Roman Barták, Charles University

A NOVEL APPROACH TO INDUCTIVE LOGIC PROGRAMMING

a joint work with Filip Železný, Ondřej Kuželka, Andrej Chovanec

What is ILP?

positive examples

[b(n0), b(n1), r(n2), b(n3), r(n4), r(n5), b(n6), r(n7), r(n8), r(n9), r(n10), r(n11), r(n12), r(n13), r(n14), r(n15), r(n16), b(n17), r(n18), r(n19), a(n10, n5), a(n19, n1), a(n5, n3), a(n7, n11), a(n5, n19), a(n0, n1), a(n0, n2), a(n4, n0), a(n3, n0), a(n8, n3), a(n0, n6), a(n2, n14), a(n7, n0), a(n0, n10), a(n0, n13), a(n17, n10), a(n0, n15), a(n3, n9), a(n5, n12), a(n0, n18), a(n2, n1), a(n2, n17), a(n1, n4), a(n11, n6), a(n6, n12), a(n6, n1), a(n5, n16), a(n7, n16), a(n4, n9), a(n13, n11), a(n5, n14), a(n1, n10), a(n16, n10), a(n3, n13), a(n8, n4), a(n19, n8), a(n2, n4), a(n2, n3), a(n2, n5), a(n6, n14), a(n15, n3), a(n8, n5), a(n9, n7), a(n2, n6), a(n17, n15), a(n18, n1), a(n9, n11), a(n7, n5)]

([r(n0), b(n1), b(n2), b(n3), b(n4), r(n5), b(n6), b(n7), b(n8), r(n9), b(n10), r(n11), r(n12), b(n13), r(n14), r(n15), b(n16), b(n17), b(n18), r(n19), a(n10, n2), a(n2, n9), a(n6, n3), a(n2, n13), a(n2, n11), a(n0, n1), a(n2, n12), a(n2, n0), a(n0, n4), a(n15, n2), a(n3, n0), a(n9, n6), a(n5, n0), a(n8, n0), a(n0, n7), a(n0, n10), a(n5, n11), a(n0, n12), a(n2, n18), a(n6, n4), a(n0, n14), a(n0, n13), a(n0, n15), a(n19, n13), a(n2, n1), a(n4, n1), a(n17, n16), a(n3, n1), a(n10, n15), a(n5, n1), a(n16, n7), a(n1, n8), a(n7, n1), a(n16, n14), a(n8, n19), a(n7, n10), a(n1, n11), a(n10, n19), a(n14, n1), a(n7, n18), a(n6, n14), a(n1, n13), a(n1, n16), a(n9, n7), a(n12, n4), a(n1, n18), a(n7, n2), a(n17, n1)]

b(N0), b(N1), b(N2), b(N3), r(N4), a(N4, N2), a(N1, N4), a(N0, N3), a(N1, N3), a(N1, N0), a(N2, N1)

hypothesis

[r(n0), b(n1), b(n2), b(n3), r(n4), r(n5), b(n6), r(n7), r(n8), b(n9), b(n10), b(n11), b(n12), b(n13), r(n14), b(n15), b(n16), r(n17), b(n18), r(n19), a(n13, n12), a(n7, n12), a(n12, n14), a(n6, n3), a(n13, n2), a(n9, n8), a(n5, n3), a(n1, n0), a(n0, n2), a(n4, n0), a(n0, n3), a(n7, n19), a(n5, n0), a(n3, n7), a(n9, n19), a(n10, n0), a(n16, n4), a(n3, n10), a(n10, n11), a(n8, n6), a(n12, n17), a(n2, n1), a(n17, n0), a(n1, n4), a(n3, n1), a(n19, n0), a(n10, n15), a(n16, n5), a(n5, n1), a(n11, n3), a(n9, n18), a(n13, n11), a(n1, n7), a(n1, n9), a(n18, n5), a(n1, n12), a(n10, n7), a(n6, n5), a(n6, n14), a(n3, n15), a(n14, n10), a(n5, n8), a(n1, n15), a(n12, n11), a(n9, n17), a(n1, n18), a(n4, n11), a(n7, n2)]

[b(n0), r(n1), r(n2), b(n3), r(n4), b(n5), r(n6), r(n7), b(n8), r(n9), r(n10), r(n11), r(n12), r(n13), r(n14), r(n15), r(n16), r(n17), r(n18), b(n19), a(n10, n18), a(n8, n10), a(n7, n12), a(n16, n19), a(n6, n3), a(n3, n19), a(n2, n13), a(n11, n16), a(n15, n9), a(n5, n3), a(n0, n1), a(n2, n0), a(n2, n12), a(n9, n6), a(n8, n11), a(n7, n0), a(n7, n3), a(n18, n2), a(n0, n14), a(n8, n6), a(n3, n9), a(n2, n1), a(n1, n4), a(n3, n1), a(n15, n10), a(n3, n11), a(n5, n1), a(n8, n1), a(n14, n3), a(n7, n1), a(n16, n18), a(n1, n10), a(n10, n16), a(n3, n13), a(n1, n9), a(n2, n4), a(n2, n3), a(n3, n16), a(n19, n10), a(n14, n1), a(n2, n6), a(n11, n17), a(n6, n13), a(n5, n17), a(n4, n11), a(n17, n3)]

[b(n0), b(n1), b(n2), b(n3), b(n4), r(n5), b(n6), r(n7), b(n8), r(n9), b(n10), r(n11), r(n12), b(n13), r(n14), r(n15), r(n16), r(n17), b(n18), r(n19), a(n3, n4), a(n12, n13), a(n9, n2), a(n14, n12), a(n12, n18), a(n0, n1), a(n4, n14), a(n0, n2), a(n0, n4), a(n0, n3), a(n2, n15), a(n19, n7), a(n9, n6), a(n6, n0), a(n5, n0), a(n3, n7), a(n13, n9), a(n9, n19), a(n0, n12), a(n5, n11), a(n10, n3), a(n10, n11), a(n8, n6), a(n17, n12), a(n12, n5), a(n0, n17), a(n12, n3), a(n7, n8), a(n6, n1), a(n2, n19), a(n9, n10), a(n14, n3), a(n16, n18), a(n14, n16), a(n18, n15), a(n10, n16), a(n3, n13), a(n7, n10), a(n1, n11), a(n16, n3), a(n6, n5), a(n12, n15), a(n9, n7), a(n15, n17), a(n15, n14), a(n7, n2), a(n7, n5)]

negative examples

ILP in detail

example

..., arc(a,b), arc(b,a), arc(a,c), arc(c,d), red(a), blue(c), blue(b) ...

Does the hypothesis entails the example?

θ-subsumption

$H\theta \subseteq E$

$\theta = \{ N1/a, N2/b \}$

hypothesis

arc(N1,N2), arc(N2,N1)

Subsumption problem
How to find an instantiation of variables in the hypothesis such that the hypothesis subsumes the example?

How to obtain the hypothesis from a given template?

$T\sigma = H$

$\sigma = \{ X4/X1, X3/X2 \}$

template

arc(X1,X2), arc(X3,X4)

A template consistency problem
How to find out which variables are unified in the template?

Talk outline

- Subsumption checking
  How to check whether the hypothesis subsumes an example?
  - Django algorithm
- Template consistency
  How to unify variables in the template to obtain a consistent hypothesis?
  - unification models with hints [AIMSA 2010]
- Template generation
  How to obtain a template?
  - stochastic history-driven generator [MICAI 2011]
- Problem decomposition
  How to improve overall efficiency?
  - a decompose-merge-refine approach [AIMSA 2012]

Subsumption problem

Does hypothesis H subsume an example E?

Recall:

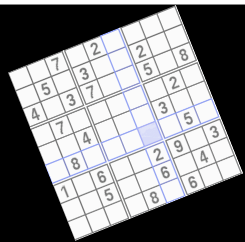
- **example** is a set of grounded atoms
  `…, arc(a,b), arc(b,a), arc(a,c), arc(c,d), red(a), blue(c), blue(b) …`
- **hypothesis** is a set of atoms with variables
  `arc(N1,N2), arc(N2,N1)`

θ-subsumption

- H subsumes E iff there exists a substitution θ such that $H\theta \subseteq E$

Using Constraint Programming techniques to check θ-subsumption.

What is CP?



Constraint Programming is a technology for solving combinatorial optimization problems modeled as constraint satisfaction problems:

- a finite set of decision **variables**
- each variable has a finite set of possible values (**domain**)
- combinations of allowed values are restricted by **constraints** (relations between variables)

Mainstream **solving approach** combines

inference (removing values violating constraints) with **search** (trying combinations of values)

(Maloberti and Sebag, 2004)

Django algorithm

- The subsumption problem is formulated as a constraint satisfaction problem.
- **Example** defines the domains of constraints
  - atoms with the same name → constraint domain
  - `…, arc(a,b), arc(b,a), arc(a,c), arc(c,d), red(a), blue(c), blue(b) …`
  - binary constraint `arc` = {(a,b), (b,a), (a,c), (c,d)}
  - unary constraint `blue` = {(c), (b)}
- **Hypothesis** formulates the CSP
  - atom with variables → constraint
  - `CSP: arc(N1,N2), arc(N2,N1)`

Template consistency

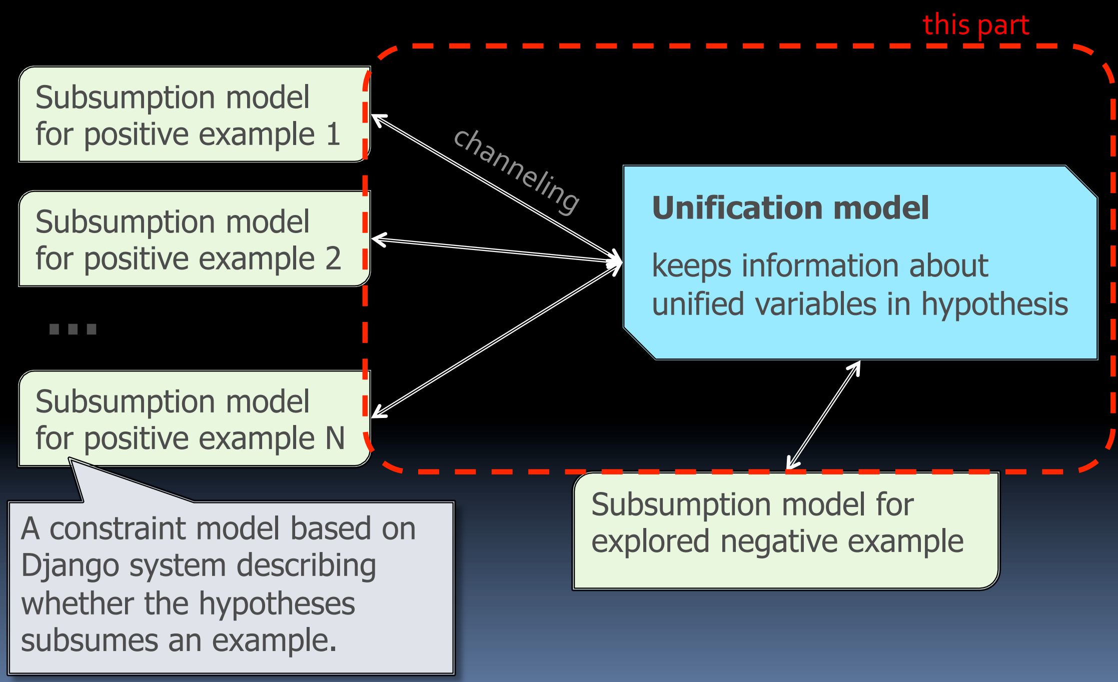
How to obtain a hypothesis (from a given template) consistent with examples?

Recall:

- **hypothesis** is a set of atoms with shared variables
  `arc(N1,N2), arc(N2,N1)`
  - hypothesis H is consistent with examples iff H subsumes all positive examples and H does not subsume any negative example
- **template** is a set of atoms with unique variables
  `arc(X1,X2), arc(X3,X4)`
- $T\theta = H$, where θ is a unification of variables

Using Constraint Programming techniques to solve the template consistency problem.

The concept



Index model (variables)

How to model which ILP variables are unified in the template?

template: arc(X1,X2), arc(X3,X4), blue(X5)

hypothesis: arc(N1,N2), arc(N2,N1), blue(N2)

- unification can be seen as a **mapping**
  - X3 → X2, X4 → X1, X5 → X2
  - always map the variable with the larger index to the variable with the smaller index
- mapping is modeled using **index CP variables**
  - `Ij` with domain `{1,…,j}`
    - ILP variable Xj maps to ILP variable X_{Ij}
  - `I1=1, I2=2, I3=2, I4=1, I5=2`

Index model (constraints)

element(I, [A_1,…,A_n], B)
models the relation **A_I = B**

- uniquen… (X3 → X2 , X5 → X2 vs. ~~X3 → X2, X5 → X3~~)
  element … **, Ij)**

arc(X1,X2), arc(X3,X4)
goes to constraint
lex([I1,I2],[I3,I4])

- "symme…
  (blue(X4), blu…5): X4 → X1, X5 → X2 vs. ~~X4 → X2, X5 → X1~~)
  I4 < I5
- channeling (from hypothesis to subsumption test)
  element(Ij, [X1,…,Xn], Xj)
- decisions (variables X2 and X3 are/aren't unified)
  I2 = I3
  I2 ≠ I3

Search framework

How to decide about the unified variables?

- Unification of variables in the template is necessary only to break subsumption of negative examples!

Do for all negative examples
  While the example is subsumed by current hypothesis
    Find subsumption θ (solution to a corresponding CSP with variables X1,…, Xn)
    Select a pair Xi, Xj such that Xi θ ≠ Xj θ
    Post constraint `Ii = Ij` (alternatively `Ii ≠ Ij`)
Find subsumption for all positive examples

Boolean model

Problem: conflict **I1=I2**, **I1≠I2** is not discovered by local (arc) consistency!

We can strengthen inference by explicitly keeping information about unified/non-unified variables in a "Boolean matrix".

	I_1	I_2	I_3	I_4	I_5	I_6
I_1	1	0	A	B	C	D
I_2	0	1	E	F	G	H
I_3	A	E	1	0	I	J
I_4	B	F	0	1	K	L
I_5	C	G	I	K	1	0
I_6	D	H	J	L	0	1

$I_1 \neq I_2$, $I_3 \neq I_4$, $I_5 \neq I_6$

$\xrightarrow{I_2 = I_3}$

1	0	0	B	C	D
0	1	1	0	G	H
0	1	1	0	G	H
B	0	0	1	K	L
C	G	G	K	1	0
D	H	H	L	0	1

Hints

- Assume that the example contains the following atoms for predicate arc:
  - arc(a,b), arc(b,a), arc(a,c), arc(c,a)
- Clearly X1 can not unify to X2 in arc(X1,X2).
- But constraints arc(X1,X2), X1=X2 are (arc) consistent!
- By exploring atoms with the same predicate symbol in positive examples, it is possible to deduce that some variables (of this atom in the hypothesis) cannot unify (hint).
  - I1≠I2

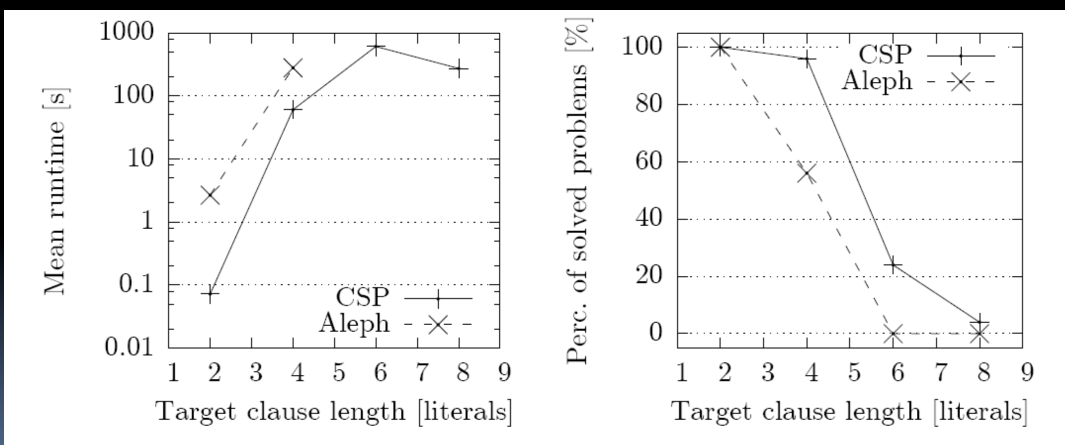
Experiments (inside)

- Comparison of runtimes (milliseconds) for identifying common structures in randomly generated structured graphs (Erdős-Rényi).

#atoms	#vars	Index	Boolean	Combined	Decoupled		
					full	no SB	no hints
7	9	0	15	16	0	16	0
8	11	0	16	15	0	15	0
8	11	31	281	31	21	343	32
8	11	94	343	109	78	421	78
9	13	94	1046	125	93	1373	94
9	13	328	1810	436	312	2309	312
9	13	1232	9626	1606	1170	12324	1185
10	15	>600000	86425	>600000	236514	110981	>600000

Experiments (outside)

- comparison to widespread ILP system Aleph



Obtaining a template

How to obtain a template?

Recall:

- template consists of atoms with fresh variables (each variable appears exactly once)

The task:

- **How many copies of each atom will appear in the template?**

Searching the space of templates and learning the most promising atoms in the template.

Existing approach

Iterative deepening search

- generate all templates of given length
- if no consistent hypothesis found then increase the length

Features:

- guarantees finding the shortest hypothesis
- too slow (generate and test)

First idea

Incremental probabilistic search

- start with template containing one atom of each predicate symbol
- add new atoms randomly with uniform distribution
- if consistent hypothesis found then remove isolated atoms (do not share variables with other atoms)

Features:

- no guarantee of finding the shortest hypothesis
- faster convergence
- ready for tuning via the probability distribution for selecting added atoms

Second idea

Stochastic history-driven tabu search

- if the added atom is successful (increased the number of broken negative examples) then add it again
- if the added atom is not successful then put it to **tabu list** and select another atom (outside the tabu list) randomly
  - tabu list is emptied if all atoms are tabu, or added atom was successful
- stochastic version
  - probability of successful atom is set to a high value
  - probability of atoms returned from the tabu list is set to a low value

Experimental results

- looking for common sub-structure in random graphs (Barabási-Réka model)
- time limit 600 seconds
- 5 runs of stochastic algorithms

IDS		IPS			SHDTS		
time[s]	length	time[s]	length	#unfinished	time[s]	length	#unfinished
1.75	**6**	**0.33**	**6**	-	0.37	**6**	-
13.97	**7**	2.34	**7**	1	**1.85**	**7**	-
13.92	**7**	1.31	**7**	1	**0.62**	**7**	-
0.27	**5**	**0.11**	**5**	1	1.17	7	-
455.43	**8**	334.51	**8**	-	**273.75**	**8**	-
10.93	**7**	1.16	**7**	1	**1.02**	**7**	-
>600	**-**	6.13	**8**	1	**2.11**	8	-
411.60	**7**	28.07	8	-	**0.46**	10	-
11.88	**7**	16.41	**7**	1	67.70	8	-
13.52	**7**	1.41	**7**	1	**0.55**	**7**	-

Experimental results (2)

- looking for common sub-structure in random graphs (Erdös-Rényi model)
- time limit 1200 seconds

	IDS		IPS		SHDTS	
nodes	time[s]	length	time[s]	length	time[s]	length
5	1.41	**6**	**0.74**	**6**	**0.74**	**6**
5	>1200	-	204.11	**9**	**59.65**	**9**
5	>1200	-	**5.23**	**9**	17.7	**9**
5	>1200	-	518.47	10	**448.72**	**10**
6	>1200	-	533.25	**9**	**241.57**	**9**
6	>1200	-	313.25	**9**	**211.78**	**9**
6	>1200	-	426.70	**10**	**366.41**	**10**
6	>1200	-	**181.28**	**9**	215.01	**9**
7	>1200	-	**716.15**	**10**	950.04	**10**
7	27.71	**7**	>1200	-	**3.98**	**7**
7	>1200	-	>1200	-	>1200	-
7	>1200	-	201.28	**10**	**164.61**	**10**

Improving efficiency

How to improve efficiency of existing ILP algorithms?

Main **idea**:

- Problem decomposition (divide-and-conquer)

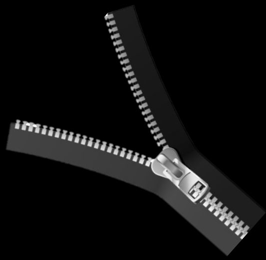
Our **approach** at glance:

- ***Decompose***
  - split examples into several groups
  - find a consistent hypothesis for each group
- ***Merge***
  - combine the hypotheses to obtain a single hypothesis consistent with all examples
- ***Refine***
  - try to shorten further the hypothesis (Ockham's razor)

Decomposition

- How to split the examples?
  - We will need to combine the obtained hypotheses!
- Our proposal
  - each group contains all positive examples and some negative examples (we only split the negative examples)
- Claim
  - union of obtained hypotheses is consistent
    - with all positive examples (subsumes them)
    - with all negative examples (does not subsume them)

Merging



- Can we do more sophisticated merging?
  - leading to a smaller hypothesis?
- Example:
  - $H_1 = \{arc_1(X_1,X_2), arc_1(X_3,X_1), arc_1(X_2,X_4), b_1(X_4)\}$
  - $H_2 = \{arc_2(Y_1,Y_3), arc_2(Y_1,Y_2), b_2(Y_2)\}$

  What if we unify $arc_1(X_2,X_4)$ and $arc_2(Y_1,Y_2)$?
  - $\{arc_1(X_1,X_2), arc_1(X_3,X_1), arc_1(X_2,X_4), b_1(X_4), arc_2(X_2,Y_3)\}$
- Claim
  - After unifying two predicates in hypotheses being merged
    - the final hypothesis is consistent with negative evidence
    - but may be no more consistent with positive evidence!
- In practice
  - we explore all possible pairs for unification
  - If consistency is violated after unification then the unification is excluded

Refinement

- Can we further shorten the hypothesis?
  - we can remove some predicate(s)
- Claim
  - After removing a predicate from the hypothesis
    - the final hypothesis is consistent with positive evidence
    - but may be no more consistent with negative evidence!
- In practice
  - we try to remove the largest subset of predicates
  - after each removal, we try to restore consistency by adding extra unifications of variables (the original template consistency algorithm)
  - if consistency is violated then the predicate is returned back

Experiments (setting)

- So, are we really better?
- Experimental setting
  - implemented in SICStus Prolog 4.1.2
  - 2.0 GHz Intel Xeon with 12 GB RAM
  - problems – looking for a common structure in graphs
    - random graphs generated using Barabási-Reka model (new nodes connected with three random arcs)
    - 20 nodes in graph, hidden structure of 5 nodes
    - 10 positive and 10 negative examples

Experiments (results)

Alg. 1		Aleph		DeMeR (10 subtasks)						
				Decomposition		Merge		Refinement		
t	len	t	len	t	len	t	len	t	len	t_{total}
17.4	**7**	>1200	-	1.7	43	2.0	11	1.1	**7**	**4.8**
436	**8**	>1200	-	7.2	54	6.3	20	30.0	10	**43.5**
460	**8**	>1200	-	54.2	60	39.3	29	30.0	10	**123.5**
508	**8**	>1200	-	23.9	53	10.9	43	30.0	**8**	**72.9**
>1200	-	>1200	-	10.2	51	6.8	15	3.5	**12**	**20.5**
>1200	-	>1200	-	23.4	55	15.4	33	30.0	**14**	**68.8**
>1200	-	>1200	-	90.9	60	31.3	31	30.0	**9**	**152.2**
>1200	-	>1200	-	88.8	61	41.8	36	30.0	**10**	**160.6**
>1200	-	>1200	-	103.1	53	41.3	28	30.0	**9**	**174.4**
>1200	-	>1200	-	99.9	57	48.3	34	30.	[illegible]	[illegible]

Only a few p[...] solved!

No problem solved!

Only a small degradation of solution quality.

Significant reduction of hypothesis size.

Experiments (looking inside)

DeMeR (5 subtasks)							DeMeR (3 subtasks)						
Decomp.		Merge		Refin.			Decomp.		Merge		Refin.		
t	len	t	len	t	len	t_{total}	t	len	t	len	t	len	t_{total}
21.4	28	2.1	7	0.1	7	23.6	21.5	19	1.1	7	0.1	7	22.7
396.8	31	2.6	8	0.2	8	399.6	394.0	21	1.7	8	0.2	8	395.9
170.9	33	7.4	19	30.0	9	208.3	519.8	22	2.1	14	6.8	8	528.7
623.7	33	3.8	27	30.0	8	657.5	892.5	23	1.0	15	25.8	8	919.3
41.2	30	1.7	12	0.3	12	43.2	29.7	18	0.8	12	0.3	12	30.8
124.5	34	5.2	21	30.0	14	159.7	624.7	21	0.7	21	30.0	15	655.4
539.1	34	11.5	22	30.0	9	580.6	>1200	-	-	-	-	-	-
>1200	-	-	-	-	-	-	>1200	-	-	-	-	-	-
786.6	32	5.4	28	30.0	9	822.0	>1200	-	-	-	-	-	-
179.4	32	4.7	31	30.0	11	214.1	147.0	20	2.4	15	30.0	8	179.4

smaller decomposition gives worse results

Summary

- What has been done?
  - checking subsumption using CP
  - finding consistent hypothesis from a template using CP
  - generating templates via search with learning
  - Improving efficiency via problem decomposition
- What is next?
  - a possible extension to noisy data
  - combination with other ILP algorithms