Carnegie Mellon University
Tepper School of Business

Decision Diagram Relaxations for Vehicle Routing

Willem-Jan van Hoeve
Carnegie Mellon University

Includes joint work with Andre Cire, Joris Kinable, Ziye Tang, and Anthony Karahalios

Overview

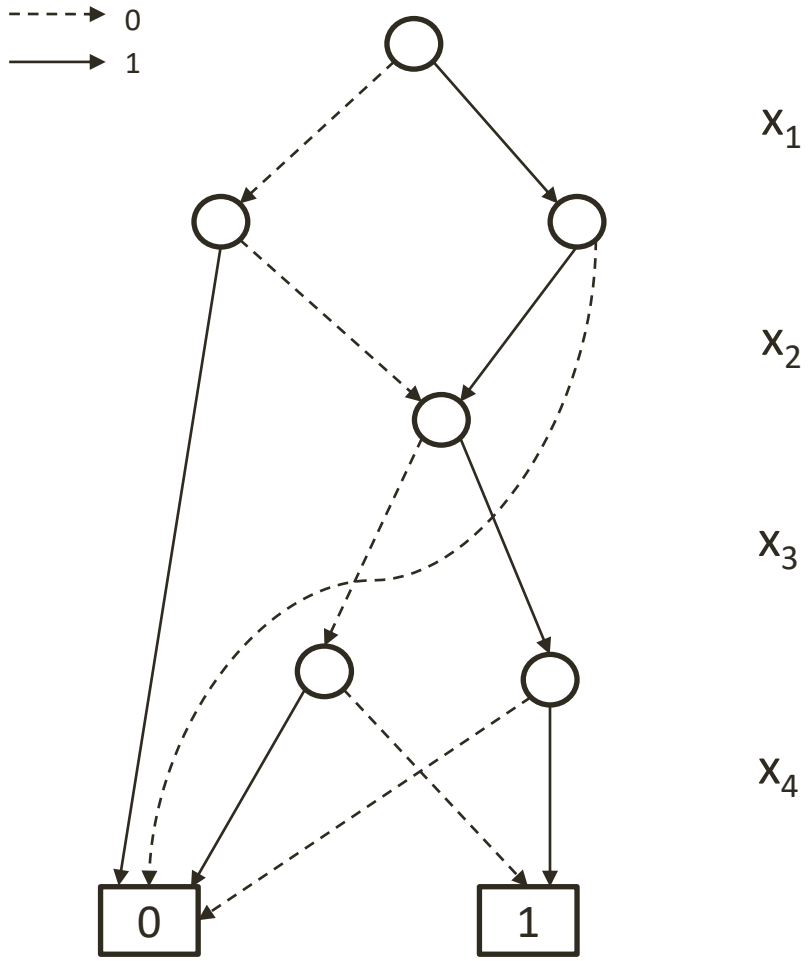
- Introduction to Decision Diagrams
- TSP
  - special case of single machine scheduling problem
  - embed DD relaxations in **constraint programming** solver
  - improve DD bound with Lagrangian relaxation and Additive Bounding
- Vehicle Routing
  - apply ‘column elimination’ to iteratively strengthen relaxed DD
  - combine with **linear programming** solver

Decision Diagrams

- Graphical representation of **Boolean functions**

$$f(x) = \left(x_1 \Leftrightarrow x_2\right) \wedge \left(x_3 \Leftrightarrow x_4\right)$$

x_1	x_2	x_3	x_4	f(x)
0	0	0	0	**1**
0	0	0	1	**0**
0	1	1	0	**0**
0	0	1	1	**1**
…	…	…	…	…

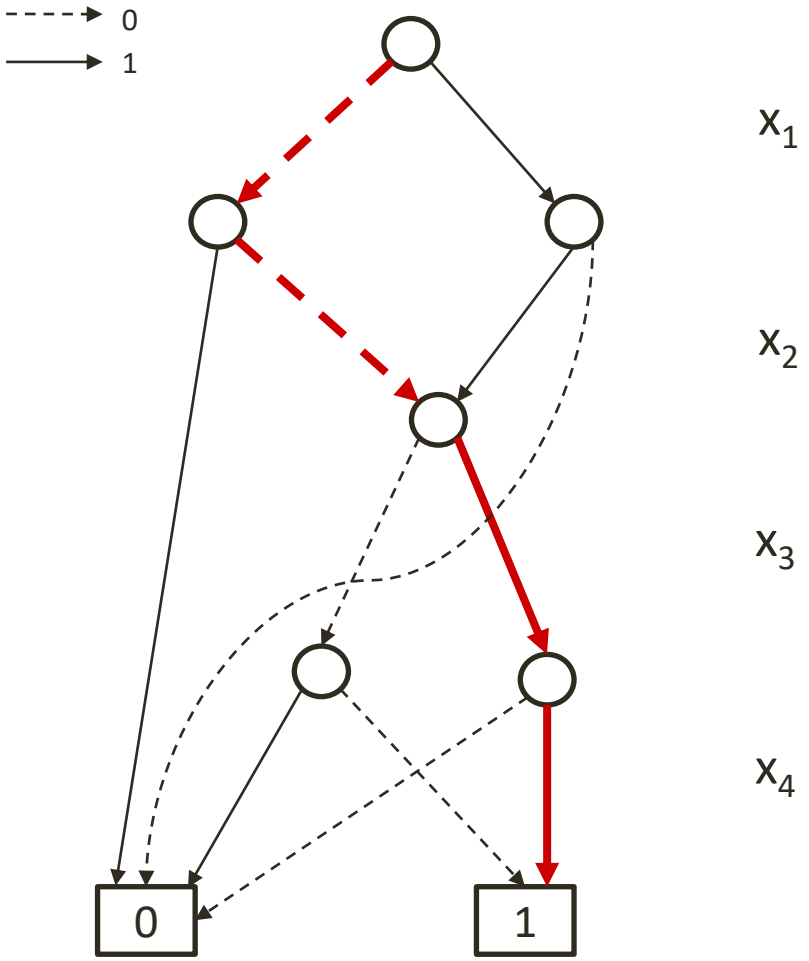


Decision Diagrams

- Graphical representation of **Boolean functions**

$$f(x) = \left(x_1 \Leftrightarrow x_2\right) \wedge \left(x_3 \Leftrightarrow x_4\right)$$

x_1	x_2	x_3	x_4	f(x)
0	0	0	0	**1**
0	0	0	1	**0**
0	1	1	0	**0**
0	**0**	**1**	**1**	**1**
…	…	…	…	…

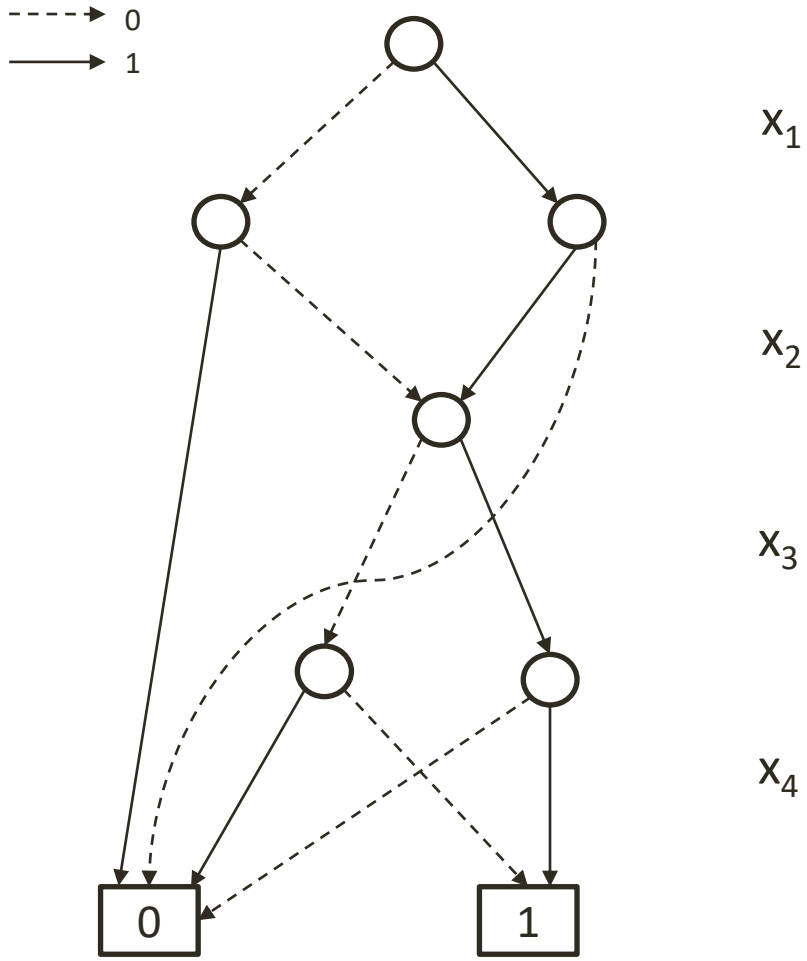


Decision Diagrams

- Graphical representation of **Boolean functions**

$$f(x) = \left(x_1 \Leftrightarrow x_2\right) \wedge \left(x_3 \Leftrightarrow x_4\right)$$

- BDD: binary decision diagram
- MDD: multi-valued decision diagram

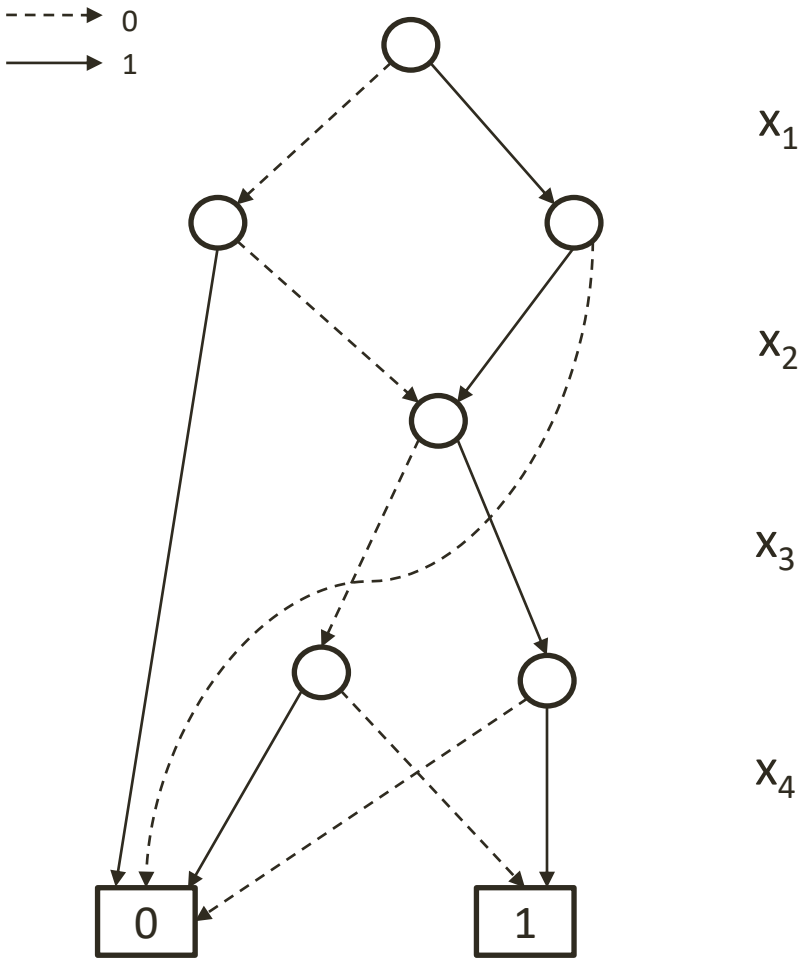


Decision Diagrams: Optimization View

- Graphical representation of **Boolean functions**

$$f(x) = \left(x_1 \Leftrightarrow x_2\right) \wedge \left(x_3 \Leftrightarrow x_4\right)$$

- Optimization perspective:
  - literals → variables
  - arcs → assignments
  - paths → solutions



Decision Diagrams: Optimization View

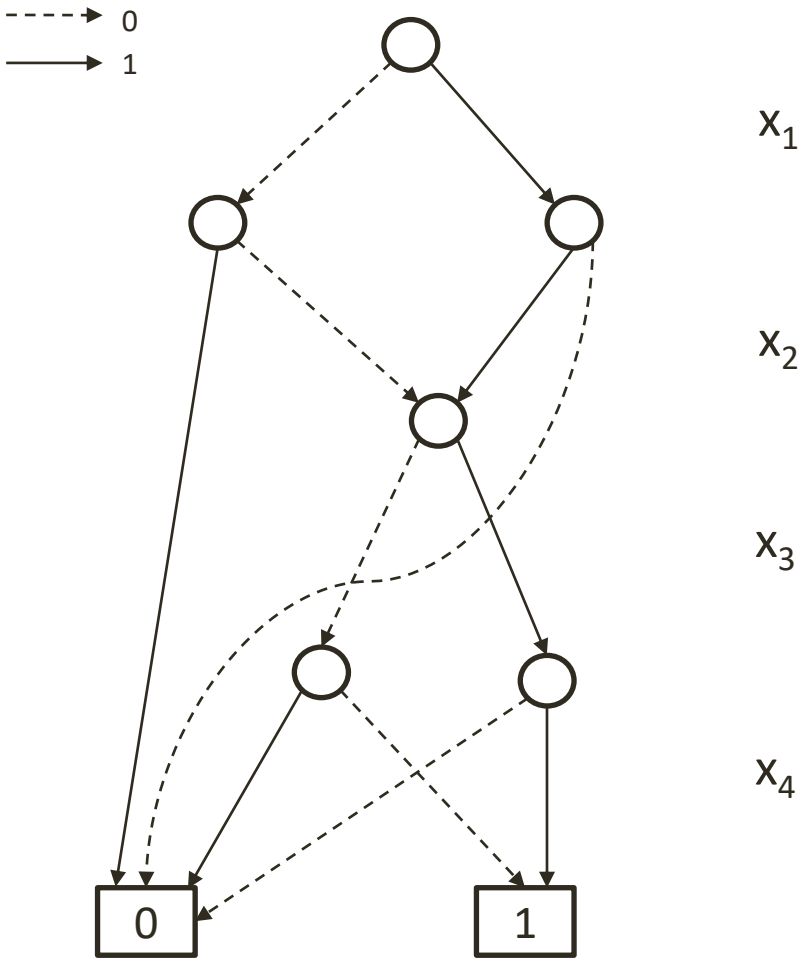
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$



Decision Diagrams: Optimization View

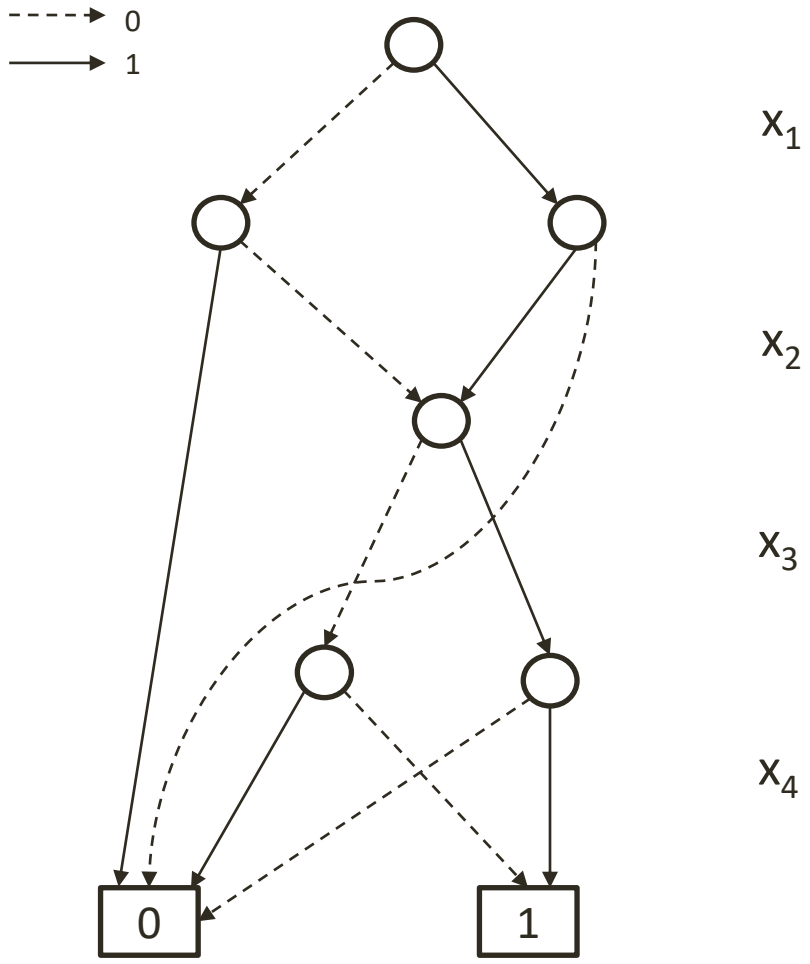
$$\max\ 2x_1 + x_2 - 4x_3 + x_4$$

subject to

$$x_1 - x_2 = 0$$

$$x_3 - x_4 = 0$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$



Decision Diagrams: Optimization View

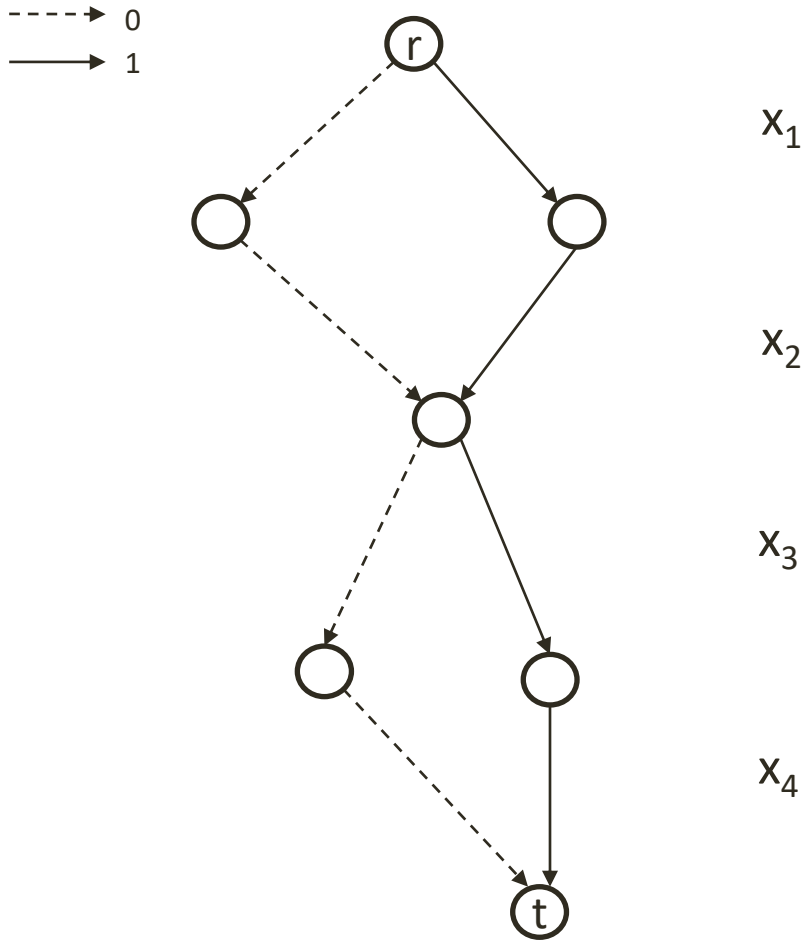
$$\max\ 2x_1 + x_2 - 4x_3 + x_4$$

subject to

$$x_1 - x_2 = 0$$

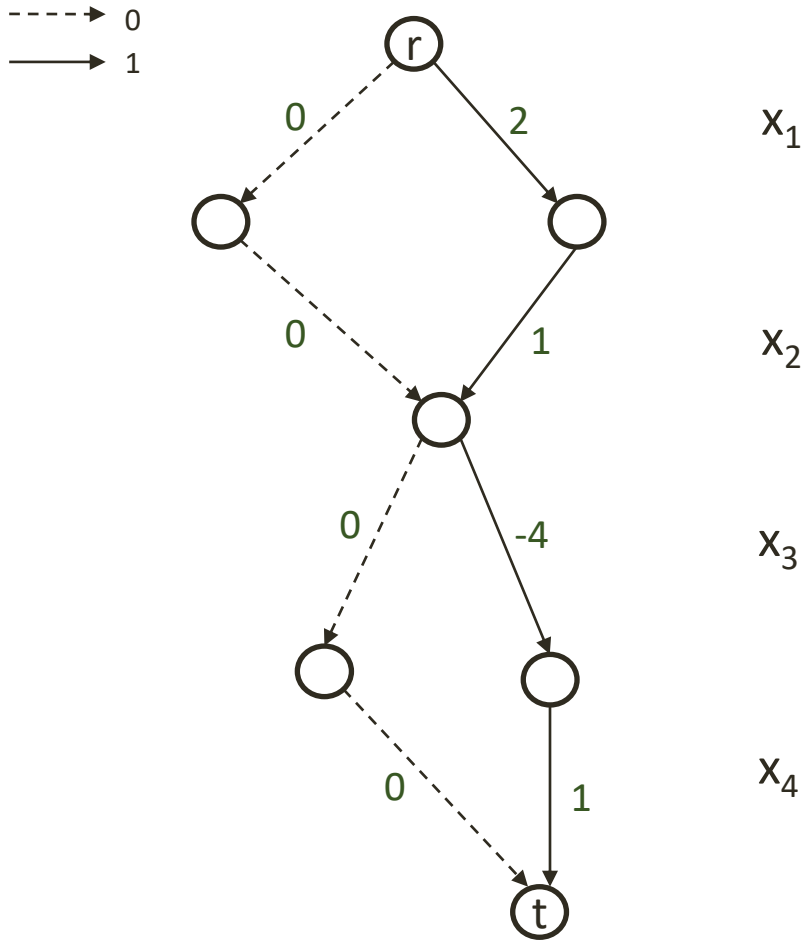
$$x_3 - x_4 = 0$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$



Decision Diagrams: Optimization View

$$\max \; 2x_1 + x_2 - 4x_3 + x_4$$
$$\text{subject to}$$
$$x_1 - x_2 = 0$$
$$x_3 - x_4 = 0$$
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$



- Maximizing a linear (or separable) function:
  - Arc lengths: contribution to the objective
  - Longest path: optimal solution

  (can also handle nonlinear functions)

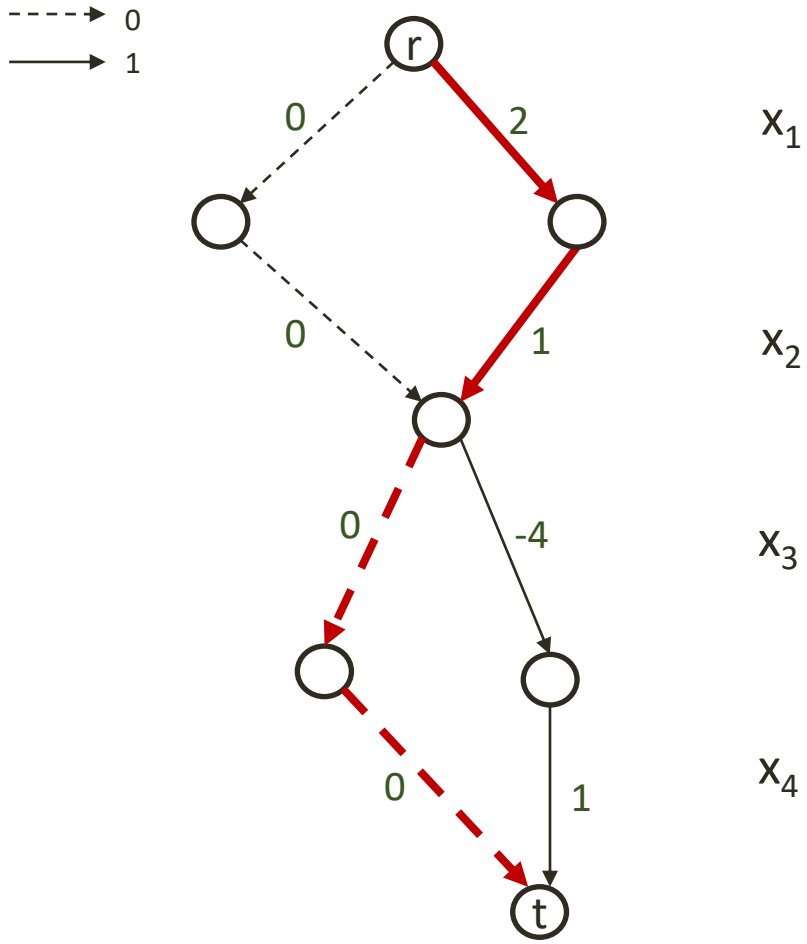
Decision Diagrams: Optimization View

max $2x_1 + x_2 - 4x_3 + x_4$
subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$



- Maximizing a linear (or separable) function:
  - Arc lengths: contribution to the objective
  - Longest path: optimal solution

  (can also handle nonlinear functions)

Compiling DDs: Top-down or iterative refinement

Top-down

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

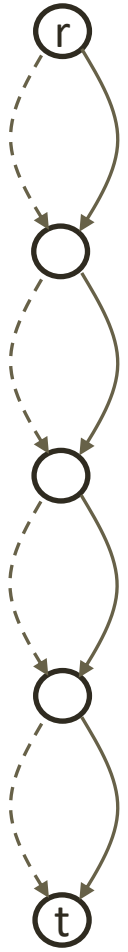
x_1

x_2

x_3

x_4

Iterative Refinement



x_1

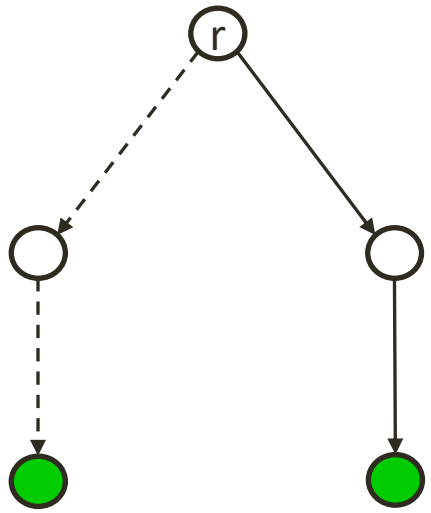
x_2

x_3

x_4

Compiling DDs: Top-down or iterative refinement

Top-down



x_1

x_2

x_3

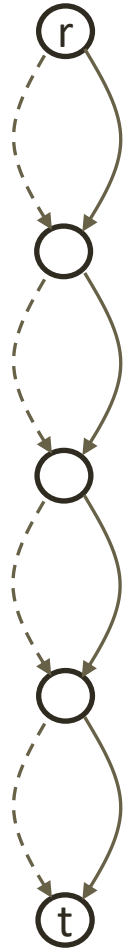
x_4

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

Iterative Refinement



x_1

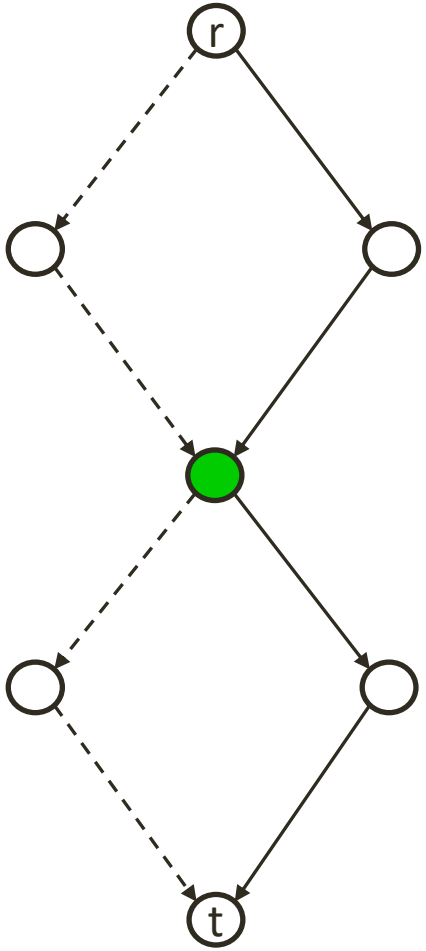
x_2

x_3

x_4

Compiling DDs: Top-down or iterative refinement

Top-down



x_1

x_2

x_3

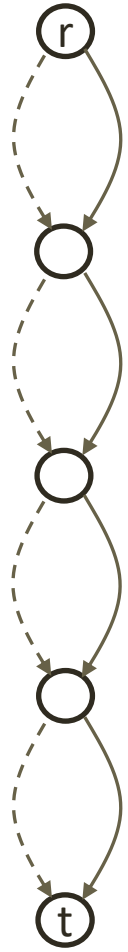
x_4

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

Iterative Refinement



x_1

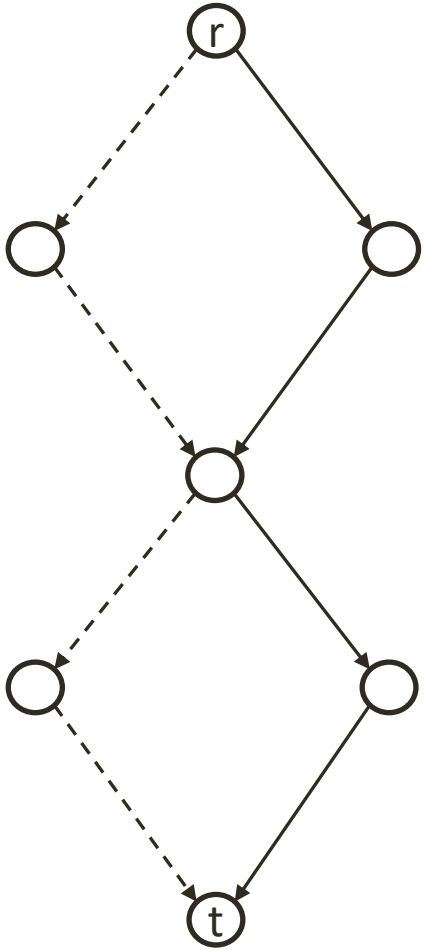
x_2

x_3

x_4

Compiling DDs: Top-down or iterative refinement

Top-down



x_1

x_2

x_3

x_4

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

Iterative Refinement

x_1

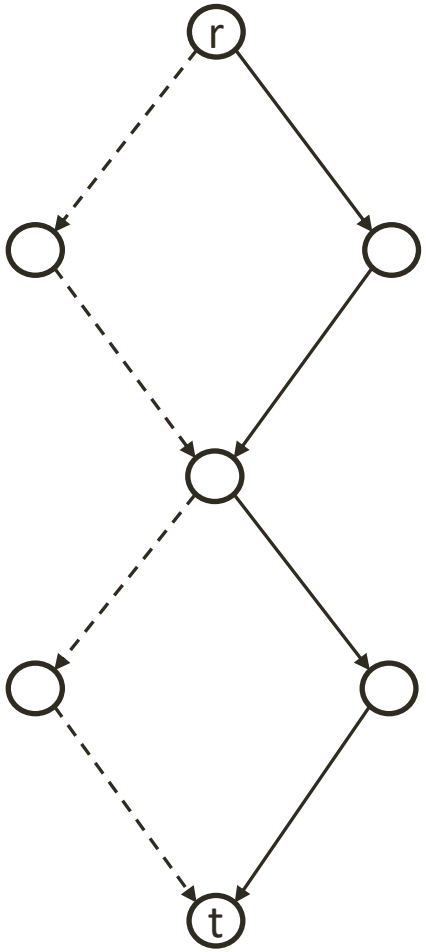
x_2

x_3

x_4

Compiling DDs: Top-down or iterative refinement

Top-down



x_1

x_2

x_3

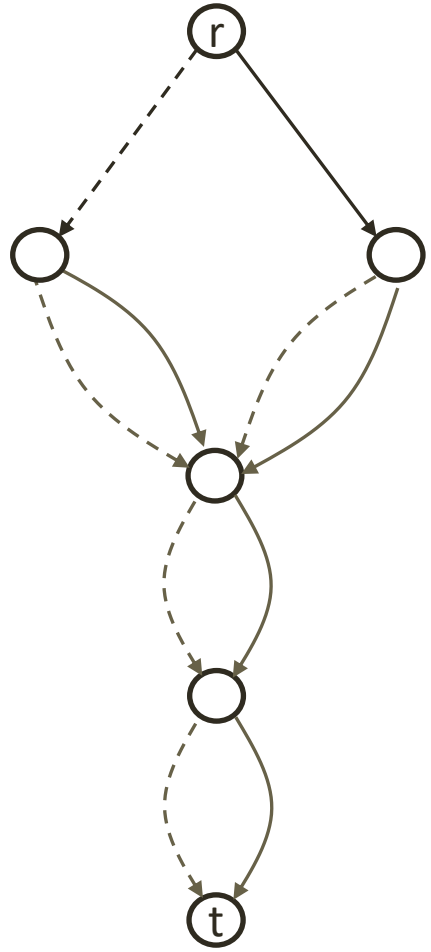
x_4

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

Iterative Refinement



x_1

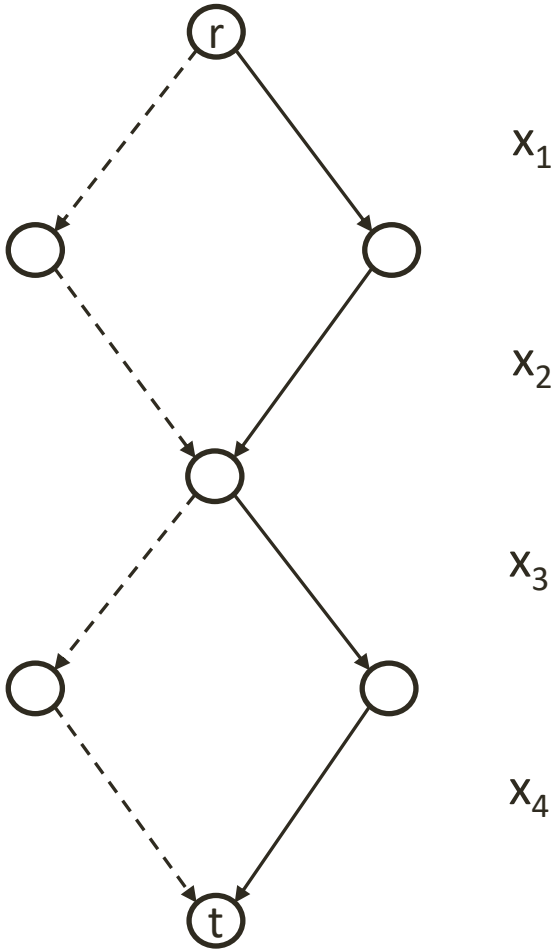
x_2

x_3

x_4

Compiling DDs: Top-down or Iterative Refinement

Top-down



x_1

x_2

x_3

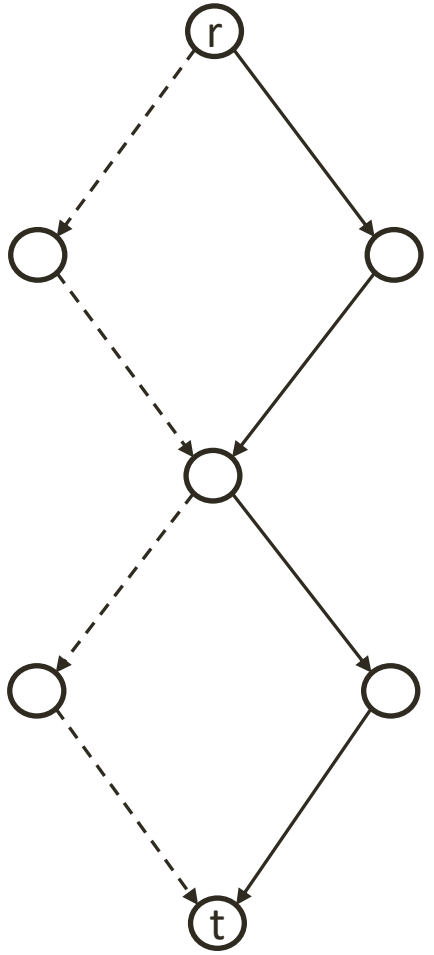
x_4

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

$x_1, x_2, x_3, x_4 \in \{0,1\}$

Iterative Refinement



x_1

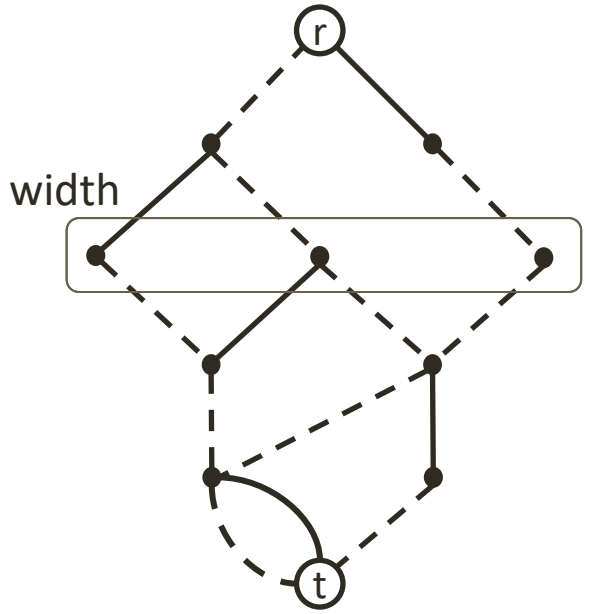
x_2

x_3

x_4

Relaxed Decision Diagrams: Polynomial Size

- Exponential size is handled by explicitly limiting the size (e.g., the width) of the diagram
- Non-equivalent nodes are merged
  - top-down compilation: need *node merging rule*
  - iterative refinement: stop when width is reached
- Requirement: no solution is lost
  - **over-approximation of the solution space**
  - provides *discrete relaxation*
  - strength is controlled by the maximum width



[Andersen, Hadzic, Hooker, Tiedemann, CP 2007]
[Bergman, Cire, vH, Hooker, CPAIOR 2011, IJOC 2016]

Categories of Successful Applications

- Constraint Programming
  - DD-based constraint propagation
- Combinatorial optimization
  - MISP, MAX-CUT, graph coloring,…
- Scheduling, routing, planning
  - machine scheduling, TSPTW, SOP, AI robotic planning,…
- Decomposition and embedding in MIP
  - nonlinear objective functions, cutting planes, column generation,…

[Andersen et al. CP2007] [Hoda et al. CP2010]
[Bergman, Cire, and/or vH, 2013-2022]
[Perez&Régin 2015-2018] [Coppé et al., CP 2022]
[Verhaeghe et al. IJCAI 2018, CPAIOR 2019]
[Gentzel et al. CP 2020, 2022]

[Bergman, Cire, vH, Hooker, 2011-2016]
[Gillard et al., IJCAI 2020] [vH, MP 2022]
[Karahalios&vH, 2022] [Coppé et al., CP 2022]

[Cire&vH, OR2013], [Kinable et al. EJOR 2017]
[O'Neil&Hoffman, ORL2019] [Bogaerdt&de Weerdt, 2019]
[Gillard&Schaus, IJCAI2022] [Rudich et al. CP 2022]
[Castro et al. 2019-2022] [Horn et al. 2019-2021]

[Bergman&Cire 2018] [Lozano et al. 2020-2022]
[Morrison et al. IJOC 2016] [Kowalczyk & Leus IJOC 2018]
[Tjandraatmadja&vH, 2019, 2021] [Davarnia&vH, MP 2021]

Textbook: Bergman, Cire, vH, Hooker [Springer 2016]

Survey paper: Castro, Cire & Beck [IJOC 2022]

Industrial DD Solver: Hop from Nextmv

nextmv Solutions Pricing Company Learn Log in Contact us Try Nextmv

New techniques

Our research into modeling techniques led us to Decision Diagrams (DD). DDs represent optimization problems as the search for a shortest (or longest) path over a layered, directed graph. They are state-based, have few restrictions on problem representation, and can outperform MIP on optimization and CP on feasibility reasoning (this, of course, depends on the model).

We've had enormous success with DDs. Some of our pickup and delivery models are orders of magnitude faster using them. We find them particularly attractive for getting started with simple models and integrating them into software stacks. There aren't any other industrial grade DD solvers, so we built Hop to be the first.

Create a free account. No credit card required.

Application: Traveling Salesperson Problem

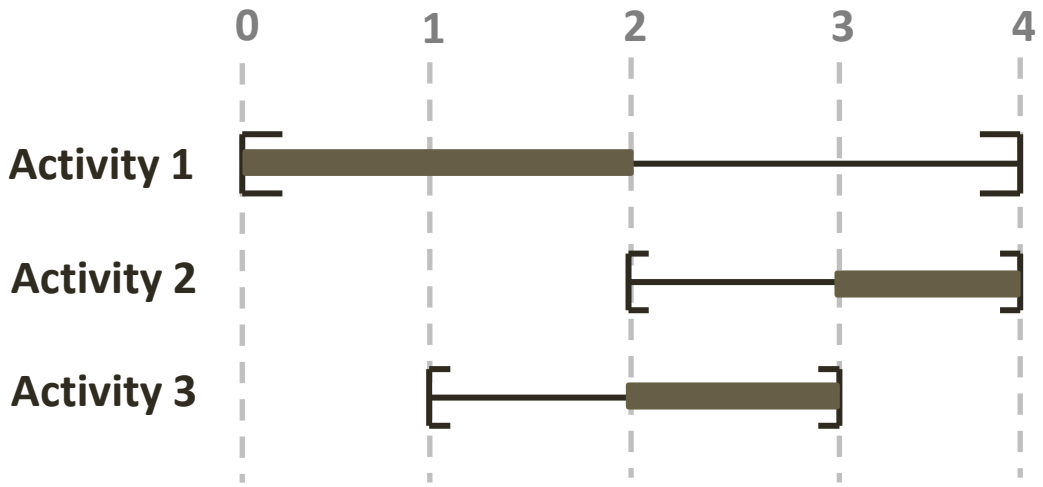
A. A. Cire and W.-J. van Hoeve. Multivalued Decision Diagrams for Sequencing Problems. *Operations Research* 61(6): 1411-1428, 2013.

D. Bergman, A. A. Cire, and W.-J. van Hoeve. Lagrangian Bounds from Decision Diagrams. *Constraints* 20(3):346-361, 2015.

J. Kinable, A. A. Cire, and W.-J. van Hoeve. Hybrid Optimization Methods for Time-Dependent Sequencing Problems. *European Journal of Operational Research* 259(3):887-897, 2017.

TSP: Special Case of Disjunctive Scheduling

- Disjunctive scheduling: Sequencing activities on a resource
- *Activities*
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
- *Resource*
  - Nonpreemptive
  - Process one activity at a time
- *Decision variables*
  - Start time $start_i$ for each activity i



Scheduling: Model Extensions

- Precedence relations $a_i \ll a_j$ between two activities
- Sequence-dependent setup times: s_{ij}
  - if a_i is followed by a_j we need at least s_{ij} time units to set up the machine
- Various objective functions
  - Makespan (=end time of last activity)
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

TSP as Scheduling Problem

TSP input

- Set of locations V
- Distance matrix D_{ij}

Other options:

- Time windows $[l_i, u_i]$
- Precedences $i \ll j$
- Visit duration h_i (can be 0)

Scheduling format

- Activity a_i for $i \epsilon V$
- Sequence-dependent setup times D_{ij}
- Release dates l_i
- Deadlines u_i
- Precedences $a_i \ll a_j$
- Processing time h_i
- *Objective: Sum of setup times*

DDs for Disjunctive Scheduling

Three main considerations:

- Representation
  - How to represent solutions of disjunctive scheduling in a DD?
- Construction
  - How to construct the DD?
- Inference techniques
  - What can we infer using the DD?

25

Decision Diagram Representation

- 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.:

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

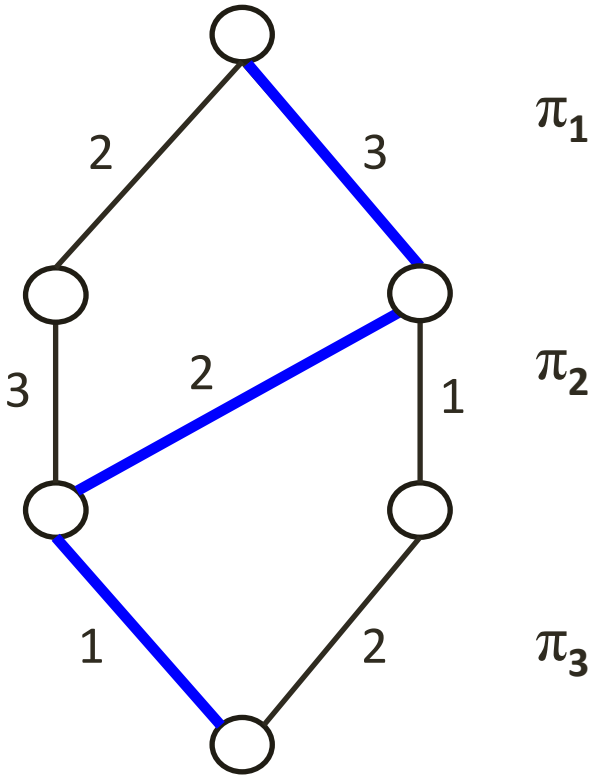
- Represent feasible permutations with **multi-valued** decision diagram (MDD)

[Cire&vH, OR 2013]

MDD Representation: Example

Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10

precedence: $3 \ll 1$



Path 3 – 2 – 1 :

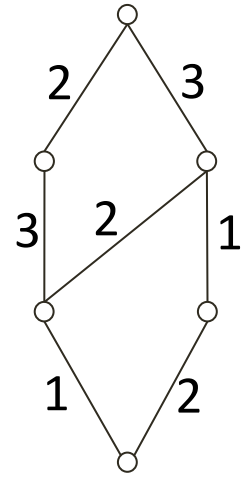
$6 \leq start_1 \leq 8$

$3 \leq start_2 \leq 5$

$1 \leq start_3 \leq 3$

MDD-based propagation

Propagation: remove infeasible arcs from the MDD



We can utilize several structures/constraints:

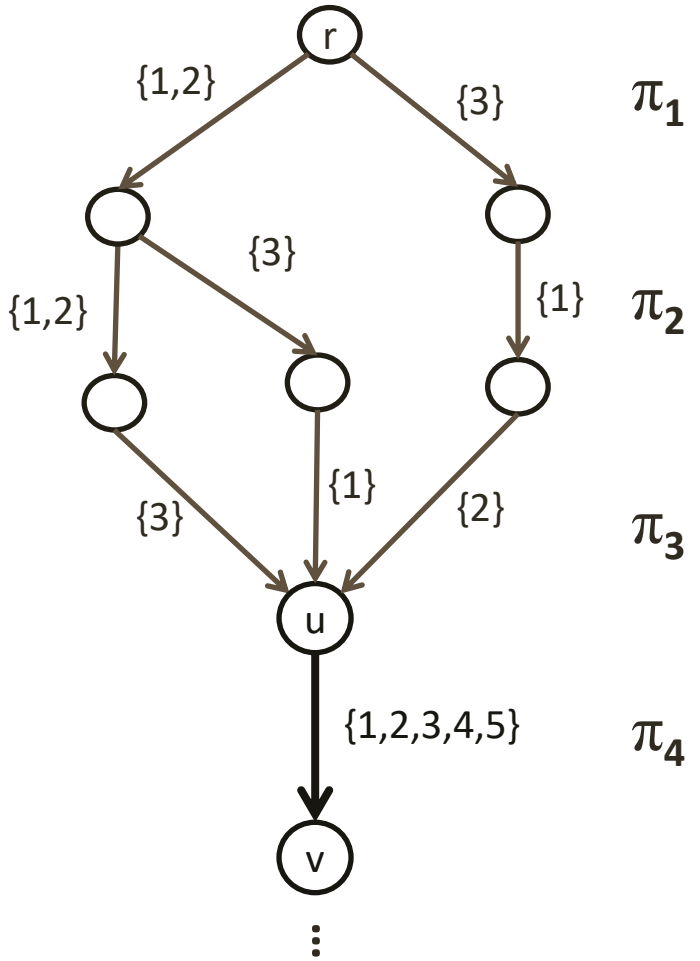
- *Alldifferent* for the permutation structure
- Earliest start time and latest end time
- Precedence relations

For a given constraint type we maintain specific ‘state information’ at each node in the MDD (both from top down and bottom up)

Propagation (cont'd)

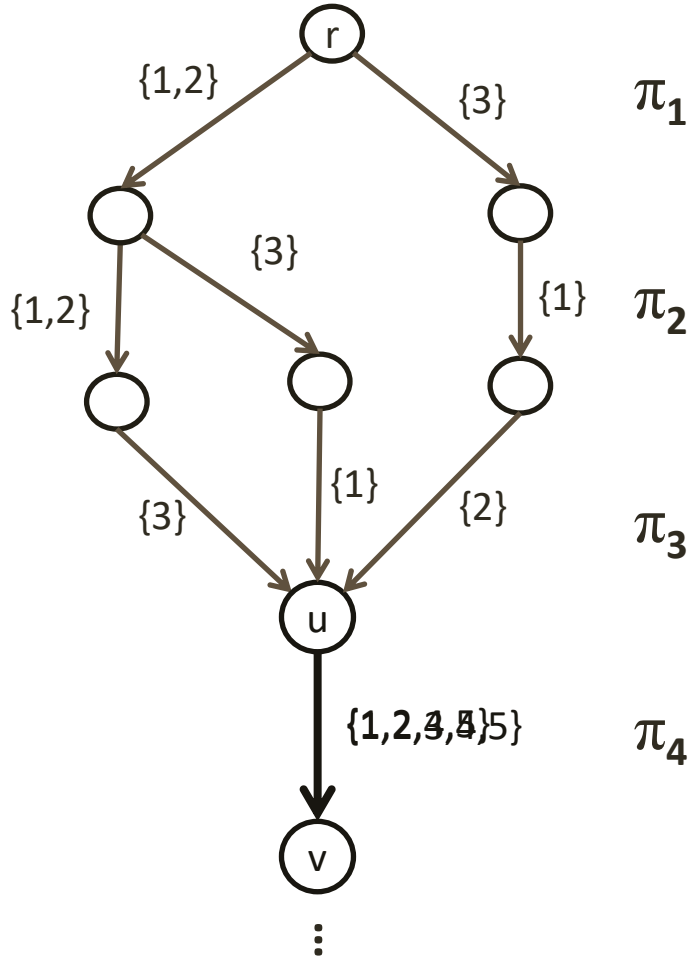
- State information at each node *i*
  - labels on *all* paths: A_i
  - labels on *some* paths: S_i
  - earliest starting time: E_i
  - latest completion time: L_i

- Top down example for arc (u,v)



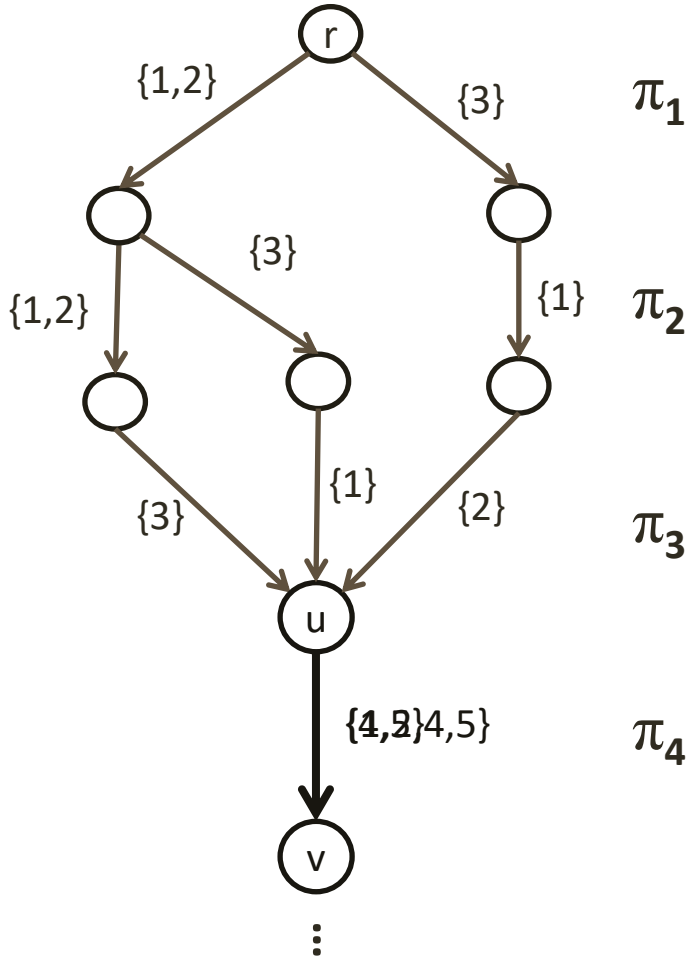
Alldifferent Propagation

- 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)



Alldifferent Propagation

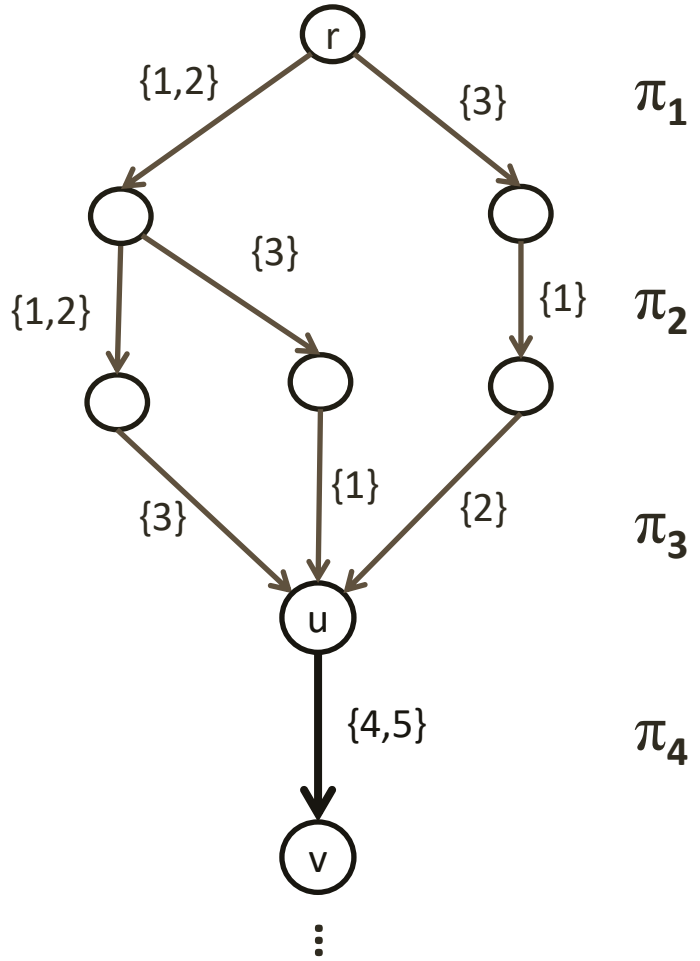
- Some-paths state: S_u
  - Labels belonging to **some** path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets (number of variables = number of values)
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

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

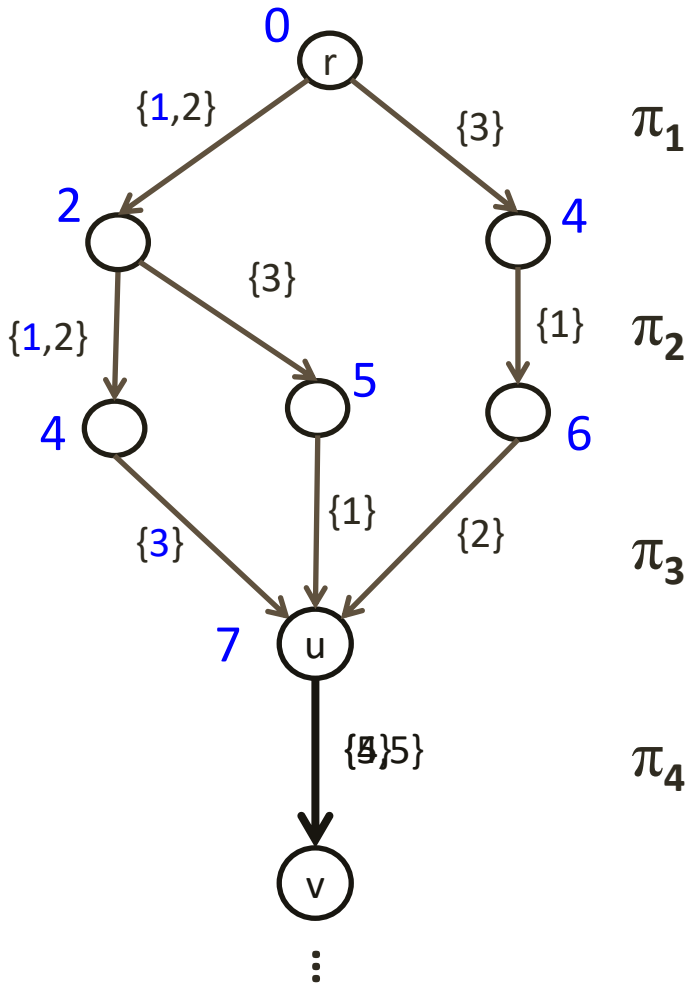
- Similarly: Latest Completion Time



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)



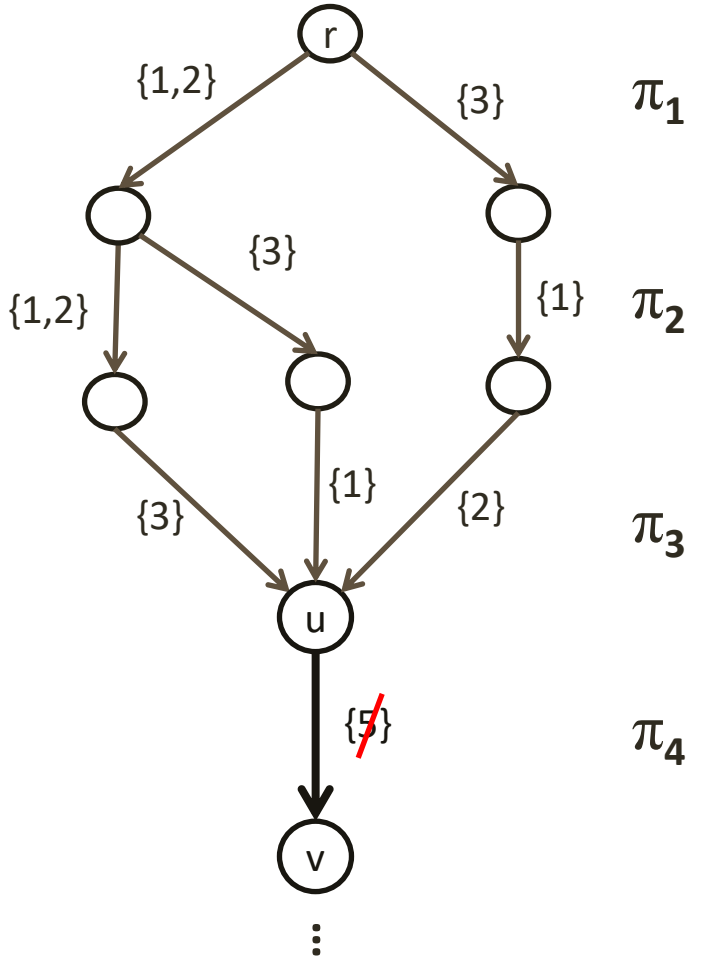
Propagate Precedence Relations

- Arc with label *j* infeasible if

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

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

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



Inference from the MDD

- **Theorem:** *Given exact MDD M, we can deduce **all** implied activity precedences in* $O(n^2|M|)$ *time*

- The algorithm can also be applied to *relaxed* MDD to find a subset of precedences
  - can be stronger than edge-finding, not-first/not-last, etc.

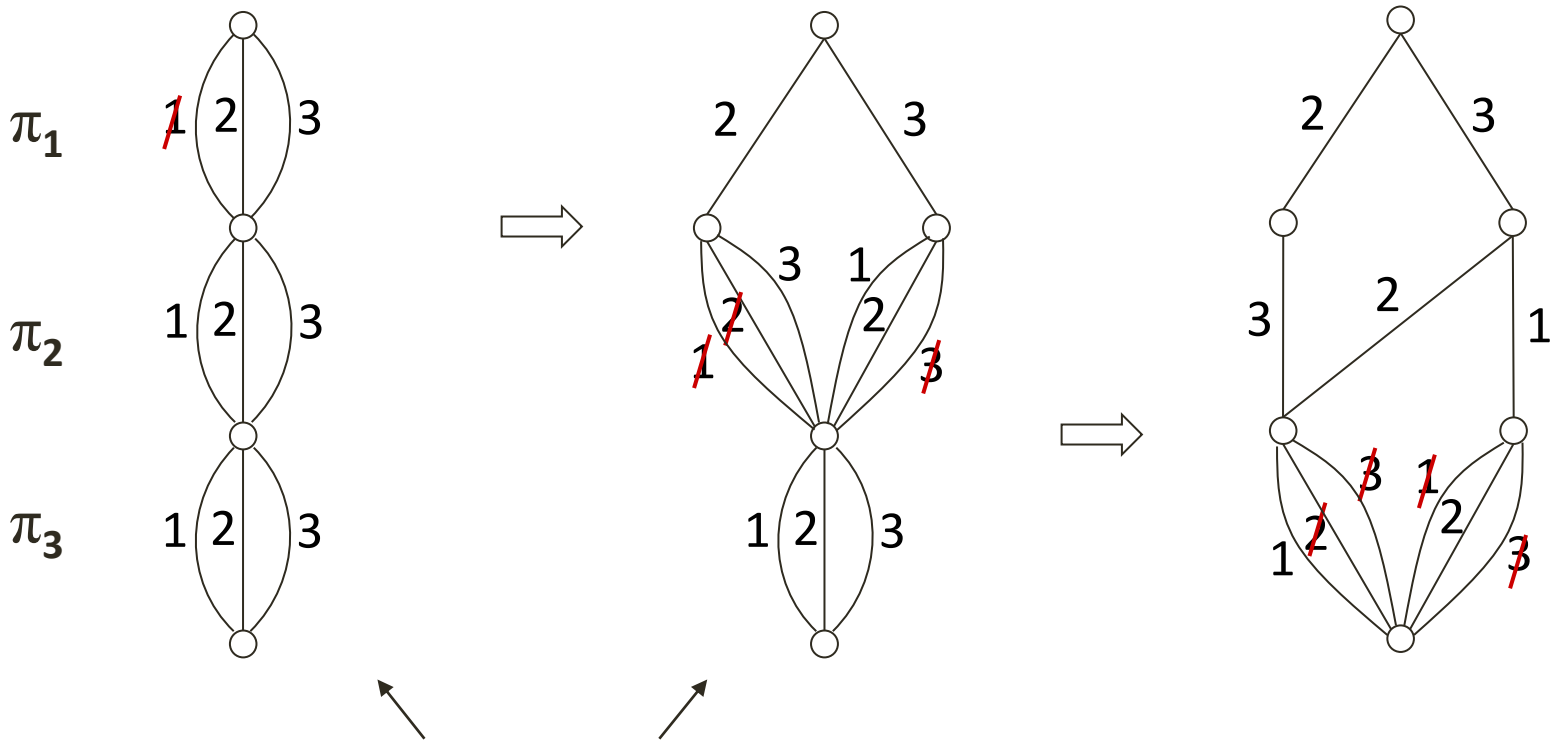
[Cire&vH, OR 2013]

Communicate Precedence Relations

1. Provide precedence relations from MDD to CP
   - update start/end time variables in CP model
   - other inference techniques may utilize them
   - may help to guide search
2. Filter the MDD using precedence relations from other (CP) techniques
3. In context of MIP, these can be added as linear inequalities

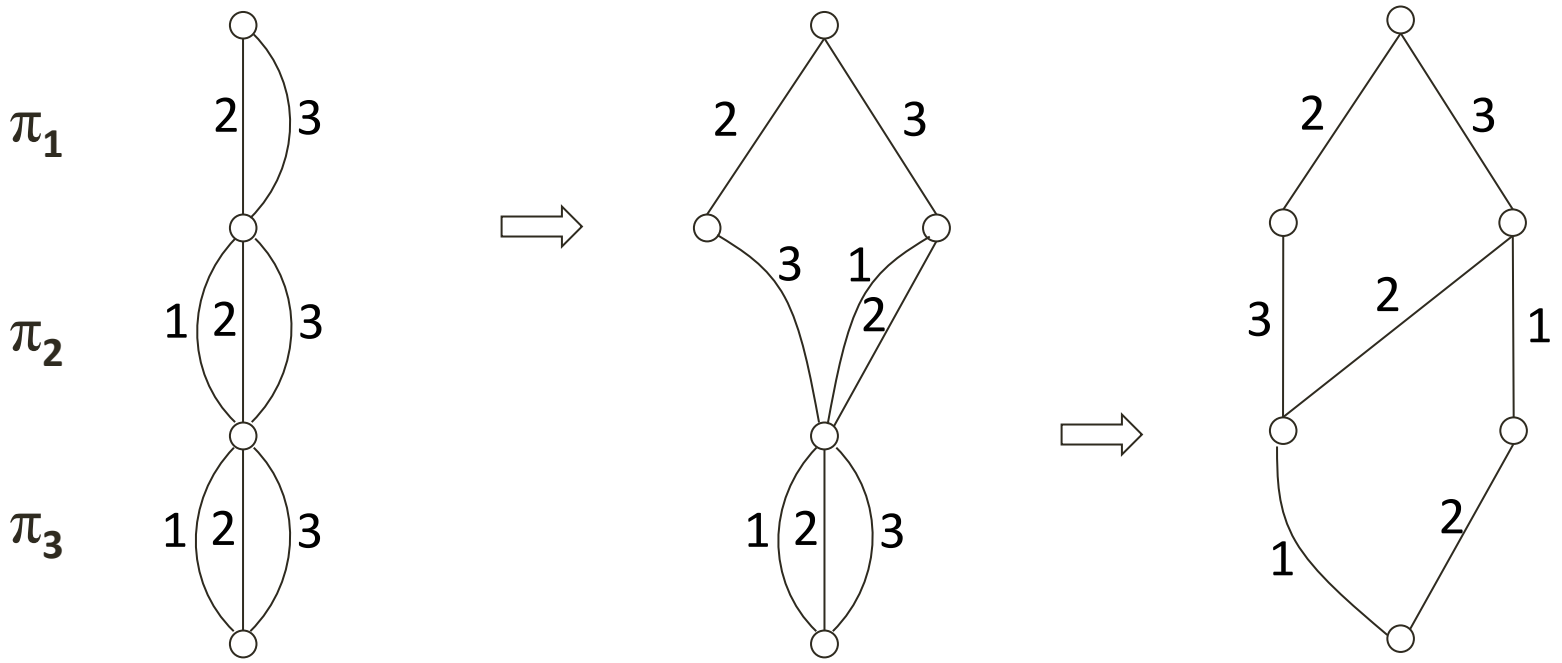
Top-down MDD compilation: Example

precedence:
$3 \ll 1$



Top-down MDD compilation: Example

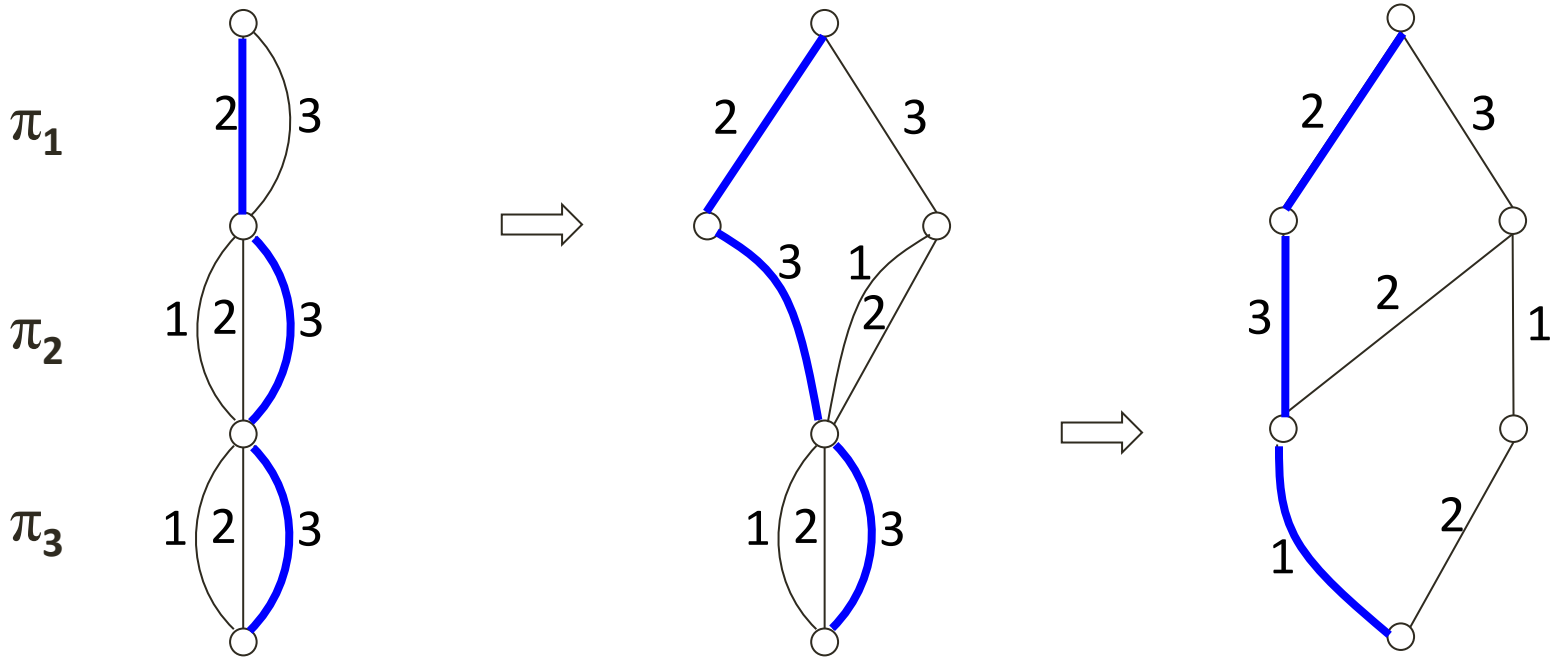
precedence:
$3 \ll 1$



Top-down MDD compilation: Example

precedence:
$3 \ll 1$

Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10



minimize makespan:

lower bound = 7

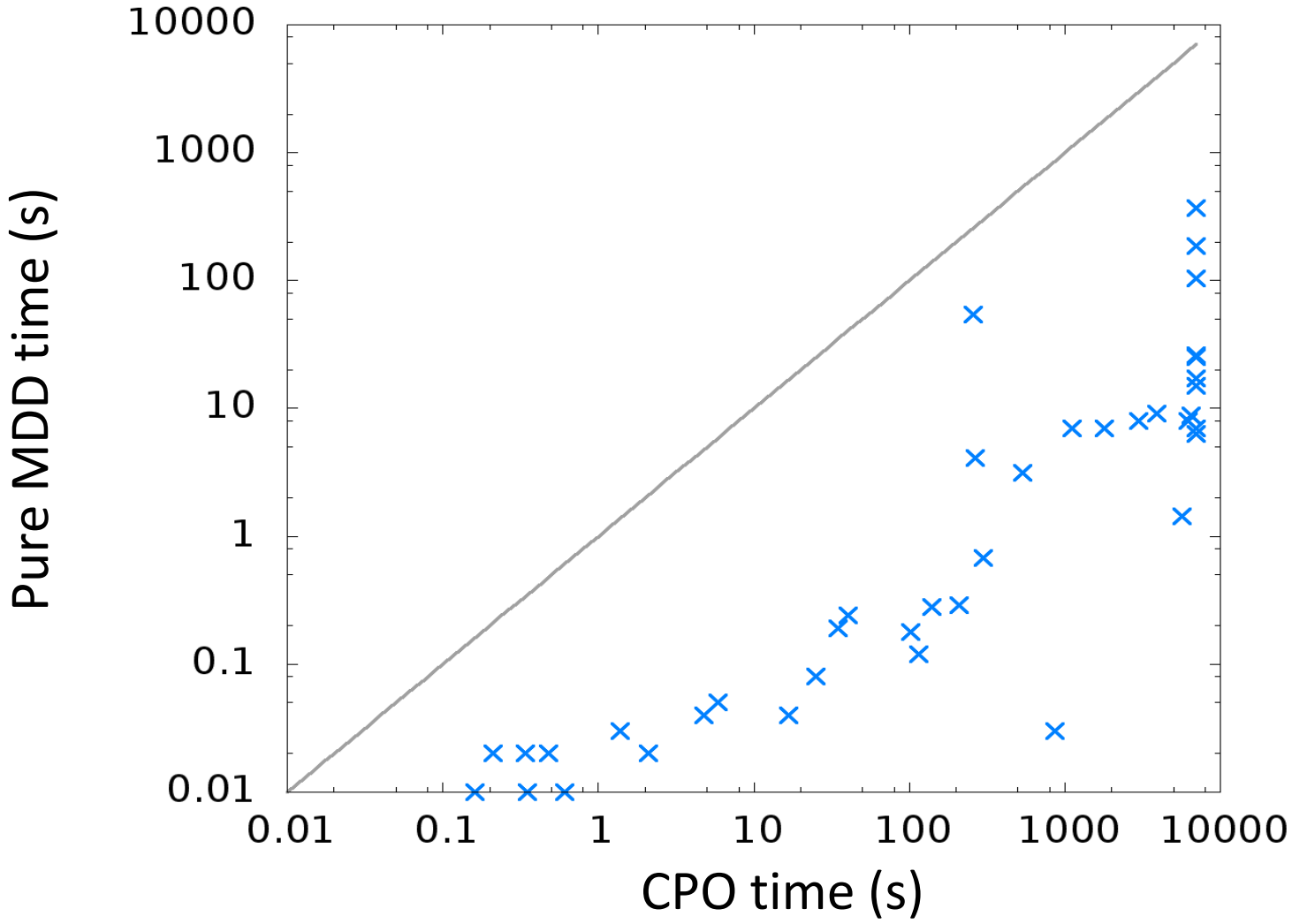
lower bound = 7

optimum = 9

Performance

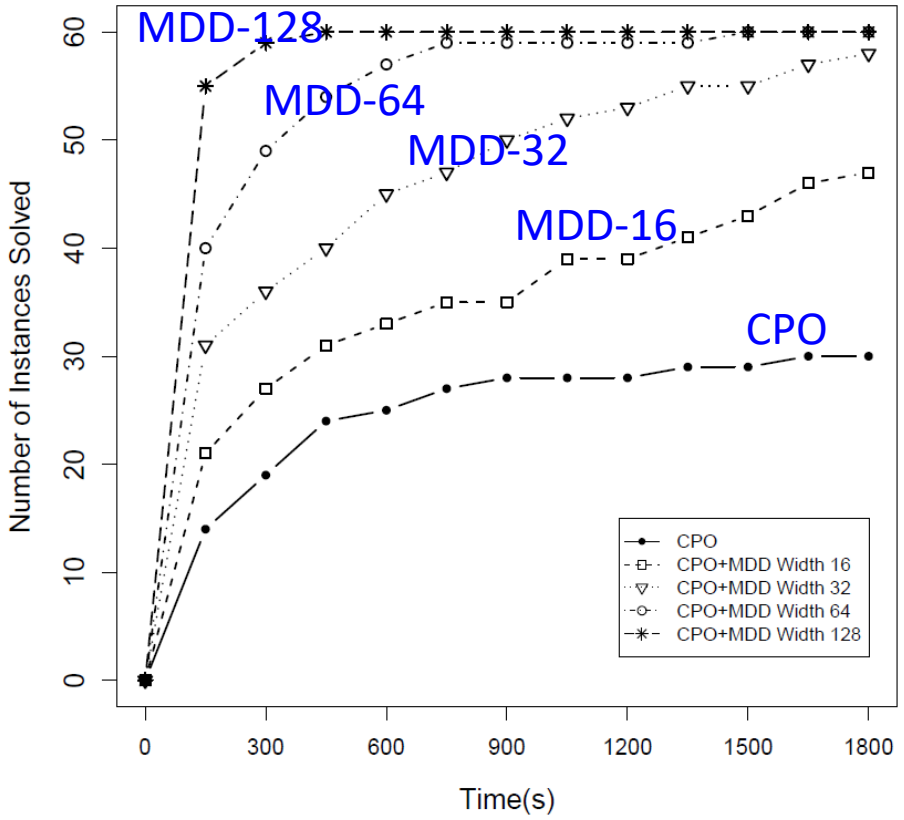
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation
  - MDD is added as user-defined propagator
- Compare three different variants
  - CPO (only use CPO propagation)
  - MDD (only use MDD propagation)
  - CPO+MDD (use both)

TSP with Time Windows

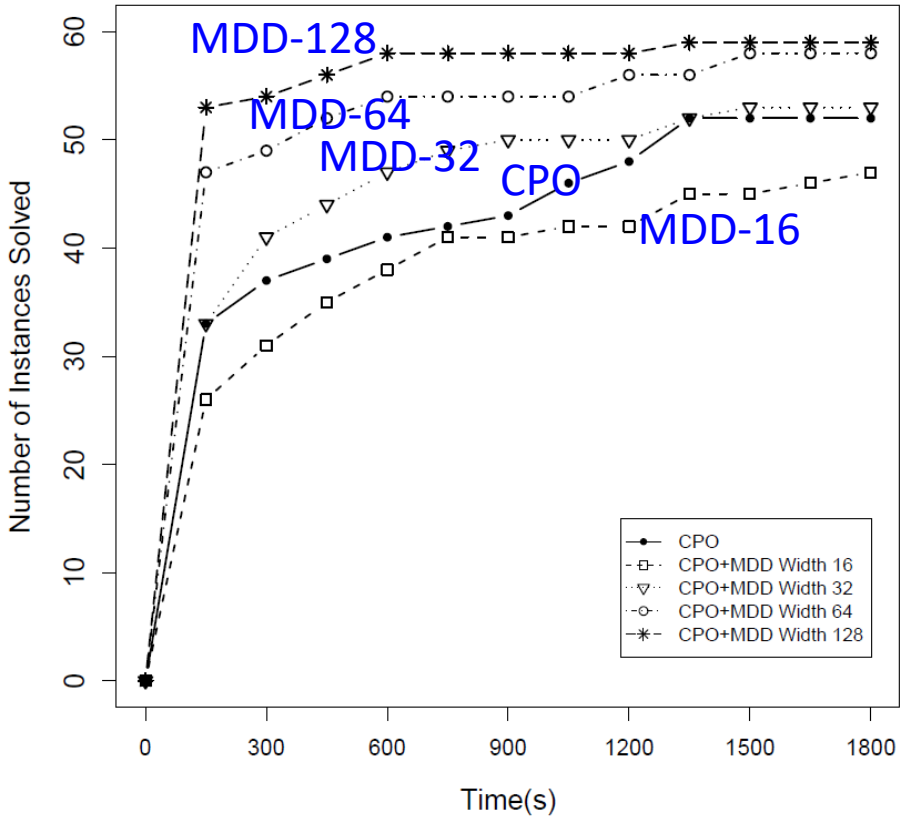


Dumas/Ascheuer instances
- 20-60 cities
- max MDD width: 16

Total Tardiness



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

Strengthening Relaxed Decision Diagrams

- Lagrangian relaxation
  - penalize constraint violations by modifying arc weights
- Additive bounding
  - incorporate dual information from LP relaxations
  - e.g., aggregate reduced costs along path from root to terminal

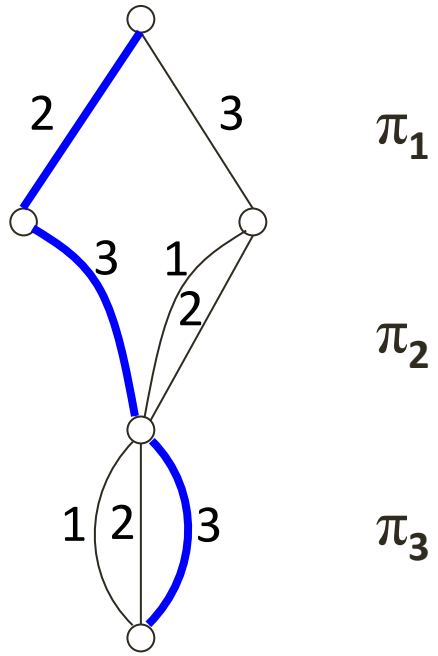
Extension: Lagrangian bounds

- Observation: MDD bounds can be very loose
  - main cause: repetition of activities
- Apply Lagrangian relaxation
  - penalize repeated activities; reward unused activities

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

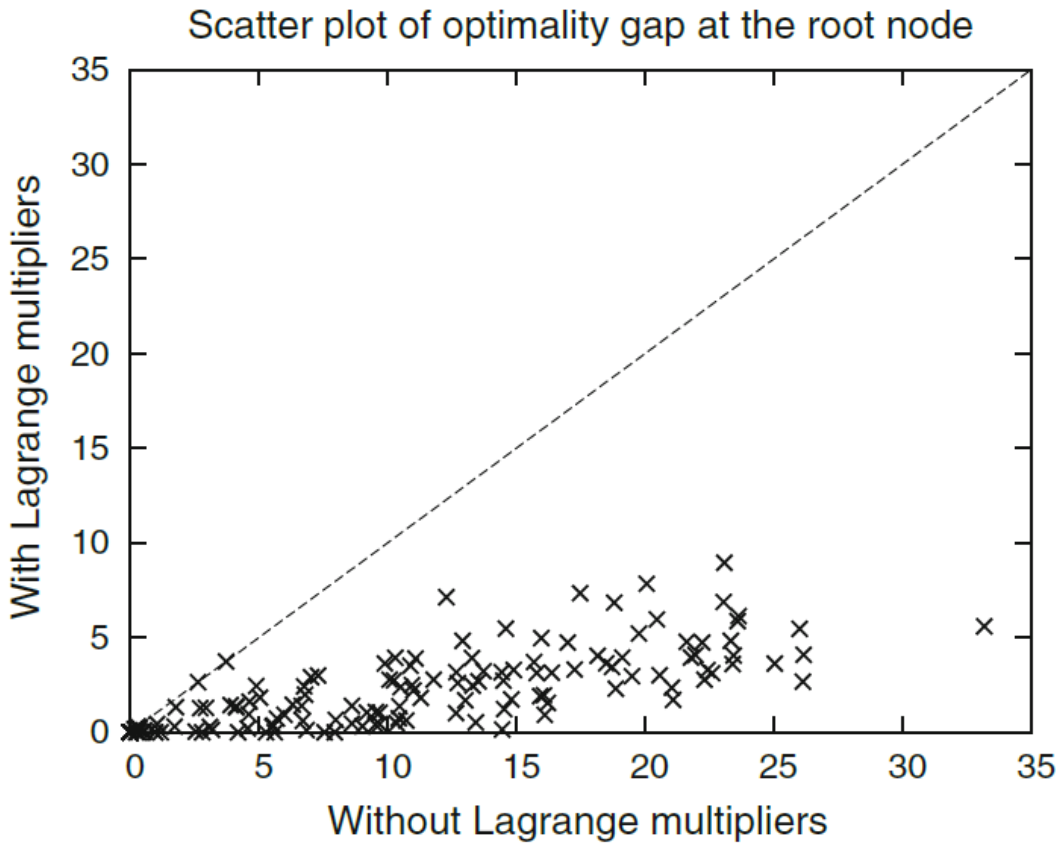
$$= z + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j (\pi_i = j) - \sum_{j=1}^{n} \lambda_j$$

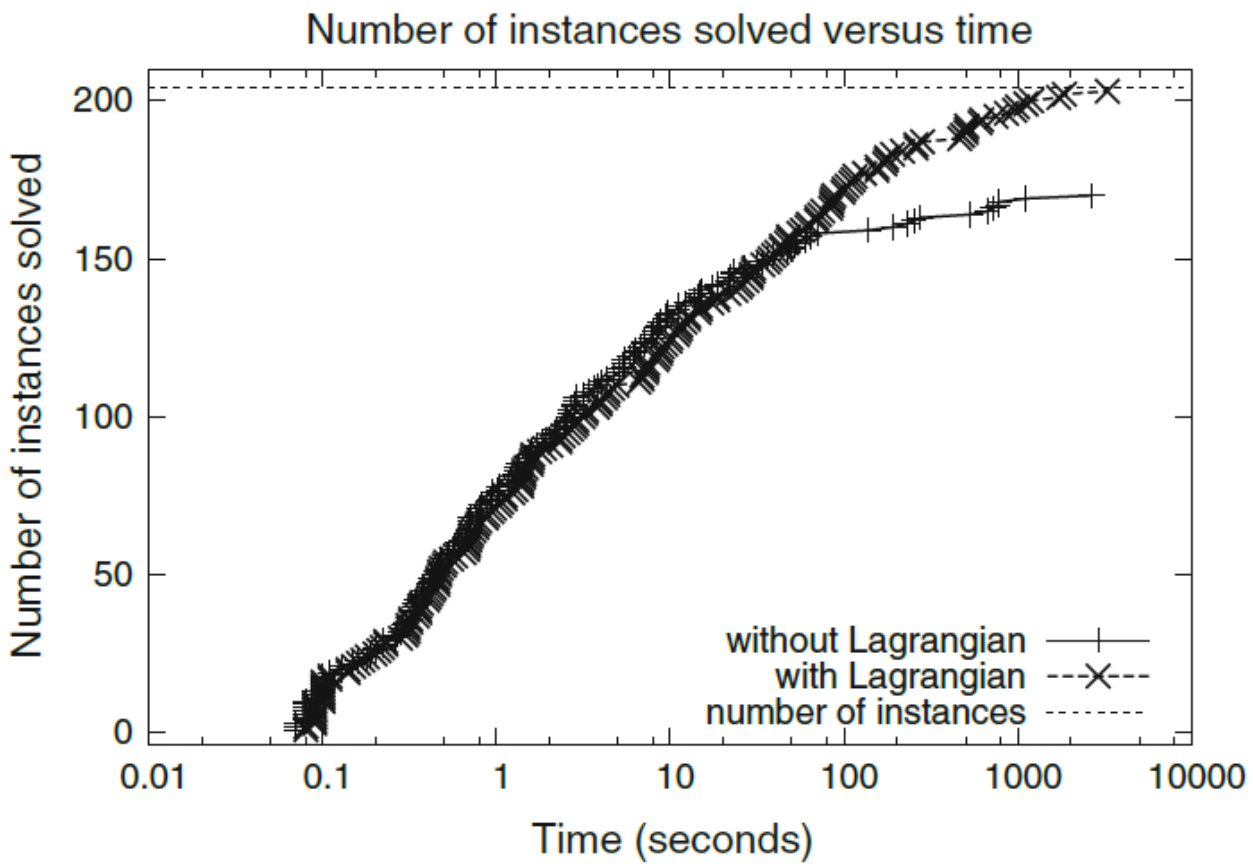
  - shortest path with updated weights



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

Impact of Lagrangian Relaxation (TSPTW)



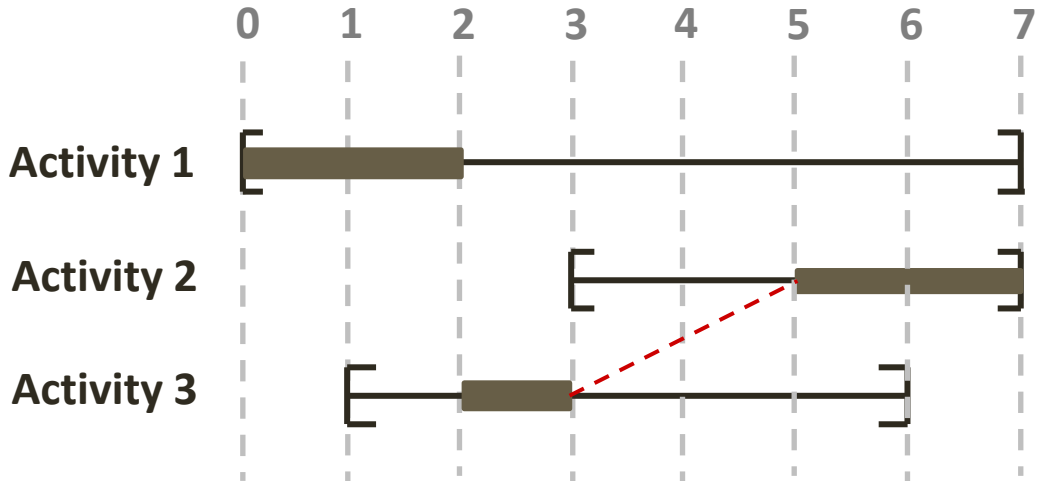


[Bergman, Cire, vH, 2015]

Extension: Additive Bounding

- Case: time-dependent sequencing
  - sequence-dependent setup times also depend on position!
  - $\delta_{i,j}^{t}$ = setup time between i and j if i is at position t

- MDD representation
  - state-dependent costs



[Kinable, Cire, vH, EJOR 2017]

Additive Bounding: LP + MDD

- Add LP reduced costs to MDD relaxation [Fischetti &Toth, 1989]
- Effectivess depends on the quality of the LP relaxation
- LP can be made stronger for specific problem class
  - TD-TSP [Picard & Queyranne, 1978] [Vander Wiel and Sahinidis, 1995] [Gouveia and Voss, 1995] [Abeledo et al. 2013] [Miranda-Bront et al., 2014]
  - TD-TSP-TW (time windows) [Miller,Tucker, Zemlin, 1960] [Desrocher & Laporte, 2014]
  - TD-SOP (precedence constraints) [Sarin, Sherali, Bhootra, 2005]

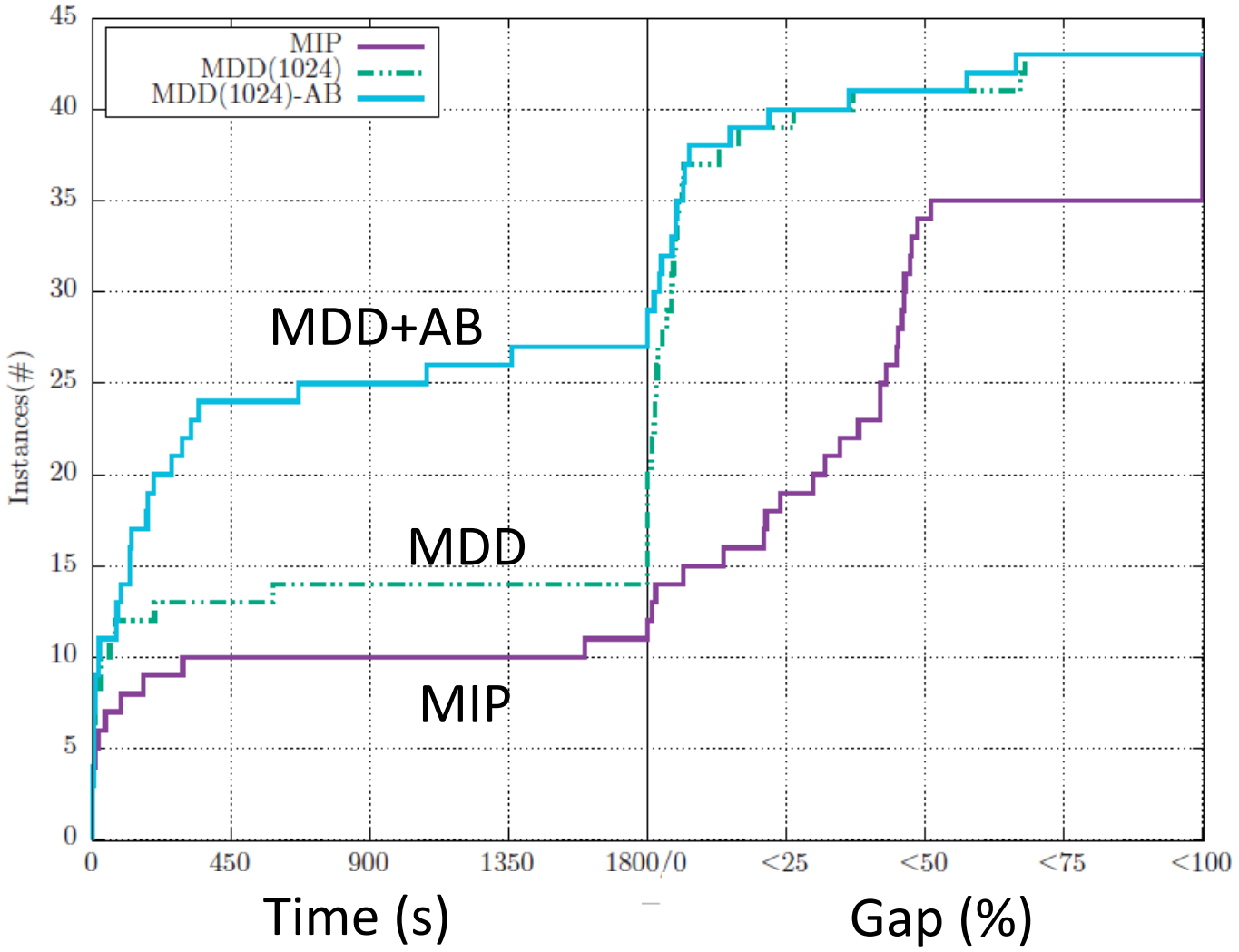
Experimental Setup

- Solvers: IBM ILOG CPLEX and CP Optimizer 12.6.3
  - MDD added to CP Optimizer (Cire & v.H., 2013)
  - maximum width 1024
  - time limit: 30 minutes

- TD-TSP 38 instances from TSPLIB (n=14-107 jobs)

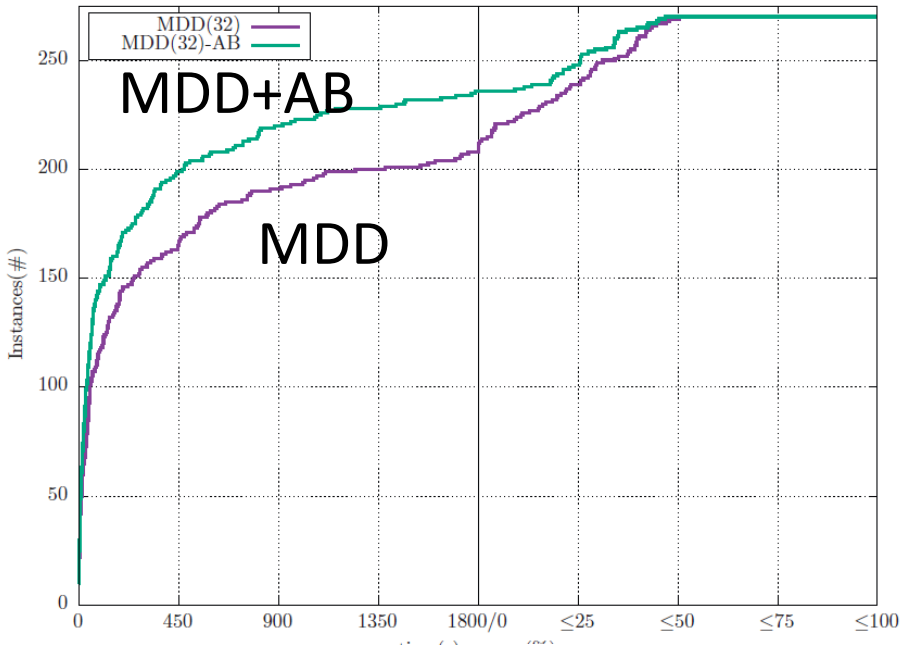
  $\delta_{i,j}^{t}$ = (n-t)*$\delta_{i,j}$ [Abeledo et al., 2013]
- TD-TSPTW based on Dumas et al. (n=30, 35, 40), 270 total
- TD-SOP 29 instances from SOP dataset in TSPLib (n=7 to 100)

TD-TSP: Performance Plot



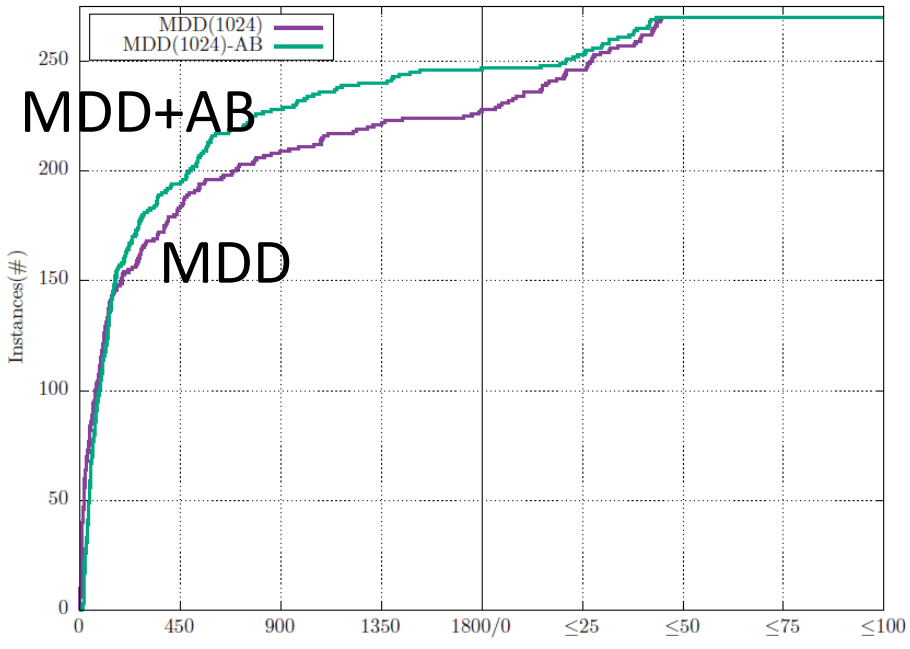
TD-TSPTW: Performance Plot

max width 32



Time (s) Gap (%)

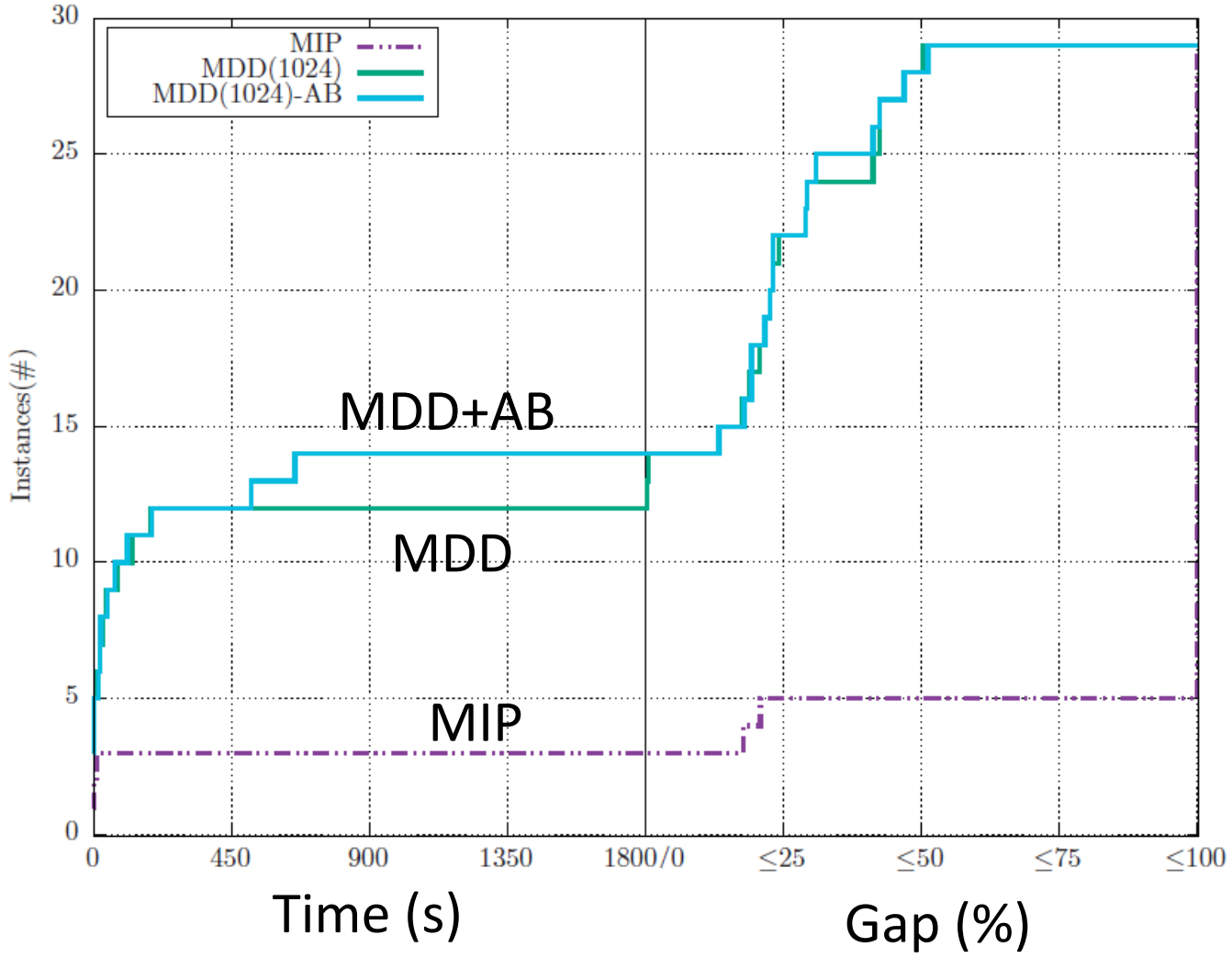
max width 1024



Time (s) Gap (%)

(MIP was unable to find any single integer solution)

TD-SOP: Performance Plot



Application: Vehicle Routing

Z. Tang and W.-J. van Hoeve. Dual Bounds from Decision Diagram-Based Route Relaxations: An Application to Truck-Drone Routing. *Transportation Science*, to appear.

Background: Column Generation for VRPs

- Branch-and-Price with Column Generation is a very effective method for solving VRPs
- It uses an *extended formulation*: binary variable for every possible truck route (in principle exponentially many)

$$\min \sum_{r \in R} d_r x_r$$

$$\text{s.t.} \sum_{r \in R} M_{ir} x_r = 1 \quad \forall i \in \{1, 2, \ldots, n\}$$

$$\sum_{r \in R} x_r = K$$

$$0 \le x_r \le 1 \qquad \forall r \in R$$

Restricted master problem with subset of routes R

LP-dual values →

← new route (add to R)

min (cost of route) – (dual values of route)
s.t. route constraints
capacity constraints
time window constraints
precedence constraints
etc.

(Often solved with Dynamic Programming)

Pricing problem: find improving route

Column ‘elimination’ instead of column generation?

- Column generation works with *restricted* set of columns
  - no valid lower bound until optimal LP basis is found *
  - stability and convergence issues due to degenerate LP solutions
  - solving LP as MIP is not sufficient—embed in branch-and-price search
- Alternative: work with *relaxed* set of columns

[vH, *IPCO* 2020]
[vH, *Math. Prog.* 2021]

  - initial relaxation includes columns that are not feasible
  - apply an iterative refinement algorithm to eliminate infeasible columns
  - use *decision diagrams* for compact representation and efficiency
  - no need for shadow prices or branch-and-price; just “MIP-it” (or use standard branch-and-bound)
  - for VRP, we can use the dynamic program of the pricing problem to compile the DD!

* But can use reduced cost information to find *approximate* LP bound

Case Study: Truck-Drone Routing

- One truck + one drone
- Possible legs include:

  truck, drone, combined
- Example route duration =

  max{1, 0.5+0.5} +

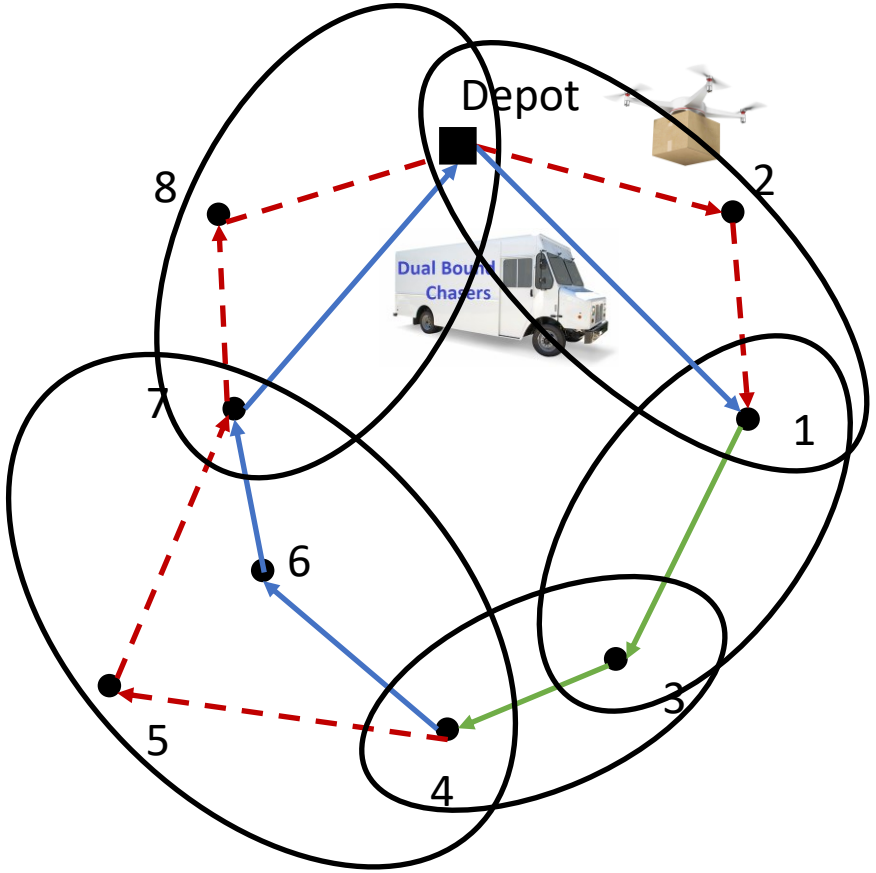
  1 +

  1 +

  max{1+1, 0.5+0.5} +

  max{1, 0.5+0.5}

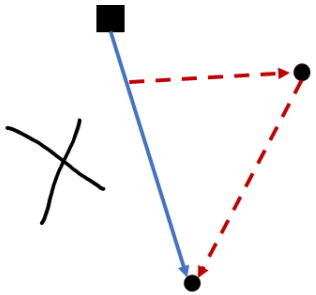
  = 6



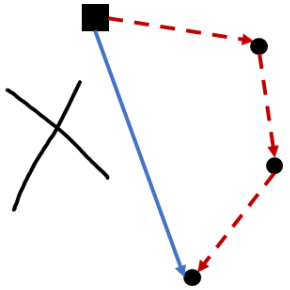
truck speed: 1 unit per edge
drone speed: 0.5 unit per edge

Definition of TSP-D

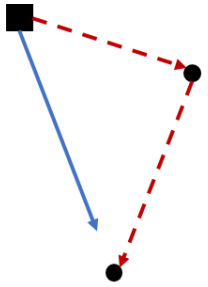
- TSP-D: Traveling Salesperson with a Drone
- Drone speed = α * truck speed (for some fixed α)
- Goal: minimize route duration
- Assumptions:



Drone cannot be dispatched from the truck while the truck is traveling



Drone can only visit one customer before rejoining with the truck



Waiting required

State of the art: Branch-and-Price

- Master LP: set partitioning model
- Pricing: DP model (with ng-route relaxation)

[Roberti & Ruthmair, TS2021]

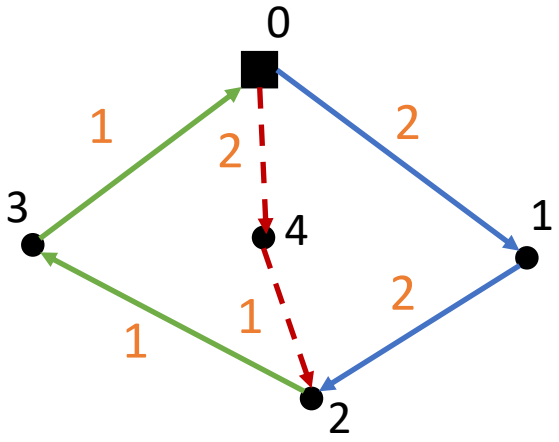
Dynamic Programming Model for TSP-D

State definition (S, LC, LT, t), where

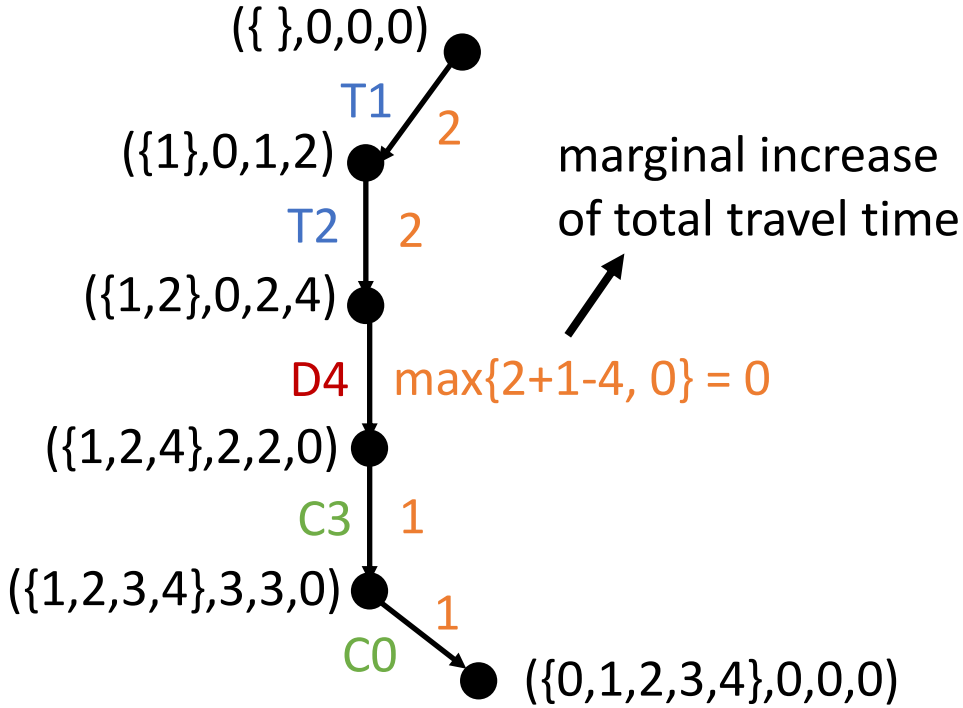
- S = customers visited so far
- LC = latest location visited by both vehicles
- LT = latest location visited by truck alone
- t = time spent by the truck traveling alone since leaving LC

Set of **controls**

- truck leg for customer i: Ti
- drone leg: Di
- combined leg: Ci



Route: T1, T2, D4, C3, C0

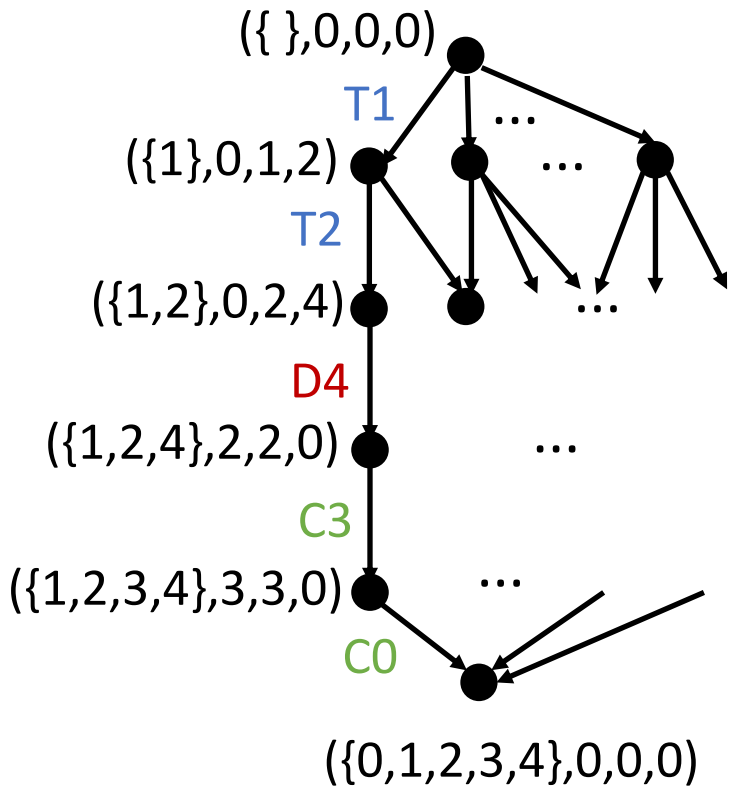


[Roberti&Ruthmair, 2021]

Decision Diagram Compilation for TSP-D

- Top-down DD compilation can be defined by state transition function of DP model

  [Bergman et al. 2016]
  - DD nodes are associated with DP states
  - DD arc labels are given by allowed controls
  - similar to state-transition graph in DP
- Apply the previous DP model for TSP-D
  - exact diagram represents all feasible solutions
  - shortest path = optimal solution, but exponential size
- How to compile relaxed decision diagram?
  - apply route relaxation DP (e.g., ng-route), or
  - define new relaxed DD via Column Elimination



Derive Bound From Constrained Network Flow

Constrained integer network flow model (NP-hard):

$$\min \quad \sum_{a \in A_D} \gamma_a y_a$$

$$\text{s.t.} \quad \sum_{a \in \delta^+(u)} y_a = \sum_{a \in \delta^-(u)} y_a, \quad \forall u \in V_D, u \neq r, t$$

$$\sum_{a \in \delta^+(r)} y_a = 1$$

$$\sum_{a \in \delta^-(t)} y_a = 1$$

$$\sum_{l(a) \text{ is a visit to customer i}} y_a = 1, \quad \forall i \in N$$

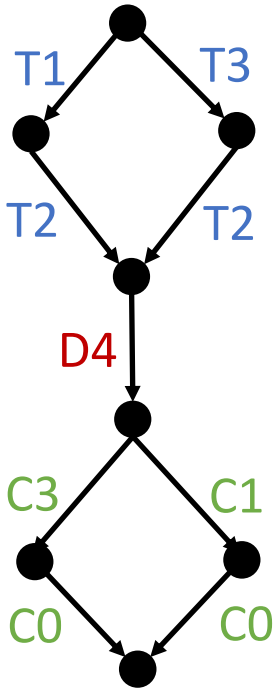
$$y_a \in \{0,1\}, \quad \forall a \in A_D$$

Lagrangian relaxation:

- Add dual variable to arc weights
- Shortest path in DD (integral)

LP relaxation:

- $0 \leq y_a \leq 1$
- Use off-the-shelf LP solver



Equivalence of Relaxation Bounds

- **Observation:** Given a DP model representing a route relaxation R, the associated decision diagram D_R contains exactly all feasible paths corresponding to R
- Let
  - SPLP(R) be the set partitioning LP model with the DP pricing problem
  - CFLP(D_R) be constrained network flow LP defined over D
  - LR(D_R) be the Lagrangian relaxation of the constrained network flow defined over D

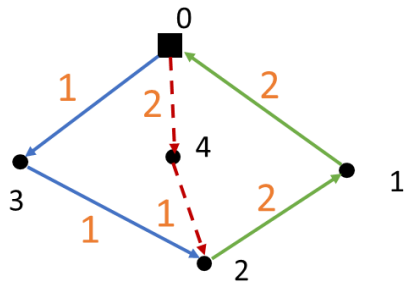
  Theorem: SPLP(R), CFLP(D_R), and LR(D_R) have the same optimal objective value

Going Beyond the ng-Route Bound

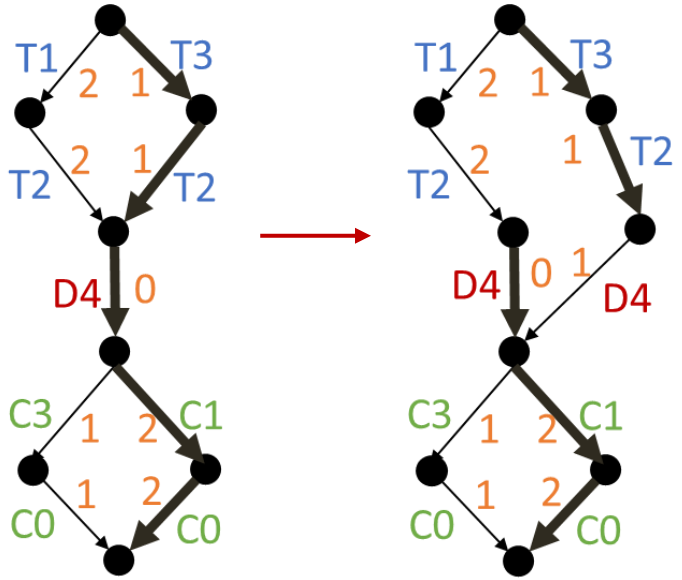
- Resolve conflicts along solution paths by refining the DD

Type 1: objective function

Route 2: T3, T2, D4, C1, C0

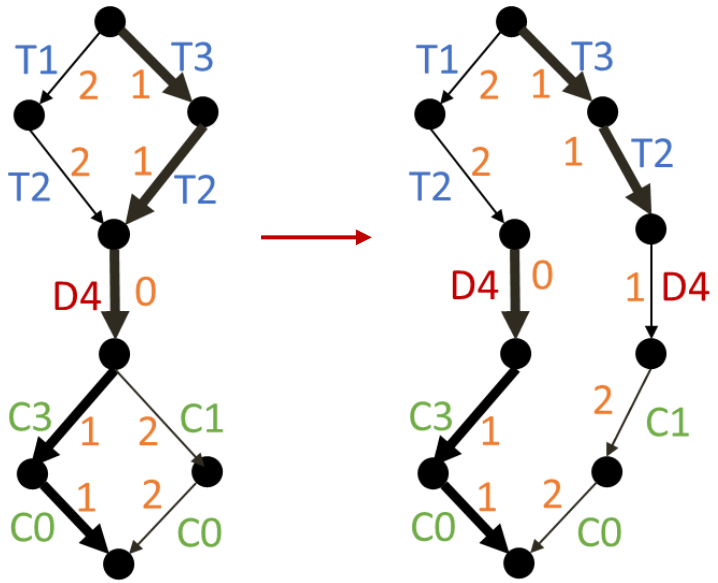


Duration = 7



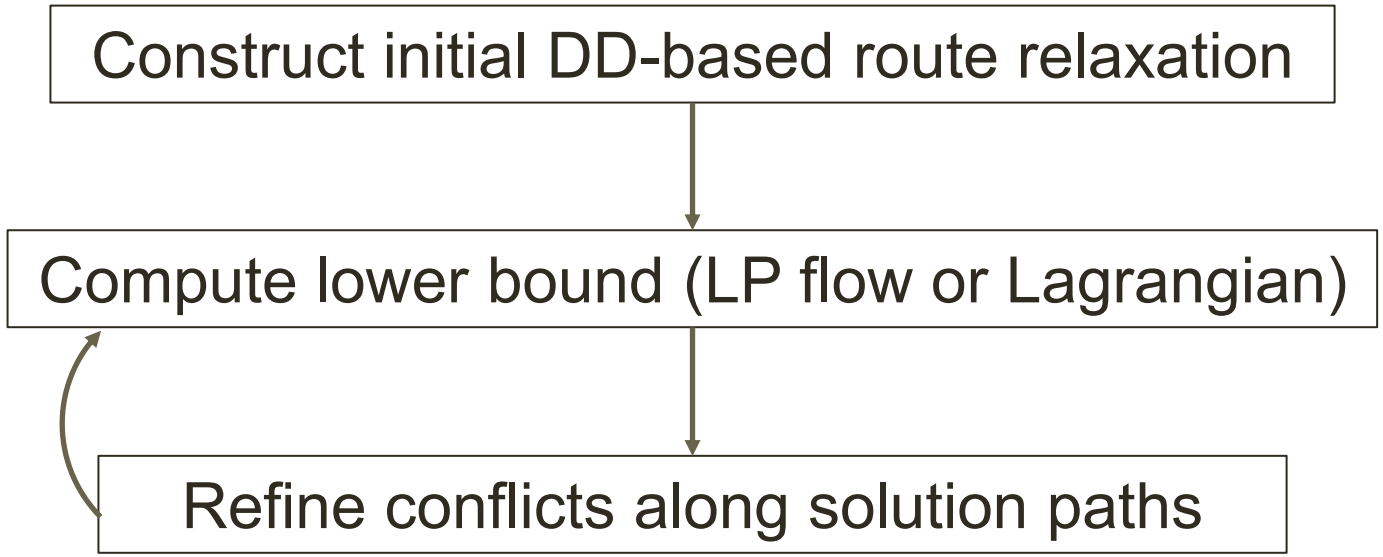
Path length = 6

Type 2: repeated visits



Customer 3 repeated

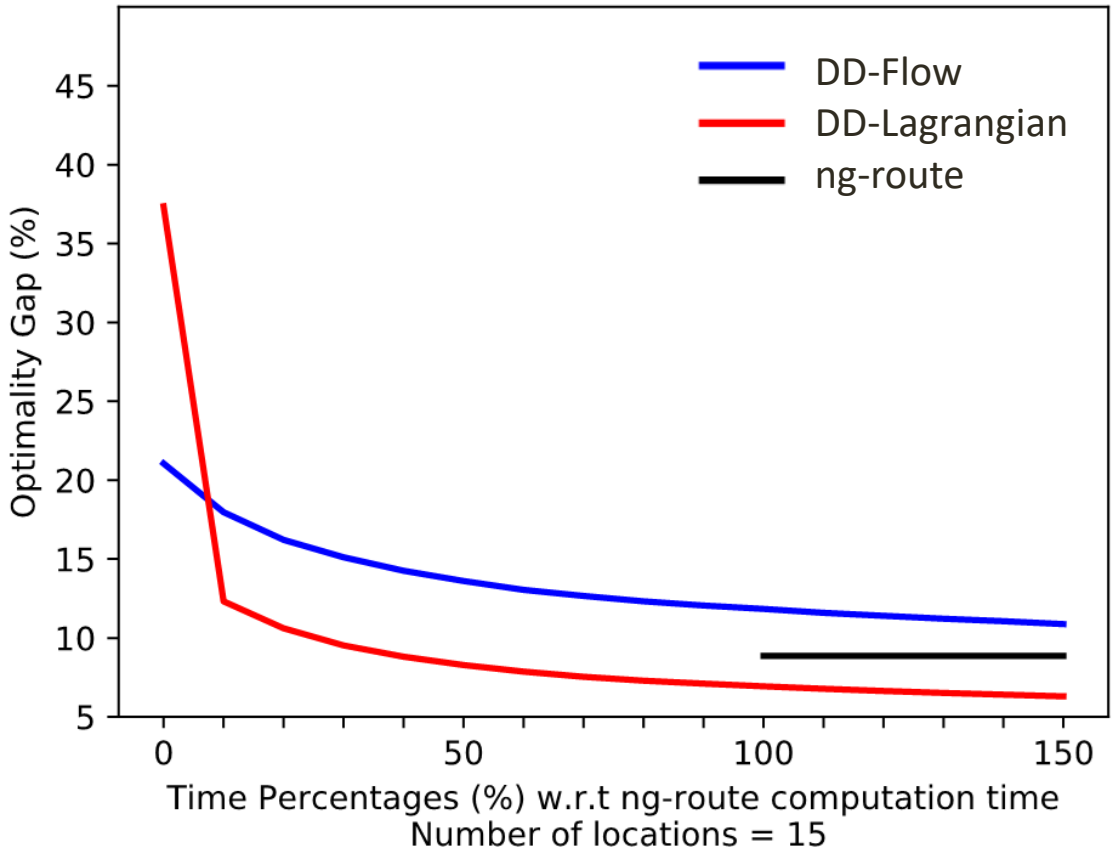
Overall Framework

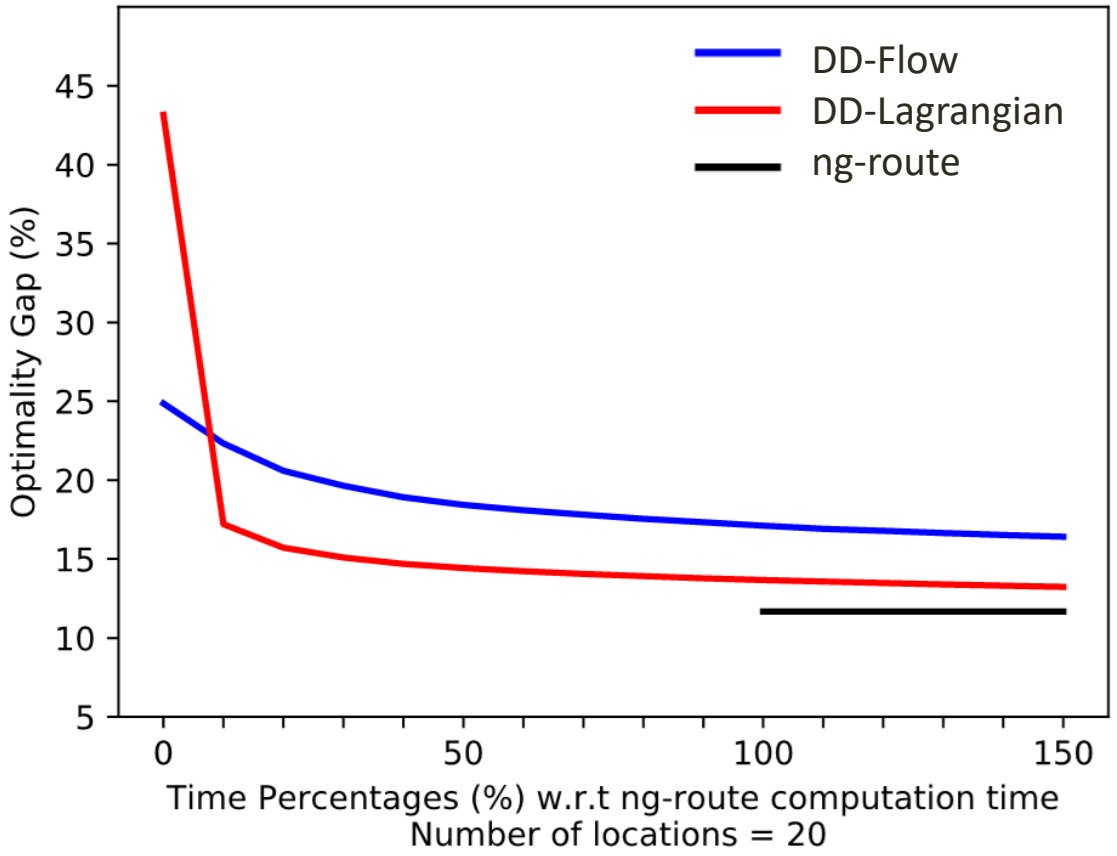


Experimental Evaluation on TSP-D

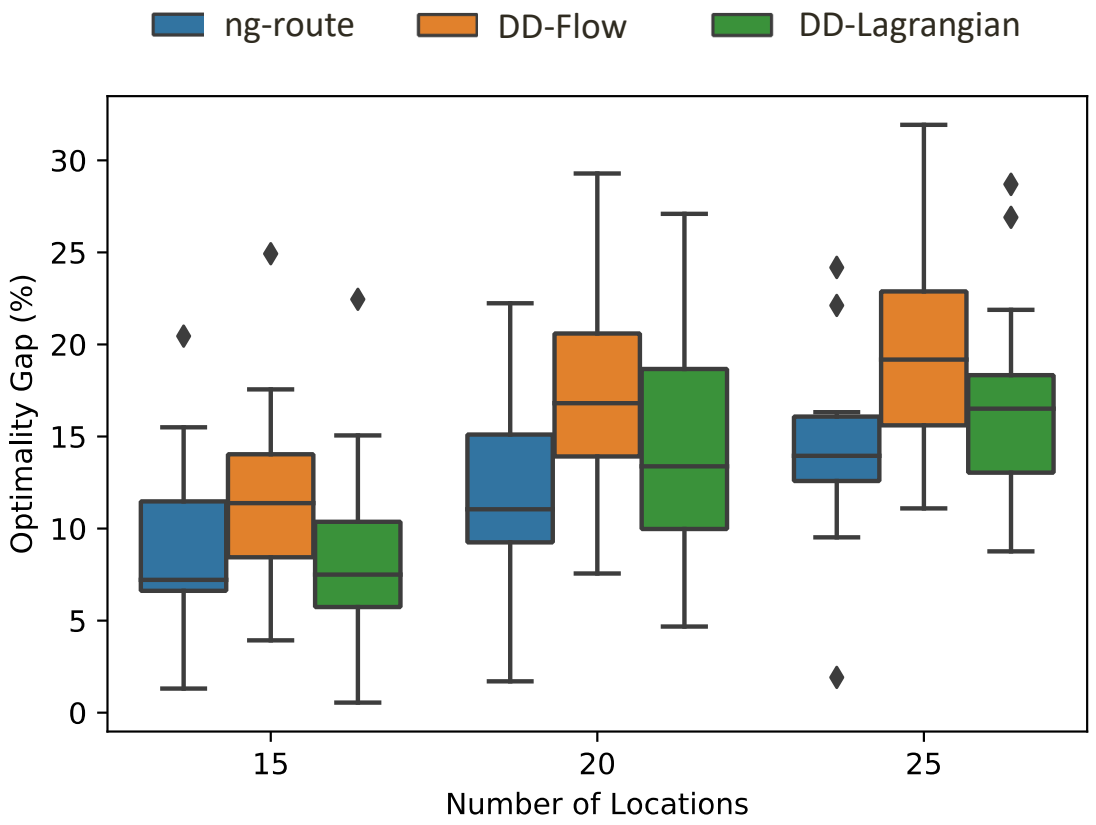
- Evaluate two variants
  - DD-Flow: lower bound from constrained network flow LP
  - DD-Lagrangian: lower bound from Lagrangian
  - both apply iterative refinement based on conflicts
- Comparison with state-of-the-art bound for TSP-D
  - column generation model from [Roberti&Ruthmair, TS2021]
  - set partitioning LP using ng-route relaxation
- Benchmark
  - random instance generation [Poikonen et al., 2019]
- Upper bound
  - best solution found by CP in 1h [Tang et al, CPAIOR2019]

Optimality gap improvement over time



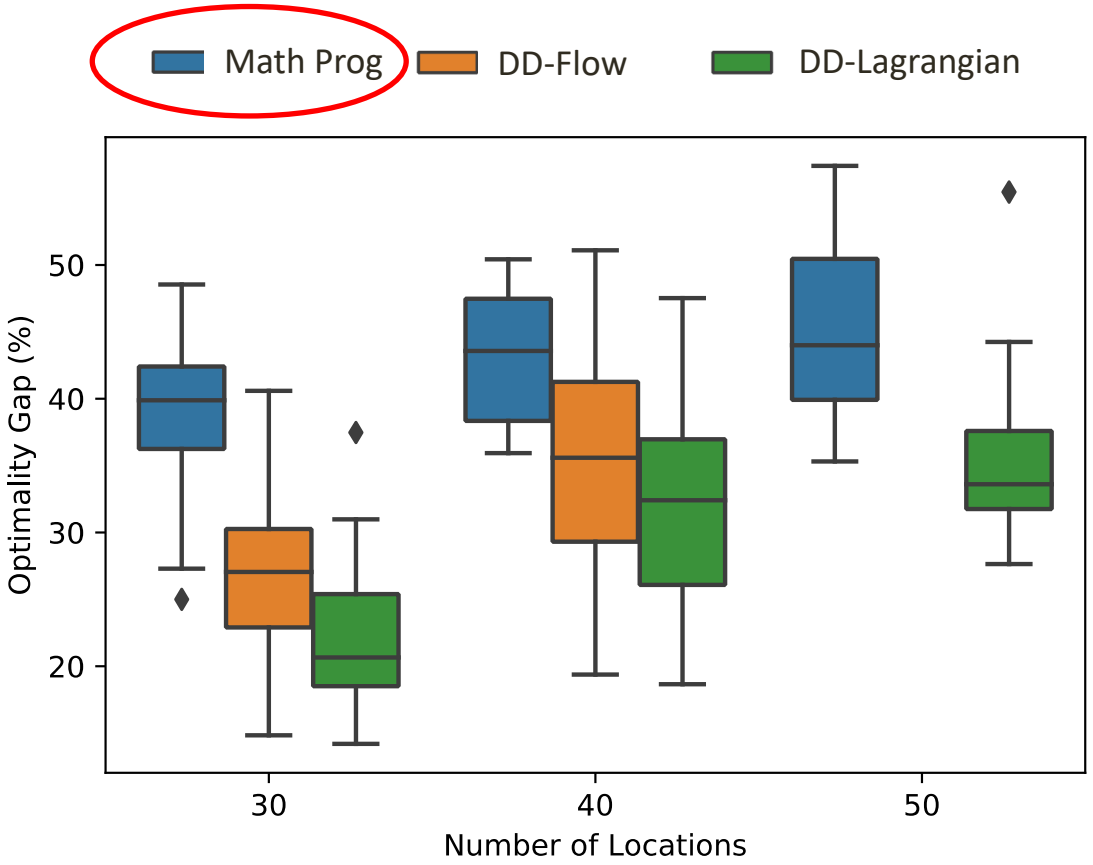


Optimality gap for varying problem sizes



(Time limit for DD methods is the ng-route solving time)

Optimality gap for larger instances



- Column generation does not scale beyond 30 locations
- We therefore compare to LP relaxation of MIP model proposed by [Roberti&Ruthmair, 2019]

Conclusions

- Decision Diagrams provide a new way to represent VRPs
- Relaxed DDs trade off size (memory) for strength of bound
  - Can be embedded in existing solvers, e.g., constraint programming
  - Or can be the basis of stand-alone solution method, e.g., column elimination
- Competitive results on variants of TSP and TSP+drone routing

DDs for general Capacitated VRPs will be presented on **Thursday**:

Karahalios and vH: Column Elimination for Capacitated Vehicle Routing Problems, CPAIOR 2023