Decision Diagrams for Sequencing and Scheduling

Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

www.andrew.cmu.edu/user/vanhoeve/mdd/

Plan

What can MDDs do for Combinatorial Optimization?

- *Compact representation* of all solutions to a problem
- Limit on size gives *approximation*
- Control strength of approximation by size limit

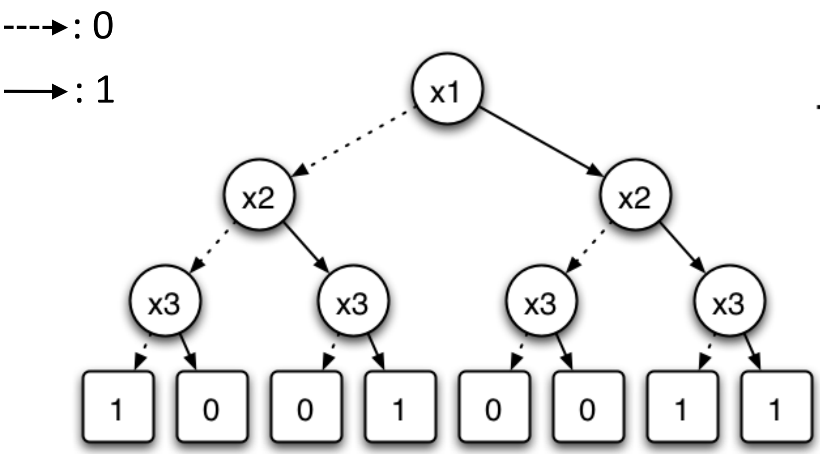
MDDs for Discrete Optimization

- 9:00am-10:30am tutorial (John Hooker)
- MDD as discrete relaxation for lower and upper bound
- Exact branch-and-bound search scheme (on MDD states)

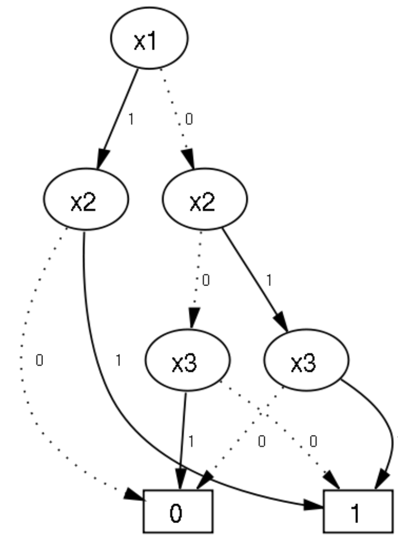
MDDs for Sequencing and Scheduling

- MDD-based constraint propagation
- Constraint-based scheduling with MDDs
- State-dependent costs

Decision Diagrams



x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



$$f(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$$

- Binary Decision Diagrams were introduced to compactly represent Boolean functions [Lee, 1959], [Akers, 1978], [Bryant, 1986]
- BDD: merge isomorphic subtrees of a given binary decision tree
- MDDs are multi-valued decision diagrams (i.e., for arbitrary finite-domain variables)

Brief background

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Original application areas: circuit design, verification
- Usually *reduced ordered* BDDs/MDDs are applied
  - fixed variable ordering
  - minimal exact representation
- Application to discrete optimization (exponential-size)
  - cut generation [Becker et al., 2005]
  - 0/1 vertex and facet enumeration [Behle & Eisenbrand, 2007]
  - post-optimality analysis [Hadzic & Hooker, 2006, 2007]
  - set bounds propagation [Hawkins, Lagoon, Stuckey, 2005]
- Scalable variant (polynomial-size)
  - relaxed MDDs

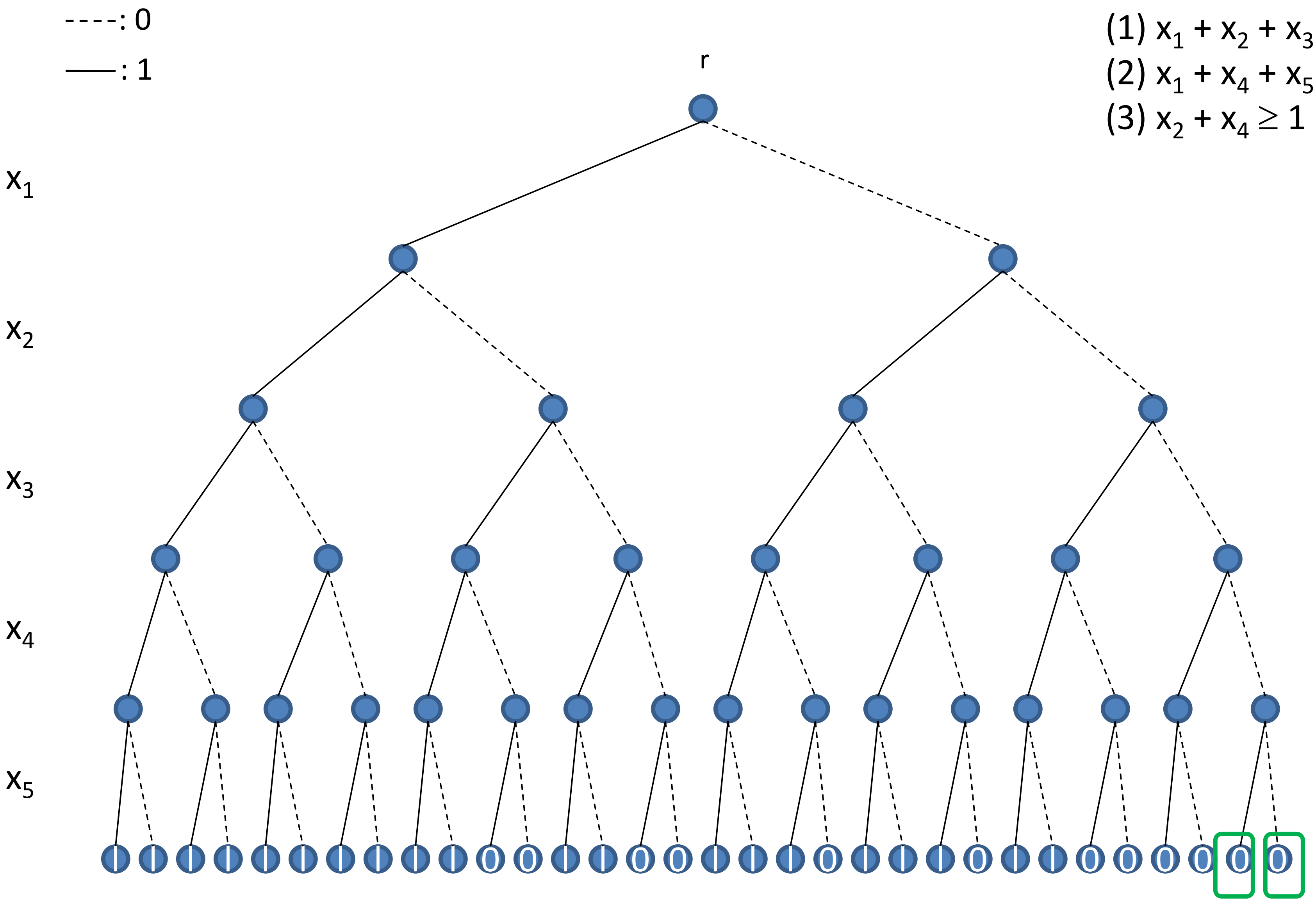
    [Andersen, Hadzic, Hooker & Tiedemann, CP 2007]

Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$

(2) $x_1 + x_4 + x_5 \geq 1$

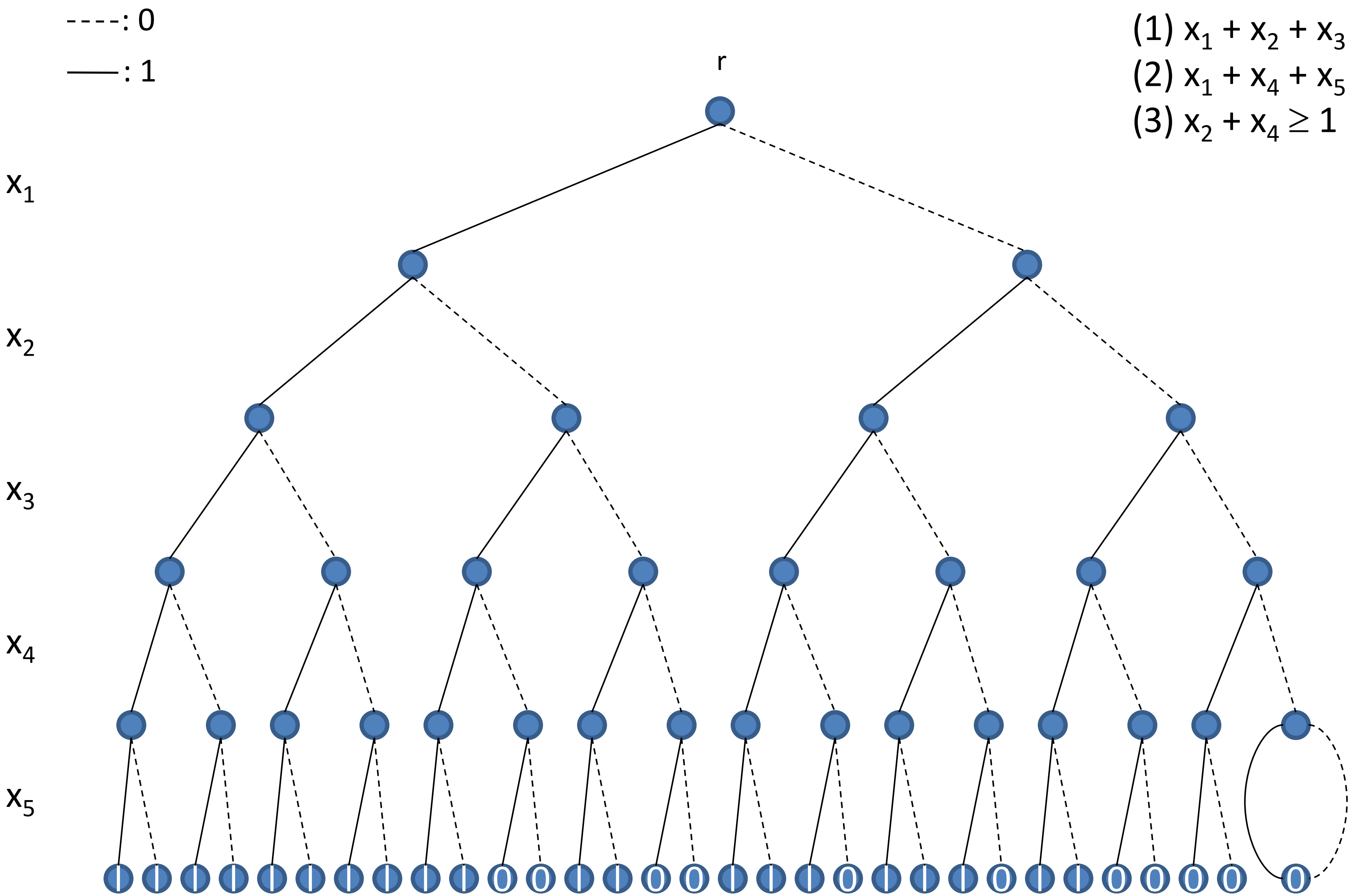
(3) $x_2 + x_4 \geq 1$



Exact MDDs for discrete optimization

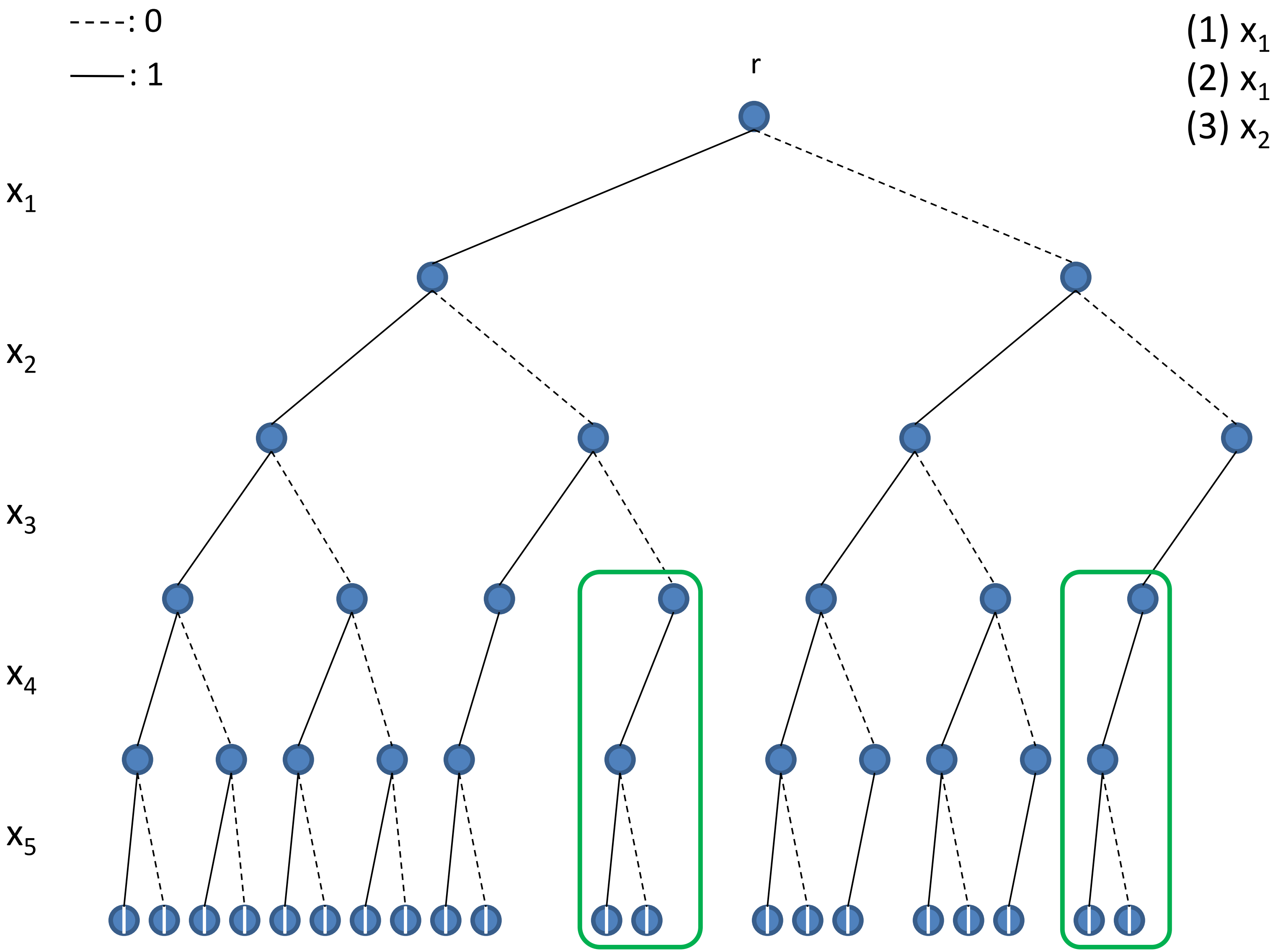
Carnegie Mellon Tepper SCHOOL OF BUSINESS

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



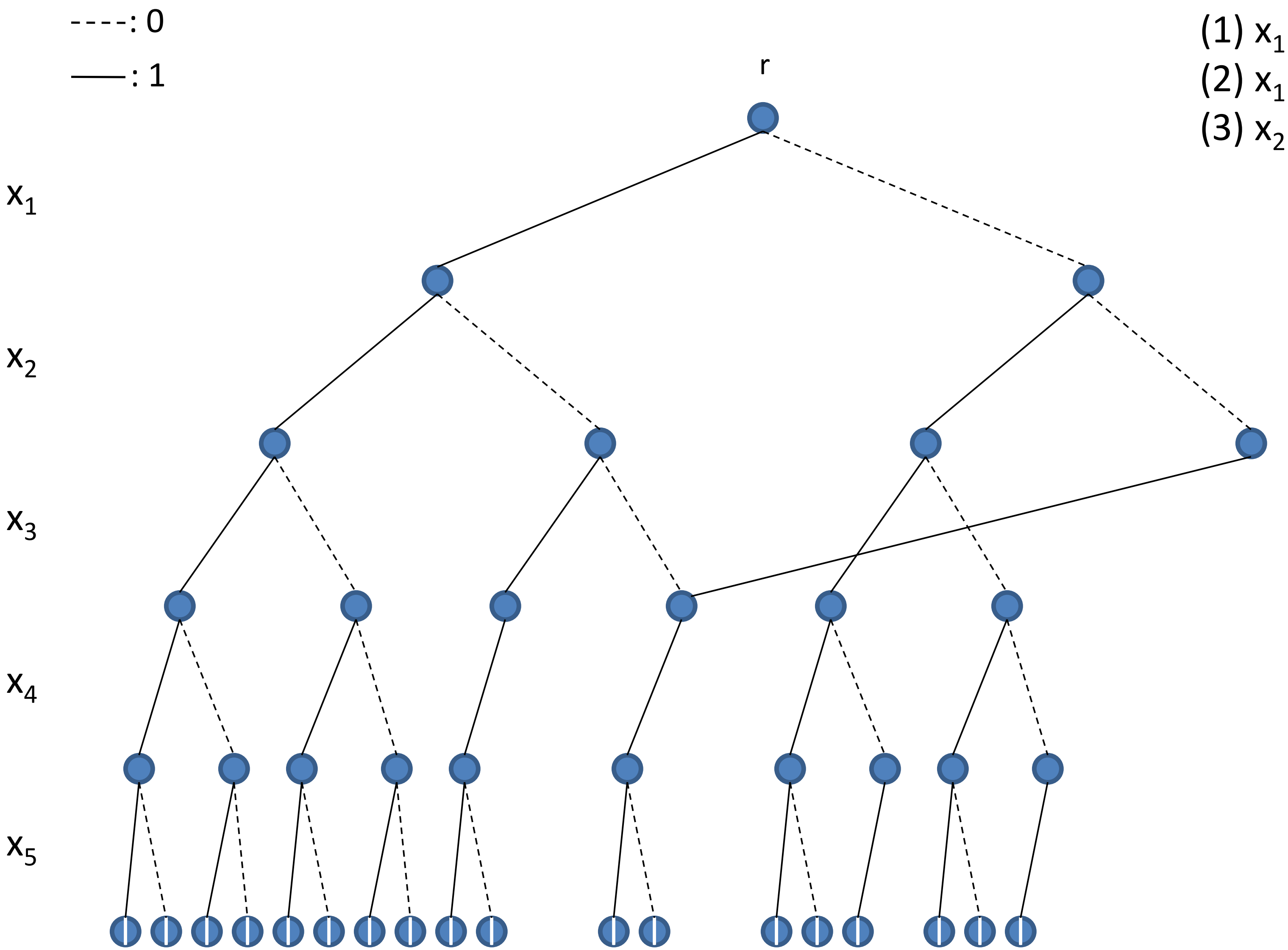
Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

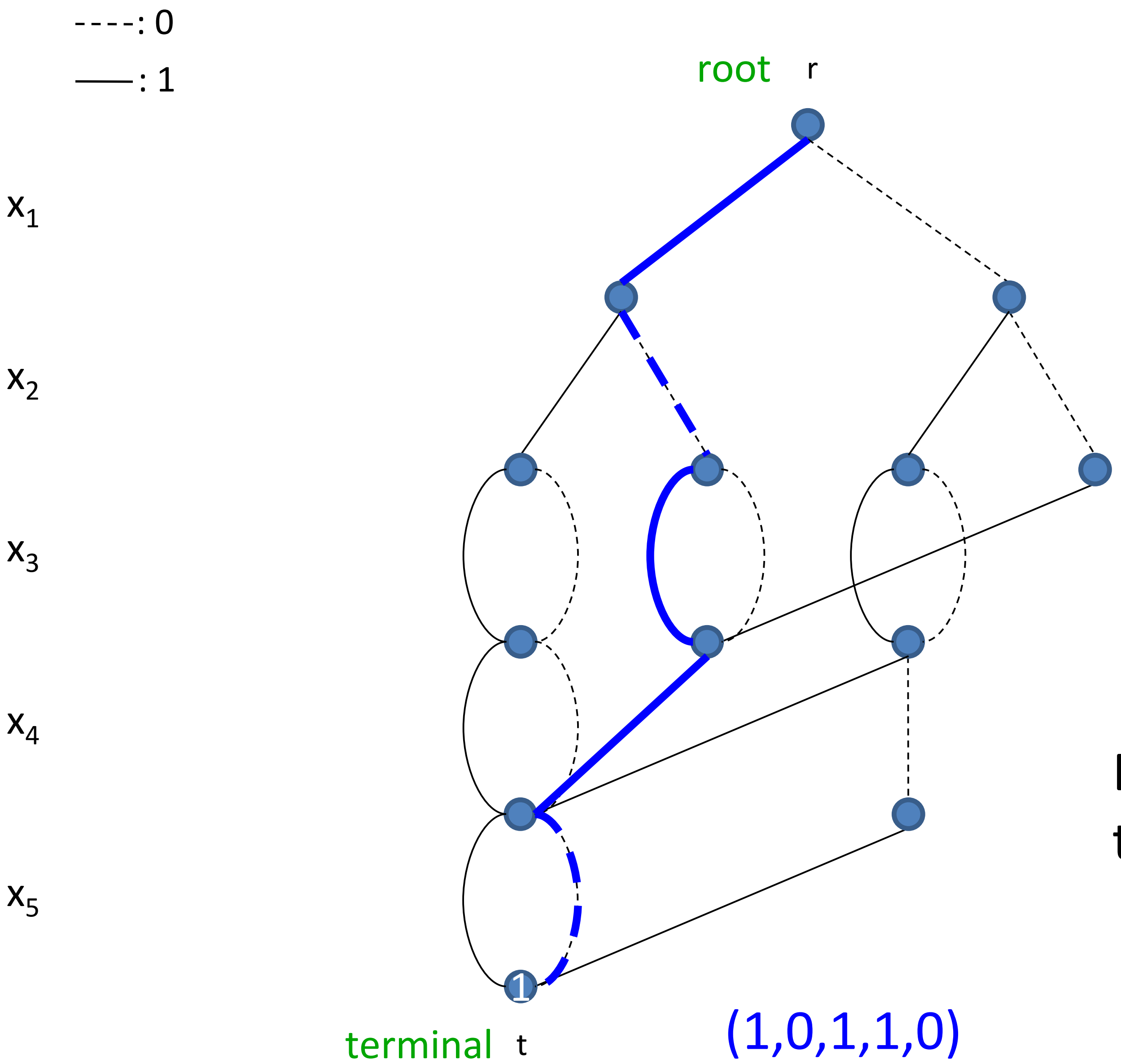


Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



Exact MDDs for discrete optimization



(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

Each path corresponds to a solution

(1,0,1,1,0)

Limited-size MDDs

- Exact MDDs can be of exponential size in general
- We can limit the size of the MDD and still have a meaningful representation:
  - First proposed by Andersen et al. [2007] for improved constraint propagation:

    Limit the *width* of the MDD (the maximum number of nodes on any layer)

MDDs for Constraint Programming

Motivation

Constraint Programming applies

- systematic search and
- inference techniques

to solve combinatorial problems

Inference mainly takes place through:

- Filtering provably inconsistent values from variable domains
- Propagating the updated domains to other constraints

$x_1 > x_2$
$x_1 + x_2 = x_3$
alldifferent(x_1, x_2, x_3, x_4)

domain propagation can be weak, however…

$x_1 \in \{\not 1, 2\}$, $x_2 \in \{\not 0, 1, \not 2, \not 3\}$, $x_3 \in \{\not 2, 3\}$, $x_4 \in \{0, \not 1\}$

Illustrative example

Carnegie Mellon Tepper SCHOOL OF BUSINESS

alldifferent(x_1,x_2,x_3,x_4)	(1)
$x_1 + x_2 + x_3 \geq 9$	(2)
$x_i \in \{1,2,3,4\}$	

(1) and (2) both domain consistent (no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
~~1~~	~~2~~	~~3~~	~~4~~
~~1~~	~~2~~	~~4~~	~~3~~
~~1~~	~~3~~	~~2~~	~~4~~
	…		
4	3	2	1

← Suppose we could evaluate (2) on this list

→ projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

alldifferent(x_1,x_2,x_3,x_4)	(1)
$x_1 + x_2 + x_3 \geq 9$	(2)
$x_i \in \{1,2,3,4\}$	

List of all solutions to *alldifferent*:

	x_1	x_2	x_3	x_4
✓	2	3	4	1
✓	2	4	3	1
✓	3	2	4	1
		…		
✓	4	3	2	1

Suppose we could evaluate (2) on this list

$D(x_1) = D(x_2) = D(x_3) = \{2,3,4\}$

projection: $D(x_4) = \{1\}$

Illustrative example (cont'd)

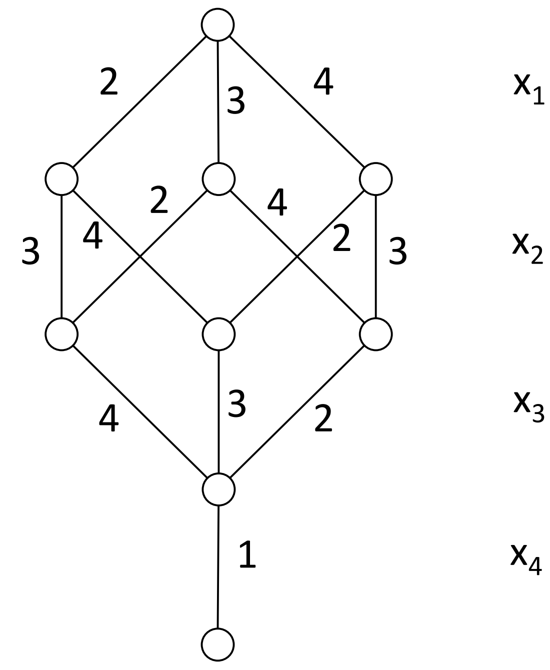
$$\textit{alldifferent}(x_1,x_2,x_3,x_4) \quad (1)$$

$$x_1 + x_2 + x_3 \geq 9 \quad (2)$$

$$x_i \in \{1,2,3,4\}$$

List of all solutions: use MDDs

x_1	x_2	x_3	x_4
2	3	4	1
2	4	3	1
3	2	4	1
	…		
4	3	2	1



Motivation for MDD propagation

- Conventional domain propagation projects all structural relationships among variables onto the domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

We can communicate more information between constraint using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space
- Limited width defines *relaxed* MDD
- Strength is controlled by the imposed width

MDD-based Constraint Programming

- Maintain limited-width MDD
  – Serves as relaxation
  – Typically start with width 1 (initial variable domains)
  – Dynamically adjust MDD, based on constraints
- Constraint Propagation
  – Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  – Node refinement: Split nodes to separate edge information
- Search
  – As in classical CP, but may now be guided by MDD

Specific MDD propagation algorithms

- Linear equalities and inequalities [Hadzic et al., 2008] [Hoda et al., 2010]
- *Alldifferent* constraints [Andersen et al., 2007]
- *Element* constraints [Hoda et al., 2010]
- *Among* constraints [Hoda et al., 2010]
- Disjunctive scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011, 2013]
- *Sequence* constraints (combination of *Amongs*) [Bergman et al., 2014]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]

Example: Among Constraints

- Given a set of variables X, and a set of values *S*, a lower bound *l* and upper bound *u*,

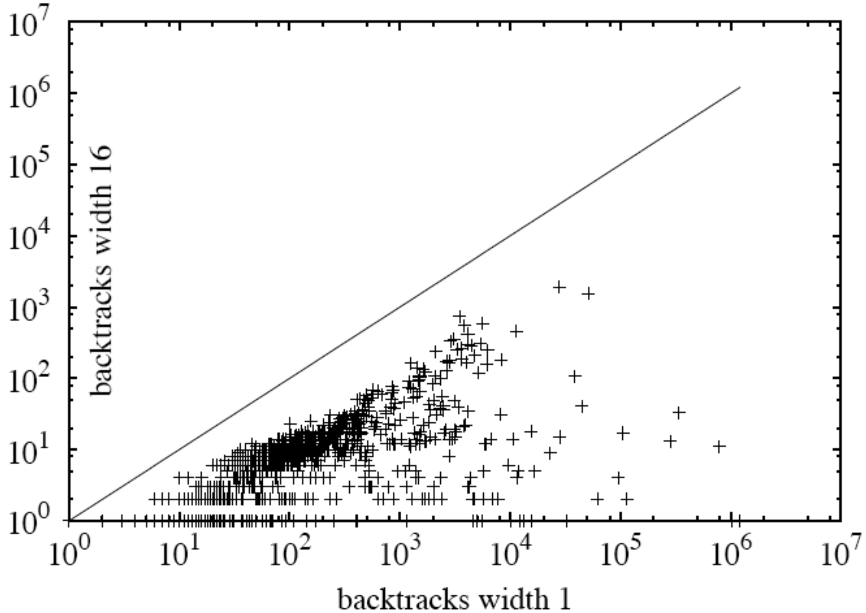
  $$\textit{Among}(\mathrm{X}, S, l, u) := \ l \leq \textstyle\sum_{\mathrm{x}\in\mathrm{X}} (\ \mathrm{x} \in S\ ) \leq u$$

  "among the variables in X, at least *l* and at most *u* take a value from the set *S*"

- Applications in, e.g., nurse scheduling
  - must work between 1 and 2 night shifts each 10 days

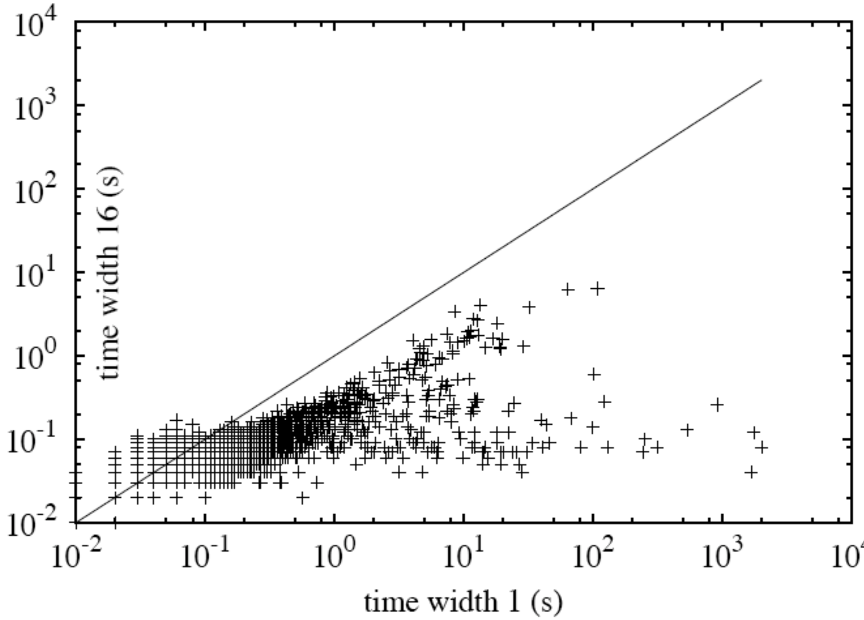
Propagating Among Constraints

backtracks



width 1 vs 16

time (s)



width 1 vs 16

(Systems of overlapping Among constraints)

Example: Sequence Constraints

Employee must work at most 7 days every 9 consecutive days

sun	mon	tue	wed	thu	fri	sat	sun	mon	tue	wed	thu
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}

$0 \le x_1 + x_2 + \ldots + x_9 \le 7$

$0 \le x_2 + x_3 + \ldots + x_{10} \le 7$

$0 \le x_3 + x_4 + \ldots + x_{11} \le 7$

$0 \le x_4 + x_5 + \ldots + x_{12} \le 7$

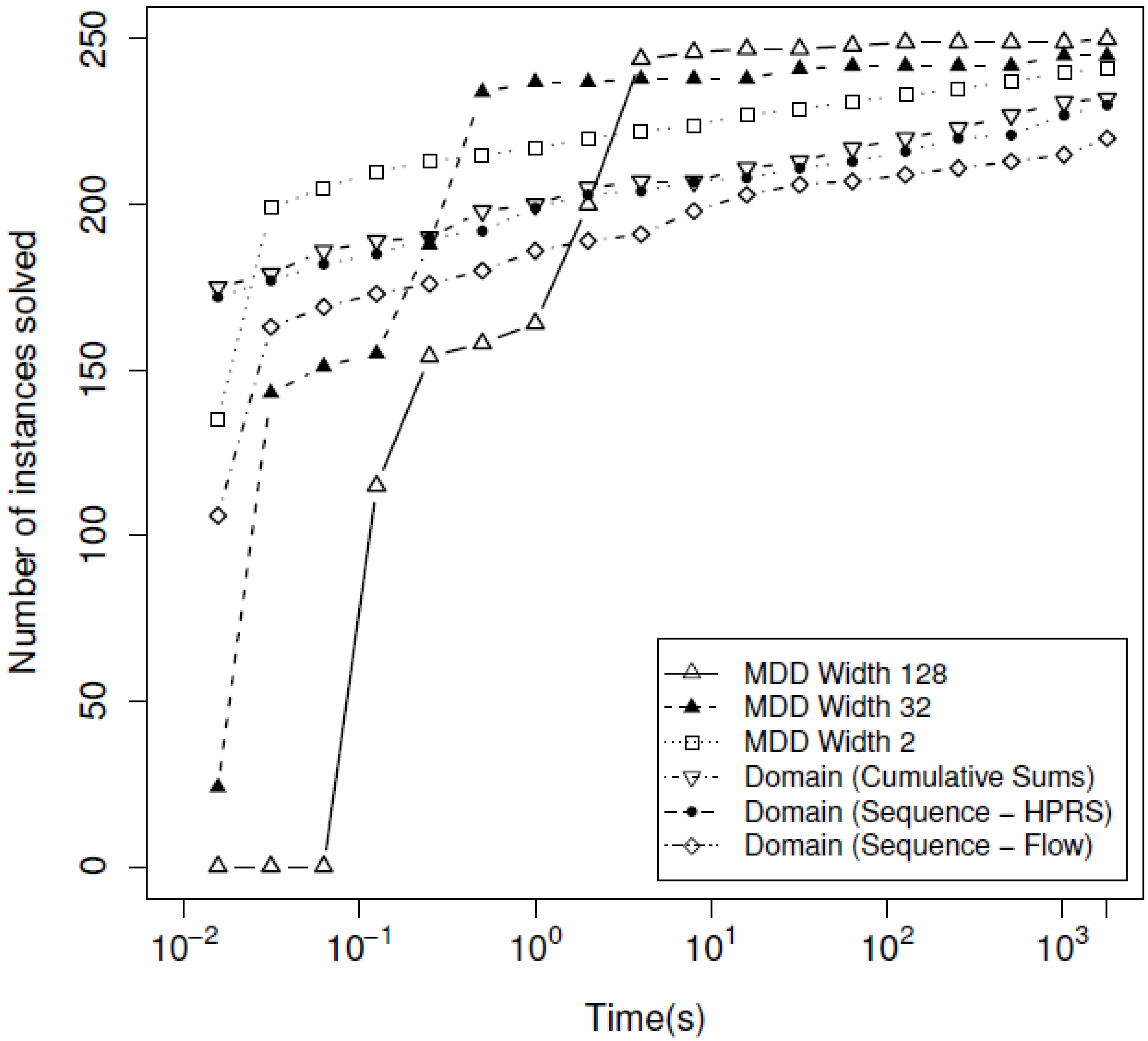
=: *Sequence*($[x_1, x_2, \ldots, x_{12}]$, q=9, S={1}, l=0, u=7)

Sequence(X, *q*, *S*, *l*, *u*) := $\bigwedge_{|X'|=q} \; l \le \sum_{x \in X'} (x \in S) \le u$

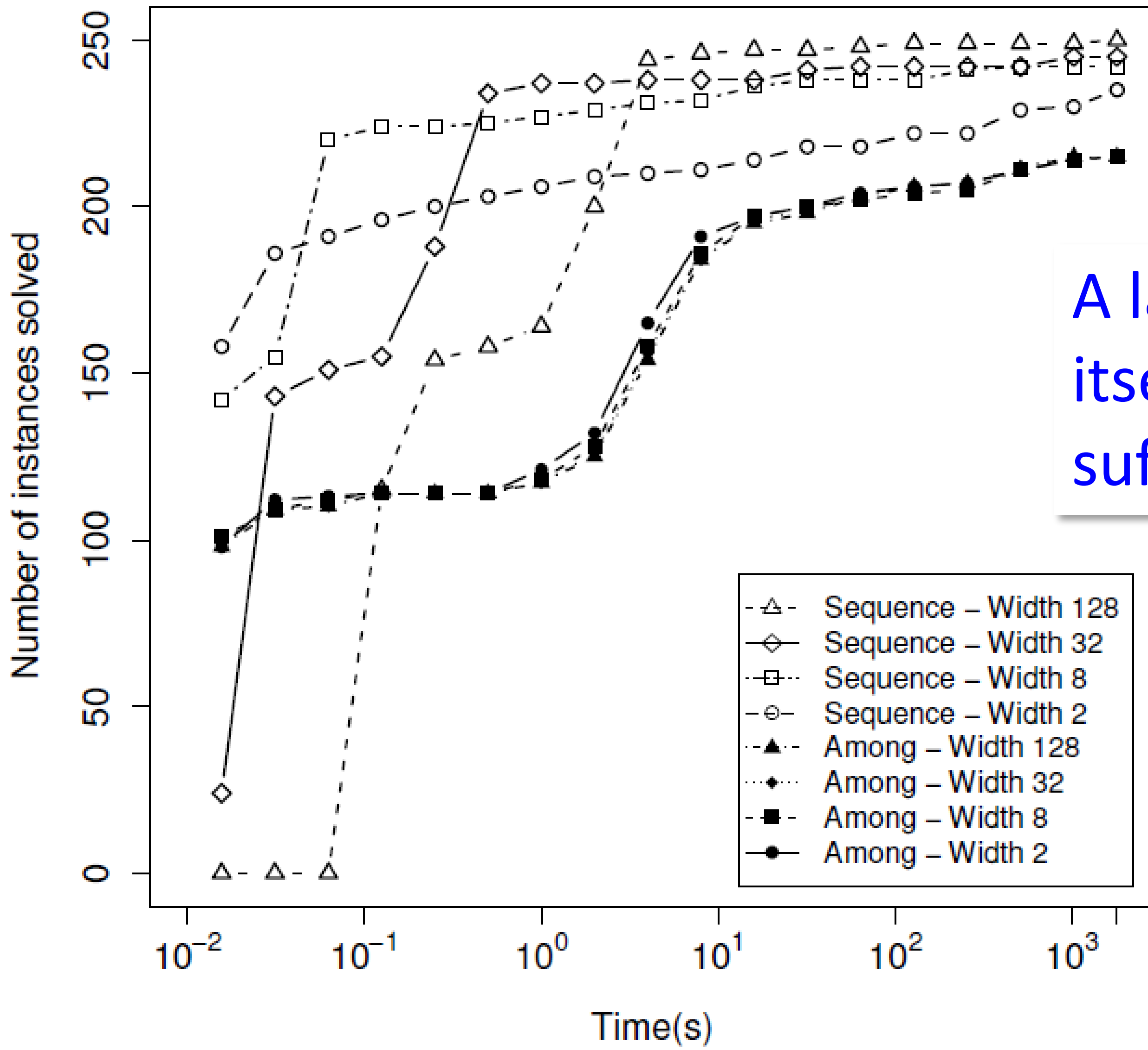
↓

Among(X, *S*, *l*, *u*)

Performance Comparison for Sequence



Sequence vs. Among

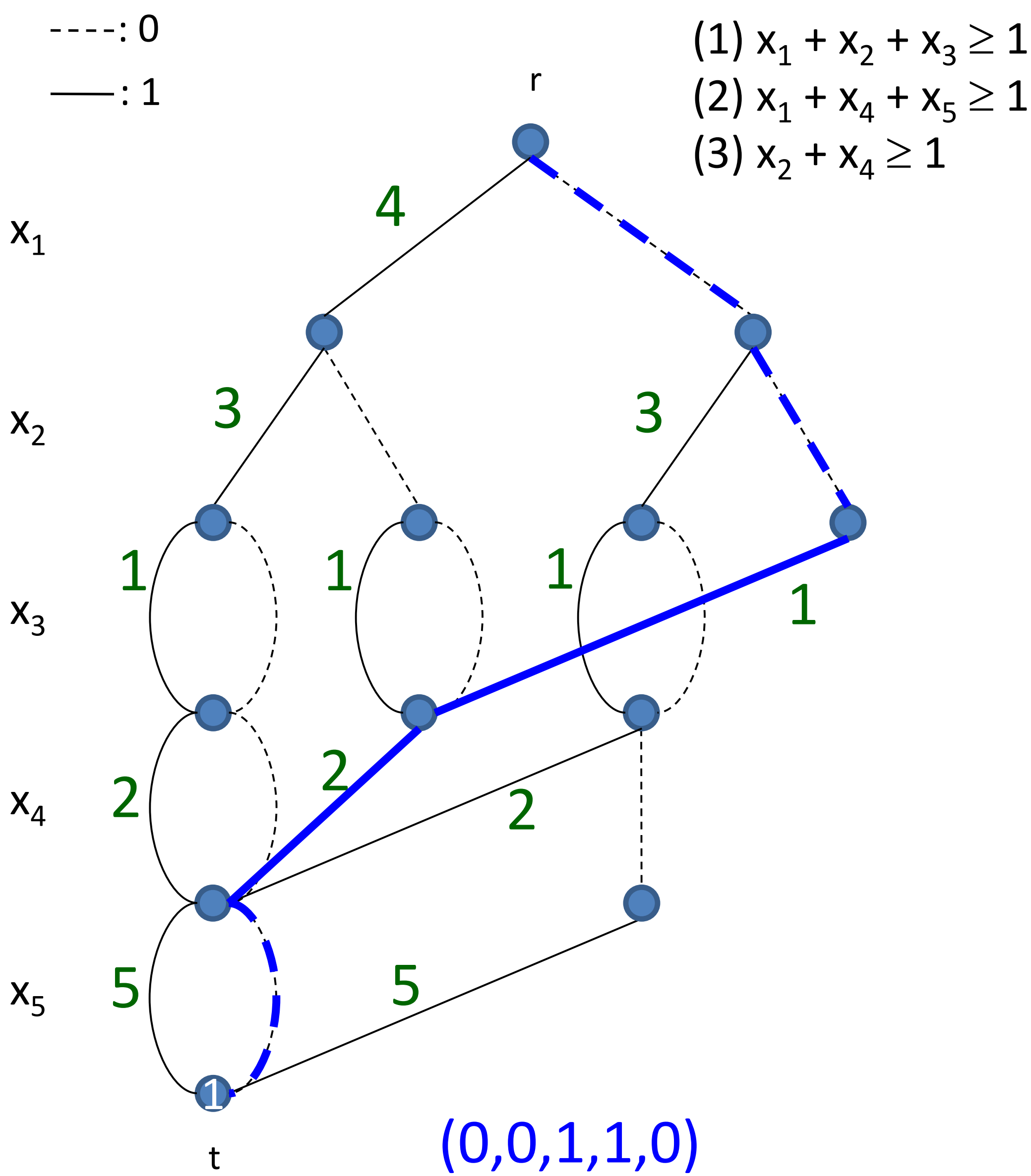


A large MDD by itself may not be sufficient!

Coming up

- MDDs can handle objective functions as well
- Important for many CP problems
  - e.g., disjunctive scheduling
  - minimize makespan, weighted completion times, etc.
- We will develop an MDD approach to disjunctive scheduling
  - combines MDD propagation and optimization reasoning

Handling objective functions



Suppose we have an objective:

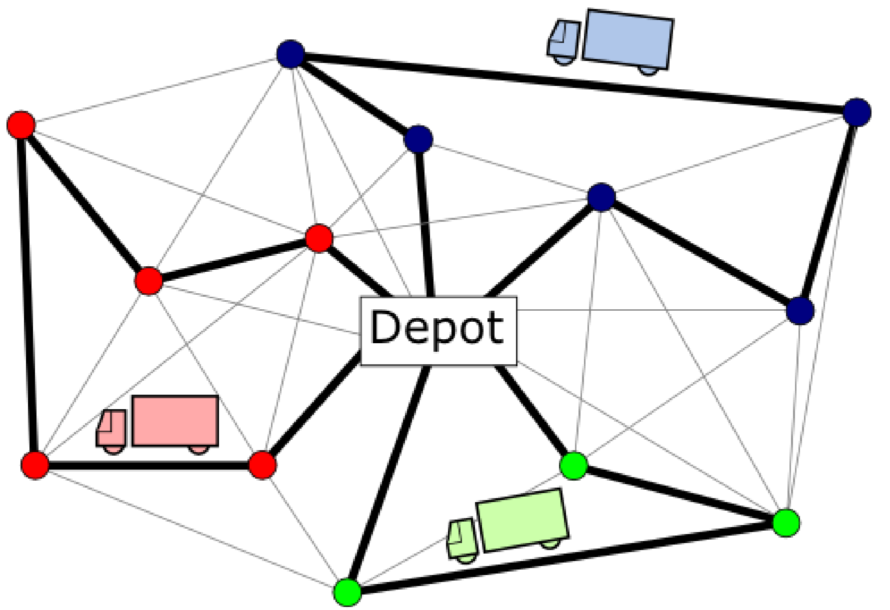
$$\min\ 4x_1+3x_2+x_3+2x_4+5x_5$$

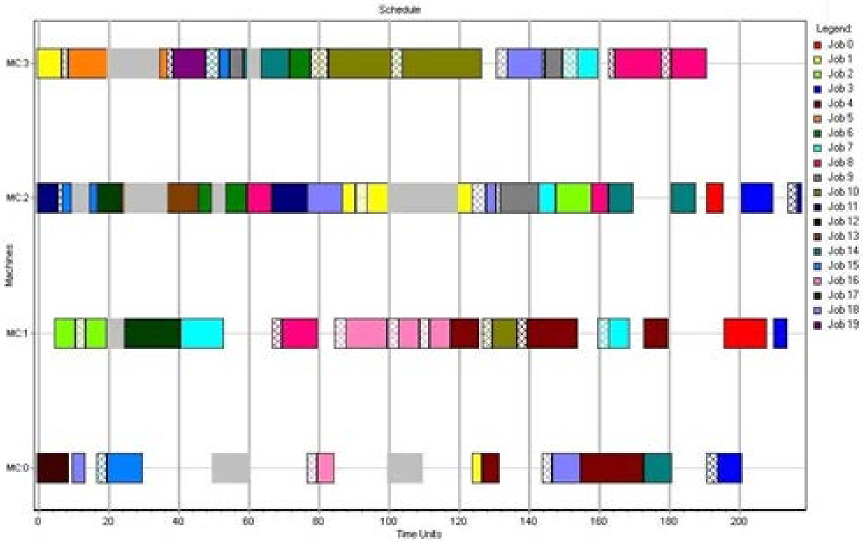
shortest path computation

MDDs for Disjunctive Scheduling

- Cire and v.H. Multivalued Decision Diagrams for Sequencing Problems. *Operations Research* 61(6): 1411-1428, 2013.

Disjunctive Scheduling

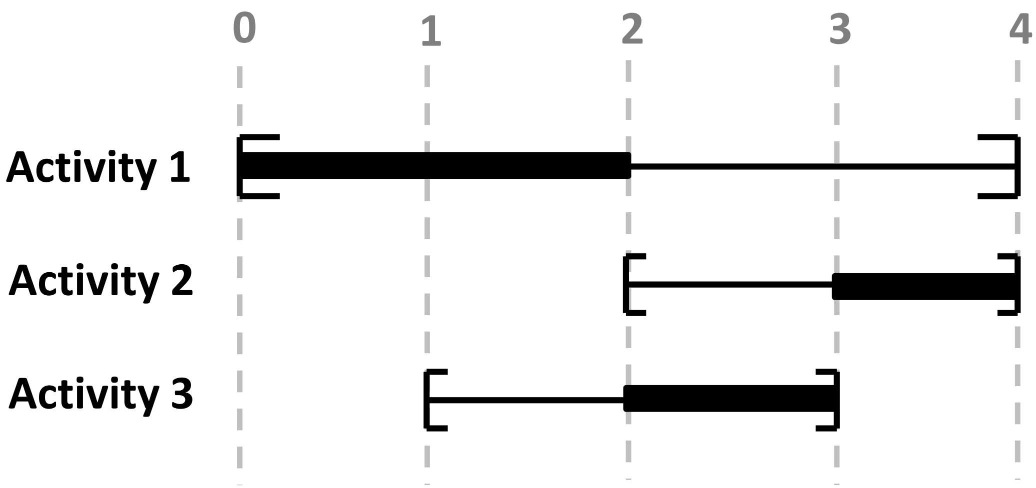




Disjunctive Scheduling in CP

- Sequencing and scheduling of activities on a resource

- *Activities*
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
  - Start time variable: s_i



- *Resource*
  - Nonpreemptive
  - Process one activity at a time

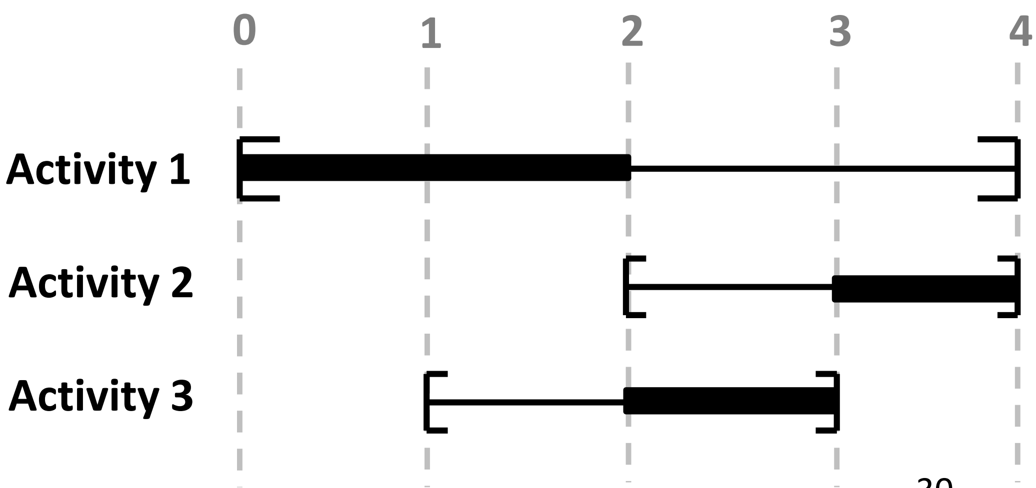
Extensions

- Precedence relations between activities
- Sequence-dependent setup times
- Various objective functions
  - Makespan
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

Inference

- Inference for disjunctive scheduling
  - Precedence relations
  - Time intervals in which an activity can be processed
- Sophisticated techniques include:
  - *Edge-Finding*
  - *Not-first / not-last rules*
- Examples: $1 \ll 3$

  $s_3 \geq 3$



Assessment of CP Scheduling

- Disjunctive scheduling may be viewed as the ‘killer application’ for CP
  - Natural modeling (activities and resources)
  - Allows many side constraints (precedence relations, time windows, setup times, etc.)
  - Among state of the art while being generic methodology
- However, CP has some problems when
  - objective is not minimize makespan (but instead, e.g., weighted sum of lateness)
  - setup times are present
  - …
- What can MDDs bring here?

optimization

MDDs for Disjunctive Scheduling

Three main considerations:

- Representation
  - How to represent solutions of disjunctive scheduling in an MDD?
- Construction
  - How to construct this relaxed MDD?
- Inference techniques
  - What can we infer using the relaxed MDD?

MDD Representation

- Natural representation as 'permutation MDD'

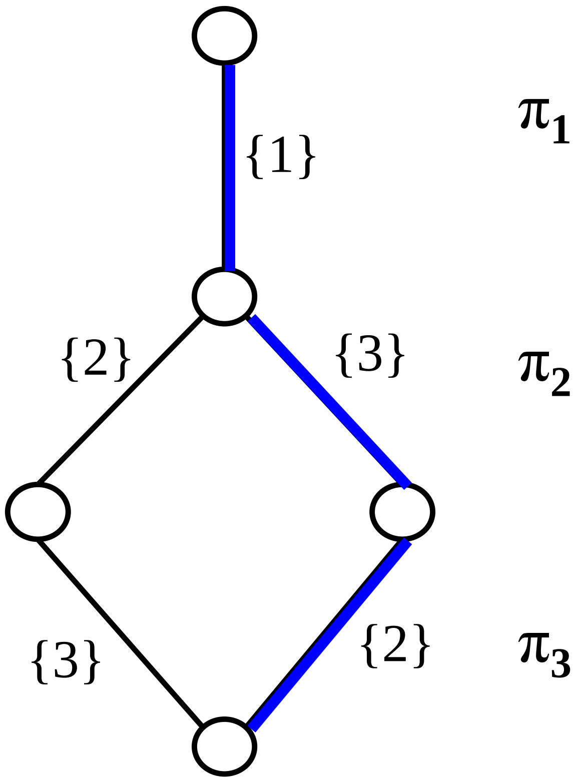
- Every solution can be written as a permutation $\boldsymbol{\pi}$

  $\pi_1, \pi_2, \pi_3, \ldots, \pi_n$: activity sequencing in the resource

- Schedule is *implied* by a sequence, e.g.:

$$start_{\pi_i} \geq start_{\pi_{i-1}} + p_{\pi_{i-1}} \qquad i = 2, \ldots, n$$

MDD Representation



Act	r_i	p_i	d_i
1	0	2	3
2	4	2	9
3	3	3	8

Path {1} – {3} – {2} :

$0 \leq start_1 \leq 1$

$6 \leq start_2 \leq 7$

$3 \leq start_3 \leq 5$

Exact MDD Compilation

Theorem: *Constructing the exact MDD for a Disjunctive Instance is an NP-Hard problem*

- We work with MDD relaxations instead
- Bounded size in specific cases, e.g. (Balas [99]):

- TSP defined on a complete graph
- Given a fixed parameter **k**, we must satisfy

$$i \ll j \quad \text{if} \quad j - i \geq k \quad \text{for cities i, j}$$

Theorem: *The exact MDD for the TSP above has $O(n2^k)$ nodes*

MDD-based propagation

Propagation: remove infeasible arcs from the MDD

We can utilize several structures/constraints:

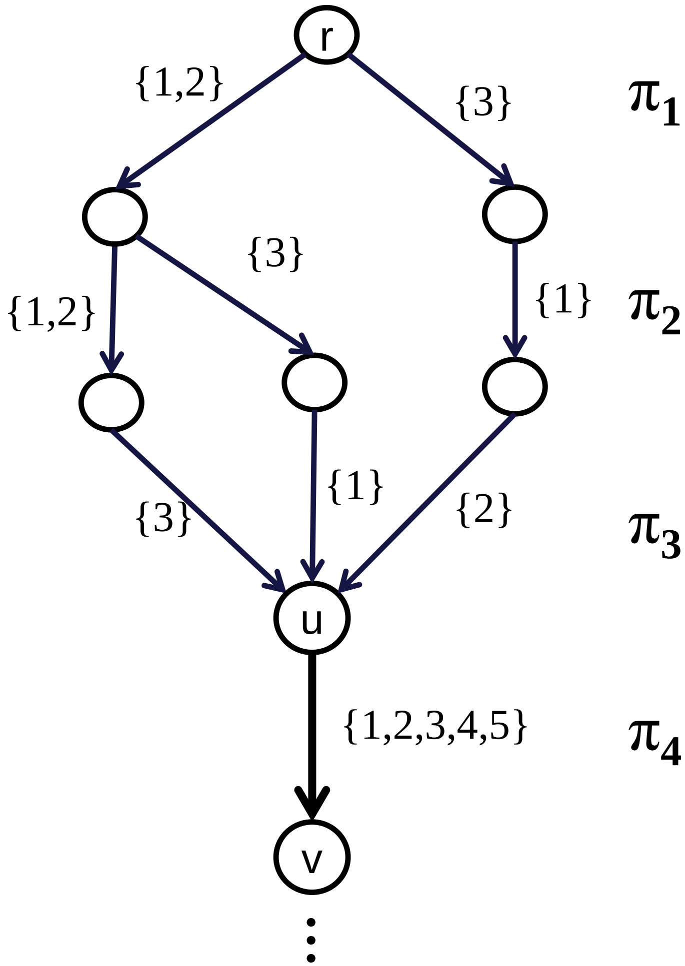
- *Alldifferent* for the permutation structure
- Earliest start time and latest end time
- Precedence relations

For a given constraint type we maintain specific ‘state information’ at each node in the MDD

  – both top-down and bottom-up

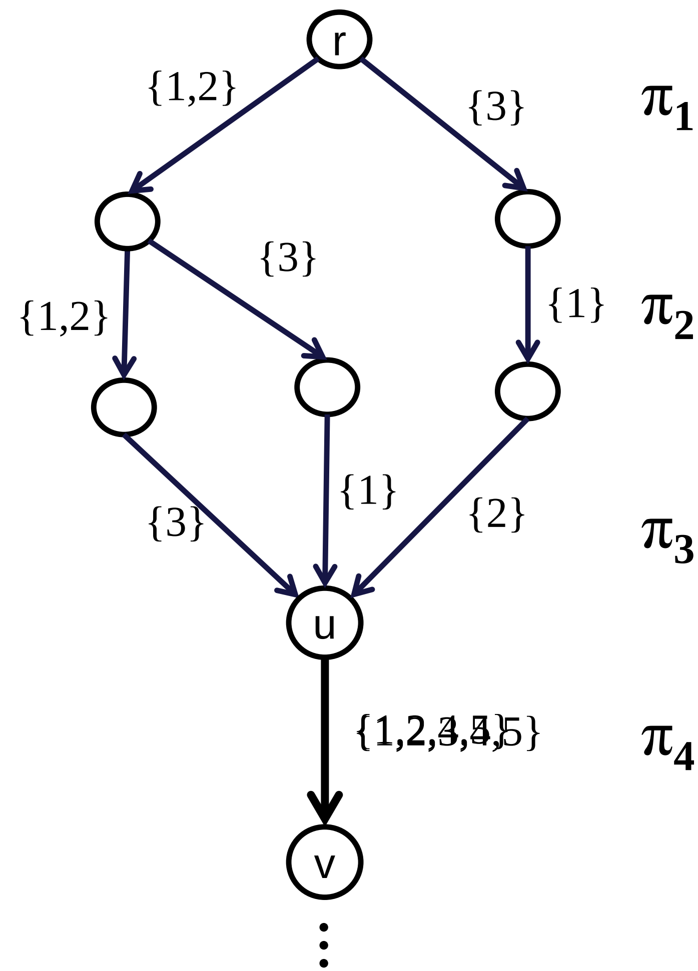
Propagation (cont'd)

- State information at each node i
  - labels on *all* paths: A_i
  - labels on *some* paths: S_i
  - earliest starting time: E_i
  - latest completion time: L_i
- Top down example for arc (u,v)



Alldifferent Propagation

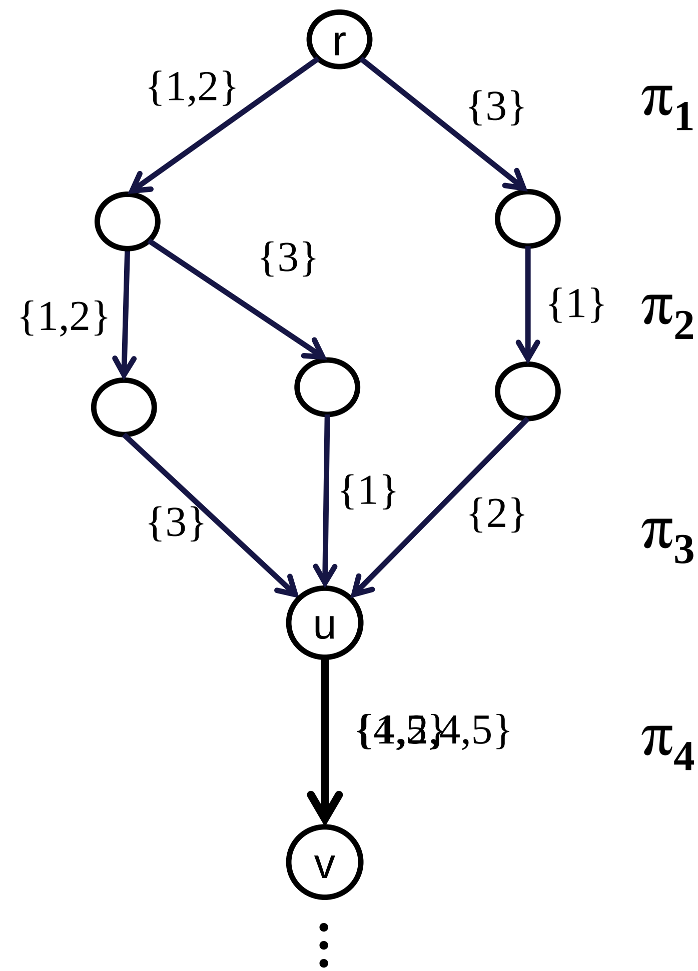
- All-paths state: A_u
  - Labels belonging to all paths from node r to node u
  - $A_u = \{3\}$
  - Thus eliminate {3} from (u,v)



[Andersen et al., 2007]

Alldifferent Propagation

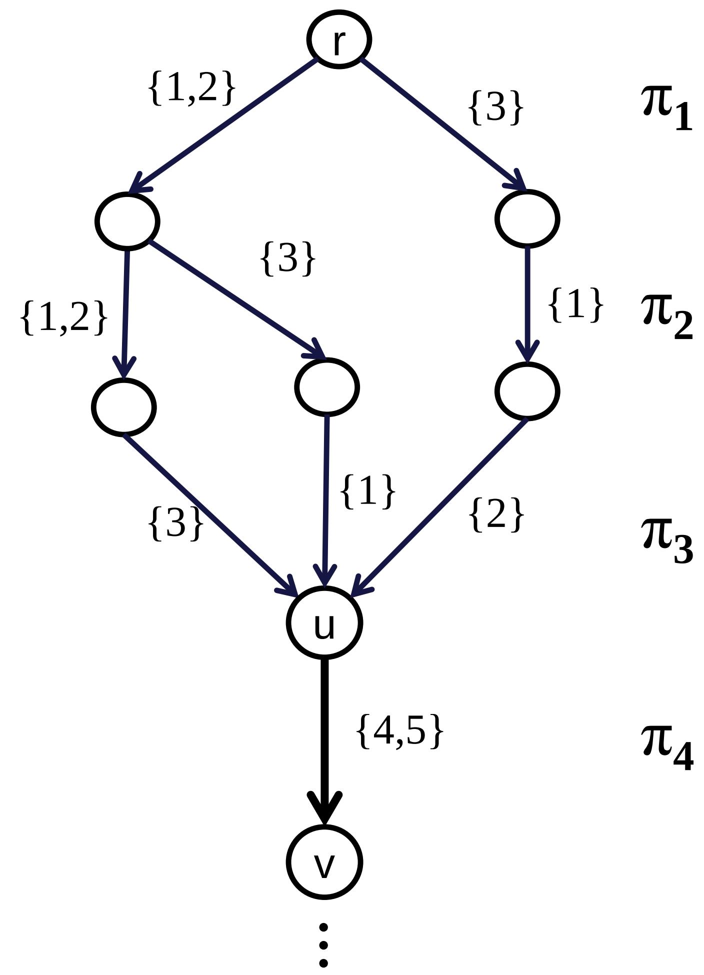
- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - S_u = {1,2,3}
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

- Earliest Completion Time: E_u
  - Minimum completion time of all paths from root to node u

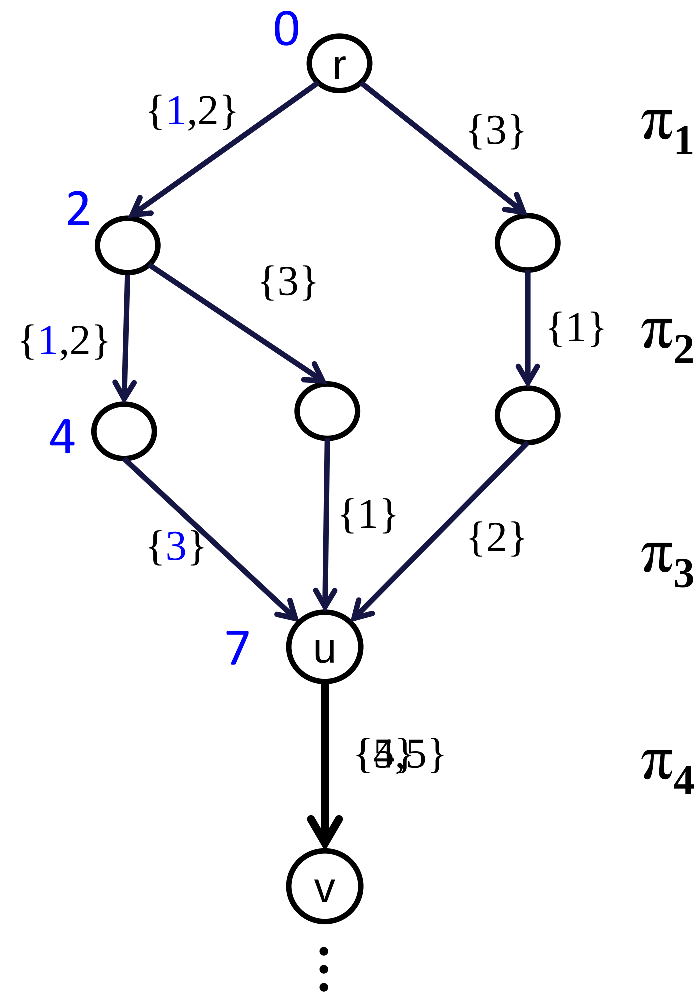
- Similarly: Latest Completion Time



Propagate Earliest Completion Time

Act	r_i	d_i	p_i
1	0	4	2
2	3	7	3
3	1	8	3
4	**5**	**6**	**1**
5	2	10	3

- $E_u = 7$
- Eliminate 4 from (u,v)



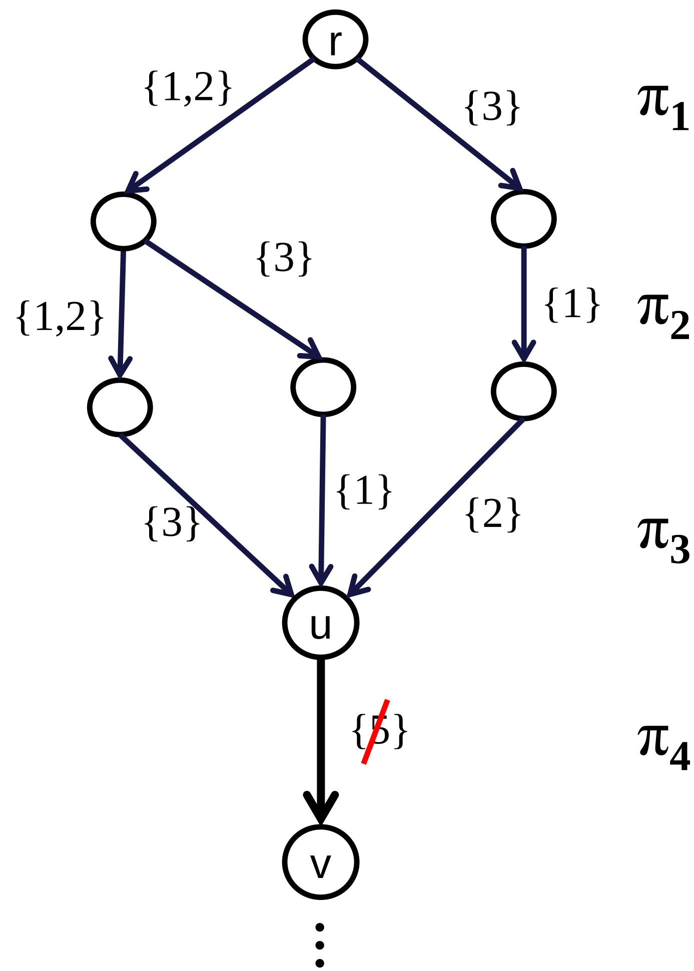
Propagate Precedence Relations

- Arc with label *j* infeasible if

  $i \ll j$ and *i* not on some path from r

- Suppose $4 \ll 5$
  - $S_u = \{1,2,3\}$
  - Since 4 not in S_u, eliminate 5 from (u,v)

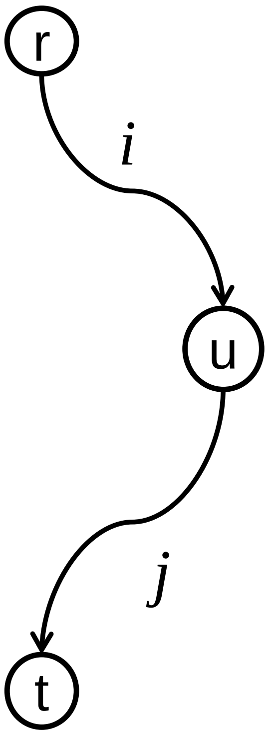
- Similarly: Bottom-up for $j \ll i$



More MDD Inference

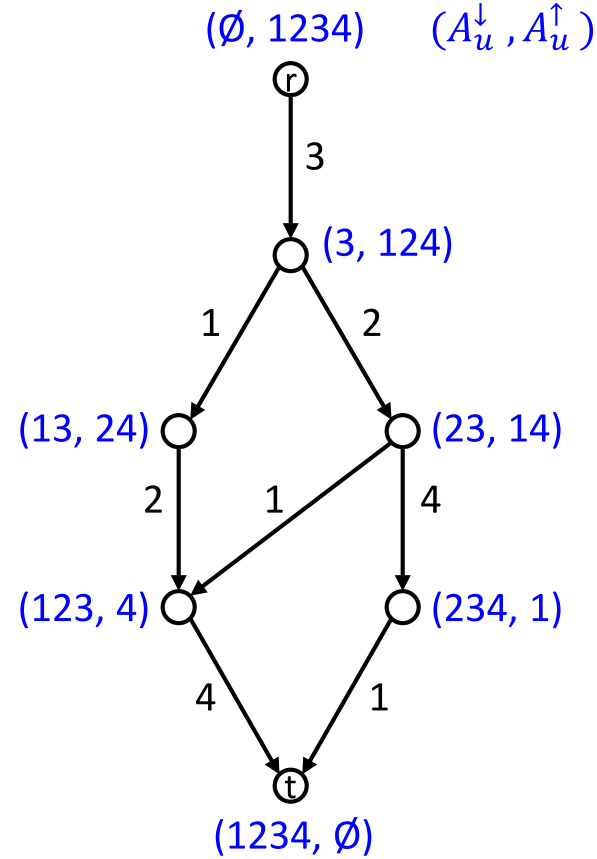
Theorem: *Given the exact MDD M, we can deduce all implied activity precedences in polynomial time in the size of M*

- For a node *u*,
  - $A_u^{\downarrow}$: values in all paths from root to *u*
  - $A_u^{\uparrow}$: values in all paths from node *u* to terminal
- *Precedence relation i ≪ j holds if and only if*
  $(j \notin A_u^{\downarrow})$ or $(i \notin A_u^{\uparrow})$ *for all nodes u in M*
- Same technique applies to relaxed MDD

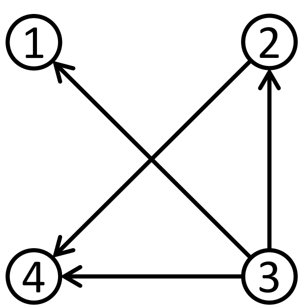


Extracting precedence relations

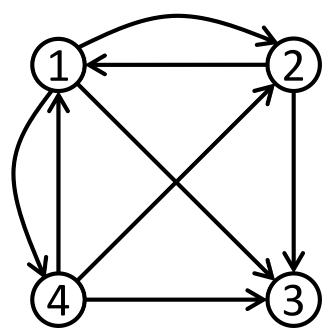
- Build a digraph G=(V, E) where V is the set of activities
- For each node u in M
  - if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$ add edge (i,j) to E
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds



$3 \ll 1$
$3 \ll 2$
$3 \ll 4$
$2 \ll 4$



$\overline{G}$



G

Extracting precedence relations

- Build a digraph G=(V, E) where V is the set of activities
- For each node u in M
  - if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$ add edge (*i*,*j*) to E
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds
- Time complexity: O(|*M*|n^2)

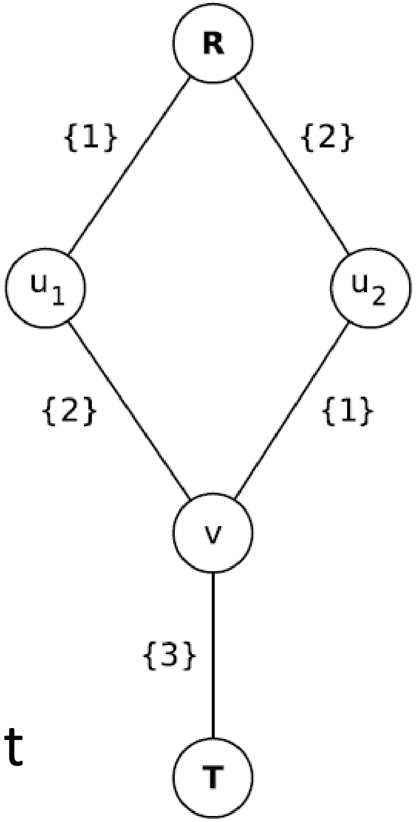
- Same technique applies to *relaxed* MDD
  - add an edge if $j \in S_u^{\downarrow}$ and $i \in S_u^{\uparrow}$
  - complement graph represents subset of precedence relations

Comparison to other methods

- Existing CP inference methods may not dominate the MDD propagation, even for small widths

Act	r_i	d_i	p_i
1	0	25	11
2	1	27	10
3	14	35	5

[Vilim, 2004]



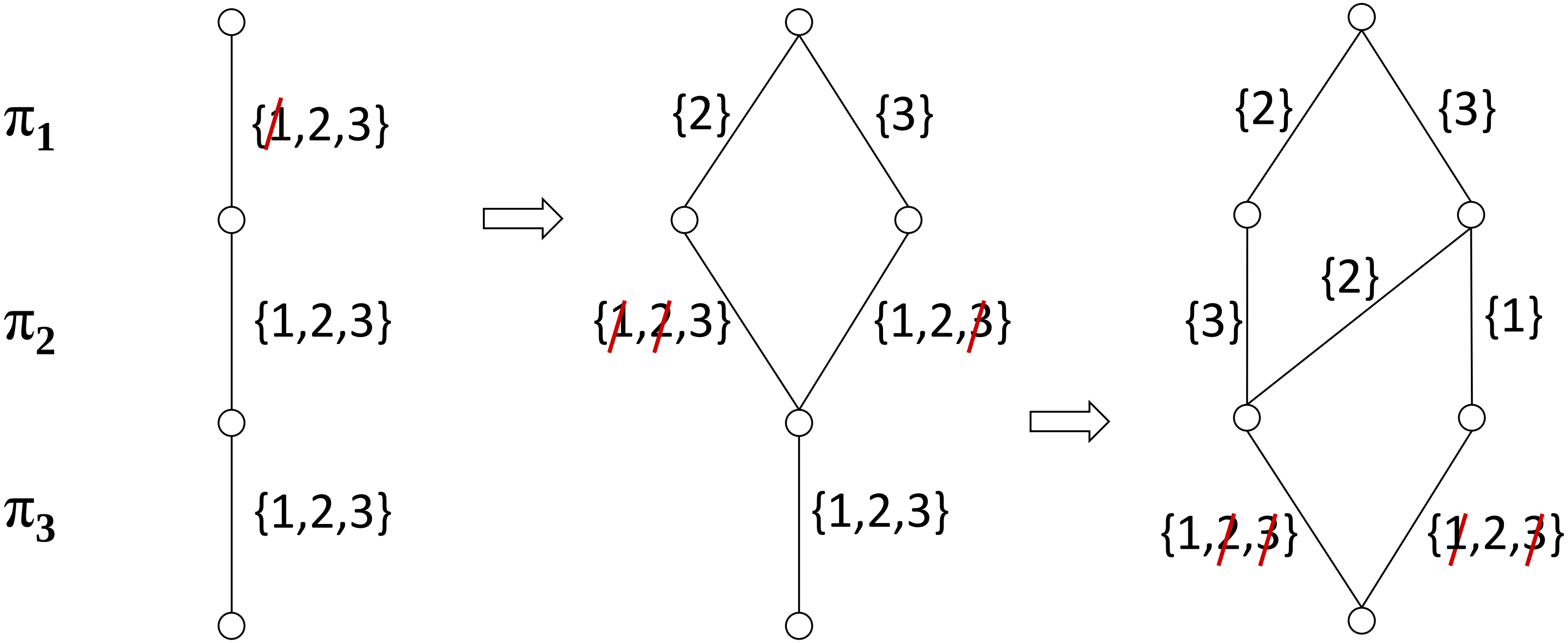
- Edge finding and not-first/not-last deduce that $1 \ll 3$ and $2 \ll 3$, but no changes in time bounds
- MDD finds the same precedences, *and* deduces that $s_3 \geq 10 + 11 = 21$

Communicate Precedence Relations

1. Provide precedence relations from MDD to CP
   - update start/end time variables
   - other inference techniques may utilize them
   - (some of the precedence relations found by the MDD may not be detected by existing CP methods)

2. Filter the MDD using precedence relations from other (CP) techniques

Top-down MDD compilation



precedence: 3 << 1

(exact MDD)

- To refine the MDD, we generally want to identify equivalence classes among nodes in a layer
  - NP-hard, but can be based on state information in practice, e.g., EST, LCT, *alldifferent* constraint (A_i and S_i states), …

Computational Evaluation

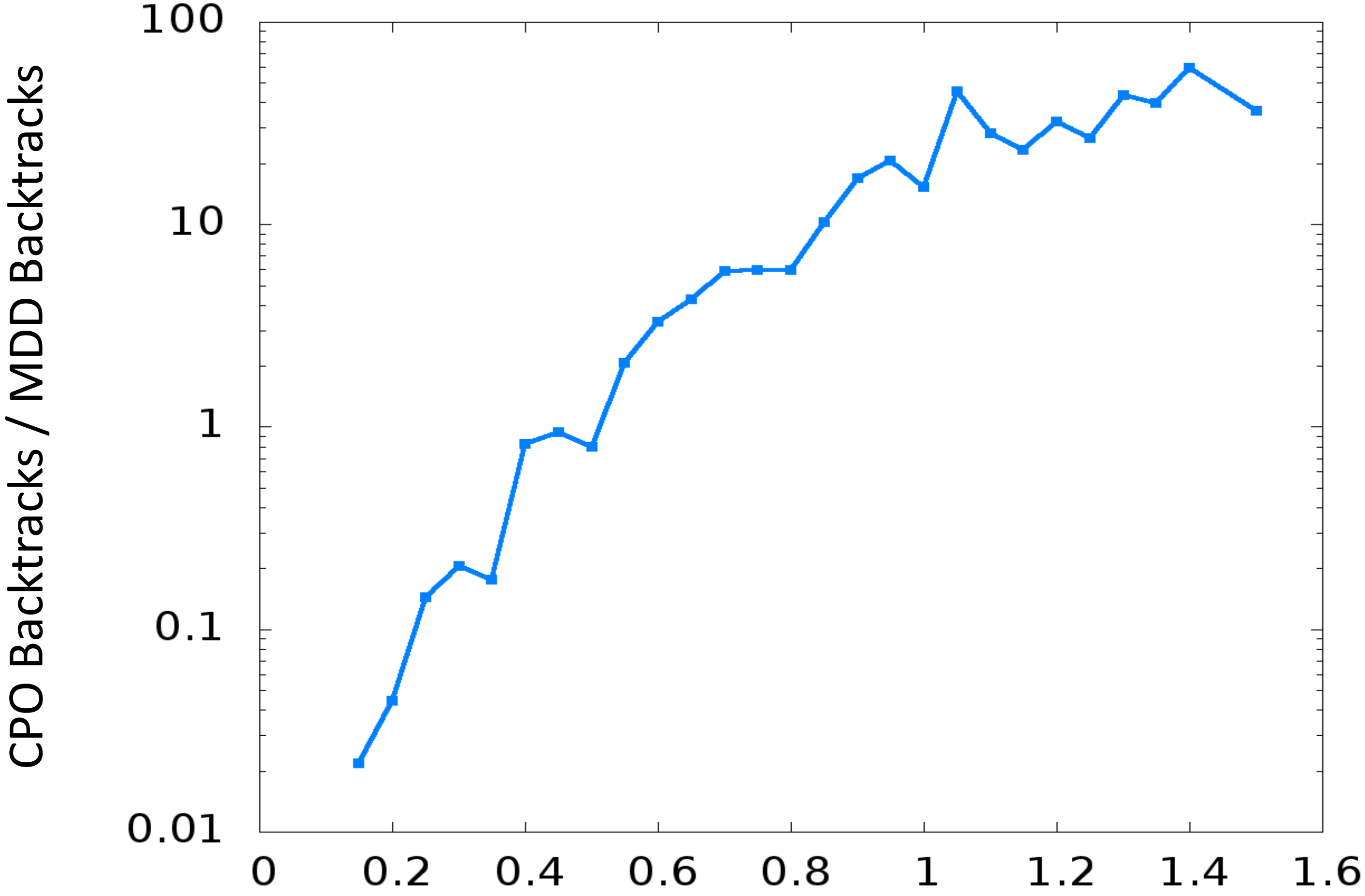
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Three different variants
  - CPO (only use CPO propagation)
  - MDD (only use MDD propagation)
  - CPO+MDD (use both)

Problem classes

- Disjunctive instances with
  - sequence-dependent setup times
  - release dates and deadlines
  - precedence relations
- Objectives
  - minimize makespan
  - minimize sum of setup times
  - minimize total tardiness
- Benchmarks
  - Random instances with varying setup times
  - TSP-TW instances (Dumas, Ascheuer, Gendreau)
  - Sequential Ordering Problem

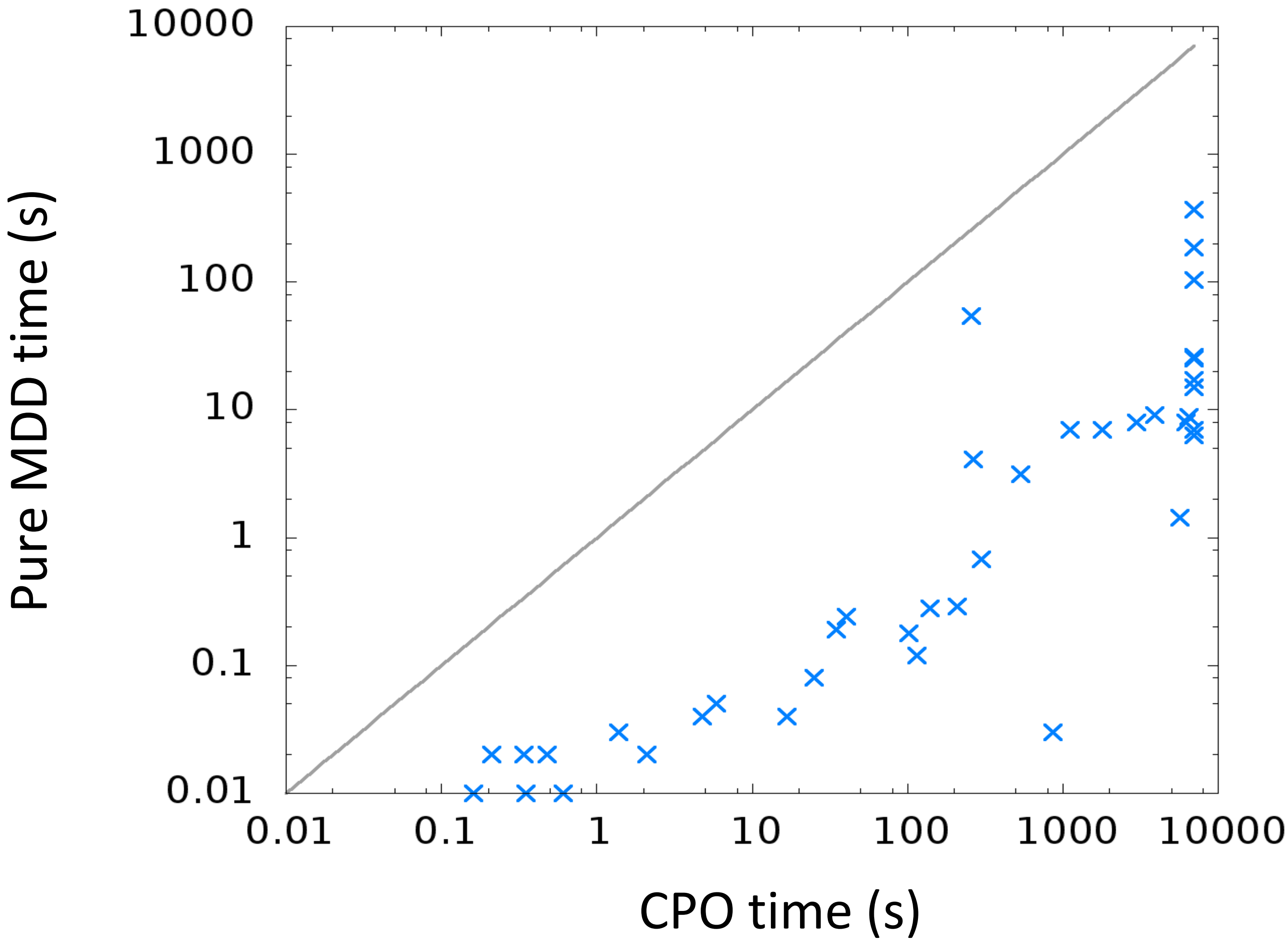
Importance of setup times



Importance of setup times
(increasing average length of setup times)

Random instances
- 15 jobs
- lex search
- MDD width 16
- min makespan

TSP with Time Windows



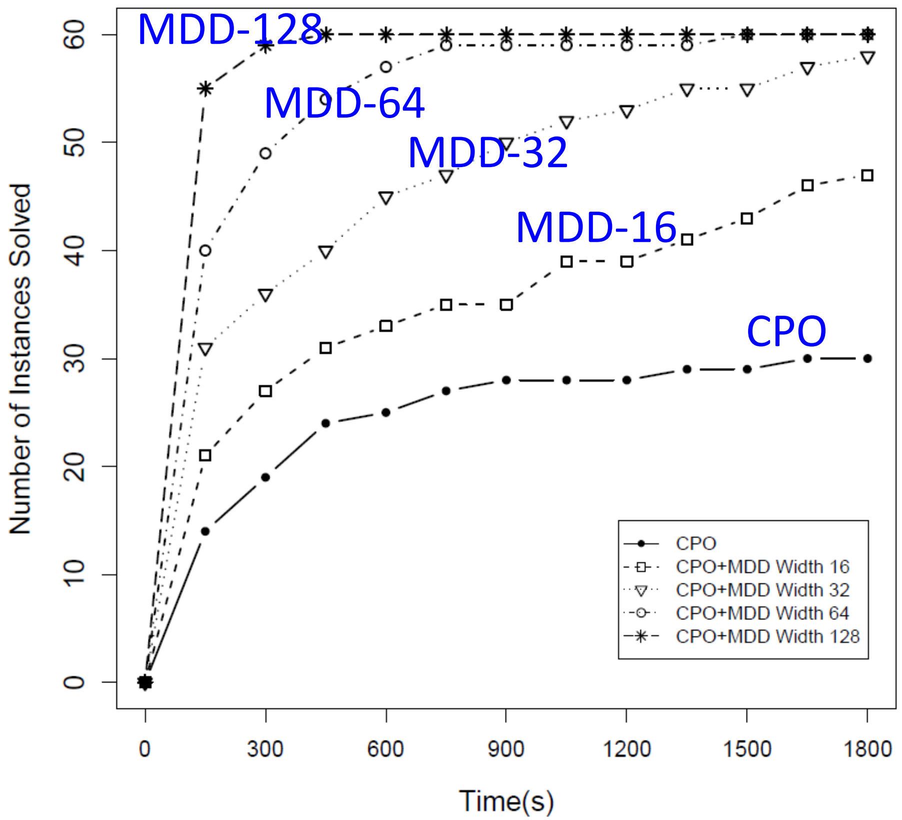
Dumas/Ascheuer instances
- 20-60 jobs
- lex search
- MDD width: 16

Minimize Total Tardiness

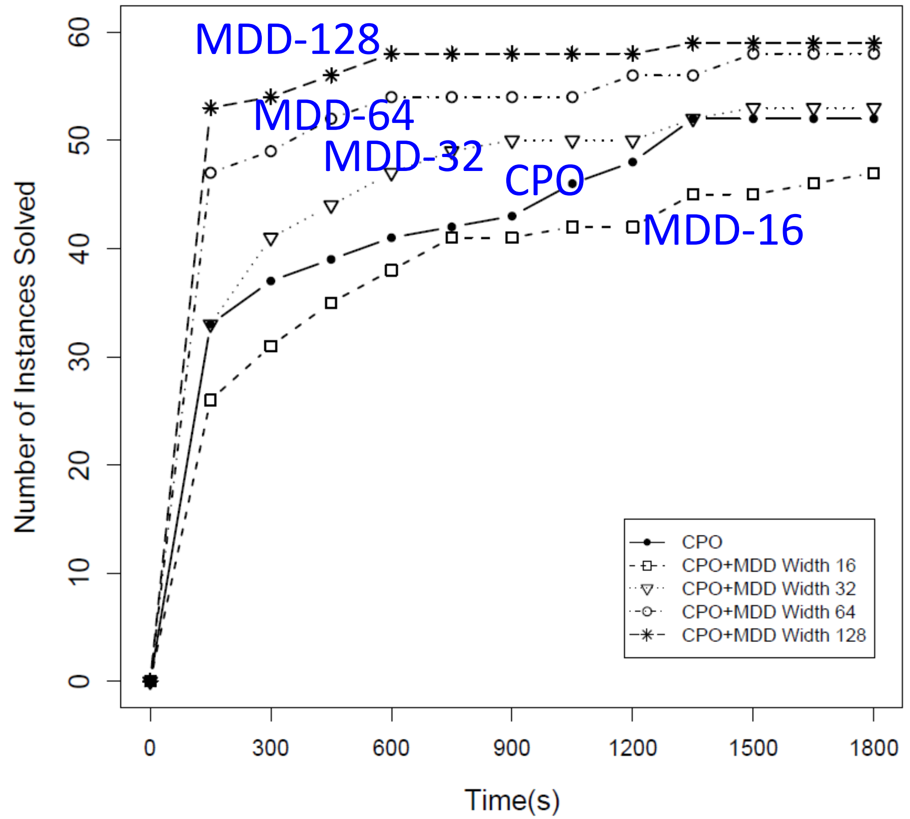
- Consider activity i with due date δ_i
  - Completion time of i: $c_i = s_i + p_i$
  - Tardiness of i: $\max\{0, c_i - \delta_i\}$
- Objective: minimize total (weighted) tardiness

- 120 test instances
  - 15 activities per instance
  - varying r_i, p_i, and δ_i, and tardiness weights
  - no side constraints, setup times (measure only impact of objective)
  - lexicographic search, time limit of 1,800s

Total Tardiness Results



total tardiness



total weighted tardiness

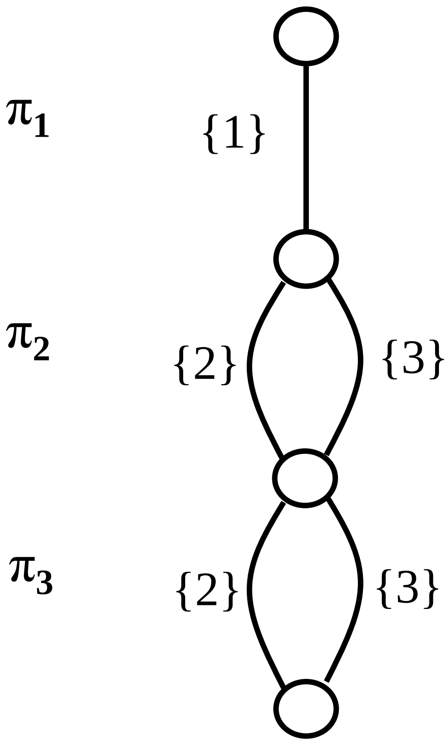
Sequential Ordering Problem (TSPLIB)

instance	vertices	bounds	CPO best	CPO time (s)	CPO+MDD, width 2048 best	CPO+MDD, width 2048 time (s)
br17.10	17	55	55	0.01	55	4.98
br17.12	17	55	55	0.01	55	4.56
ESC07	7	2125	2125	0.01	2125	0.07
ESC25	25	1681	1681	TL	1681	48.42
p43.1	43	28140	28205	TL	28140	287.57
p43.2	43	[28175, 28480]	28545	TL	**28480**	**279.18** *
p43.3	43	[28366, 28835]	28930	TL	**28835**	**177.29** *
p43.4	43	83005	83615	TL	83005	88.45
ry48p.1	48	[15220, 15805]	18209	TL	16561	TL
ry48p.2	48	[15524, 16666]	18649	TL	17680	TL
ry48p.3	48	[18156, 19894]	23268	TL	22311	TL
ry48p.4	48	[29967, 31446]	34502	TL	**31446**	**96.91** *
ft53.1	53	[7438, 7531]	9716	TL	9216	TL
ft53.2	53	[7630, 8026]	11669	TL	11484	TL
ft53.3	53	[9473, 10262]	12343	TL	11937	TL
ft53.4	53	14425	16018	TL	14425	120.79

* solved for the first time

Extension: Lagrangian bounds

- Observation: MDD bounds can be very loose



$$\sum_{i=1}^{n} (\pi_i = j) = 1 \quad \forall j$$

Main cause: repetition of activities

Proposed remedy:

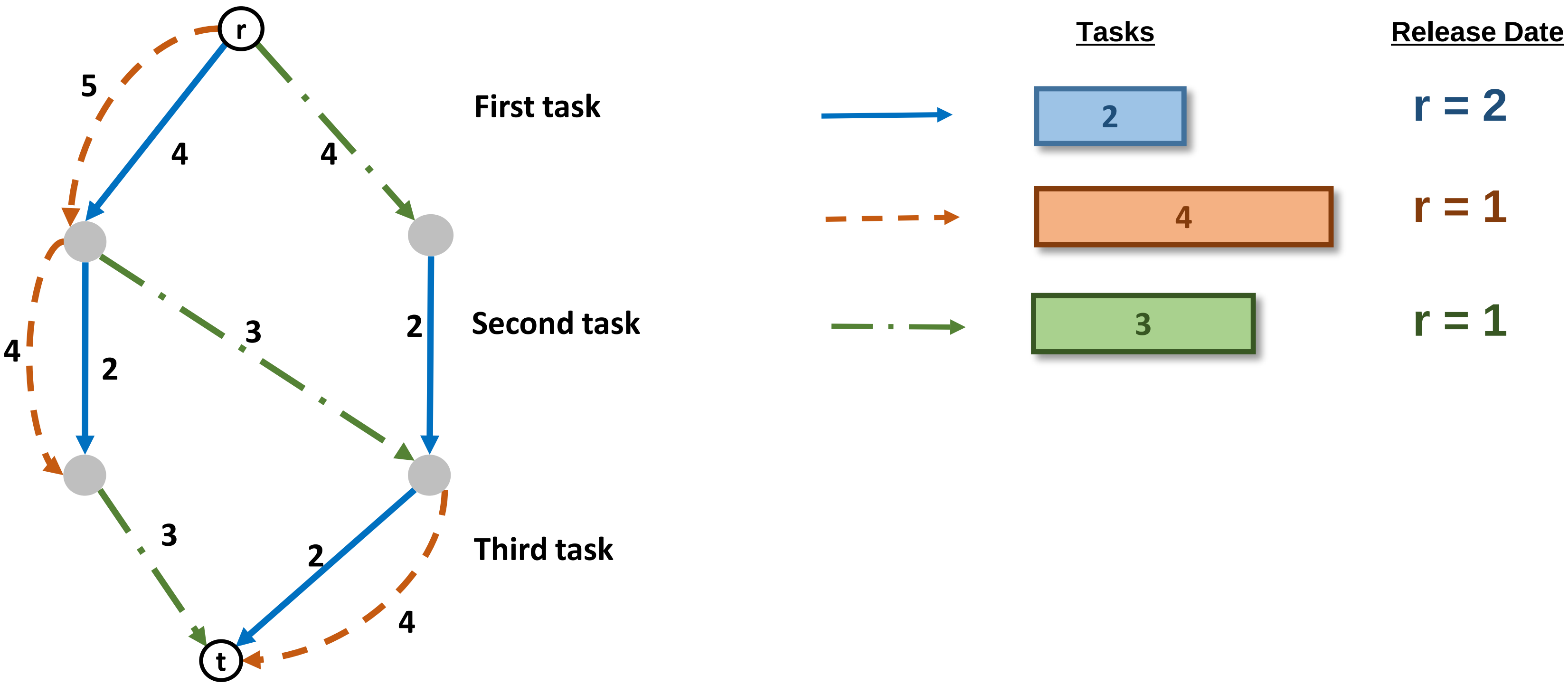
- add Lagrangian relaxation
- penalize repeated activities

$$\min z + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{n} (\pi_i = j) - 1 \right)$$

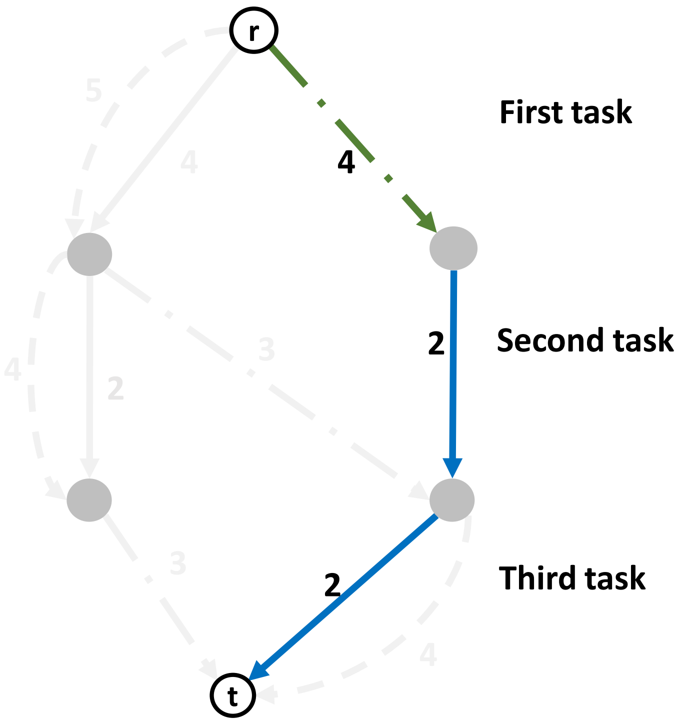
$$= z + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j (\pi_i = j) - \sum_{j=1}^{n} \lambda_j$$

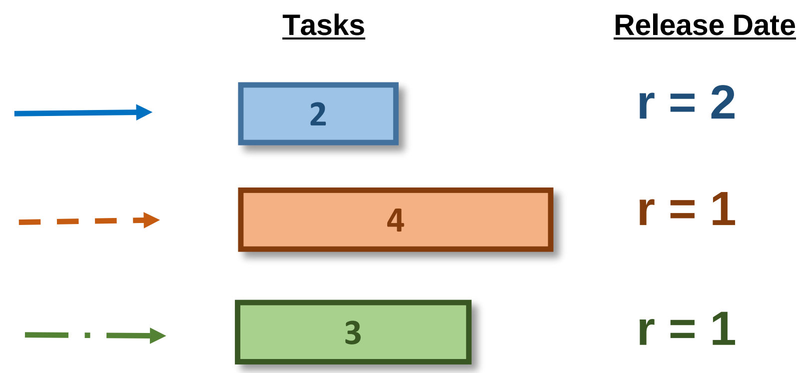
- Shortest path with updated weights

Example: Relaxed Decision Diagram



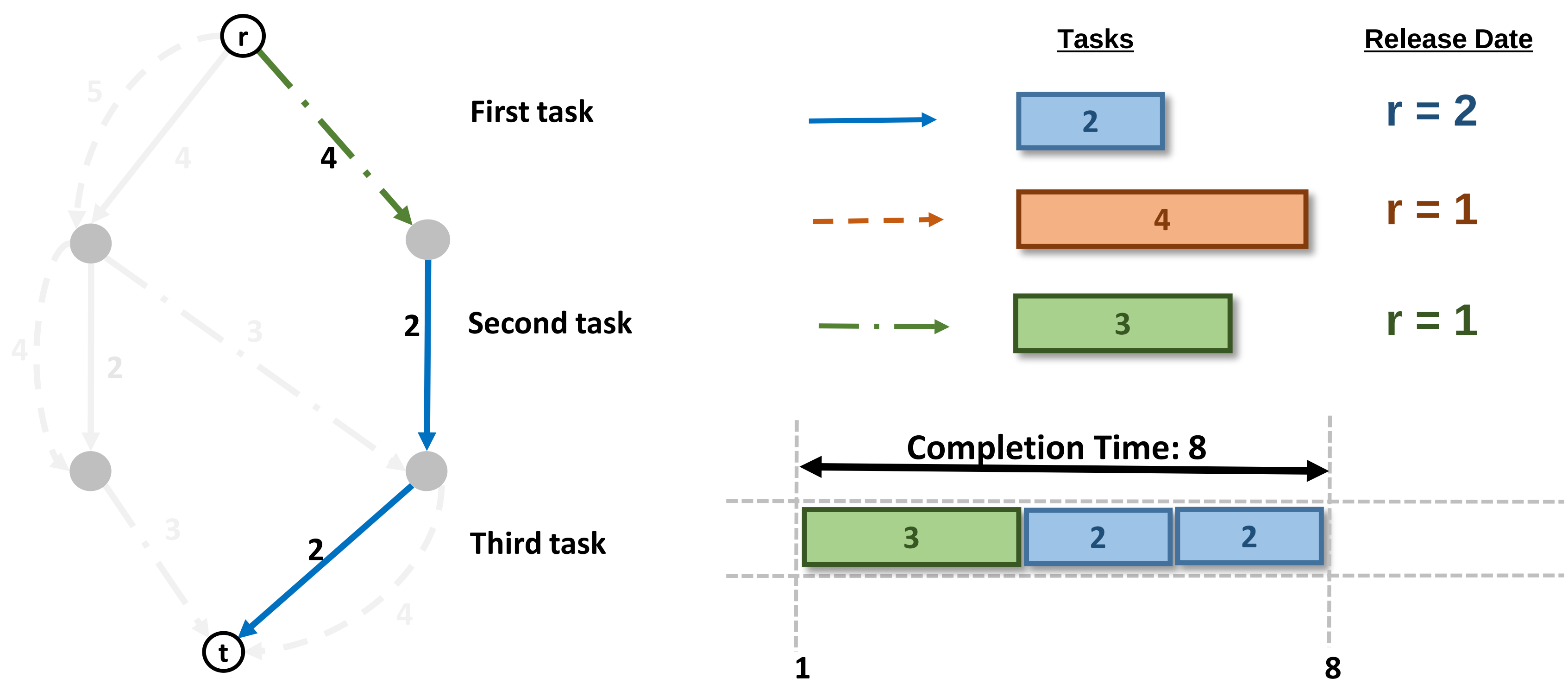
Shortest Path: Lower Bound



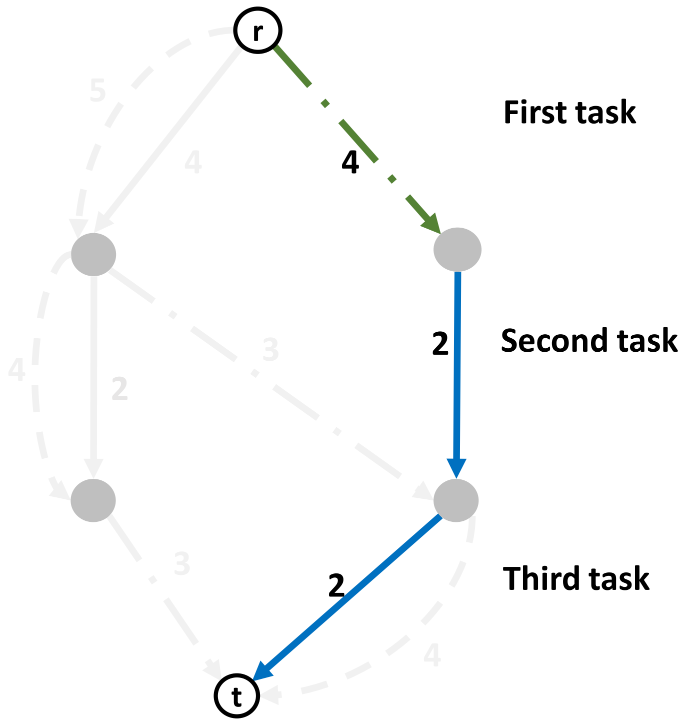


- **Shortest path**
  - Length: **Lower bound** on the optimal solution value

Shortest Path: Lower Bound



Issues



- Solutions of a relaxed DD may violate several constraints of the problem

- **Violation**: "All tasks performed once"

$$\sum_{e|v(e)=i} x_e = 1 \quad \text{for all tasks i}$$

Remedy: Lagrangian Relaxation

[Bergman et al., 2015]

min z = shortest path

s.t. $\sum_{e|v(e)=i} x_e = 1$, for all tasks i → Lagrangian multipliers λ_i

(+other problem constraints)

min z = shortest path + $\sum_i \lambda_i (1 - \sum_{e|v(e)=i} x_e)$

s.t. (other problem constraints)

This is done by updating shortest path weights!

General Approach

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- We penalize infeasible solutions in a relaxed DD:

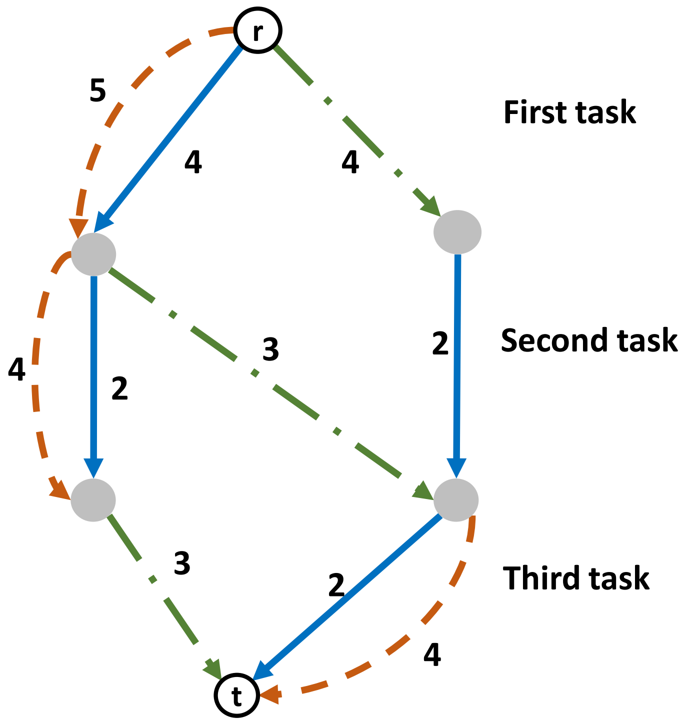
  Any separable constraint of the form

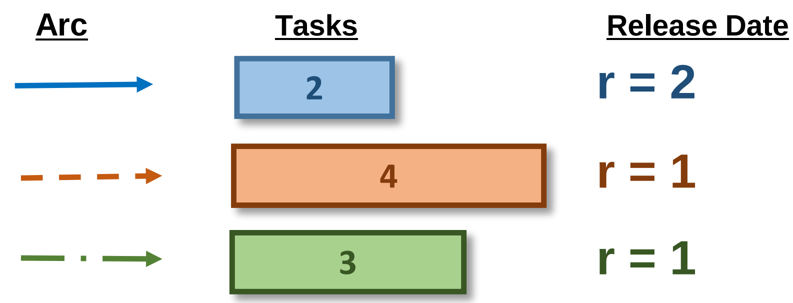
  $$f_1(x_1) + f_2(x_2) + \dots + f_n(x_n) \leq c$$

  that **must** be satisfied by solutions of an MDD can be dualized

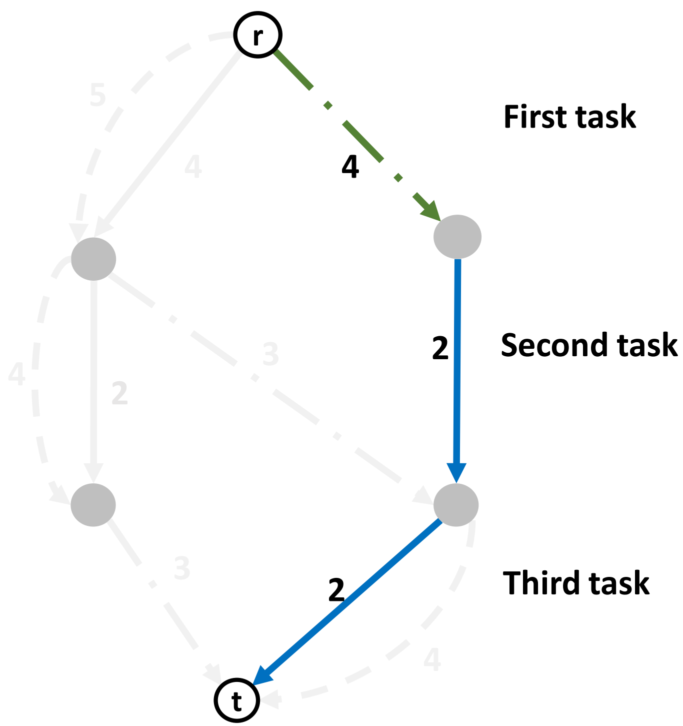
- We need only to focus on the **shortest path solution**
  - Identify a violated constraint and penalize
  - Systematic way directly adapted from LP
  - Shortest paths are **very fast** to compute

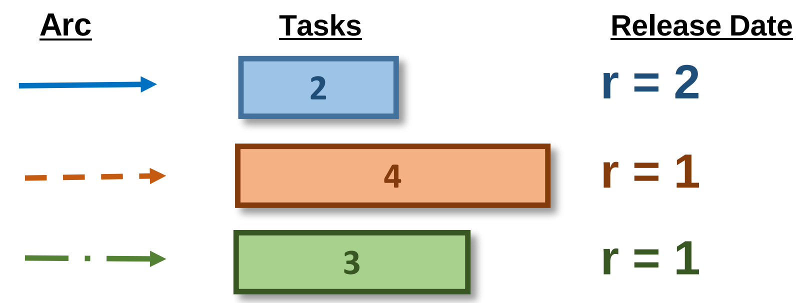
Improving Relaxed Decision Diagram

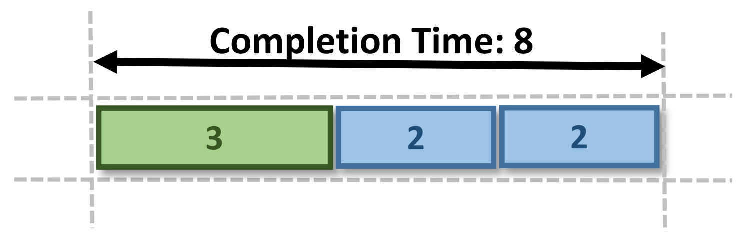




Improving Relaxed Decision Diagram



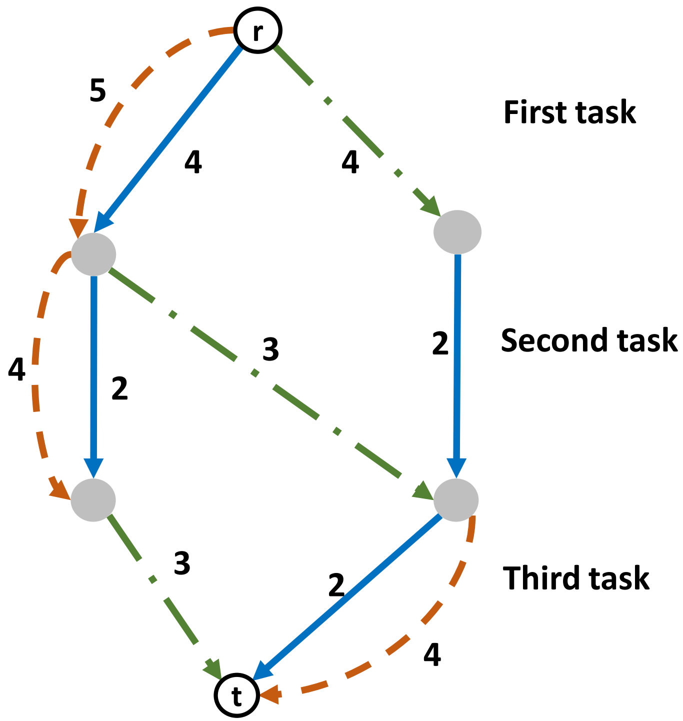


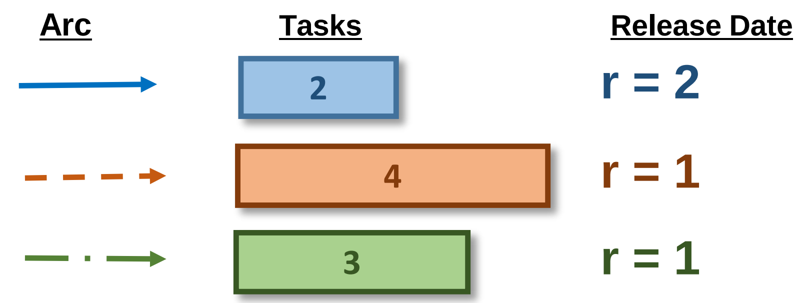


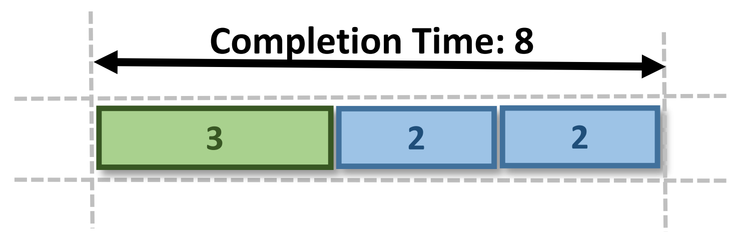
Penalization:

- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Improving Relaxed Decision Diagram



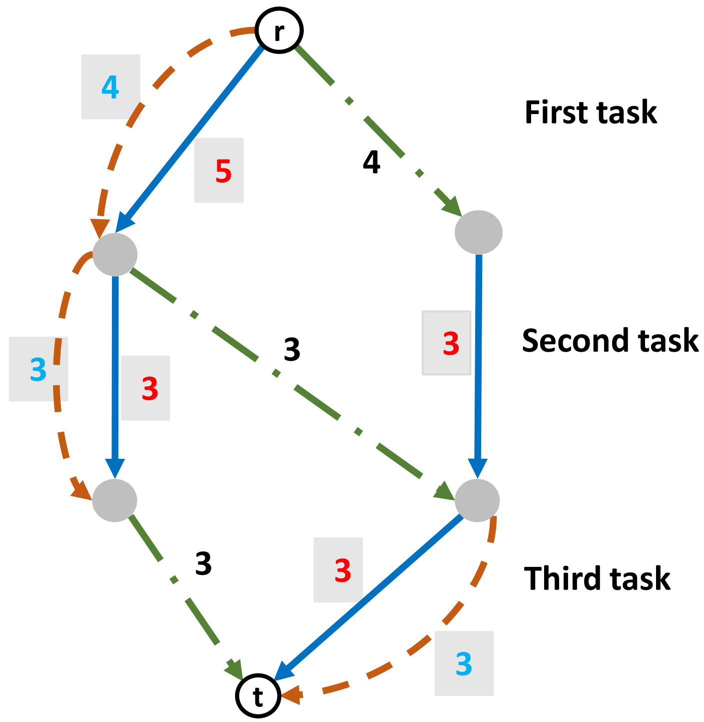


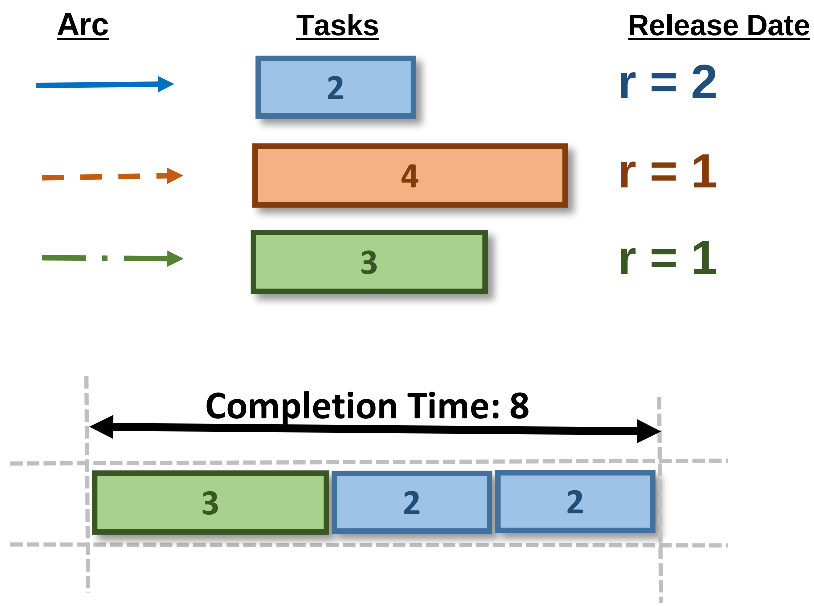


Penalization:

- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Improving Relaxed Decision Diagram

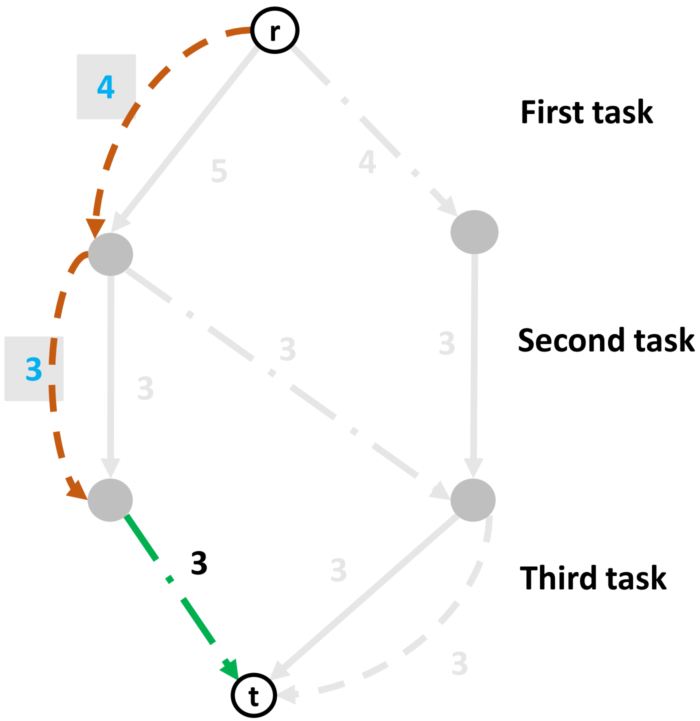


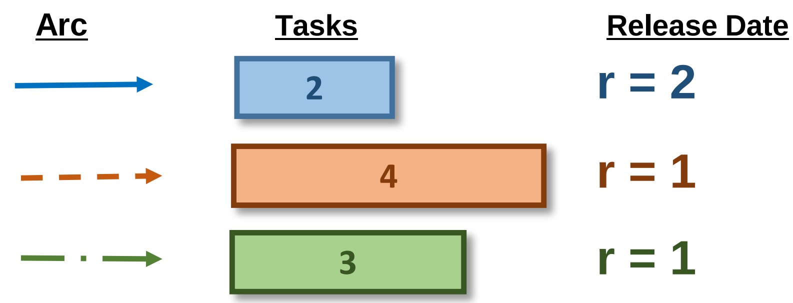


Penalization:

- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

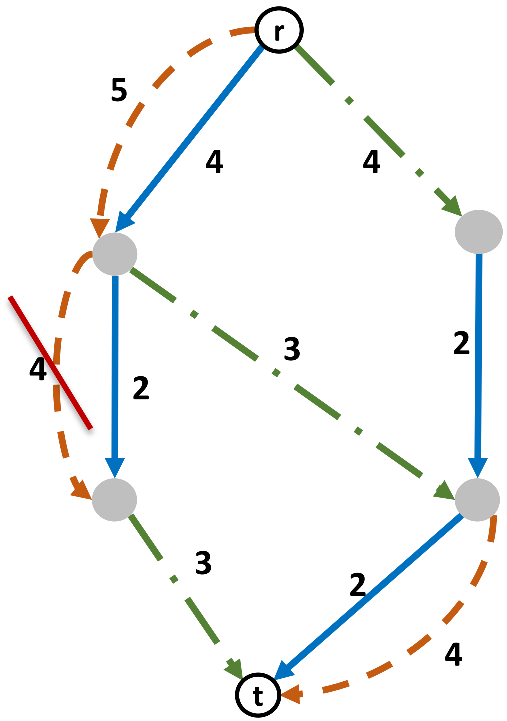
Improving Relaxed Decision Diagram





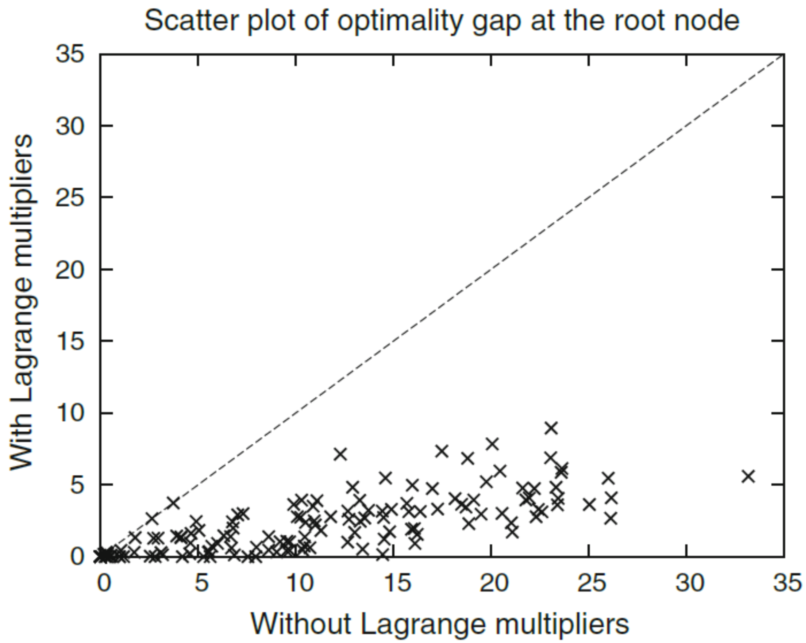
- **New shortest path: 10**
  - Guaranteed to be **a valid lower bound** for any penalties

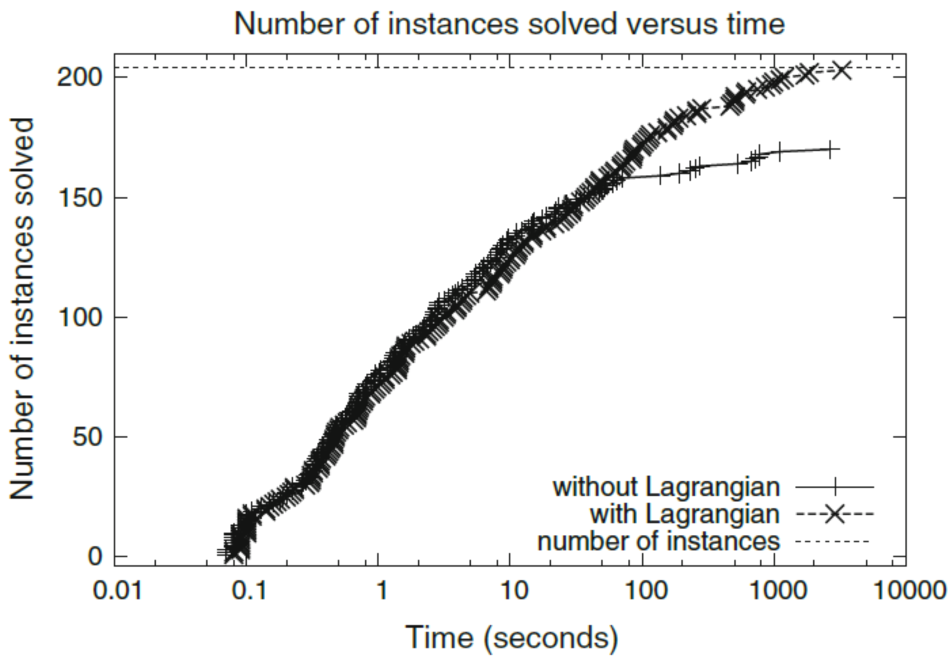
Cost-Based Filtering



- If minimum solution value through an arc exceeds max(D(z)) then arc can be deleted
- Suppose a solution of value 10 is known
- MDD filtering extends to Lagrangian weights: More filtering possible

Impact on TSP with Time Windows





TSPTW instances

(*Constraints*, 2015)

State-Dependent Costs

- Kinable, Cire and v.H. Hybrid Optimization for Time-Dependent Sequencing. *Under Review*.

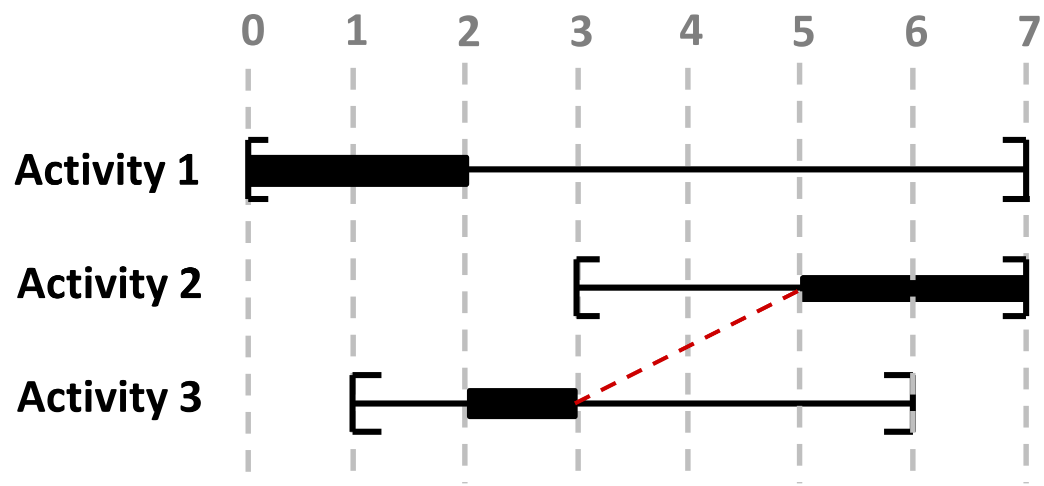
Context and Motivation

Carnegie Mellon Tepper School of Business

- Time-dependent sequencing
  - machine scheduling, routing
- Challenging problem
  - best results so far use dedicated methods
  - not easy to extend with side constraints
- Utilize constraint programming framework?
  - strengthened constraint propagation with MDDs
  - improved bounds via additive bounding with LP
  - evaluate on TD-TSP and TD-SOP

Time-Dependent Sequencing

- Activities
  - processing time p_i
  - released date r_i
  - deadline d_i



- Resource
  - non-preemptive
  - process one activity at a time
  - sequence-dependent setup times: also depend on position!

    $\delta_{i,j}^{t}$ = setup time between i and j if i is at position t

Constraint Programming Model

- Variables π_i : label of i^th activity in the sequence
  L_i : position of activity i in the sequence

$$
\begin{array}{lll}
\min & \sum_{i=0}^{n} \delta^i_{\pi_i,\pi_{i+1}} & \\
\text{s.t.} & \text{AllDiff}(\pi_1,\ldots,\pi_n) & \\
 & L_{\pi_i} = i & \forall i = 1,\ldots,n \\
 & L_i < L_j & \forall (i \ll j) \in P \\
 & L_i \in \{1,\ldots,n\} & \forall i = 1,\ldots,n \\
 & \pi_i \in \{1,\ldots,n\} & \forall i = 1,\ldots,n
\end{array}
$$

- Weak model: objective and AllDiff are decoupled

MDD-based propagation

Update MDD propagation algorithms:

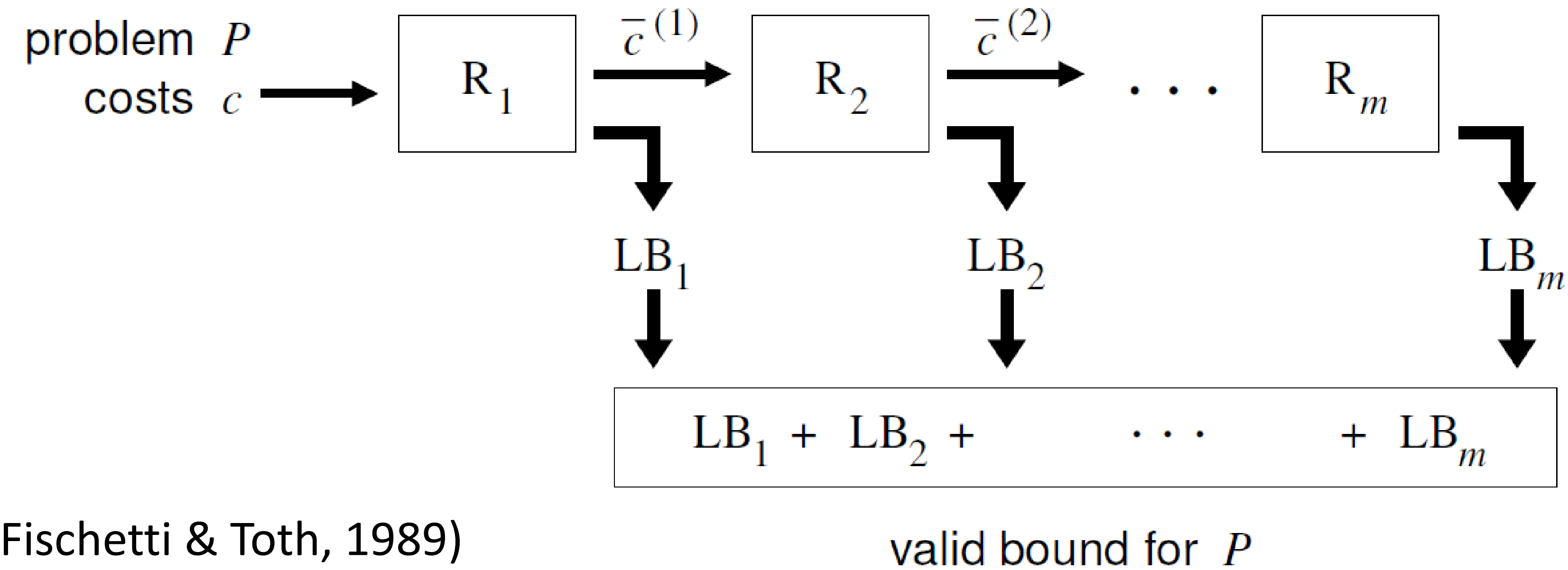
- *Alldifferent* for the permutation structure
  - unchanged
- Precedence relations
  - unchanged
- Earliest start time and latest end time
  - adapt rule: $\delta_{i,j}$ becomes $\delta^{t}_{i,j}$
- Objective
  - minimize sum of setup times

Updated CP Model

$$
\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & \text{AllDiff}(\pi_1, \ldots, \pi_n) \\
& \text{MDDconstr}(\pi_1, \ldots, \pi_n, W, z, \delta^t, P) \\
& L_{\pi_i} = i && \forall i = 1, \ldots, n \\
& L_i < L_j && \forall (i \ll j) \in P \\
& L_i \in \{1, \ldots, n\} && \forall i = 1, \ldots, n \\
& \pi_i \in \{1, \ldots, n\} && \forall i = 1, \ldots, n \\
& z \in \{0, \ldots, \infty\}
\end{aligned}
$$

Stronger model: objective handled within MDD constraint

Additive Bounding

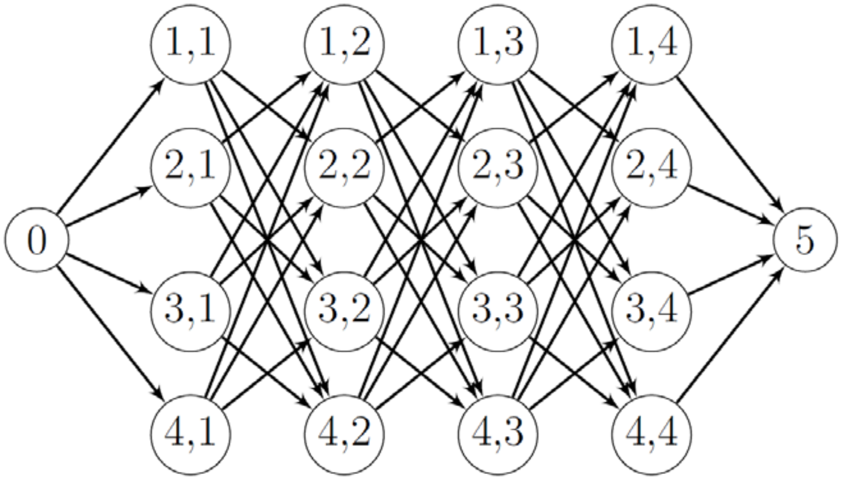


(Fischetti & Toth, 1989)

Add LP reduced costs to MDD relaxation

- Continuous LP relaxation ‘discretized’ through MDD
- Stronger bounds
- Improved cost-based filtering

MIP and LP relaxation



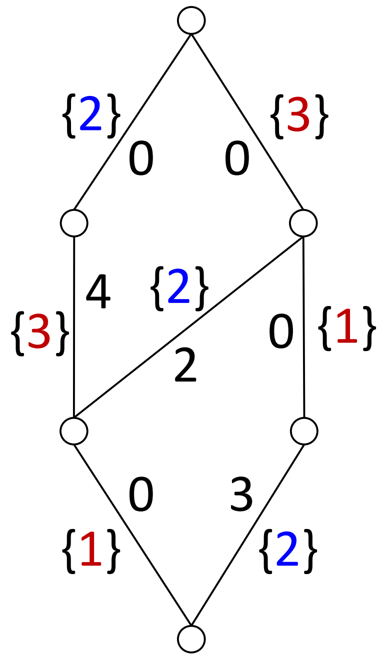
- Time-space network model (Picard & Queyranne, 1978)
- Variables

$$x_{i,j}^t = \begin{cases} 1 & \text{if i is performed at t and followed by j} \\ 0 & \text{otherwise} \end{cases}$$

- Constraints: flow conservation; perform each activity
- Valid inequalities: subtour and 4-cycle elimination

Embedding reduced costs in MDD

- State information at each node *i*
  - shortest path from root to *i* with respect to $\overline{c}^{t}_{i,j}$
  - root node initialized with LP objective value
- Since MDD is relaxation, shortest path is valid bound
  - filter edges that do not participate in improving shortest path

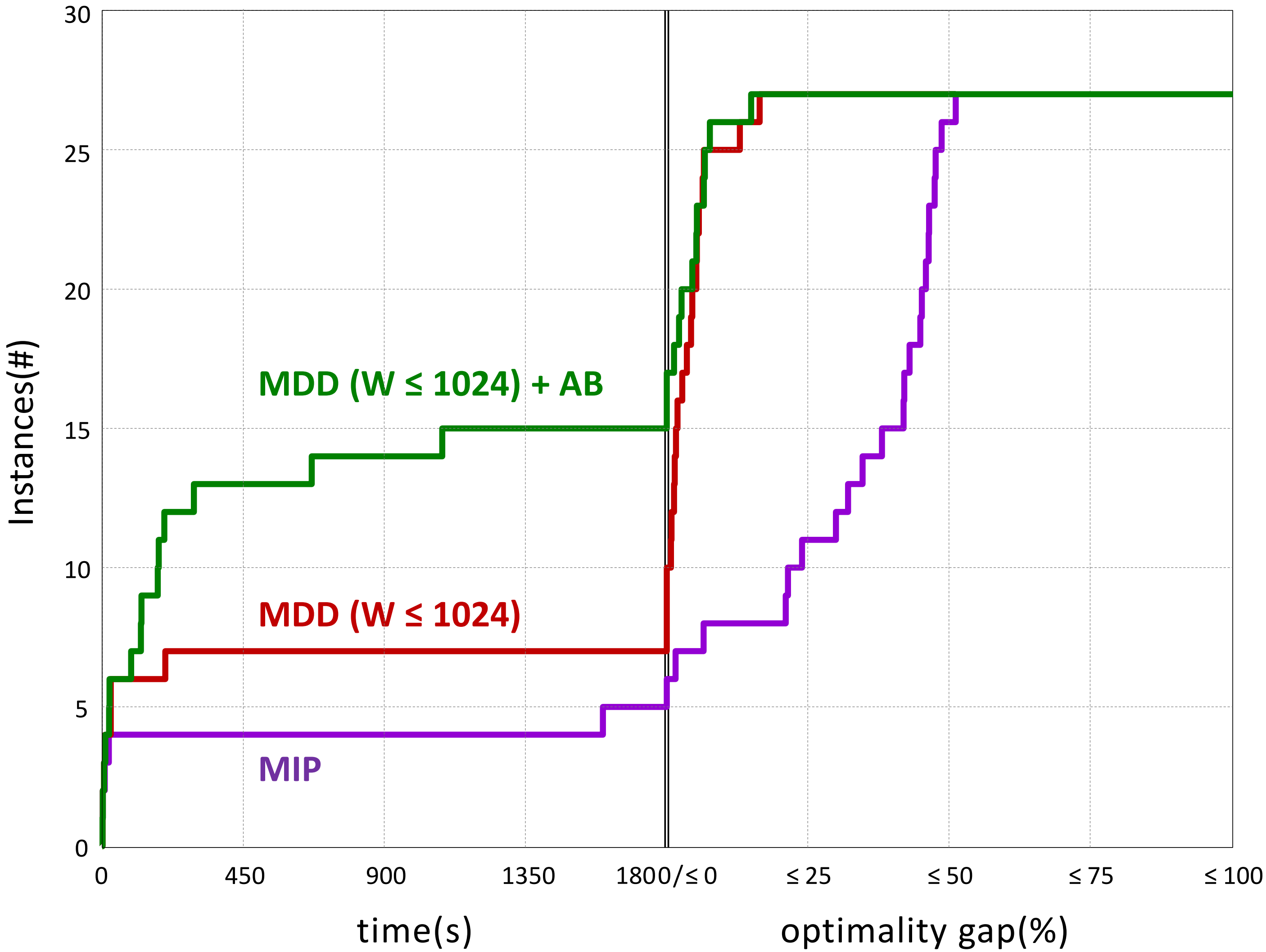


- MDD maintains both the original objective and this new 'additive bound' constraint

Experiments

- Time-dependent TSP and SOP benchmarks
  - 38 instances from TSPLIB (14-107 jobs)
  - $\delta^t_{i,j}$ = (n–t)*$\delta_{i,j}$ [Abeledo et al. 2013]
- Time limit: 30 minutes
- MDD added to IBM ILOG CP Optimizer 12.4
  - maximum width 1024
- MIP model (IBM ILOG CPLEX 12.4)
  - state-space integer program
  - subtour and 4-cycle elimination constraints
  - LP relaxation takes several hours for ≥90 vertices

Results on Time-dependent TSP



Note: Dedicated branch, price and cut algorithm (Abeledo et al., 2013) solves more TD-TSP instances optimally

Results on Time-dependent SOP

	#Solved
MIP	6/30
Pure CP	5/30
CP + MDD + Additive Bounding	10/30

On average, additive MDD+LP bound improves

- LP root node bound by 51.41%
- MDD root node bound by 9.54%

Conclusion

- MDD propagation natural generalization of domain propagation
  - Strength of MDD relaxation can be controlled by the width
  - Huge reduction in solution time is possible
- For sequencing/disjunctive scheduling problems
  - MDD can handle all side constraints and objectives from existing CP scheduling systems
  - Polynomial cases (e.g., Balas variant)
  - MDD propagation algorithms (alldifferent, time windows, …)
  - Extraction of precedence constraints from MDD
  - Can be enriched with math programming relaxations
  - Great addition to constraint-based systems