Exploiting the Power of MIP Solvers in MAXSAT

Jessica Davies[1] and Fahiem Bacchus[2]

[1]MIAT, INRA, Toulouse, France
[2]Department of Computer Science, University of Toronto

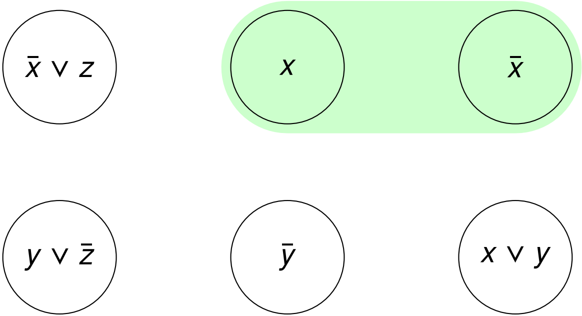
Outline

1. Background
2. The MaxHS Approach
3. Exploiting CPLEX
4. Empirical Results

The MAXSAT Problem

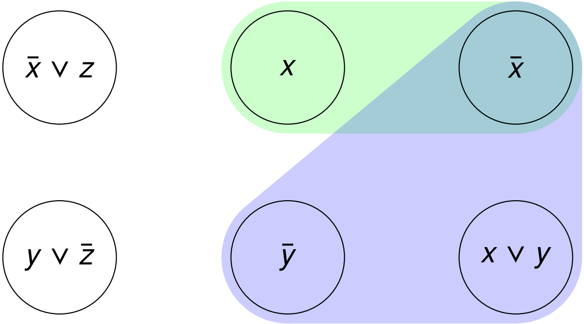
- MAXSAT is an optimization version of SAT
- An instance of the MAXSAT problem is given by a **CNF** formula $\mathcal{F}$ and a **cost** $wt(C) \in \mathbb{N} \cup \{\infty\}$ associated with each clause C
- A truth assignment π has cost equal to the sum of the costs of the clauses it falsifies
- Goal: find an optimal truth assignment, i.e., a truth assignment of minimum cost $mincost(\mathcal{F})$
- Clauses with $wt(C) = \infty$ are **hard**, all others are **soft**

Cores



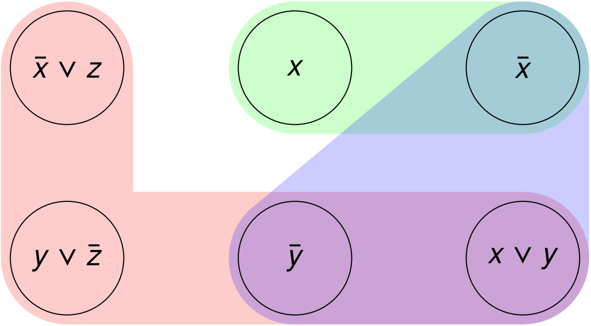
- A **core** is any subset of the soft clauses that is inconsistent with the hard clauses
- This instance $\mathcal{F}$ has 4 cores

Cores



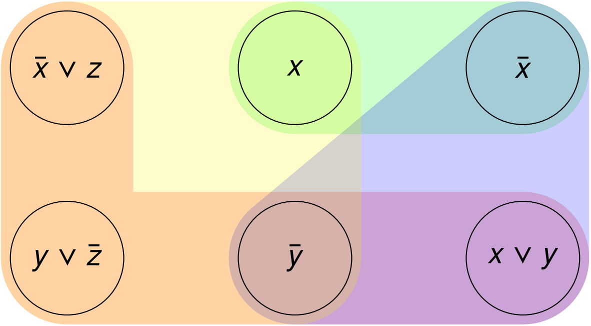
- A **core** is any subset of the soft clauses that is inconsistent with the hard clauses
- This instance $\mathcal{F}$ has 4 cores

Cores



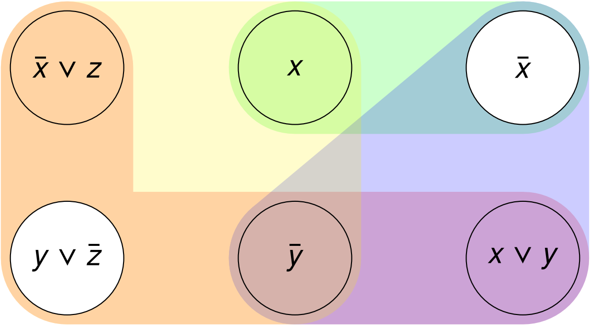
- A **core** is any subset of the soft clauses that is inconsistent with the hard clauses
- This instance $\mathcal{F}$ has 4 cores

Cores



- A **core** is any subset of the soft clauses that is inconsistent with the hard clauses
- This instance $\mathcal{F}$ has 4 cores

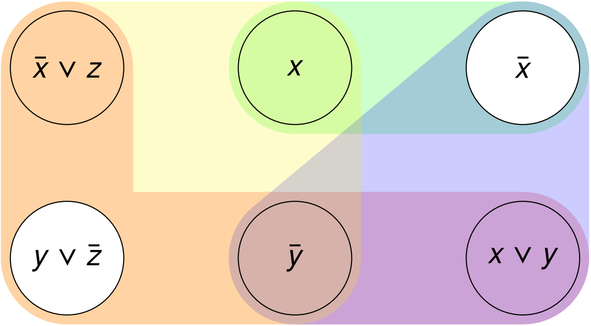
Hitting Sets



$$\pi = \{x, \bar{y}, z\}$$

- The clauses *falsified* by π are a **hitting set** of the cores

MaxHS Theorem



By the theorem, $\pi = \{x, \bar{y}, z\}$ is a solution

Theorem: if π satisfies $\mathcal{F} \setminus hs$ where hs is a minimum cost hitting set of a collection of cores, then π is a solution

Outline

1. Background

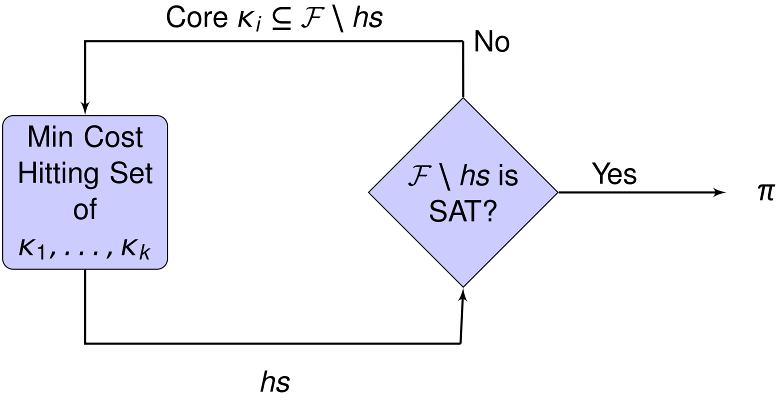
2. The MaxHS Approach

3. Exploiting CPLEX

4. Empirical Results

The MaxHS Approach

- In this paper we extend this existing approach for solving MAXSAT



The SAT Model

- The SAT solver works with a relaxed formula

$$\mathcal{F}^b = hard(\mathcal{F}) \cup \{C_i \vee b_i | C_i \in soft(\mathcal{F})\}$$

- The b_i are the relaxation variables, each appearing in only one clause
- To test if $\mathcal{F} \setminus hs$ is SAT, we use the set of assumptions

$$A_{hs} = \{b_i | C_i \in hs\} \cup \{\neg b_i | C_i \notin hs\}$$

- Applying these assumptions to $\mathcal{F}^b$ produces $\mathcal{F} \setminus hs$

Core Generation

	$\mathcal{F}^b$		
C_1	$\bar{x} \lor z \lor b_1$		
C_2	$x \lor b_2$		
C_3	$\bar{x} \lor b_3$		
C_4	$y \lor \bar{z} \lor b_4$		
C_5	$\bar{y} \lor b_5$		
C_6	$x \lor y \lor b_6$		

Core Generation

	$\mathcal{F}^b$	Known Cores
C_1	$\bar{x} \vee z \vee b_1$	$\kappa_1 = \{C_2, C_3\}$
C_2	$x \vee b_2$	$\kappa_2 = \{C_3, C_5, C_6\}$
C_3	$\bar{x} \vee b_3$	$\kappa_3 = \{C_1, C_4, C_5, C_6\}$
C_4	$y \vee \bar{z} \vee b_4$	
C_5	$\bar{y} \vee b_5$	
C_6	$x \vee y \vee b_6$	

Core Generation

	$\mathcal{F}^b$	Known Cores
C_1	$\bar{x} \lor z \lor b_1$	$\kappa_1 = \{C_2, C_3\}$
C_2	$x \lor b_2$	$\kappa_2 = \{C_3, C_5, C_6\}$
C_3	$\bar{x} \lor b_3$	$\kappa_3 = \{C_1, C_4, C_5, C_6\}$
C_4	$y \lor \bar{z} \lor b_4$	
C_5	$\bar{y} \lor b_5$	Hitting Set
C_6	$x \lor y \lor b_6$	$hs = \{C_3, C_6\}$

Core Generation

	$\mathcal{F}^b$	Known Cores	Assumptions A_{hs}
C_1	$\bar{x} \vee z \vee b_1$	$\kappa_1 = \{C_2, C_3\}$	$\neg b_1$
C_2	$x \vee b_2$	$\kappa_2 = \{C_3, C_5, C_6\}$	$\neg b_2$
C_3	$\bar{x} \vee b_3$	$\kappa_3 = \{C_1, C_4, C_5, C_6\}$	b_3
C_4	$y \vee \bar{z} \vee b_4$		$\neg b_4$
C_5	$\bar{y} \vee b_5$	Hitting Set	$\neg b_5$
C_6	$x \vee y \vee b_6$	$hs = \{C_3, C_6\}$	b_6

Core Generation

	$\mathcal{F}^b\|_{A_{hs}}$	Known Cores	Assumptions A_{hs}
C_1	$\bar{x} \vee z$	$\kappa_1 = \{C_2, C_3\}$	$\neg b_1$
C_2	x	$\kappa_2 = \{C_3, C_5, C_6\}$	$\neg b_2$
		$\kappa_3 = \{C_1, C_4, C_5, C_6\}$	b_3
C_4	$y \vee \bar{z}$		$\neg b_4$
C_5	$\bar{y}$	Hitting Set	$\neg b_5$
		$hs = \{C_3, C_6\}$	b_6

Core Generation

$\mathcal{F}^b\|_{A_{hs}}$		Known Cores	Assumptions A_{hs}
C_1	$\bar{x} \vee z$	$\kappa_1 = \{C_2, C_3\}$	$\neg b_1$
C_2	x	$\kappa_2 = \{C_3, C_5, C_6\}$	$\neg b_2$
		$\kappa_3 = \{C_1, C_4, C_5, C_6\}$	b_3
C_4	$y \vee \bar{z}$		$\neg b_4$
C_5	$\bar{y}$	Hitting Set	$\neg b_5$
		$hs = \{C_3, C_6\}$	b_6

Conflict Clause $b_1 \vee b_2 \vee b_4 \vee b_5$

Core Generation

- The conflict clause $(b_1 \vee b_2 \vee b_4 \vee b_5)$ intuitively means that one of the corresponding clauses must be falsified:
  - $\kappa_4 = \{C_1, C_2, C_4, C_5\}$ is a new core
- The b-variables appear only positively in $\mathcal{F}^b$
  - Positive b-variables in the assumptions only satisfy clauses, and cannot contribute to conflicts
  - $\therefore$ all conflict clauses derived by the SAT solver correspond to cores

Core Generation

- The conflict clause $(b_1 \vee b_2 \vee b_4 \vee b_5)$ intuitively means that one of the corresponding clauses must be falsified:
  - $\kappa_4 = \{C_1, C_2, C_4, C_5\}$ is a new core
- The *b*-variables appear only positively in $\mathcal{F}^b$
  - Positive *b*-variables in the assumptions only satisfy clauses, and cannot contribute to conflicts
  - ∴ all conflict clauses derived by the SAT solver correspond to cores

Hitting Set IP Model

Objective: $\min \Sigma_i b_i wt(C_i)$

Constraints: $\Sigma_{b_i | C_i \in \kappa_j} b_i \geq 1$ for all known cores κ_j

↓

CPLEX

↓

A_{hs}

MaxHS Performance

- The existing MaxHS solver performs well but is not state-of-the-art [Davies and Bacchus, CP-11]
- The time required to solve the hitting set problems dominates
- In this paper we present methods that improve the performance of MaxHS
- These methods involve giving CPLEX more information, in order to
  - reduce the **difficulty** of solving the IP model
  - reduce the **number of times** the IP model is solved

Outline

CPLEX as MAXSAT Solver

	minimaxsat	bincd	cplex	wpm1
Industrial	1637	2251	1779	2152
Crafted	933	534	1019	711
Total	2570	2785	2798	2863

(Number solved out of a total of 3826 non-random instances)

- MAXSAT can be translated to IP using a standard encoding
- CPLEX is actually a very good MAXSAT solver, especially on Crafted instances
- MaxHS uses CPLEX, so can we further exploit CPLEX?

CPLEX as MAXSAT Solver

	minimaxsat	bincd	**cplex**	wpm1
Industrial	1637	2251	1779	2152
Crafted	933	534	**1019**	711
Total	2570	2785	2798	2863

(Number solved out of a total of 3826 non-random instances)

- MAXSAT can be translated to IP using a standard encoding
- CPLEX is actually a very good MAXSAT solver, especially on Crafted instances
- MaxHS uses CPLEX, so can we further exploit CPLEX?

SAT Model with Equivalences

- In the MaxHS approach, setting a relaxation variable b_i to *true* represents falsifying C_i
- However, this relationship is not fully captured by the SAT model
- We modify the SAT model $\mathcal{F}^b$ by adding **equivalence clauses** that enforce $b_i \equiv \bar{C}_i$

$\mathcal{F}^b_{eq} = \mathcal{F}^b$	$\cup$ Equivalence Clauses
$(\neg x \vee z \vee b_1)$	$(\neg b_1 \vee x), (\neg b_1 \vee \neg z)$
$(x \vee b_2)$	$(\neg b_2 \vee \neg x)$
$(\neg x \vee b_3)$	$(\neg b_3 \vee x)$
$(y \vee \neg z \vee b_4)$	$(\neg b_4 \vee \neg y), (\neg b_4 \vee z)$
$(\neg y \vee b_5)$	$(\neg b_5 \vee y)$
$(x \vee y \vee b_6)$	$(\neg b_6 \vee \neg x), (\neg b_6 \vee \neg y)$

SAT Model with Equivalences

- In the MaxHS approach, setting a relaxation variable b_i to *true* represents falsifying C_i
- However, this relationship is not fully captured by the SAT model
- We modify the SAT model $\mathcal{F}^b$ by adding **equivalence clauses** that enforce $b_i \equiv \bar{C}_i$

$\mathcal{F}^b_{eq} = \mathcal{F}^b$	$\cup$ Equivalence Clauses
$(\neg x \vee z \vee b_1)$	$(\neg b_1 \vee x), (\neg b_1 \vee \neg z)$
$(x \vee b_2)$	$(\neg b_2 \vee \neg x)$
$(\neg x \vee b_3)$	$(\neg b_3 \vee x)$
$(y \vee \neg z \vee b_4)$	$(\neg b_4 \vee \neg y), (\neg b_4 \vee z)$
$(\neg y \vee b_5)$	$(\neg b_5 \vee y)$
$(x \vee y \vee b_6)$	$(\neg b_6 \vee \neg x), (\neg b_6 \vee \neg y)$

Non-Core Constraints

- Now, the *b*-variables appear both positively and negatively in the SAT model
- Hence, propagating $b_i = true$ can contribute to a contradiction
- Conflict clauses returned by the SAT solver can now contain both positive **and negative** *b*-literals

Non-Core Constraints

- E.g., $(b_1 \vee b_2 \vee \neg b_3)$
  - This clauses says that either C_1 or C_2 must be falsified OR C_3 must be truthified, in any MAXSAT solution
- These clauses no longer represents cores, but can still be added to the IP model to guide CPLEX
- CPLEX no longer solves a pure hitting set problem

Non-Core Constraints

- The non-core constraints also capture the previously defined **Realizability** condition on the hitting sets [Davies and Bacchus, CP-11]
- Mutual falsifiability:
  - Clauses with clashing literals can not be falsified at the same time

    e.g., $(\neg x \vee z \vee b_1)$ and $(x \vee y \vee b_6)$
  - The equivalence clauses $(\neg b_1 \vee x)$, $(\neg b_6 \vee \neg x)$ allow us to derive the non-core constraint $(\neg b_1 \vee \neg b_6)$ to enforce this condition
- Compatibility with *hard*$(\mathcal{F})$: automatically checks that falsifying the clauses in the hitting set is compatible with satisfying the hard clauses

Seeding

- In MaxHS, CPLEX starts with no constraints, and then receives cores only one at time
- We can give CPLEX more information to begin with, using the technique of **seeding**
- Seeding involves deriving initial constraints for CPLEX
- We propose 3 different seeding techniques

Eq-Seeding

- Many MAXSAT instances contain unit soft clauses
- $C_i = (x)$ means $b_i \equiv \neg x$
- Given these equivalencies, check if any clauses of $\mathcal{F}^b$ can be rewritten as clauses over only b-literals
- E.g., given

$$\begin{array}{lcrcl} C_2 = (x \vee b_2) & \rightarrow & \neg b_2 & \equiv & x \\ C_5 = (\neg y \vee b_5) & \rightarrow & b_5 & \equiv & y \end{array}$$

from $(x \vee y \vee b_6)$ we obtain

$$(\neg b_2 \vee b_5 \vee b_6)$$

- Such clauses can be added to CPLEX initially

Implication Seeding

- Given the equivalence theory $\mathcal{F}_{eq}^{b}$ we can unit propagate each b-literal (probing)
- For each b_i (and $\neg b_i$) we collect the set of b-literals it implies, $\{b_i^1, \ldots, b_i^k\}$
- This represents k binary clauses
- We can add a **single** linear constraint to CPLEX:

$$-k * b_i + b_i^1 + \cdots + b_i^k \geq 0$$

Reverse Implication Seeding

- During Implication Seeding, when we unit propagate each b-literal we also find implied **original** literals

  e.g., $b_1 \rightarrow x$, $b_2 \rightarrow y$
- Unlike in Eq-Seeding, these relationships are not equivalences
- However, we can replace x and y in a clause $(\neg x, \neg y) \in \mathcal{F}^b$ to obtain a b-variable clause $(\neg b_1, \neg b_2)$

Other Methods

- We improve the information given to CPLEX via two additional methods
    1. Strengthen the constraints: reduce the conflict clauses to be **minimal** using a greedy MUS algorithm
    2. More initial constraints for CPLEX: greedily compute a set of **disjoint** cores
- See the paper for more details

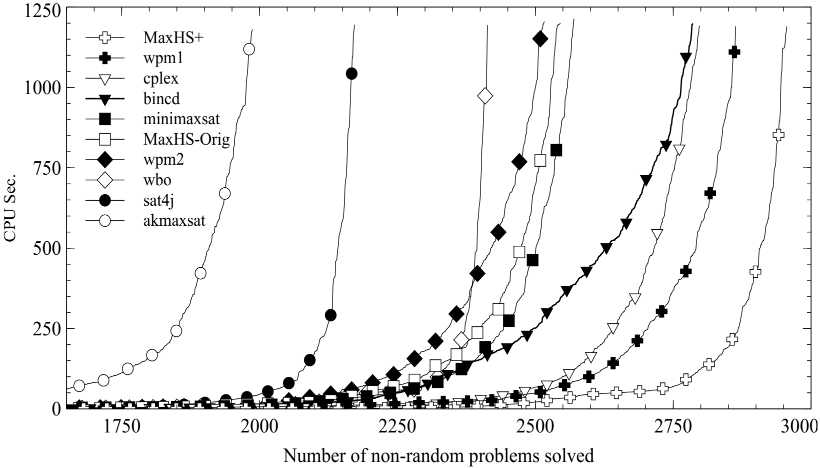
Outline

1. Background
2. The MaxHS Approach
3. Exploiting CPLEX
4. Empirical Results

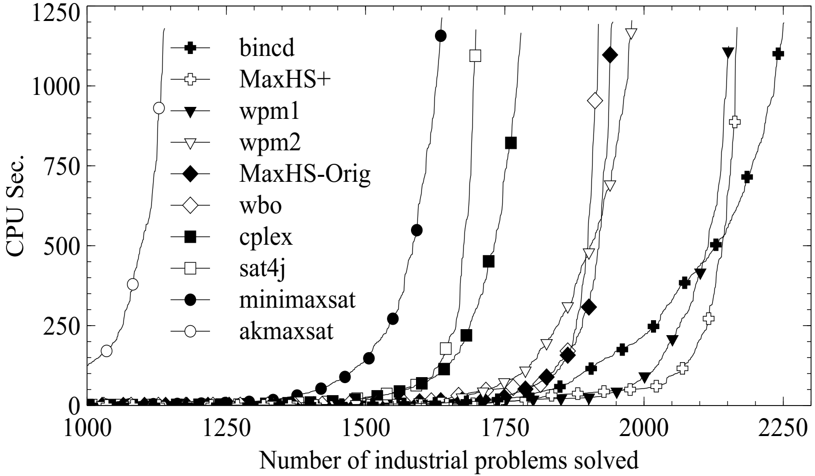
Experimental Setup

- All crafted and industrial instances from the previous seven MAXSAT Evaluations, with duplicates removed
- We removed 17 families that we felt are better classified as random. This leaves 3826 instances out of 4502.
- 2.1 GHz CPUs, 2.5 GB, 1200 sec. timeout
- Note that the previous two MAXSAT Evaluations were limited to 0.5GB machines

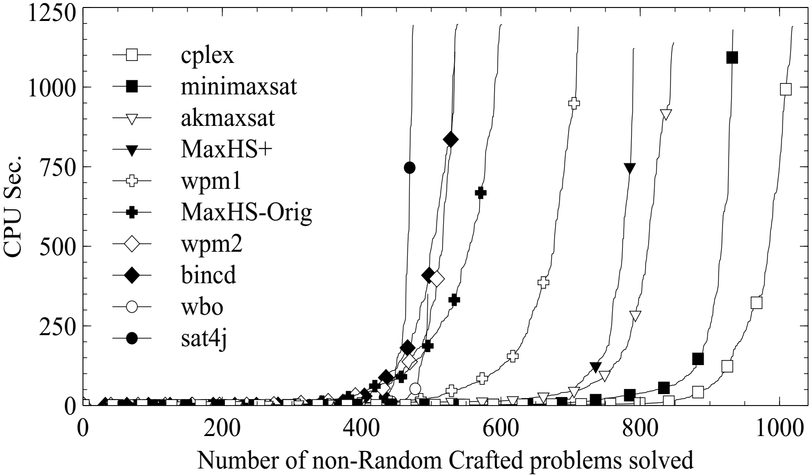
Overall Results



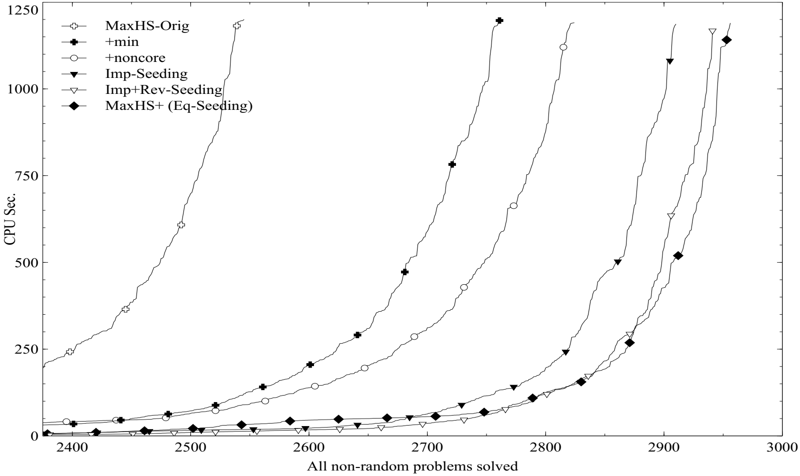
Industrial Instances



Crafted Instances



Comparison of Our New Methods



Divergence of Performance

	MaxHS+	wpm1	cplex	bincd	minimaxsat
MaxHS+		**399**	**439**	**325**	**584**
wpm1	**292**		508	324	756
cplex	**303**	449		546	529
bincd	**143**	232	502		566
minimaxsat	**187**	457	289	421	

- Entry (i, j) in the table shows the number of problems solved by i in 600 sec. that j failed to solve within twice as much time
- Each of the top 5 solvers outperforms the others on a non-trivial number of instances
- Indicates that each of these solvers embeds useful ideas

Conclusion

- The basic approach of MaxHS involves splitting the problem between two solvers, a SAT solver and a MIPS solver
- The approach is very flexible and in this paper we have exploited some of this flexibility to split the task between the two solvers in a different way
- Using approximations to avoid solving the IP model to optimality yields even better performance [to appear, CP-2013]