Cumulative Scheduling and Pseudo-Boolean Constraints

Deduction at Scale

Albert Oliveras

(+ Ignasi Abío, Roberto Asín, Robert Nieuwenhuis, Enric Rodríguez, Javier Larrosa, ...)

Barcelogic Research Group, Tech. Univ. Catalonia, Barcelona

Motivation

Assume a company needs to complete the following 15 jobs:

# Job	Transport	Unload	Process	Pack
1	1 truck	10 unloaders	5 technicians	3 packers
2	1 truck	5 unloaders	3 technicians, 1 supervisor	2 packers
3	1 car	2 unloaders	1 technicians , 1 supervisor	1 packer
...				
15	1 truck	5 unloaders	3 technicians	2 packers

- Tasks cannot be interrupted (non-preemptive)
- Task precedences: transport $\rightarrow$ unload $\rightsquigarrow$ process $\rightsquigarrow$ pack
- Each task has a certain duration
- Resource availability:
  2 trucks, 4 cars, 30 unloaders, 13 techn., 3 superv., 8 packers

GOAL: complete all jobs as early as possible

Overview of the talk

- Cumulative scheduling
- SAT encodings for PB constraints
  - SAT encodings vs SMT
  - Existing encodings
- Standard BDD-based encoding
- New results for BDD-based encodings
  - CNFs for monotonic functions
  - Probably BDDs for some PB constraints are non-poly
  - Avoiding non-polynomial BDDs
- Conclusions and Future Work

Problem statement

INPUT DATA:

- For each resource $r \in \mathcal{R}$, an integer $avail(r)$
- For each task $t \in \mathcal{T}$:
  - $d(t)$: duration of the task
  - $est(t)$: earliest starting time
  - $lct(t)$: latest completion time
  - List of consumptions $\{(r, units(t,r)), \ldots, \}$
- List of precedences $t_1 \rightsquigarrow t_2$, and immediate ones $t_1 \rightarrow t_2$

GOAL: find a valid schedule that minimizes makespan
(i.e. minimizes the maximum completion time)

Solving method

Best approach: encoding to SAT + PB-constraints ([StuckeyCP09]).

- Consider **propositional** variables:

$$s_{t,i} \equiv start\_time(t) \leq i$$

and add clauses:

- $s_{t,0} \rightarrow s_{t,1}$ $\quad s_{t,1} \rightarrow s_{t,2}$ $\quad \ldots$ **[integer consistency]**
- $\neg s_{t,est(t)-1}$ $\quad s_{t,lct(t)-d(t)}$ **[satisfy *est* and *lct*]**
- $s_{t_2,0} \rightarrow s_{t_1,0}$ $\quad s_{t_2,1} \rightarrow s_{t_1,1}$ $\quad \ldots$ **[t_1 precedes t_2]**

Solving method

Best approach: encoding to SAT + PB-constraints ([StuckeyCP09]).

- Consider **propositional** variables:

  $$s_{t,i} \equiv start\_time(t) \leq i$$

  and add clauses:

  - $s_{t,0} \rightarrow s_{t,1}$ $\quad s_{t,1} \rightarrow s_{t,2}$ $\quad \ldots$ **[integer consistency]**
  - $\neg s_{t,est(t)-1}$ $\quad s_{t,lct(t)-d(t)}$ **[satisfy *est* and *lct*]**
  - $s_{t_2,0} \rightarrow s_{t_1,0}$ $\quad s_{t_2,1} \rightarrow s_{t_1,1}$ $\quad \ldots$ **[t_1 precedes t_2]**

- Consider **propositional** variables:

  $$a_{t,i} \equiv \text{task } t \text{ is active at time } i$$

  and define them with:

  - $a_{t,i} \quad \leftrightarrow \quad start\_time(t) \leq i \quad \wedge \quad start\_time(t) + d(t) \geq i + 1$

Solving method

Best approach: encoding to SAT + PB-constraints ([StuckeyCP09]).

- Consider **propositional** variables:

  $$s_{t,i} \equiv start\_time(t) \leq i$$

  and add clauses:
  - $s_{t,0} \rightarrow s_{t,1}$ $\quad s_{t,1} \rightarrow s_{t,2}$ $\quad \ldots$ **[integer consistency]**
  - $\neg s_{t,est(t)-1}$ $\quad s_{t,lct(t)-d(t)}$ **[satisfy *est* and *lct*]**
  - $s_{t_2,0} \rightarrow s_{t_1,0}$ $\quad s_{t_2,1} \rightarrow s_{t_1,1}$ $\quad \ldots$ **[t_1 precedes t_2]**
- Consider **propositional** variables:

  $$a_{t,i} \equiv \text{task } t \text{ is active at time } i$$

  and define them with:
  - $a_{t,i} \quad \leftrightarrow \quad start\_time(t) \leq i \quad \wedge \quad start\_time(t) + d(t) \geq i + 1$
- Add **PB-constraints** of the form:

  $$\sum_{t \text{ s.t. } t \text{ uses } r} units(t,r) * a_{t,i} \leq avail(r), \qquad \text{for each } r \text{ and time } i$$

Solving method

Best approach: encoding to SAT + PB-constraints ([StuckeyCP09]).

- Consider **propositional** variables:

  $$s_{t,i} \equiv start\_time(t) \leq i$$

  and add clauses:
  - $s_{t,0} \rightarrow s_{t,1}$ $\quad s_{t,1} \rightarrow s_{t,2}$ $\quad \ldots$ **[integer consistency]**
  - $\neg s_{t,est(t)-1}$ $\quad s_{t,lct(t)-d(t)}$ **[satisfy est and lct]**
  - $s_{t_2,0} \rightarrow s_{t_1,0}$ $\quad s_{t_2,1} \rightarrow s_{t_1,1}$ $\quad \ldots$ **[t_1 precedes t_2]**
- Consider **propositional** variables:

  $$a_{t,i} \equiv \text{task } t \text{ is active at time } i$$

  and define them with:
  - $a_{t,i} \quad \leftrightarrow \quad start\_time(t) \leq i \quad \wedge \quad start\_time(t) + d(t) \geq i + 1$
- Add **PB-constraints** of the form:

  $$2x_1 + 7x_2 + 5x_3 + 10x_4 \leq 12$$

Overview of the talk

- Cumulative scheduling
- SAT encodings for PB constraints
  - SAT encodings vs SMT
  - Existing encodings
- Standard BDD-based encoding
- New results for BDD-based encodings
  - CNFs for monotonic functions
  - Probably BDDs for some PB constraints are non-poly
  - Avoiding non-polynomial BDDs
- Conclusions and Future Work

SAT Encoding for PB-Constraints

- SAT + PB-constraints problems solvable with an SMT system
- But ... recent experience with SAT + cardinality constraints (PB with all coeffs. 1):

	Speed-up factor if encoding					Slow-down factor if encoding				
Suite	**TO**	**4**	**2**	**1.5**	**TOT.**	**1.5**	**2**	**4**	**TO**	**TOT.**
MSU4	168	54	14	7	*243*	7	24	215	12	*258*
DES	321	53	20	16	*410*	11	4	0	0	*15*

MSU4: unsat-based max-SAT solving

DES: discrete event system diagnosis problems

- How to decide whether SAT or SMT is the best choice?
- Why can SAT **sometimes** beat SMT?

How could SAT beat SMT?

SMT solvers can generate unsat proofs, which come in two parts:

- A resolution refutation from:
  - the clauses of the input CNF
  - the generated explanations (clauses)
- For each explanation clause, an independent proof in (its) T.

So, after all, SMT generates a SAT encoding, but lazily.

SMT solver runtime $\geq$ size of smallest resolution proof.

How could SAT beat SMT? (2)

In "artificial-like" problems:

- SMT's lazy SAT encoding could end up being a full one
- And... this full encoding could be a rather naive one.

Example: T = cardinality constraints. T-solver is just a counter.

Unsat instance: $x_1 + \ldots + x_n \leq k$ and $x_1 + \ldots + x_n > k$

Refutation requires all $\binom{n}{k+1}$ explanations like, e.g.,

$$\overline{x_1} \vee \overline{x_2} \vee \ldots \vee \overline{x_{k+1}}$$

Here a good SAT encoding with auxiliary vars works better.

Splitting on aux vars can give expon. speedup: Extended Resol.

But... some constraints admit no P-size domain-consistent SAT encoding, e.g., alldiff [BessiereEtal'09].

PB-constraints SAT Encodings

Desirable properties of a SAT Encoding:

- Polynomial size w.r.t. the number of vars of the constraint
- Consistent: take $2x_1 + 4x_2 + 2x_3 \leq 5$ and $I = \{x_1, x_2\}$.
  Unit propagation should detect a false clause.
- Arc-consistent: take $2x_1 + 3x_2 + 7x_3 \leq 5$ and $I = \{x_1\}$.
  Unit propagation should propagate $\overline{x_3}$

Existing encodings:

Encoding	**Size**	**Consistent**	**Arc-consistent**
Tree-like	Exponential	yes	yes
BDD	Not known	yes	yes
Adders	$O(\sum \log a_i)$	no	no
Sorting	$O((\sum \log a_i) \log^2 \sum \log a_i)$	yes	no
WatchDog	$O(n^2 \log n \log a_{max})$	yes	yes

Why yet another encoding?

- Existing arc-consistent polyn. encoding depends on coeffs.
  - Take constraints:

    $$3x_1 + 2x_2 + 4x_3 \leq 5$$
    $$30001x_1 + 19999x_2 + 39998x_3 \leq 50007$$

  - They both represent the same Boolean function:

    $$\neg(x_1 \wedge x_3) \quad \wedge \quad \neg(x_2 \wedge x_3)$$

  - 2nd constr. would have an unnecessarily large encoding

- A BDD-based encoding would solve this problem

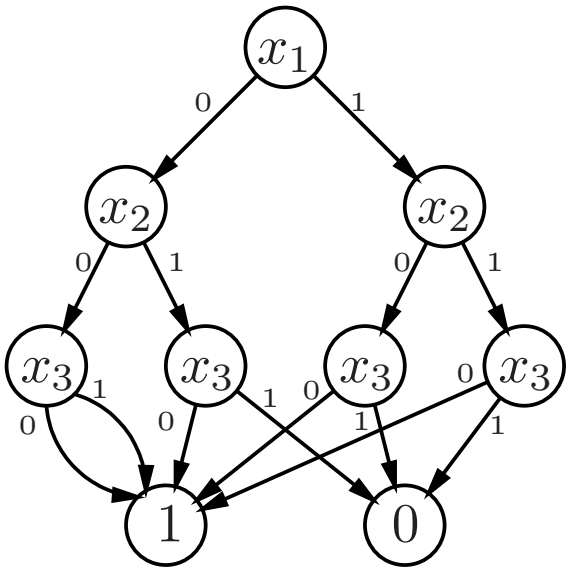
- Question: what is the **BDD size**?

Overview of the talk

- Cumulative scheduling
- SAT encodings for PB constraints
  - SAT encodings vs SMT
  - Existing encodings
- Standard BDD-based encoding
- New results for BDD-based encodings
  - CNFs for monotonic functions
  - Probably BDDs for some PB constraints are non-poly
  - Avoiding non-polynomial BDDs
- Conclusions and Future Work

BDD construction process

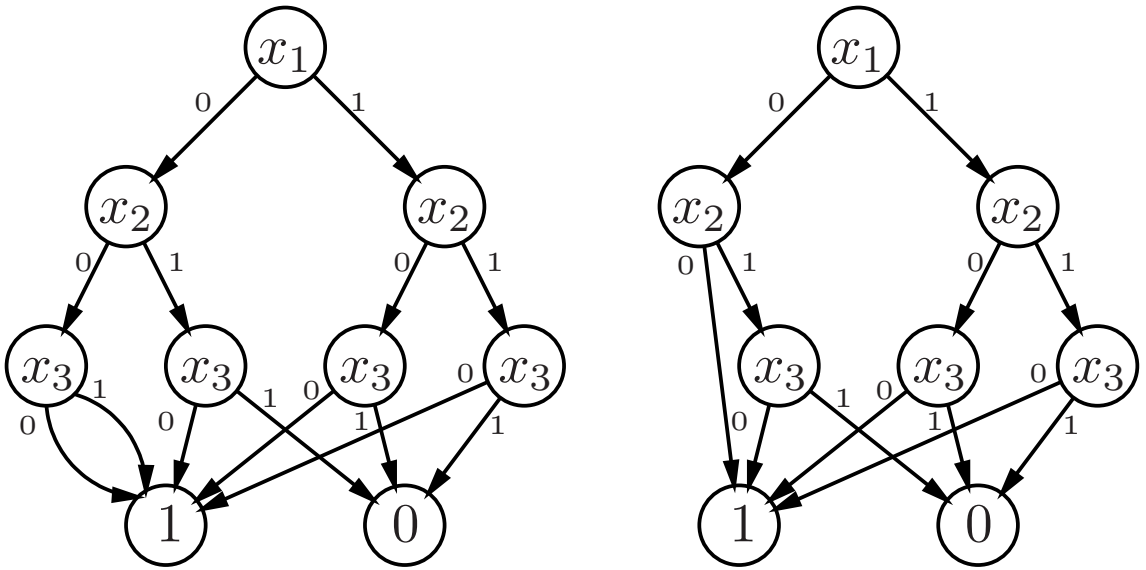
Take constraint $2x_1 + 3x_2 + 5x_3 \leq 6$ a construct the diagram:



Ordering $x_1 < x_2 < x_3$ is previously chosen

BDD construction process

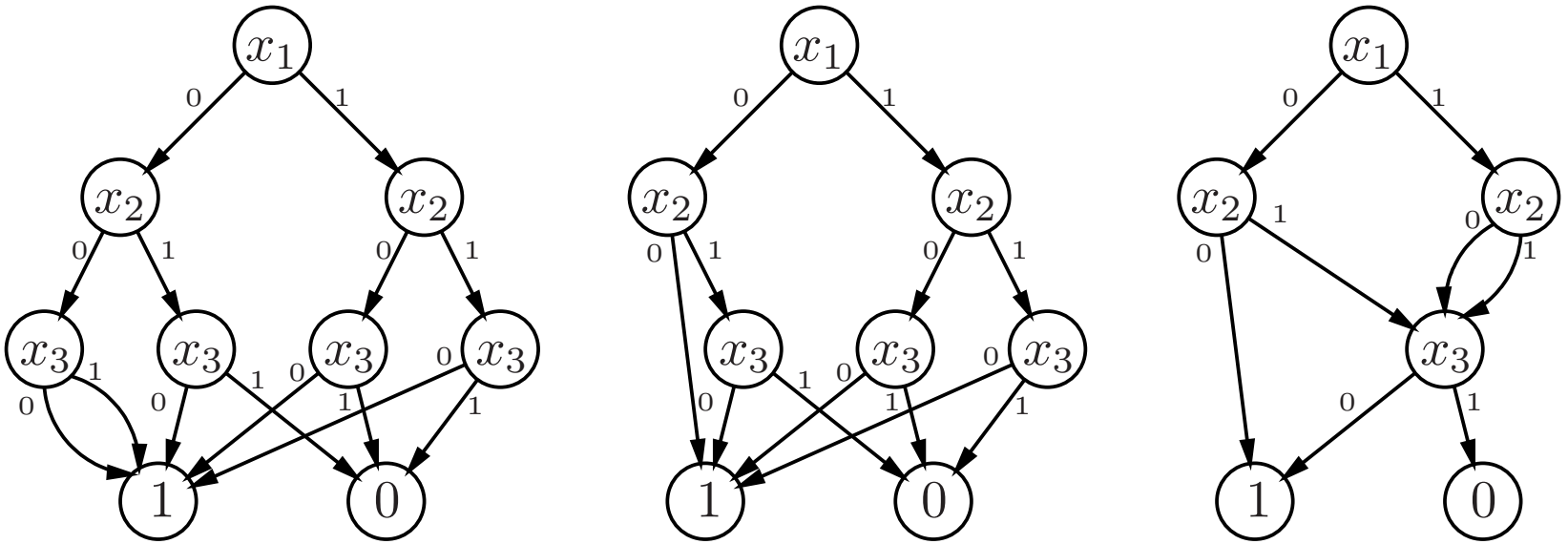
Take constraint $2x_1 + 3x_2 + 5x_3 \leq 6$ a construct the diagram:



Nodes with identical children are removed

BDD construction process

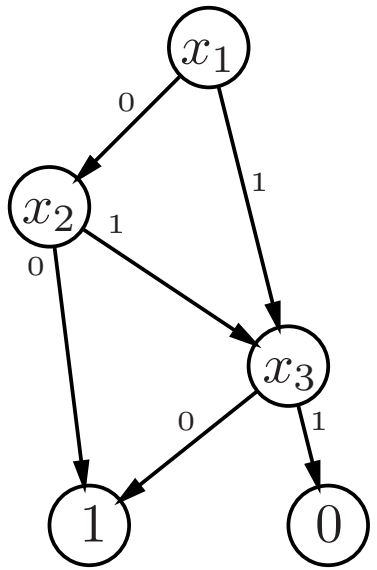
Take constraint $2x_1 + 3x_2 + 5x_3 \leq 6$ a construct the diagram:



Isomorphic subtrees are merged

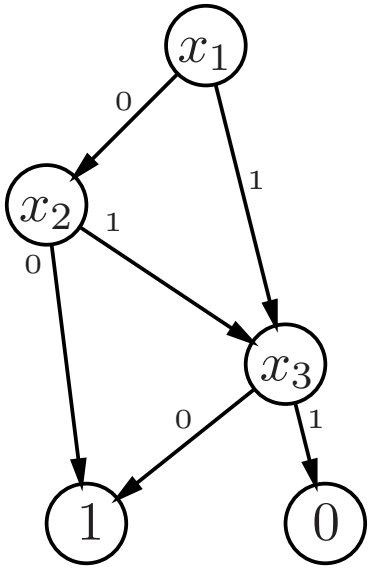
BDD construction process

Take constraint $2x_1 + 3x_2 + 5x_3 \leq 6$ a construct the diagram:



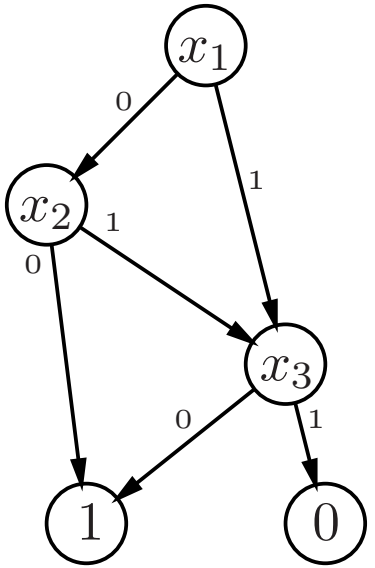
Canonical BDD for the constraint and ordering $x_1 < x_2 < x_3$

CNF conversion of BDDs



- Add auxiliary (output) variable for each node: o_1, o_2 and o_3

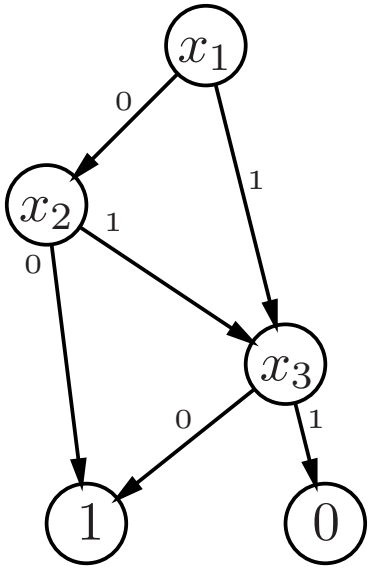
CNF conversion of BDDs



- Add auxiliary (output) variable for each node: o_1, o_2 and o_3
- Specify behaviour of each node. For example:

$$\overline{x_1} \wedge \overline{o_2} \rightarrow \overline{o_1} \qquad x_1 \wedge \overline{o_3} \rightarrow \overline{o_1} \qquad \overline{o_2} \wedge \overline{o_3} \rightarrow \overline{o_1}$$
$$\overline{x_1} \wedge o_2 \rightarrow o_1 \qquad x_1 \wedge o_3 \rightarrow o_1 \qquad o_2 \wedge o_3 \rightarrow o_1$$

CNF conversion of BDDs



- Add auxiliary (output) variable for each node: o_1, o_2 and o_3
- Specify behaviour of each node. For example:

$$\overline{x_1} \wedge \overline{o_2} \rightarrow \overline{o_1} \qquad x_1 \wedge \overline{o_3} \rightarrow \overline{o_1} \qquad \overline{o_2} \wedge \overline{o_3} \rightarrow \overline{o_1}$$
$$\overline{x_1} \wedge o_2 \rightarrow o_1 \qquad x_1 \wedge o_3 \rightarrow o_1 \qquad o_2 \wedge o_3 \rightarrow o_1$$

- Add unit clause o_1

Overview of the talk

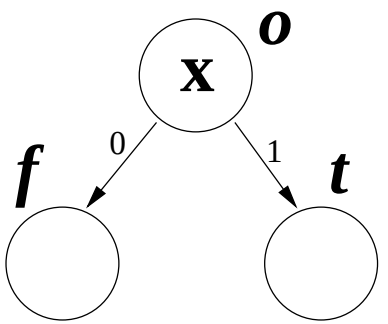
- Cumulative scheduling
- SAT encodings for PB constraints
  - SAT encodings vs SMT
  - Existing encodings
- Standard BDD-based encoding
- New results for BDD encodings
  - CNFs for monotonic functions
  - Probably BDDs for some PB constraints are non-poly
  - Avoiding non-polynomial BDDs
- Conclusions and Future Work

CNF for BDDs of Monotonic Functions

- PB-constraints are monotonic in the following sense:
  if $M[x = 1]$ is a model, so is $M[x = 0]$.

  Consequence: only negated vars can be propagated

- We can exploit monotonicity and use only 2 clauses per node:



$$\overline{f} \rightarrow \overline{o} \qquad\qquad \overline{t} \wedge x \rightarrow \overline{o}$$

plus a unit clause with the root variable

- Encoding is arc-consistent for any monotonic function

Exponential BDD for a given ordering

$$
\begin{array}{rcl}
 & & \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{2n} \\
a_1 & = & 0\ 0\ 0\ \ 0\ 0\ \cdots\ 0\ \mathbf{1} \qquad = \quad 1 \\
a_2 & = & 0\ 0\ 0\ \ 0\ 0\ \cdots\ \mathbf{1}\ 0 \qquad = \quad 2 \\
\cdots & & \qquad\qquad\qquad\ \cdots \\
a_{2n-1} & = & 0\ 0\ 0\ \ 0\ \mathbf{1}\ \cdots\ 0\ 0 \\
a_{2n} & = & 0\ 0\ 0\ \ \mathbf{1}\ 0\ \cdots\ 0\ 0 \qquad = \quad 2^{2n-1} \\
 & & \\
a_{2n+1} & = & \mathbf{1}\ 0\ 0\ \ 0\ 0\ \cdots\ 0\ \mathbf{1} \\
a_{2n+2} & = & \mathbf{1}\ 0\ 0\ \ 0\ 0\ \cdots\ \mathbf{1}\ 0 \\
\cdots & & \qquad\qquad\qquad\ \cdots \\
a_{4n-1} & = & \mathbf{1}\ 0\ 0\ \ 0\ \mathbf{1}\ \cdots\ 0\ 0 \\
a_{4n} & = & \mathbf{1}\ 0\ 0\ \ \mathbf{1}\ 0\ \cdots\ 0\ 0 \\
 & & \\
K & = & \underbrace{d_m\ \ldots\ d_0}_{n \text{ in binary}}\ 0\ 0\ \ \mathbf{1}\ \mathbf{1}\ \cdots\ \mathbf{1}\ \mathbf{1}
\end{array}
$$

FIRST FACT: any selection of n coeffs. from the first $2n$ can be completed using the second half to add exactly K

Exponential BDD for a given ordering

						$2n$						
a_1	$=$		0	0	0	0	0	$\cdots$	0	**1**	$=$	1
a_2	$=$		0	0	0	0	0	$\cdots$	**1**	0	$=$	2
$\cdots$								$\cdots$				
a_{2n-1}	$=$		0	0	0	0	**1**	$\cdots$	0	0		
a_{2n}	$=$		0	0	0	**1**	0	$\cdots$	0	0	$=$	2^{2n-1}
a_{2n+1}	$=$		**1**	0	0	0	0	$\cdots$	0	**1**		
a_{2n+2}	$=$		**1**	0	0	0	0	$\cdots$	**1**	0		
$\cdots$								$\cdots$				
a_{4n-1}	$=$		**1**	0	0	0	**1**	$\cdots$	0	0		
a_{4n}	$=$		**1**	0	0	**1**	0	$\cdots$	0	0		
K	$=$	$\underbrace{d_m \ldots d_0}_{n \text{ in binary}}$	0	0	**1**	**1**	$\cdots$	**1**	**1**			

SECOND FACT: any two different selections of n coefficients form the first $2n$ have different sums

Exponential BDD for a given ordering

					$2n$						
a_1	$=$		0	0	0	0	0	$\cdots$	0	**1**	$=$ 1
a_2	$=$		0	0	0	0	0	$\cdots$	**1**	0	$=$ 2
$\cdots$								$\cdots$			
a_{2n-1}	$=$		0	0	0	0	**1**	$\cdots$	0	0	
a_{2n}	$=$		0	0	0	**1**	0	$\cdots$	0	0	$=$ 2^{2n-1}
a_{2n+1}	$=$		**1**	0	0	0	0	$\cdots$	0	**1**	
a_{2n+2}	$=$		**1**	0	0	0	0	$\cdots$	**1**	0	
$\cdots$								$\cdots$			
a_{4n-1}	$=$		**1**	0	0	0	**1**	$\cdots$	0	0	
a_{4n}	$=$		**1**	0	0	**1**	0	$\cdots$	0	0	
K	$=$	$\underbrace{d_m \ldots d_0}_{n \text{ in binary}}$	0	0	**1**	**1**	**1**	$\cdots$	**1**	**1**	

PROOF: consider assignm. I_1 and I_2 to the first $2n$ vars. s.t.

- They assign exactly n variables to true
- $\text{sum}(I_1) < \text{sum}(I_2)$

Exponential BDD for a given ordering

$$
\begin{array}{rcccccccccccc}
 & & & & & & \overbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{2n} & & & & & & \\
a_1 & = & & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & = & 1 \\
a_2 & = & & 0 & 0 & 0 & 0 & 0 & \cdots & \mathbf{1} & 0 & = & 2 \\
\cdots & & & & & & & & \cdots & & & & \\
a_{2n-1} & = & & 0 & 0 & 0 & 0 & \mathbf{1} & \cdots & 0 & 0 & & \\
a_{2n} & = & & 0 & 0 & 0 & \mathbf{1} & 0 & \cdots & 0 & 0 & = & 2^{2n-1} \\
 & & & & & & & & & & & & \\
a_{2n+1} & = & & \mathbf{1} & 0 & 0 & 0 & 0 & \cdots & 0 & \mathbf{1} & & \\
a_{2n+2} & = & & \mathbf{1} & 0 & 0 & 0 & 0 & \cdots & \mathbf{1} & 0 & & \\
\cdots & & & & & & & & \cdots & & & & \\
a_{4n-1} & = & & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & \cdots & 0 & 0 & & \\
a_{4n} & = & & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & \cdots & 0 & 0 & & \\
 & & & & & & & & & & & & \\
K & = & \underbrace{d_m \ldots d_0}_{n \text{ in binary}} & 0 & 0 & \mathbf{1} & \mathbf{1} & \cdots & \mathbf{1} & \mathbf{1} & & &
\end{array}
$$

There exists an assignment A to the last $2n$ vars such that $\text{sum}(I_1 \cup A) = K$ and hence $\text{sum}(I_2 \cup A) > K$. Hence I_1 and I_2 lead to different BDD nodes $\Rightarrow$ at least $\binom{2n}{n}$ nodes

Interval of a PB-constraint

We need an ingredient to prove the main result about BDD sizes.
Consider again $2x_1 + 3x_2 + 5x_3 \leq 6$.

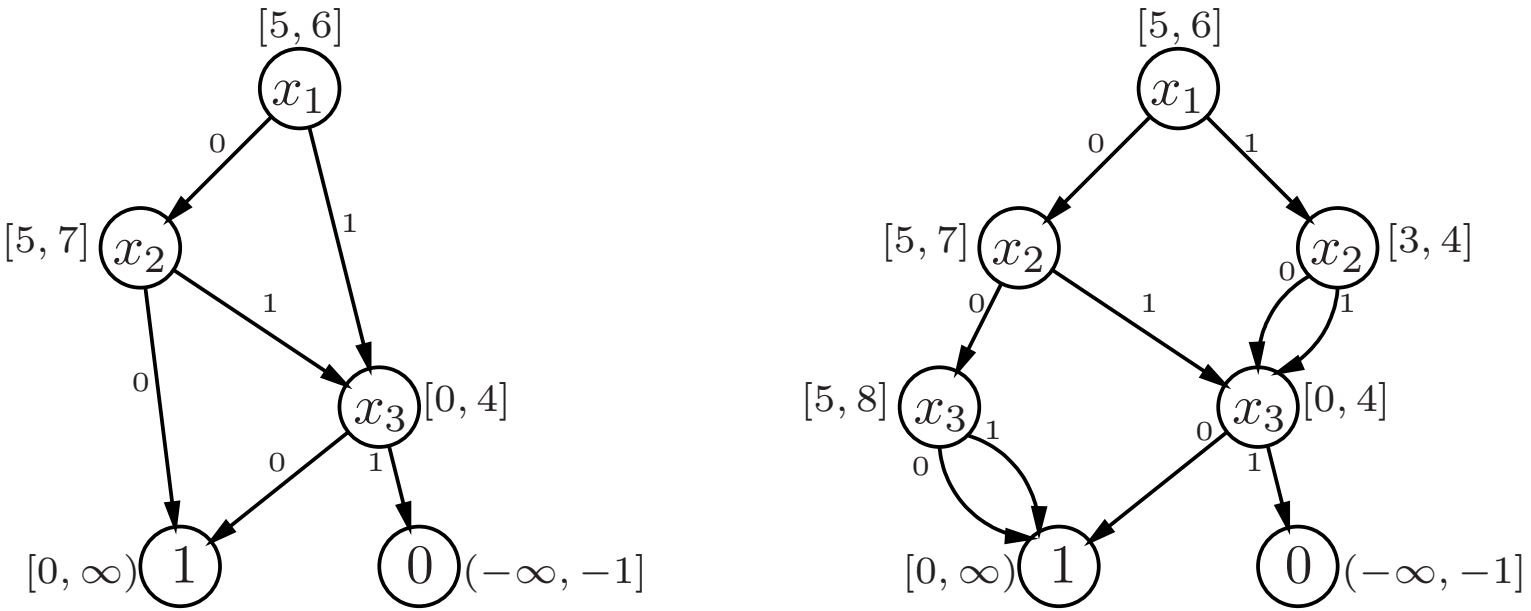
- First observation:
  - Note that no combination of coefficients can add 6
  - Hence, constraint is equivalent to $2x_1 + 3x_2 + 5x_3 \leq 5$
  - In this case, we cannot repeat this process
- Second observation:
  - Note that there is a combination that adds 7
  - Hence, constraint is **not** equivalent to $2x_1 + 3x_2 + 5x_3 \leq 7$

CONCLUSION: constraint is equivalent to $2x_1 + 3x_2 + 5x_3 \leq [5, 6]$.

COMPLEXITY: solve subset sum problems (NP-complete)
However, given the BDD, this can be easily computed.

Interval of a PB-constraint

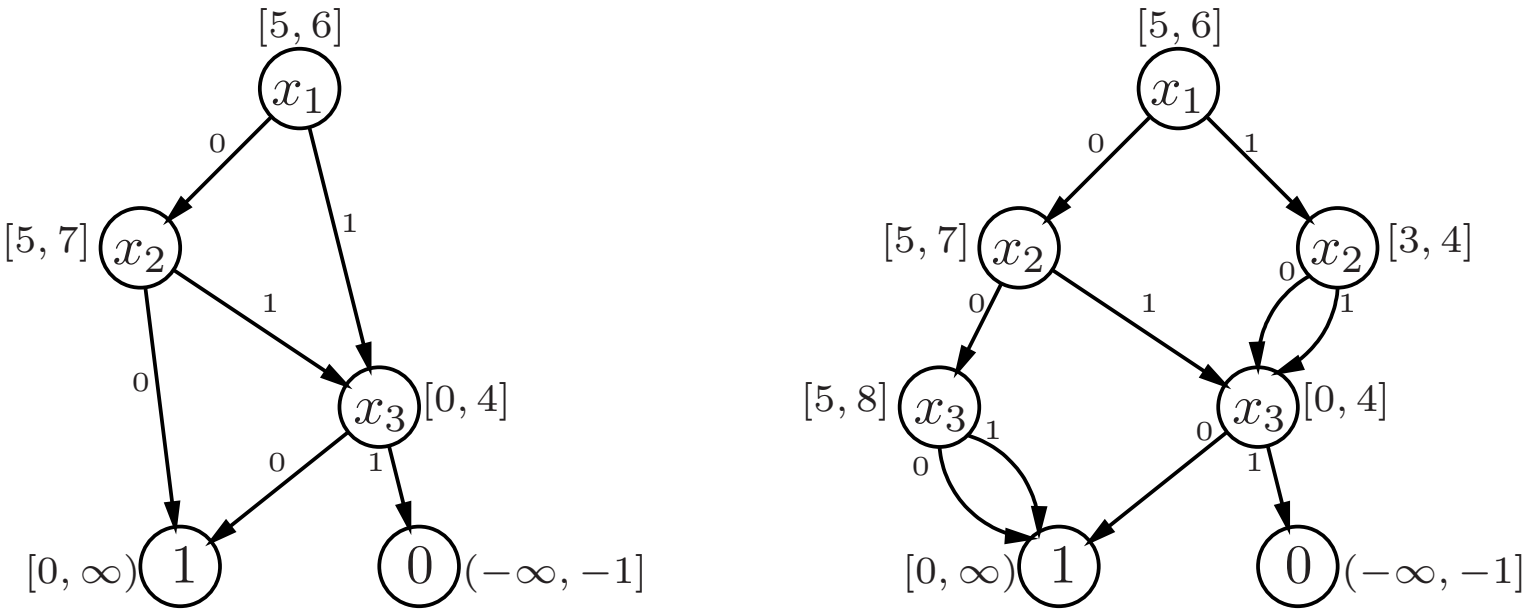
Take BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$ and a modified version:



We will compute intervals bottom up on the modified version
(same could be done on the original one)

Interval of a PB-constraint

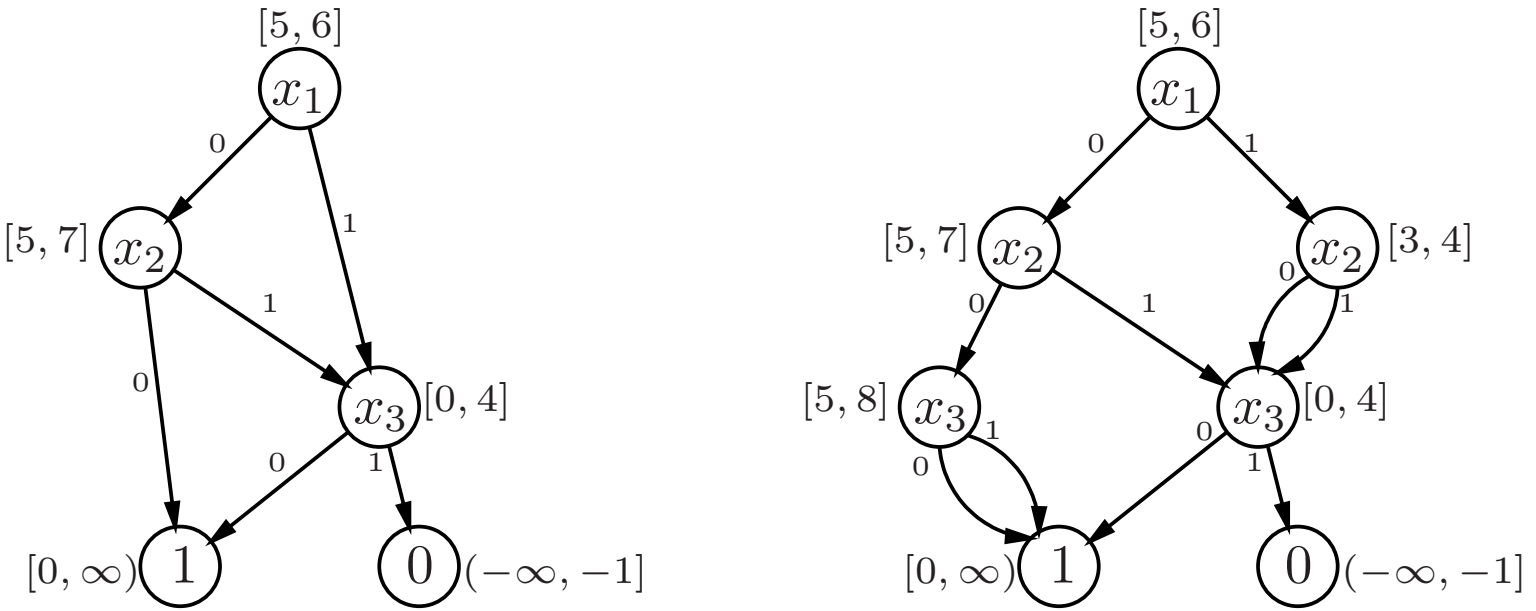
Take BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$ and a modified version:



- True node is equivalent to $0 \leq [0, \infty)$
- False node is equivalent to $0 \leq (-\infty, -1]$

Interval of a PB-constraint

Take BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$ and a modified version:

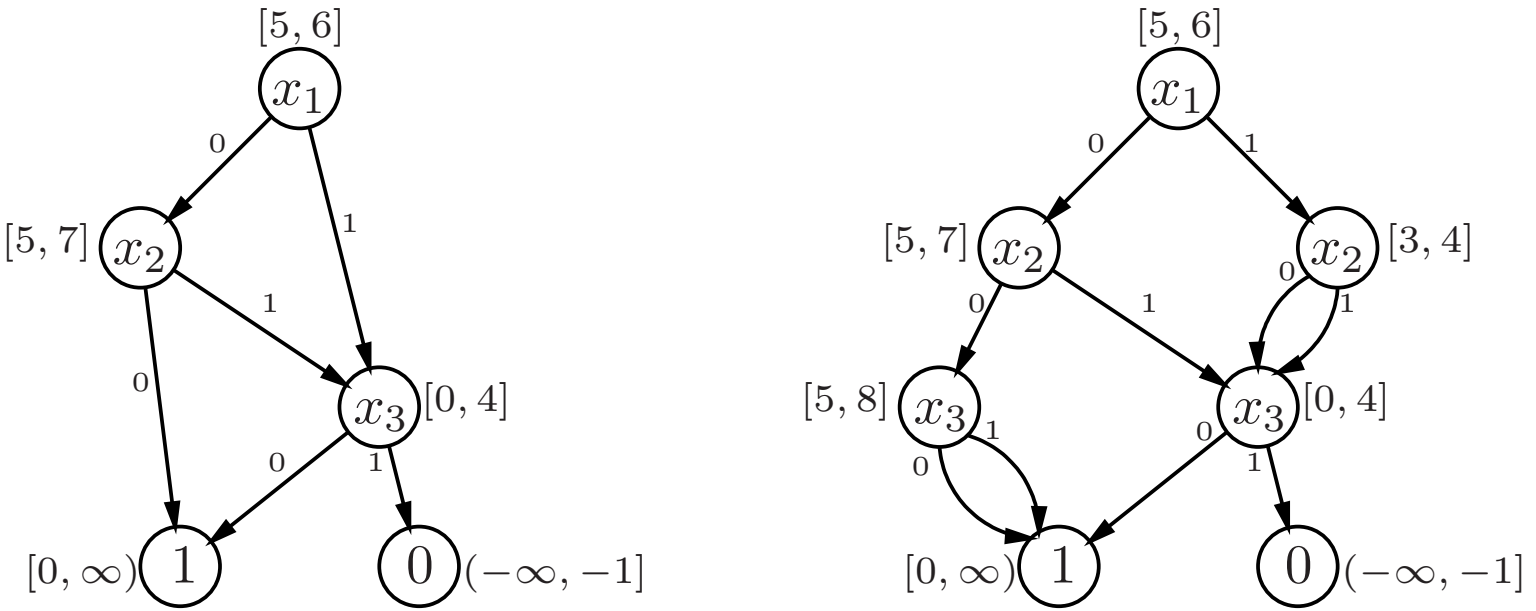


Let's compute the interval for the root node:

- False child represents $3x_2 + 5x_3 \leq [5, 7]$.
- True child represents $3x_2 + 5x_3 \leq [3, 4]$

Interval of a PB-constraint

Take BDD for $2x_1 + 3x_2 + 5x_3 \leq 6$ and a modified version:



Let's compute the interval for the root node:

- False child represents $2x_1 + 3x_2 + 5x_3 \leq [5,7]$ (since $x_1 = 0$)
- True child represents $2x_1 + 3x_2 + 5x_3 \leq [5,6]$ (since $x_1 = 1$)

Hence root node represents $2x_1 + 3x_2 + 5x_3 \leq [5,7] \cap [5,6] = [5,6]$

Size of BDDs for PB constraints

Theorem. Unless NP=co-NP,
there are PB constr. for which all BDDs are non-poly

Proof. Assume poly BDDs always exist.

Take UNSAT subset sum pbl: $\exists$ subset of $\{a_1, \ldots, a_n\}$ with sum K?
(NP-complete problem)
Let's show there is a poly UNSAT certificate verifiable in poly time:

We know that pbl is UNSAT iff $a_1x_1 + \ldots + a_nx_n \leq K$ and
$a_1x_1 + \ldots + a_nx_n \leq K-1$ are equiv.

Take poly BDD for $a_1 + \ldots a_n \leq K$ (will act as the certificate)

Checking that constraints are equivalent (i.e. pbl UNSAT) amounts:

1. Computing the interval I of the root (poly time)
2. Checking that $K - 1 \in I$

Avoiding Non-Polynomial BDDs

- Consider the constraint $9x_1 + 8x_2 + 3x_3 \leq 10$
- We can rewrite it using powers of two:

$$(8y_1 + 1y_2) + 8x_2 + (2z_1 + z_2) \leq 10$$

  and add clauses for $y_1 = y_2 = x_1$ and $z_1 = z_2 = x_3$
- Good news:
  BDDs for power-of-two PB constraints are polynomial
- Bad news:
  Arc-consistency for the original constraint is lost
- But...
  - Nice consistency property is preserved:
    $\mathcal{A}$ cannot be extended to a model $\Rightarrow \overline{r}$ unit-propagated
    (where r is the BDD root)
  - This allows us to recover arc-consistency [WalshIJCAI'09]

Recovering Arc-consistency

- Very simple idea:
  - Take constraint $C : 2^{e_1} x_1 + \ldots + 2^{e_n} x_n \leq K$
  - Build BDD-encoding for
    $C_i : 2^{e_1} x_1 + \ldots + 2^{e_i} \cdot 1 + \ldots + 2^{e_n} x_n \leq K$ with root var r_i
  - Add clauses $\overline{r_i} \rightarrow \overline{x_i}$

- It is easy to see that this recovers arc-consistency:
  - Take assigment $\mathcal{A}$ that can only be extended with $\overline{x_i}$
  - Hence C_i is inconsitent with $\mathcal{A}$ and $\overline{r_i}$ is unit propagated
  - $\overline{x_i}$ is propagated due to clause $\overline{r_i} \rightarrow \overline{x_i}$

- Upper bound on the final encoding size: $O(n \cdot n^2 \log a_{max})$

Overview of the talk

- Cumulative scheduling
- SAT encodings for PB constraints
  - SAT encodings vs SMT
  - Existing encodings
- Standard BDD-based encoding
- New results for BDD encodings
  - CNFs for monotonic functions
  - Probably BDDs for some PB constraints are non-poly
  - Avoiding non-polynomial BDDs
- Conclusions and Future Work

Conclusions and Future Work

Conclusions:

- Shown polynomial and arc-consistent BDD-based encoding
- Proved that, unless NP=co-NP, PB constraints have non-polynomial BDDs

Future work:

- Find PB constraint family that exhibits exponential behavior
- Try to achieve arc-consistency without replicating constraints
- Experimental evaluation on diverse benchmarks