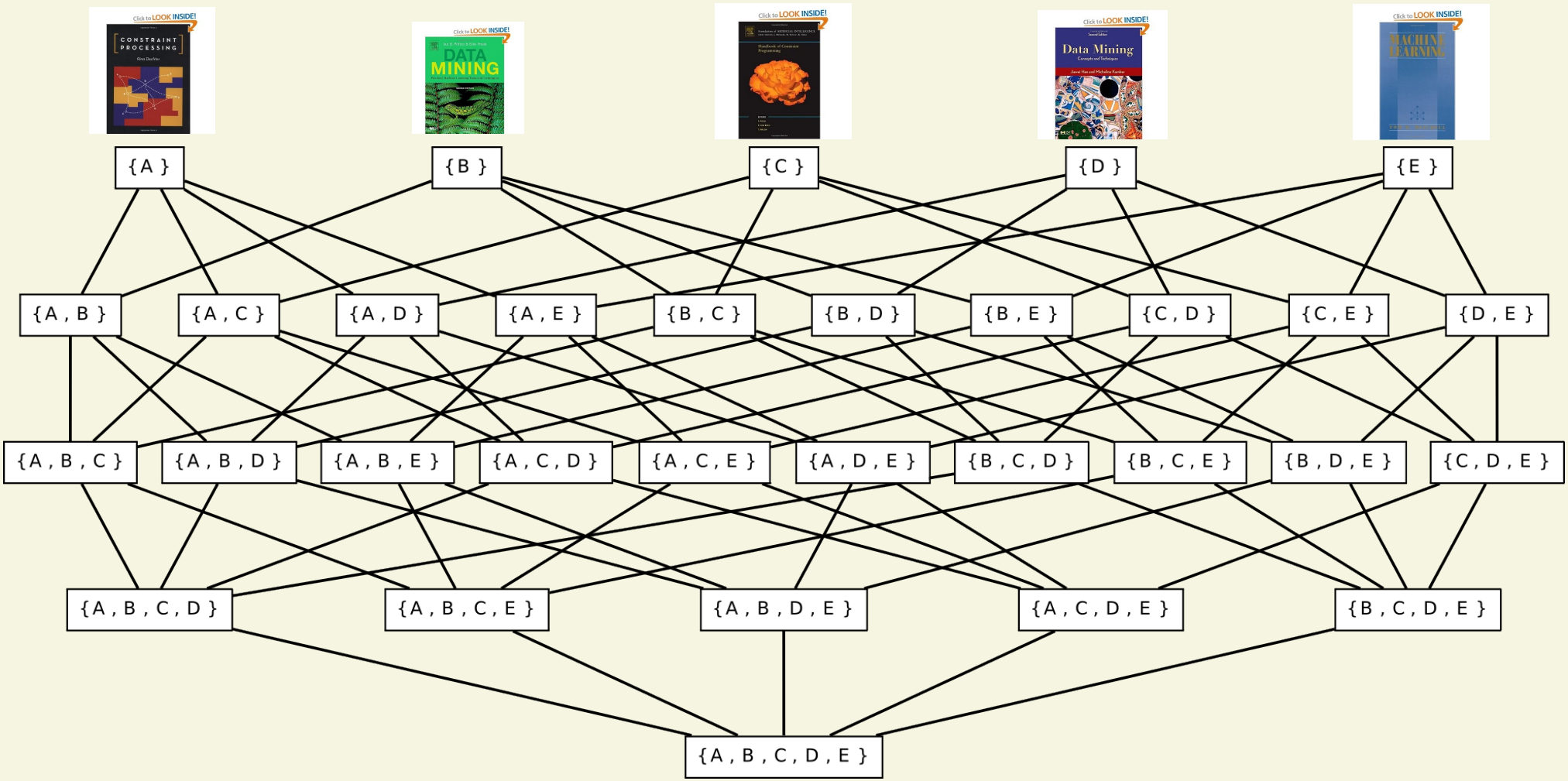
Graph Mining



Most fundamental setting

Itemset Mining

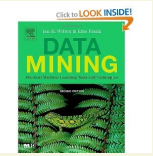
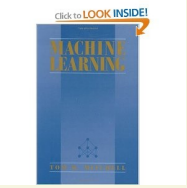
sets of items



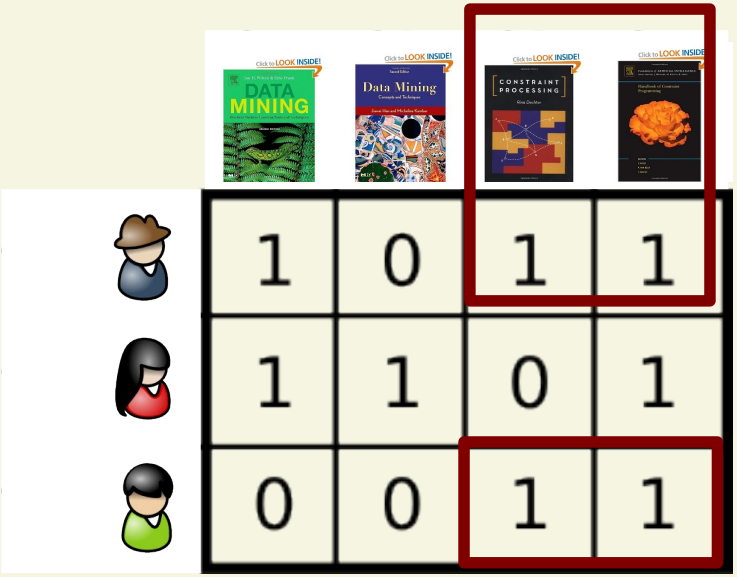
Itemset Mining

Find: set of *items* appearing frequently

Example:

{  ,  }: frequency = 2

Declarative approach



coverage(,) = { , }

frequency(,) = 2

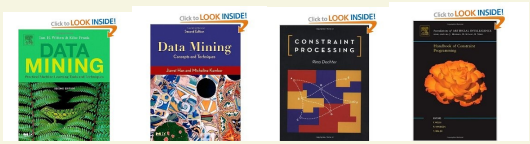
CP for Itemset Mining

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

coverage: $\forall T_t: \quad T_t = 1 \Leftrightarrow (\{I_1, \ldots I_n\} \subseteq row_t)$

frequency: $\sum_t T_t \geq Freq$

CP for Itemset Mining



	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

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

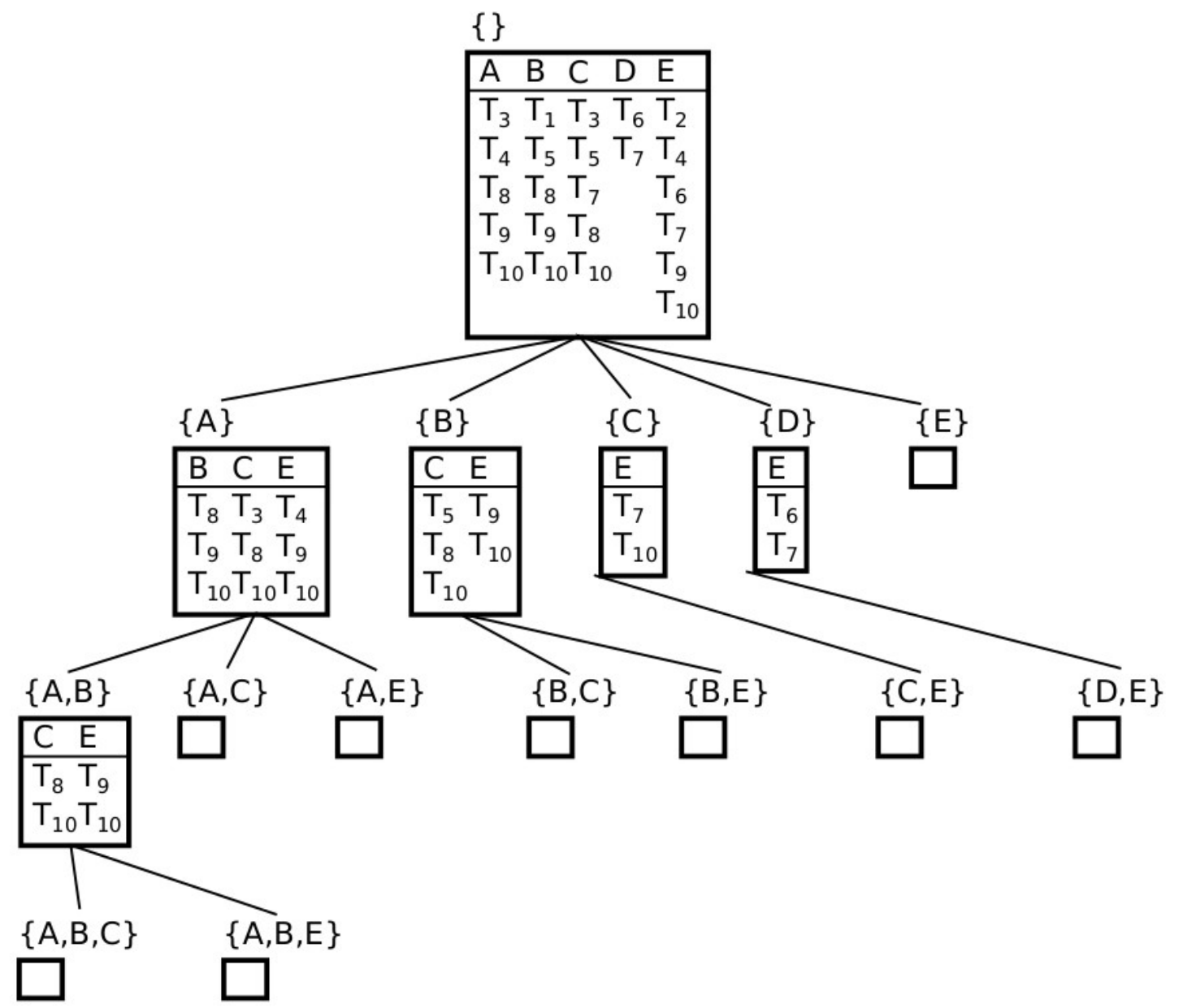
frequency: $\forall I_i: \quad I_i = 1 \Rightarrow \sum_{t, D_{ti}=1} T_t \geq Freq$

CP4IM, basic model

Algorithm 1 Fim_cp's frequent itemset mining model, in Essence'

```
given NrT, NrI : int
given TDB : matrix indexed by [int(1..NrT),int(1..NrI)] of int
given Freq : int

find Items : matrix indexed by [int(1..NrI)] of bool
find Trans : matrix indexed by [int(1..NrT)] of bool

such that

$ encode TDB: every Trans its complement has no supported Items
forall t: int(1..NrT).
  Trans[t] <=> ((sum i: int(1..NrI). Items[i]*(1-TDB[t,i])) = 0),

$ frequency: every Item is supported by sufficently many Trans
forall i: int(1..NrI).
  Items[i] => ((sum t: int(1..NrT). Trans[t]*TDB[t,i]) >= Freq)
```

Traditional search: proj. database



CP for Itemset Mining

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

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

	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

CP for Itemset Mining

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

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

- propagate i2

Intuition: infrequent
i2 can never be part of freq. superset

	i1 0/1	i2 0	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

CP for Itemset Mining

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

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

- propagate i2
- propagate t1

Intuition: unavoidable
t1 will always be covered

	i1	i2	i3	i4
	0/1	0	0/1	0/1
t1 1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

CP for Itemset Mining

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

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

- propagate i2
- propagate t1
- branch i1=1

		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0/1	0	0	1	1

CP for Itemset Mining

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

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

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

		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0	0	0	1	1

More constraints

- Coverage (required) $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$
- Frequent $I_i = 1 \Rightarrow \sum_t T_t D_{ti} \geq Freq$
- Maximal $I_i = 1 \Leftrightarrow \sum_t T_t D_{ti} \geq Freq$
- Closed $I_i = 1 \Leftrightarrow \sum_t T_t (1 - D_{ti}) = 0$
- Delta-closed $I_i = 1 \Leftrightarrow \sum_t T_t (1 - \delta - D_{ti}) = 0$
- ...

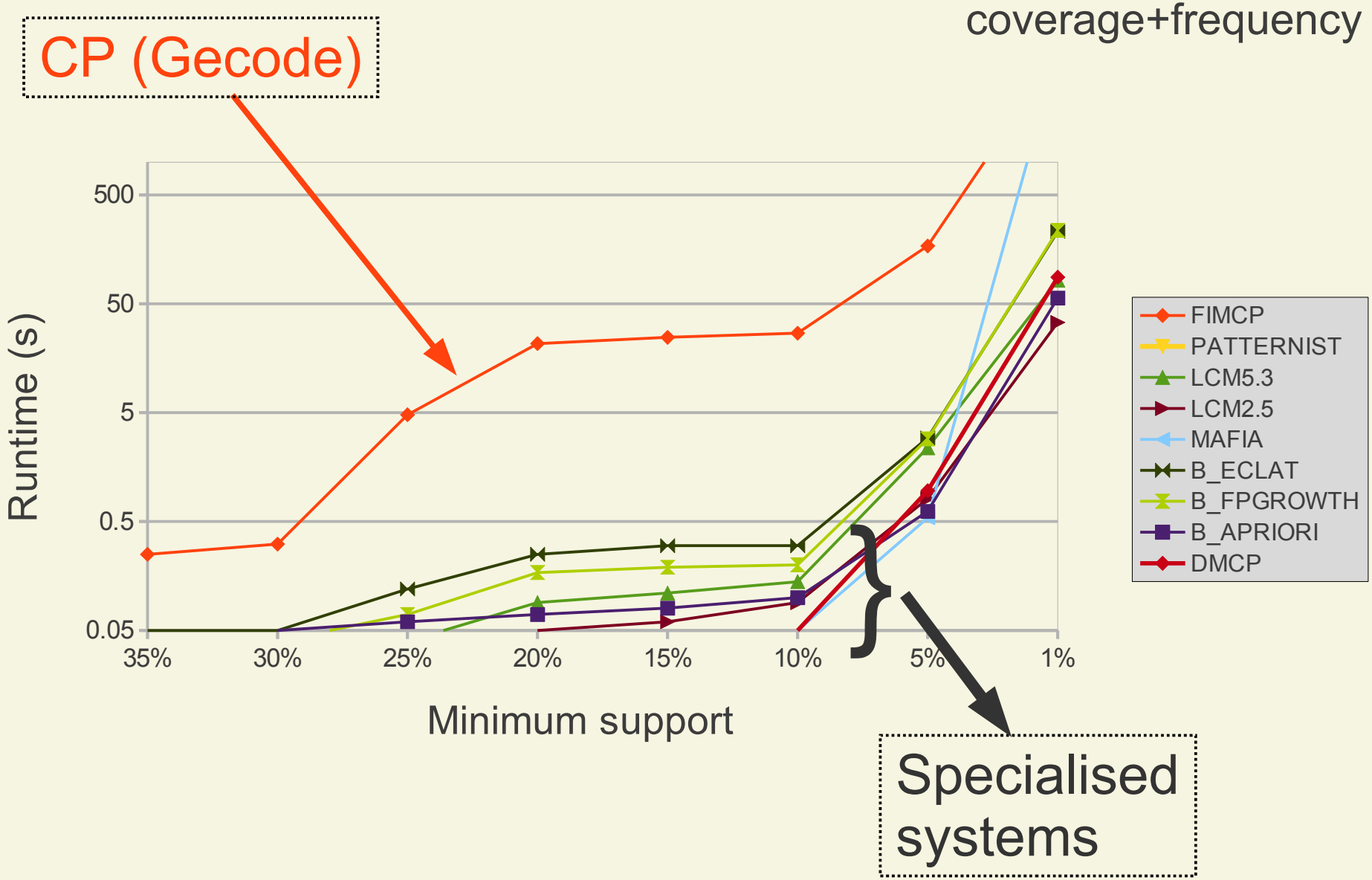
+ combinations

Generality

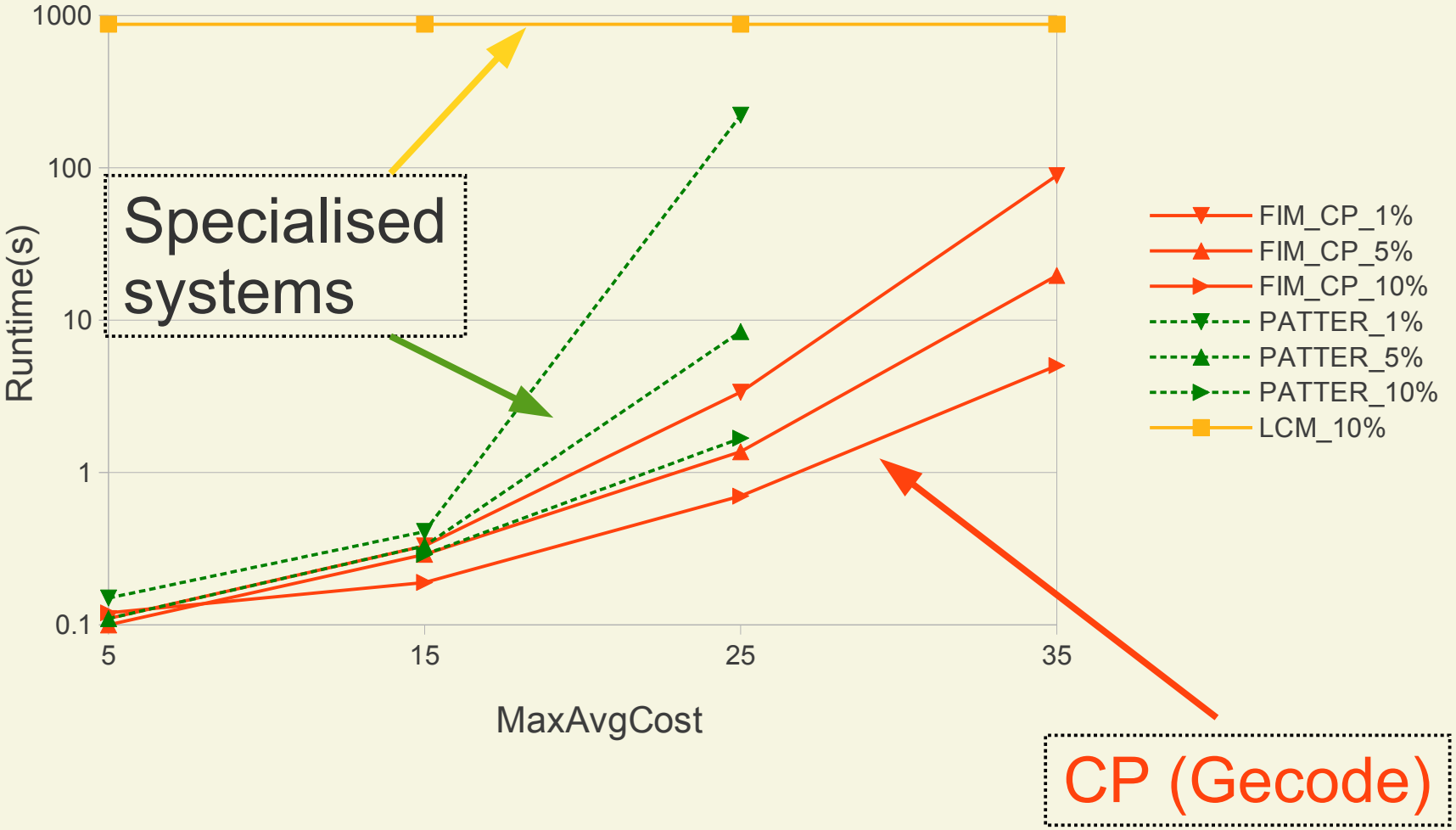
	LCM [15]	MAFIA [6]	ExAMiner [4]	DualMiner [5]	DDPrune [12]	CP4IM
Constraints on data						
Minimum frequency	X	X	X	X		X
Maximum frequency				X		X
Emerging patterns						X
Condensed Representations						
Maximal	X	X		X		X
Closed	X	X				X
$\delta-$Closed						X
Constraints on syntax						
Max/Min total cost			X	X		X
Minimum average cost			X			X
Max/Min size	X	X	X	X		X
Constraints on labelled data						
Minimum correlation					X	X
Maximum correlation						X

Simple Itemset Mining

coverage+frequency



Constraint-based mining



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

 : {➕ , ❌}  ,  : {➕➕ , }

Correlation constraint

$$f\left(\sum_{t \in P} T_t, \sum_{t \in N} T_t\right) \geq Bound$$

- 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 $I_i = 1 \Rightarrow \ldots$

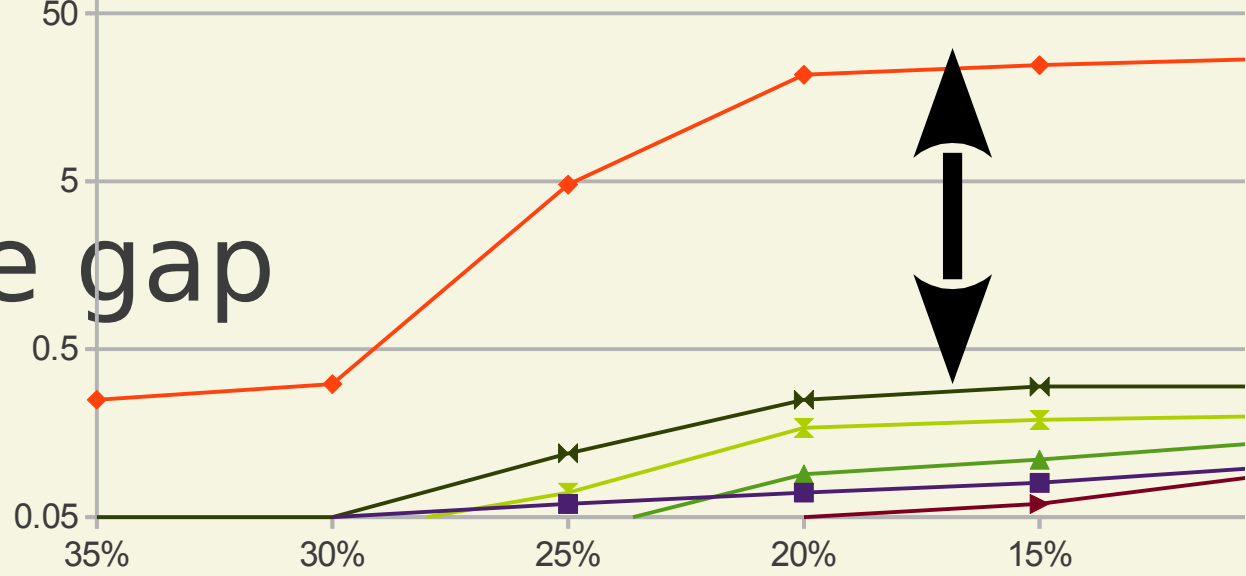
much stronger propagation !

Correlated itemset mining

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

Runtime in seconds

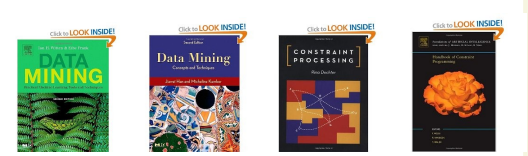
Decreasing the gap



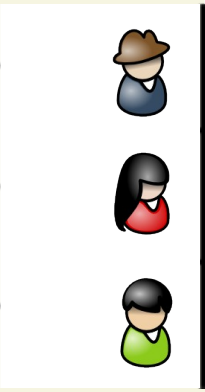
Hypothesis: unnecessary overhead in CP solver

An integrated CP solver would:

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



Itemset Mining principles



1	0	1	1
1	1	0	1
0	0	1	1

- Search strategy *Level-wise, BFS, DFS*
- Representation of data

1	0	1	1
1	1	0	1
0	0	1	1

1	0	1	1
1	1	0	1
0	0	1	1

1	0	1	1
1	1	0	1
0	0	1	1

- Representation of sets

1	0	1	1

{1, 3, 4}

0	1	0	0

{2}

Integration 1/3

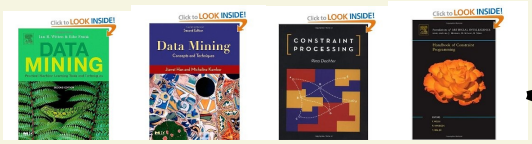
	Eclat Miner	Gecode CP Solver	**DMCP** CP Solver
Search strategy	DFS	DFS (binary)	DFS (binary)
Repres. of data	Shared, vertical	In constraints (up to 4 copies)	Shared matrix (default: vertical)

- Data shared (read-only) by constraints
- Horizontal, positive and negative *views* available

Integration 2/3

	Eclat Miner	Gecode CP Solver	Our **DMCP** CP Solver
Repres. of sets	Sparse or Dense	Sparse	Sparse or Dense
Types of vars.	Boolean vector (set)	Bool, Int, Set, ...	Boolean vector (set)

- Represented by lower and upper bound:



0, 0/1, 0/1, 1

Min: {0, 0, 0, 1}

Max: {0, 1, 1, 1}

Integration 3/3

	Eclat Miner	Gecode CP Solver	Our **DMCP** CP Solver
Constraints	Few, hard to combine	Many, easy to add/combine	Some, easy add/combine
Constraint activ.	Strict order (in algorithm)	On domain change	Change of lower/upper bound

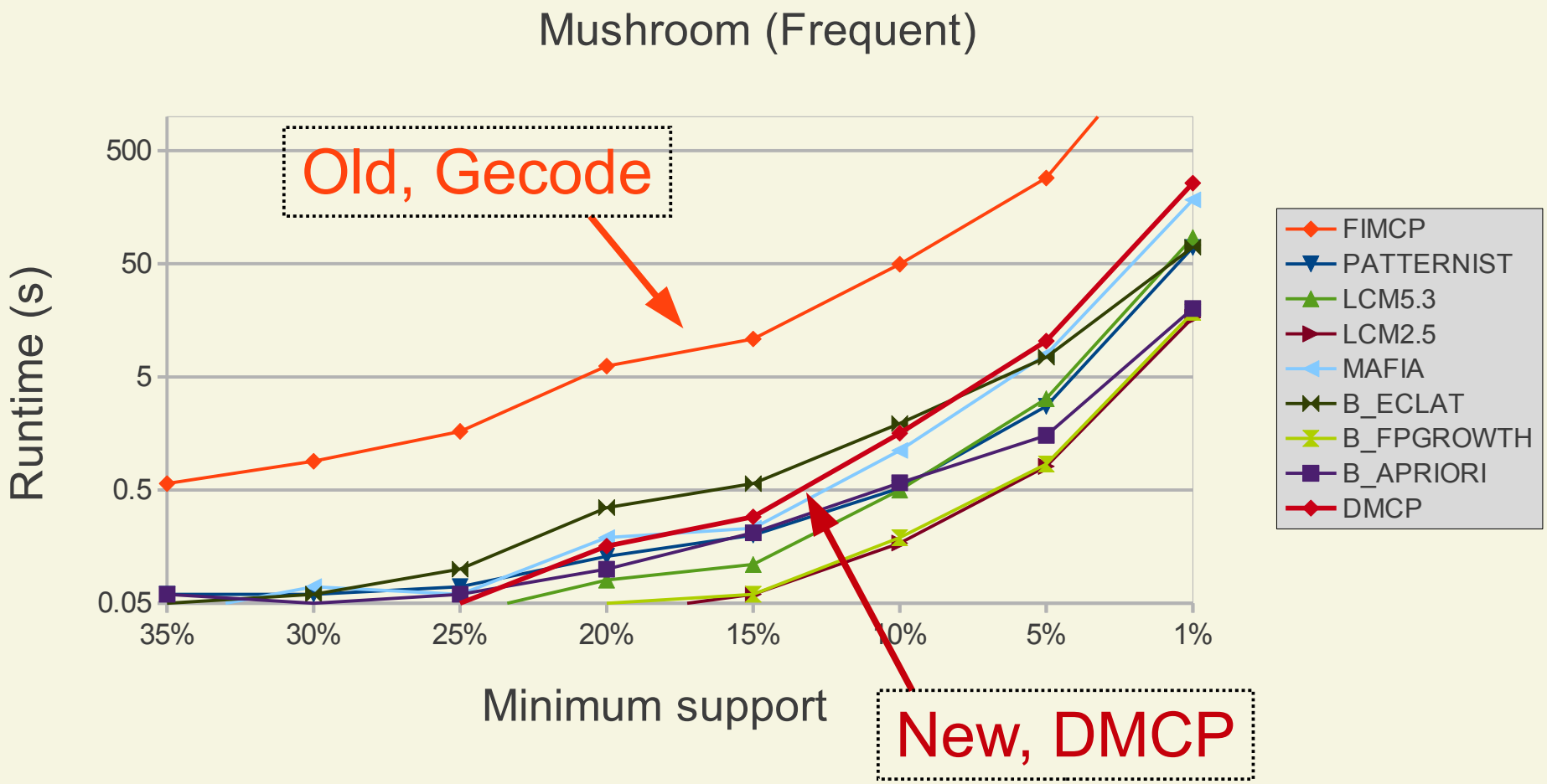
- General matrix constraint:

Data representation (matrix)

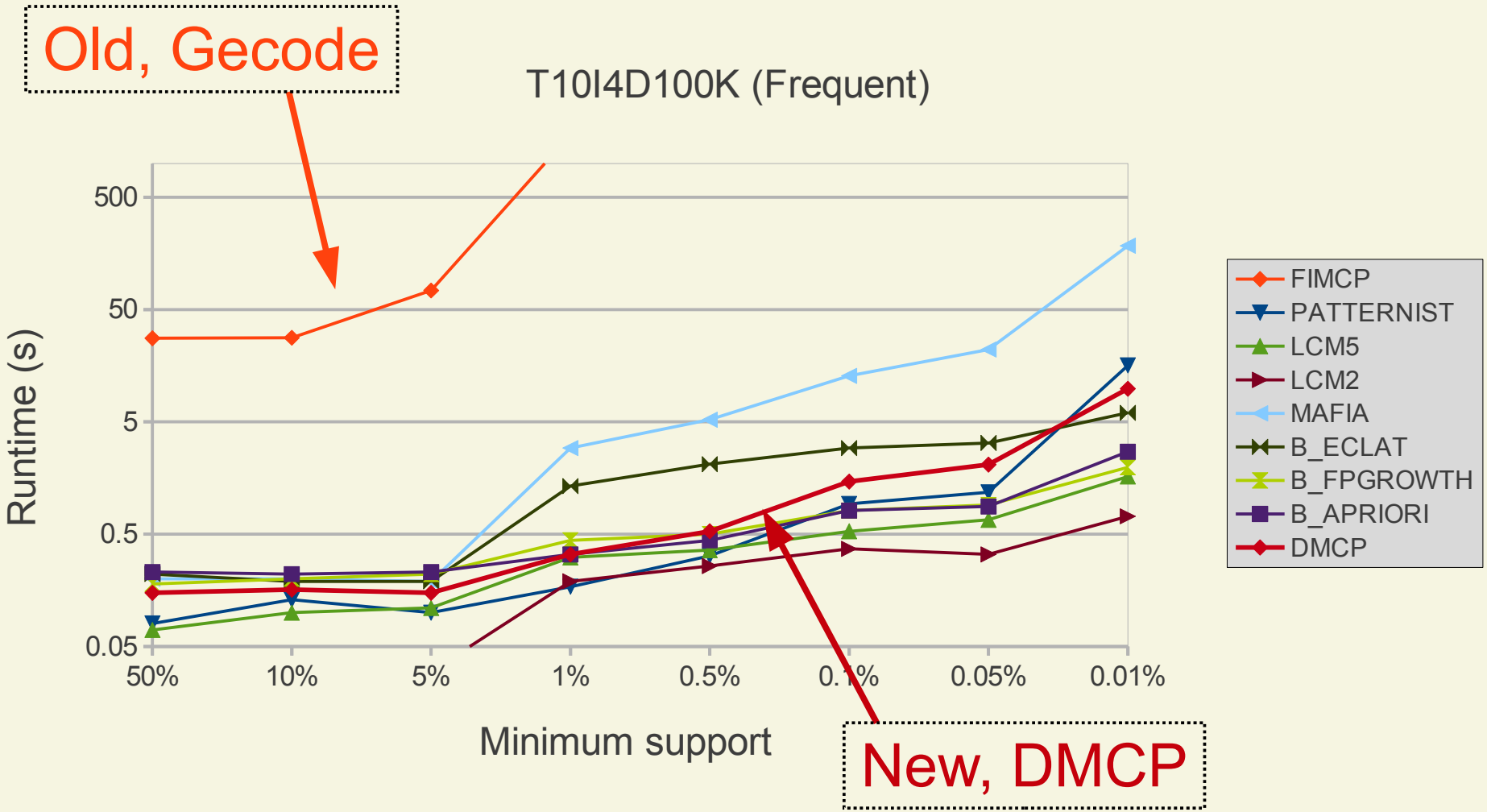
$$\boldsymbol{X} \geq_1 \mathbf{1}_{\geq_2 \theta}(\mathcal{A} \cdot \boldsymbol{Y});$$

Boolean vectors

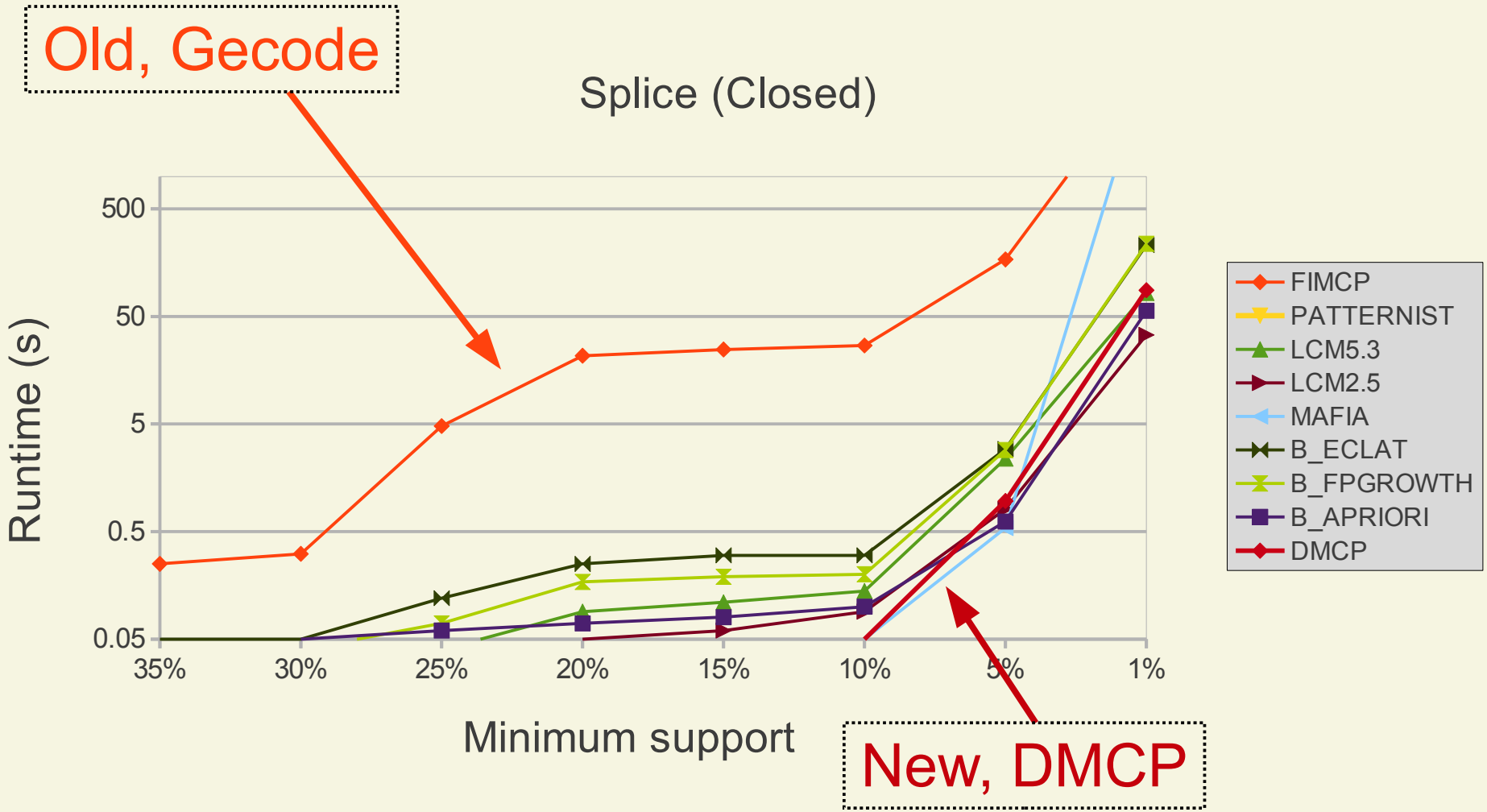
Frequent Itemset Mining



Frequent Itemset Mining, scaling



Closed Itemset Mining



CP for Itemset Mining

Model

Advantages of CP modelling:

- Easily add new constraints
- Freely combine constraints

Search

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], ...

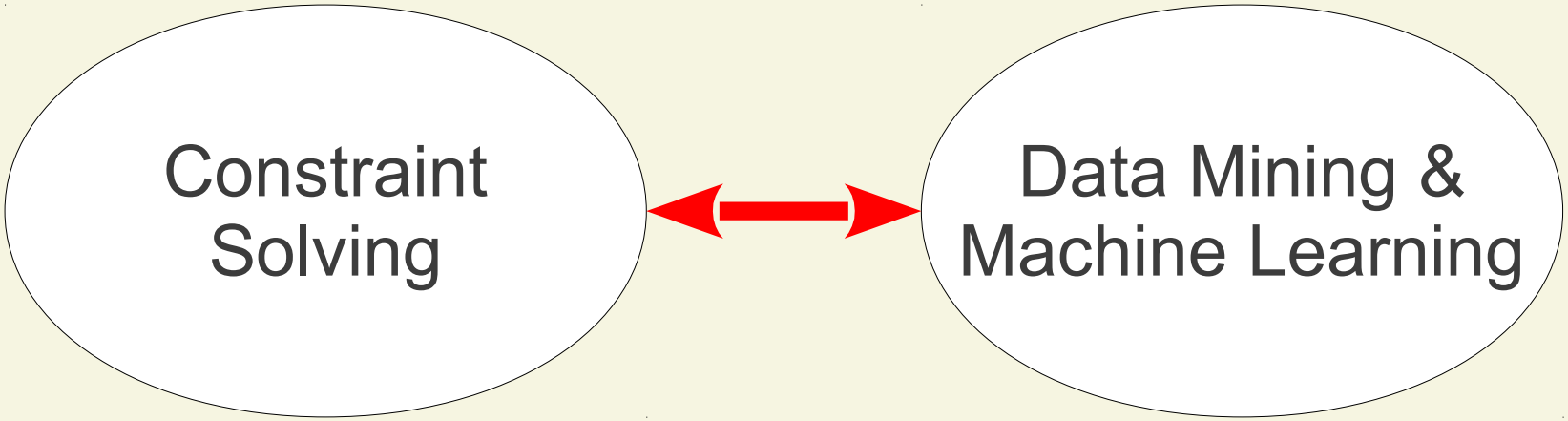
Mostly using hand-craft algorithms: Ripper, C4.5, k-means, Apriori, ...

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



How can the two fields benefit from each-other?

- Part A: Introduction
- Part B: Solving in ML & DM
- <u>Part C: Learning in constraint solvers</u>

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?
$\rightarrow$ Machine Learning

No Free Lunch theorem

No single algorithm is best on all problem instances

$\rightarrow$ 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?

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

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

$\rightarrow$ 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

<u>3. Calculate *features* on instance/data</u>

- What features to use?
  $\rightarrow$ 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)

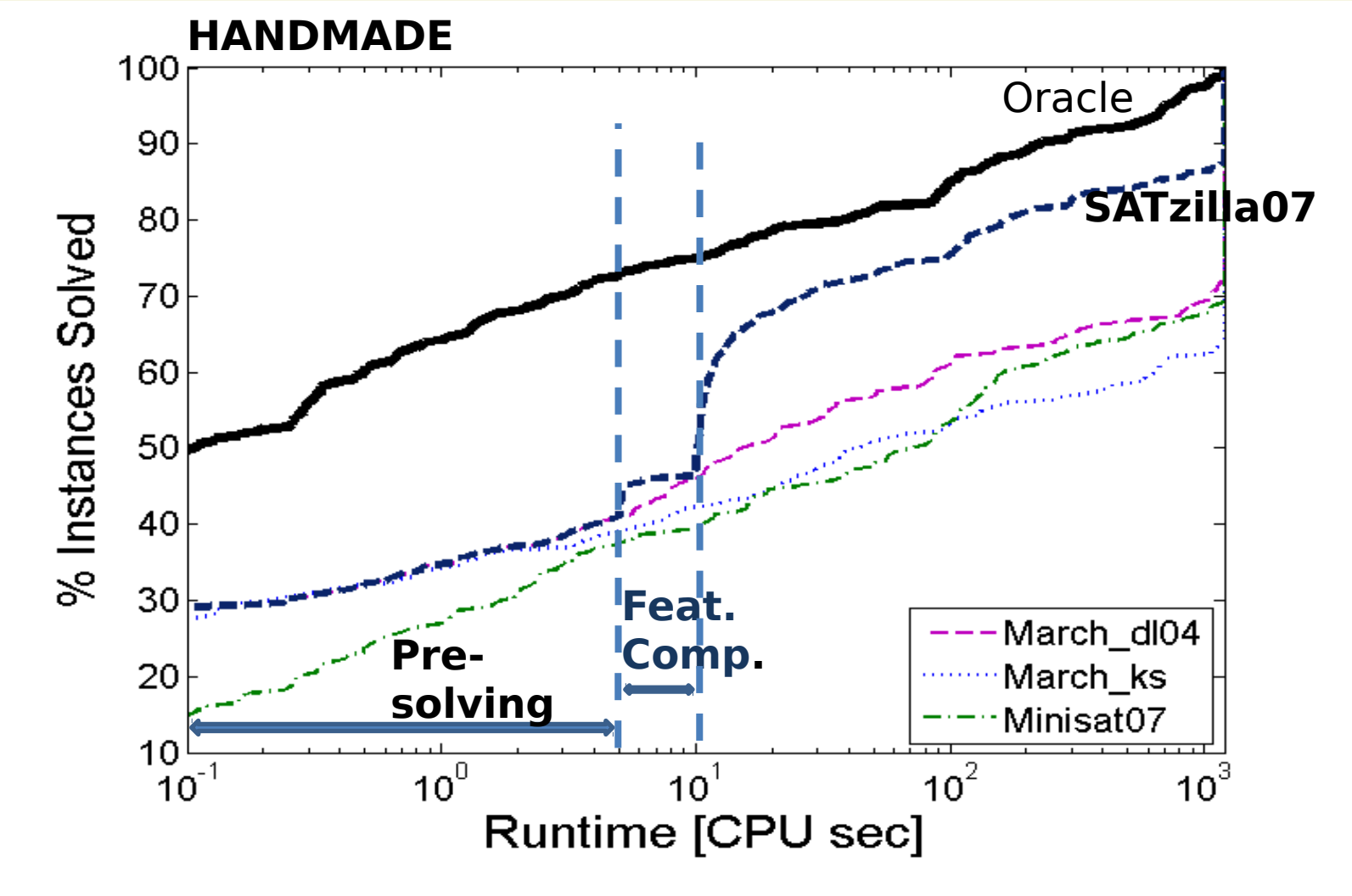
2. Problem instances/data
   - From previous competitions

SATZilla 2007: winning the SAT competition

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

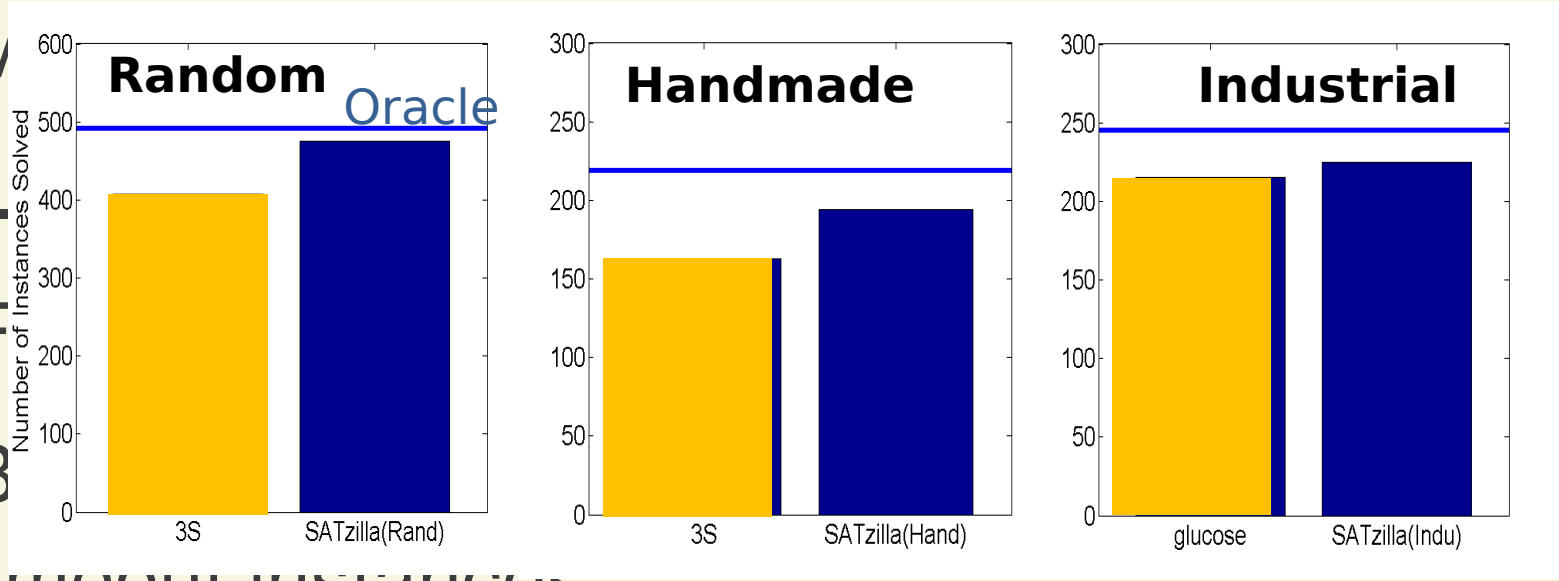
4. Build predictive model
   - logistic regression of *log*(runtime)
   - censored data/timeouts
   - hierarchical: predict sat/unsat

SATZilla 2007: winning the SAT competition



SATzilla 2011: more success

- M
- L
- F atures)
- B eature-
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 <u>variance</u>

- 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)

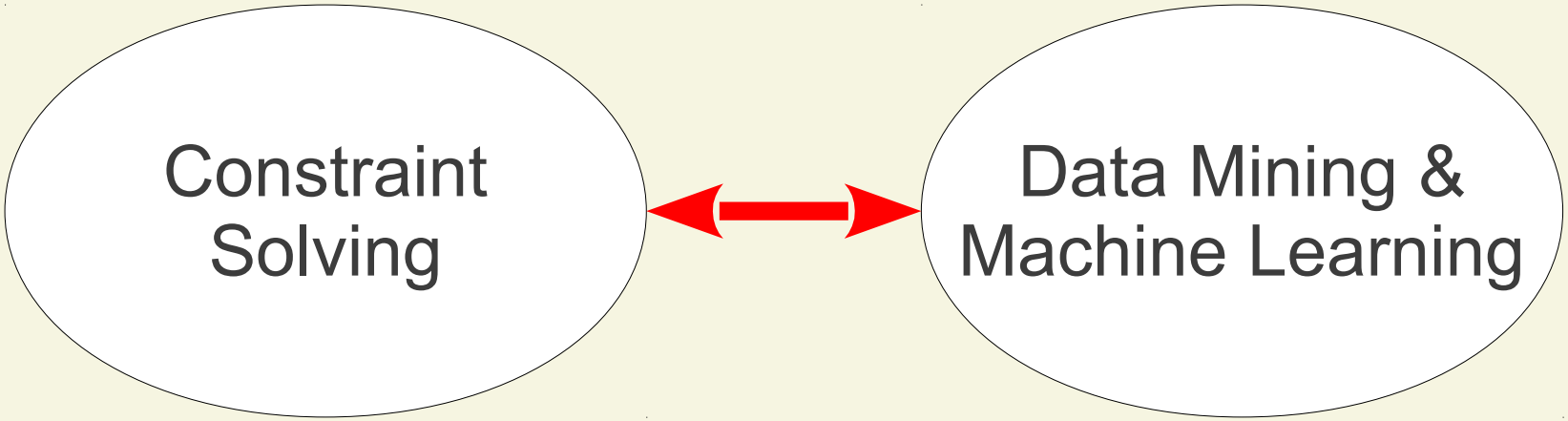
Open Questions

- 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]

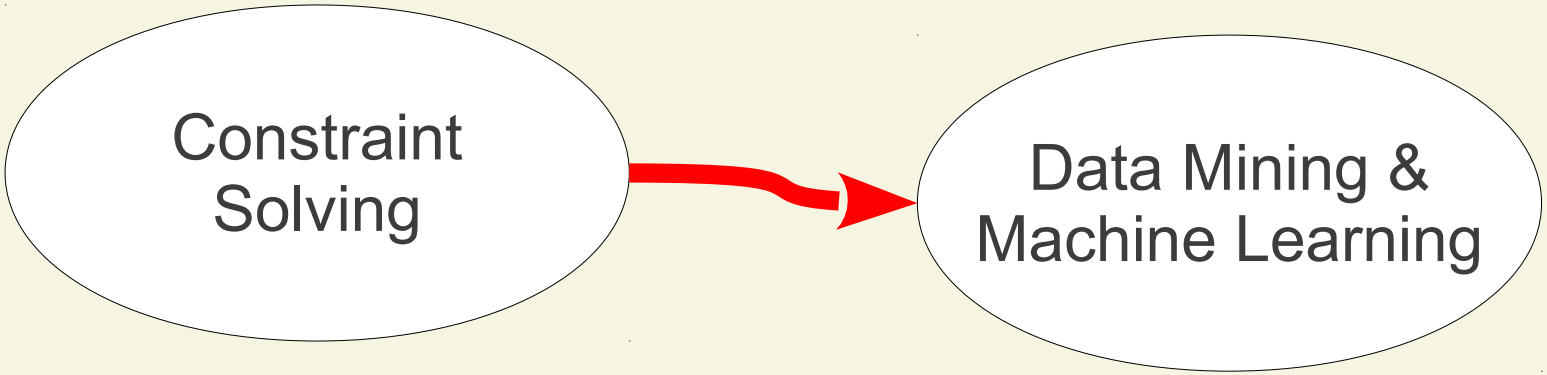
This lecture



How can the two fields benefit from each-other?

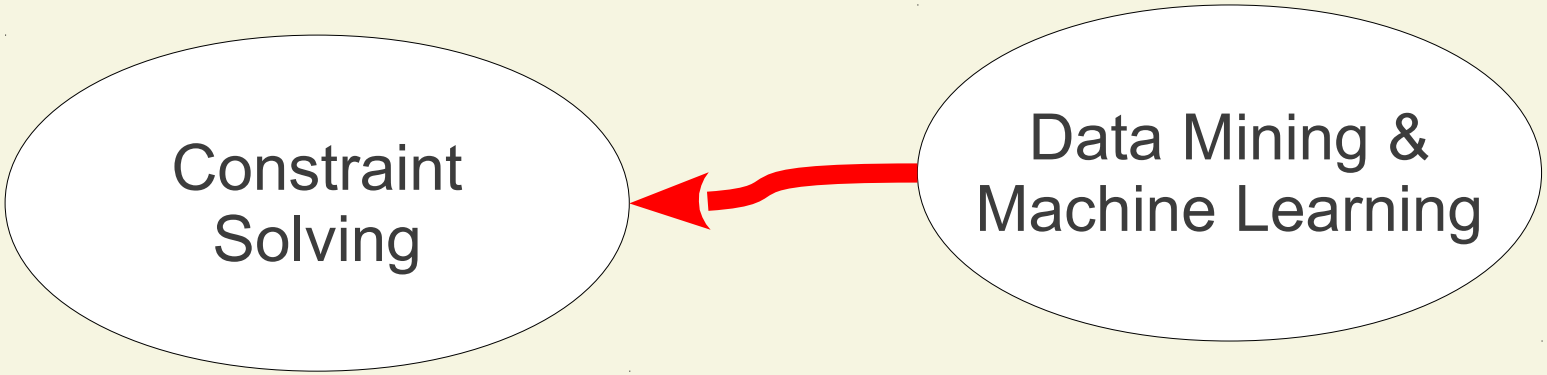
- Part A: Introduction
- Part B: Solving in ML & DM
- Part C: Learning in constraint solvers

What CS offers to DM & ML



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

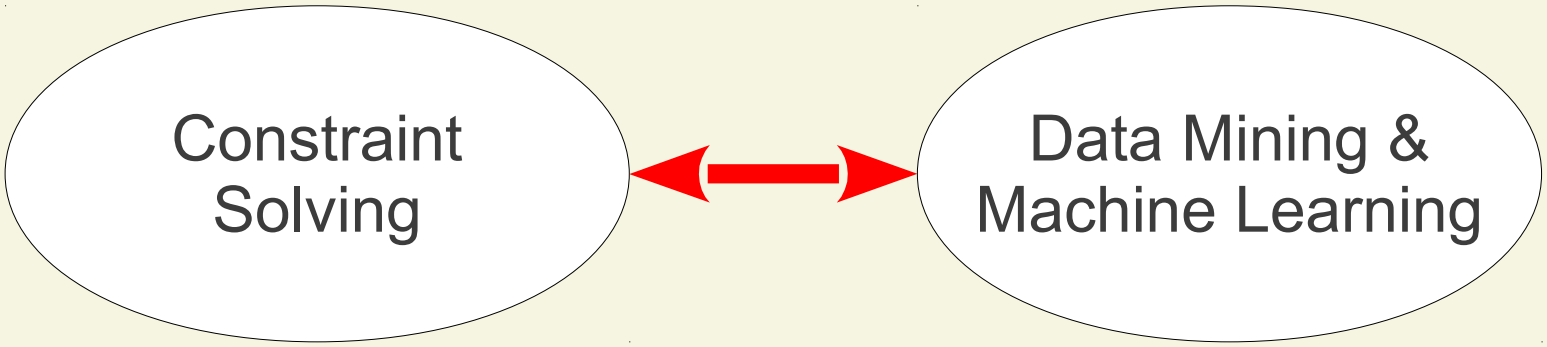
What DM & ML offers to CS



The use of historical data:

- Learning/improving models (constraints)
- Learning/improving search strategies, solver selection, heuristics, parameters, etc

Thank you for listening



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?