Carnegie Mellon University
Tepper School of Business

Column Elimination

Willem-Jan van Hoeve
Carnegie Mellon University

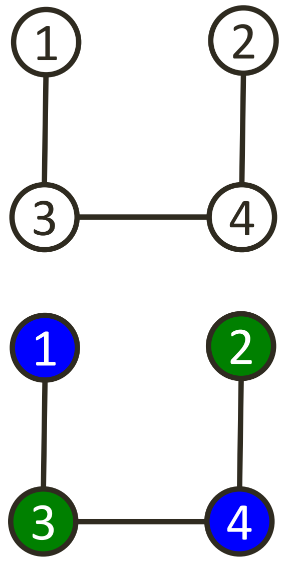
Includes joint work with Ziye Tang and Anthony Karahalios

Partially supported by ONR Grants N00014-18-1-2129 and N00014-21-1-2240, and NSF Award #1918102

Plan

- Column generation: brief introduction
  - graph coloring
- Decision diagrams: an alternative approach
  - column elimination
  - graph coloring
- More structural connections
  - vehicle routing

Graph Coloring



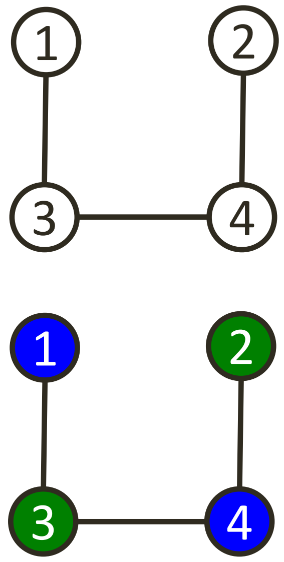
- Assign a color to each vertex
- Adjacent vertices are colored differently
- Minimize the number of colors needed

- Fundamental combinatorial optimization problem
- Many applications, e.g., rostering, scheduling, …
- Challenge for exact methods: good lower bounds

MIP formulation: Work with color classes

- Let I be the set of all independent sets (color classes)
- Binary variable x_i : use independent set i
- Ensure that each vertex is colored
- Comparatively strong LP relaxation

$$\begin{aligned} \min \quad & \sum_{i \in I} x_i \\ \text{s.t.} \quad & \sum_{i \in I} a_{ij} x_i = 1 \quad \forall j \in V \\ & x_i \in \{0,1\} \qquad \forall i \in I \end{aligned}$$



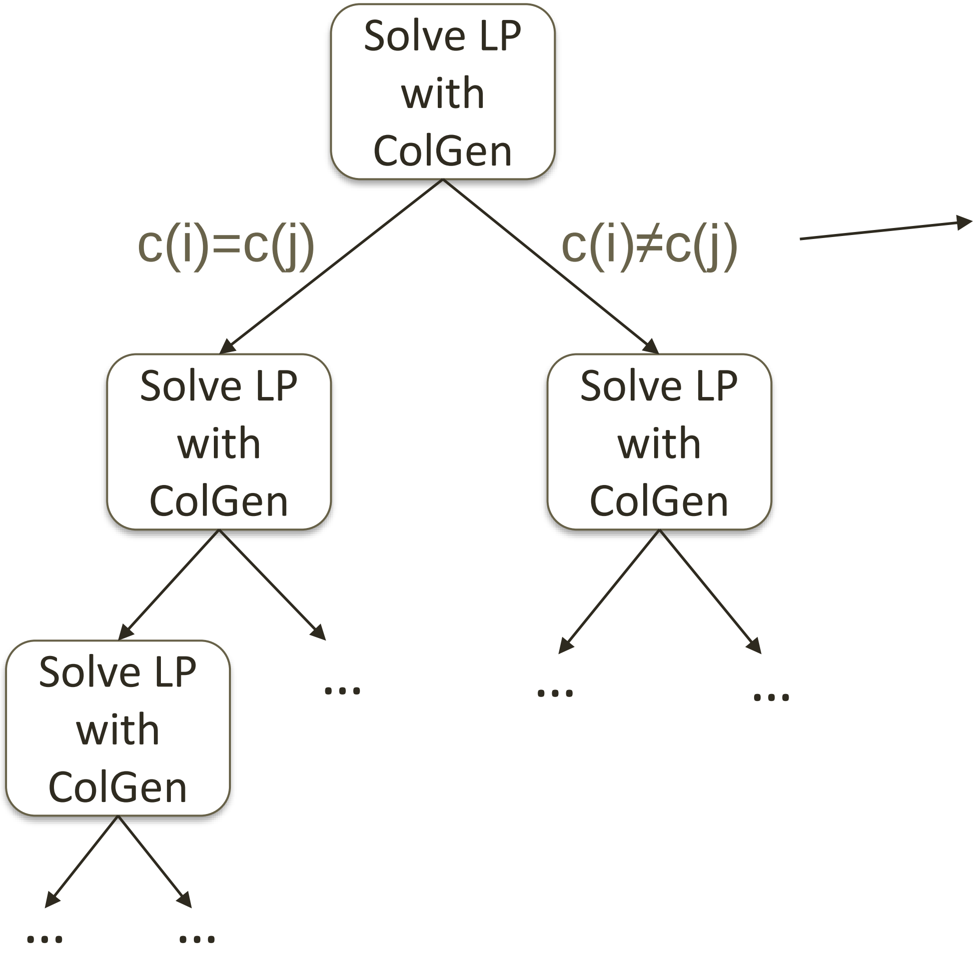
$I = \{\{1\},\{2\},\{3\},\{4\},$
$\{1,2\},\{1,4\},\{2,3\}\}$

drawback: I has exponential size

Solve LP via Column Generation

- Master Problem
  - Restricted set I of variables (‘columns’)
  - Initialize to ensure feasibility, e.g., {{1},{2},{3},{4}}
  - Solve LP relaxation: shadow price π_i for vertex i
- Pricing Problem
  - Find new LP variable (an independent set) with negative reduced cost: $1 - \sum_i \pi_i y_i < 0$
  - This is an integer program (binary y_i)
  - Add to I if it exists, otherwise Master LP solution is optimal
- Repeat until Master LP is optimal

Integer Optimality: Branch-and-Price



Branching constraint:
vertices i and j have the
same color vs. *different* color

Branch-and-Price for graph coloring:
[Mehrotra&Trick 1996] [MMT2011] [GM2012] [HCS2012] [MSJ2016] …

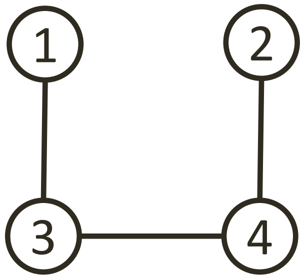
Column ‘elimination’ instead of column generation?

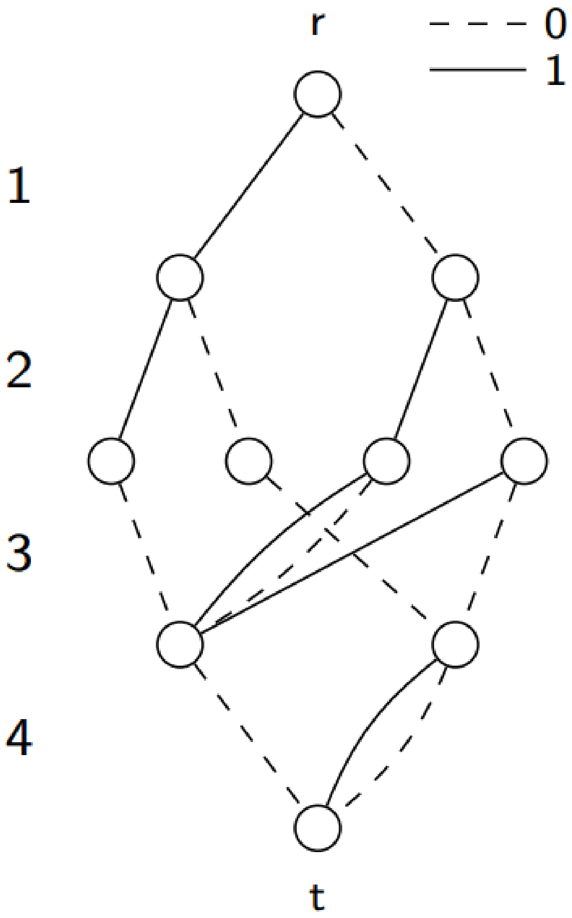
- Column generation works with *restricted* set of columns
  - no valid lower bound until optimal LP basis is found *
  - stability and convergence issues due to degenerate LP solutions
  - solving LP as MIP is not sufficient—embed in branch-and-price search
- Alternative: work with *relaxed* set of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

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

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

Representing all independent sets as decision diagram

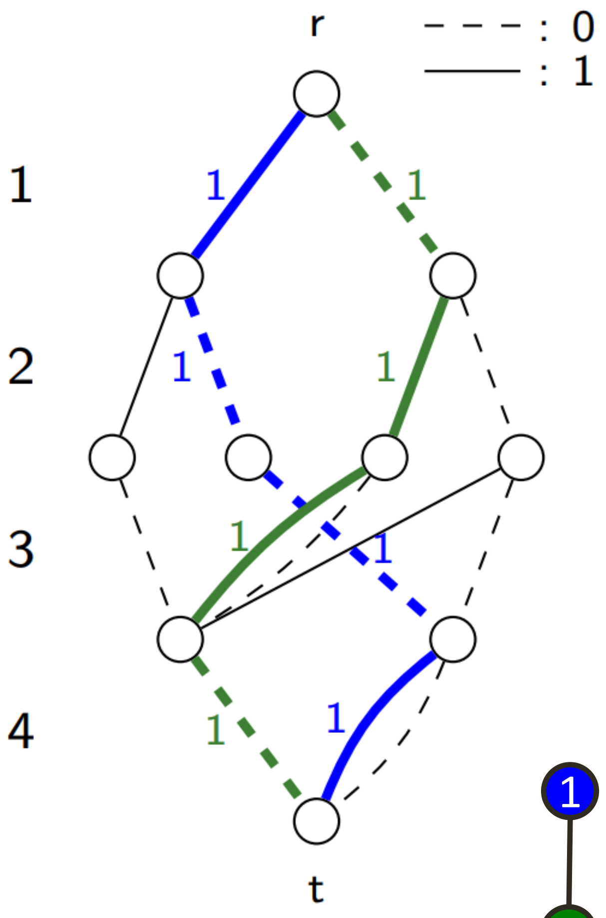


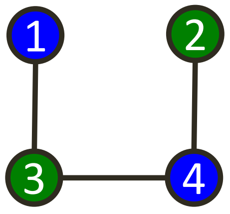


- Exact decision diagram: each r-t path corresponds to an independent set
- Prior work: compilation method that builds the unique minimum size diagram

[Bergman, Cire, vH, Hooker, 2012, 2014]

Reformulating the MIP model





- Integer variable y_a : 'flow' through arc *a*

$$(F) = \min \sum_{a \in \delta^+(r)} y_a$$ — minimize number of paths (colors)

$$\text{s.t.} \sum_{a=(u,v) \mid L(u)=j, \ell(a)=1} y_a = 1 \quad \forall j \in V$$ — one 1-arc per vertex

$$\sum_{a \in \delta^-(u)} y_a - \sum_{a \in \delta^+(u)} y_a = 0 \quad \forall u \in N \setminus \{r, t\}$$ — 'flow conservation'

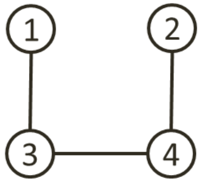
$$y_a \in \{0, 1, \ldots, n\} \quad \forall a \in A$$ — integrality

Two Main Challenges

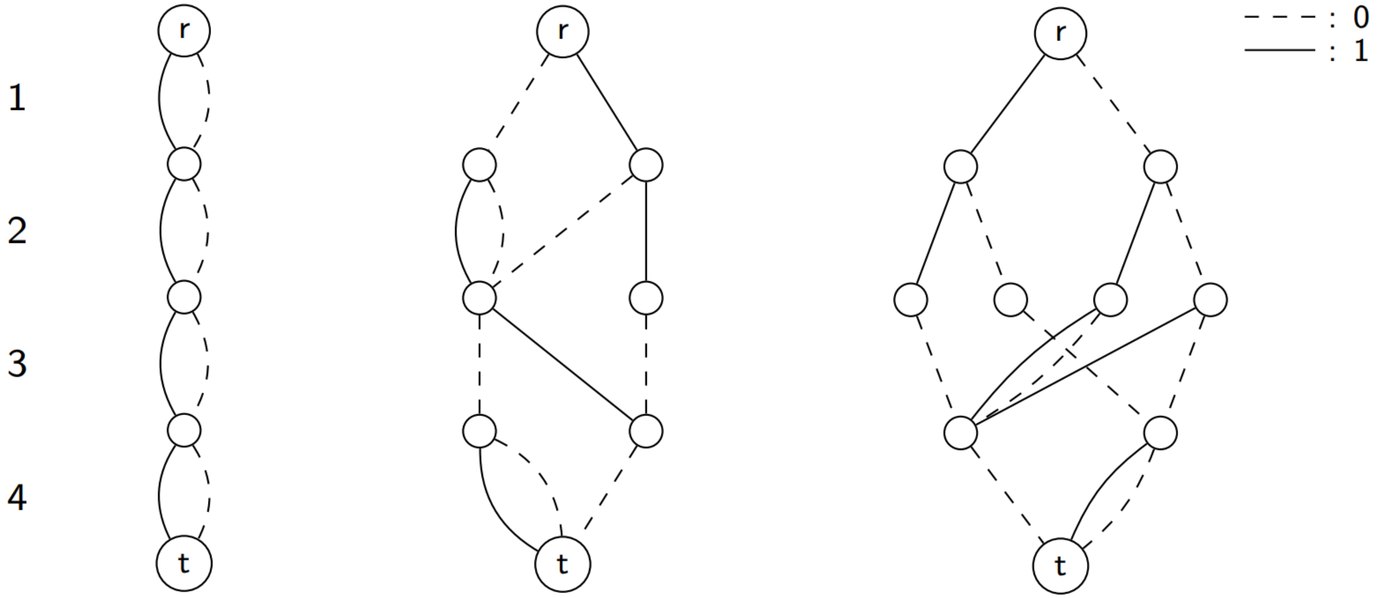
1. Exact decision diagrams can be of exponential size (in the size of the input graph)
   - Use *relaxed* decision diagrams instead
   - Provides lower bound on coloring number
2. Solving the constrained integer flow problem is NP-hard
   - Less relevant in practice: MIP solvers scale well
   - But we can also use LP relaxation (polynomial)

Exact and Relaxed Decision Diagrams

- Decision diagram *D* for problem *P* is $\begin{cases} \textit{exact} & \text{if } \mathrm{Sol}(D) = \mathrm{Sol}(P) \\ \textit{relaxed} & \text{if } \mathrm{Sol}(D) \supseteq \mathrm{Sol}(P) \end{cases}$



input graph

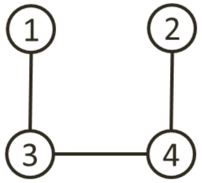


relaxed
(5 nodes)

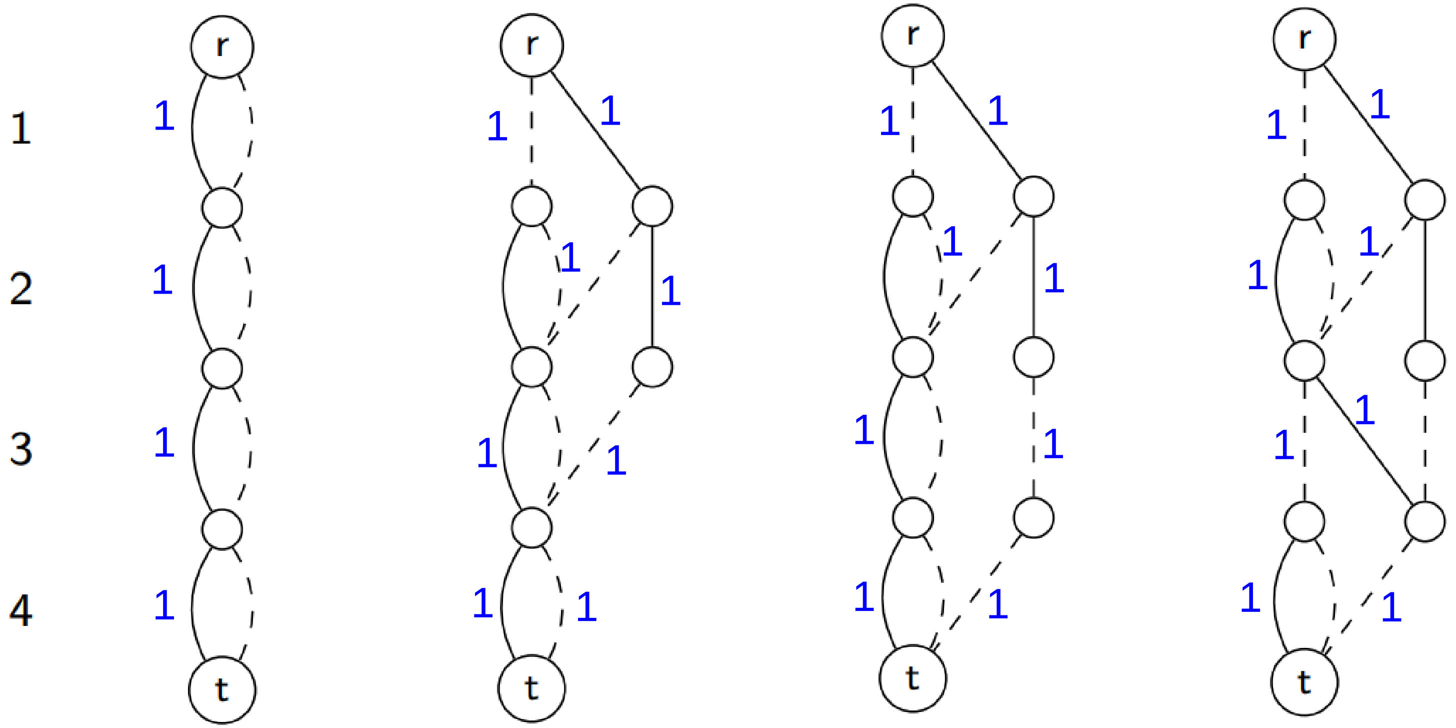
relaxed
(8 nodes)

exact
(10 nodes)

Incremental Refinement by Eliminating Conflicts



input graph



Optimal!

lower bound:	1	2	2	2
flow paths:	(1,1,1,1)	(1,1,0,1) (0,0,1,0)	(1,1,0,0) (0,0,1,1)	(1,0,0,1) (0,1,1,0)
conflicting edge:	(1,3)	(2,4)	(3,4)	-

Analysis of overall procedure

Lemma: Conflicts can be found in polynomial time (in the size of the diagram) via a path decomposition of the flow

Lemma: Eliminating *k* conflicts yields diagram of at most O(*kn*) size

- Eliminating one conflict increases each layer by at most one node

Lemma: In each iteration, compilation via conflict elimination produces a valid lower bound

Lemma: Eliminating all conflicts yields the unique exact diagram

Theorem: Algorithm terminates with an optimal solution (if time permits)

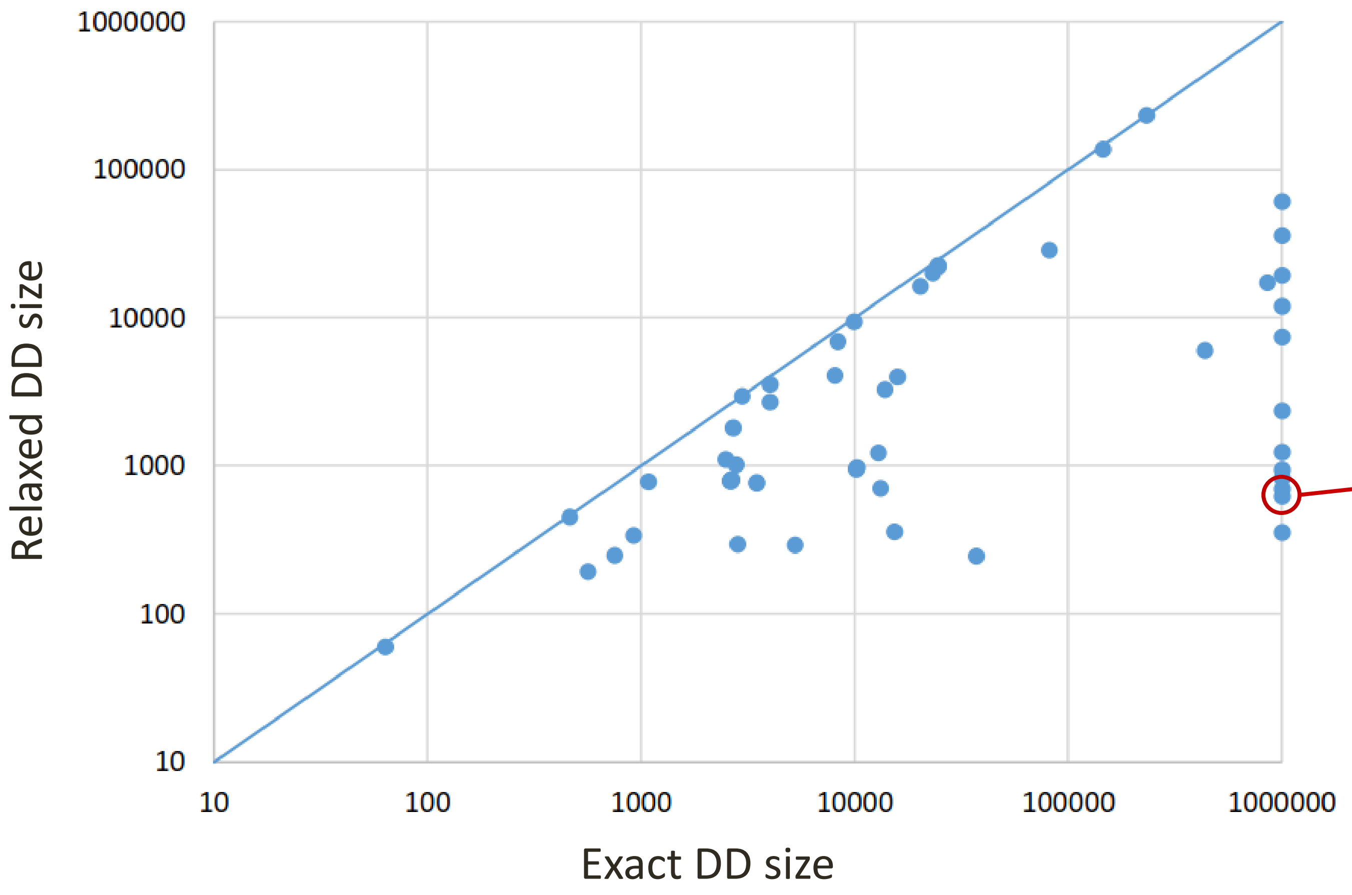
Is there any hope that this might work? Yes!

- **Theorem:** Relaxed decision diagram can be *exponentially smaller* than exact decision diagram for proving optimality

Proof sketch:

- There exists a graph coloring instance class (i.e., paths),
- and associated vertex ordering, such that
- the exact decision diagram is of exponential size
- while a polynomial-size relaxed decision diagram exists that proves optimality

Evaluation on DIMACS benchmark instances



- Relaxed decision diagram can be orders of magnitude smaller than exact decision diagram to prove optimality, but not always
- DSJR500.1 (n=500, m=3,555)
  - Exact DD: ≥1M nodes
  - Relaxed DD: 627 nodes

(Each instance is solved to optimality by at least one of the two methods)

Column Elimination: How to prove optimality faster?

1. Add upper bound heuristics

[vH, *Math. Prog.* 2021]

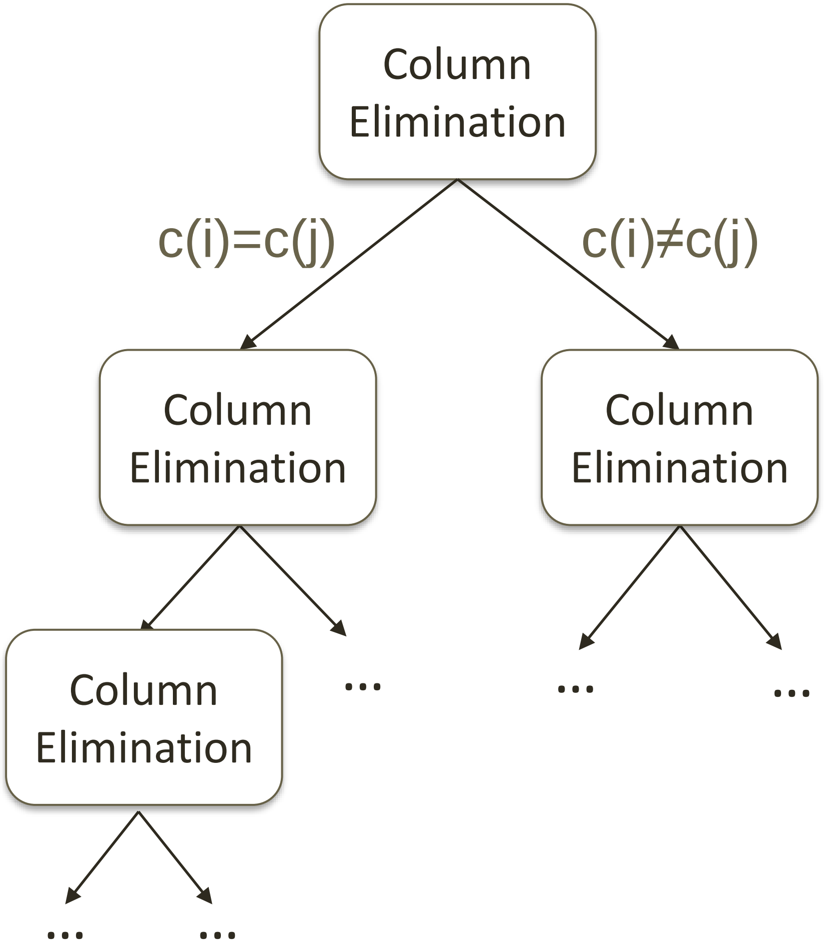
2. Two phases: first solve LPs, then solve MIPs

3. Run portfolio approach over multiple orderings
   – Vertex ordering can have dramatic impact

[Karahalios & vH, *Constraints* 2022]

4. Embed column elimination in branch-and-bound

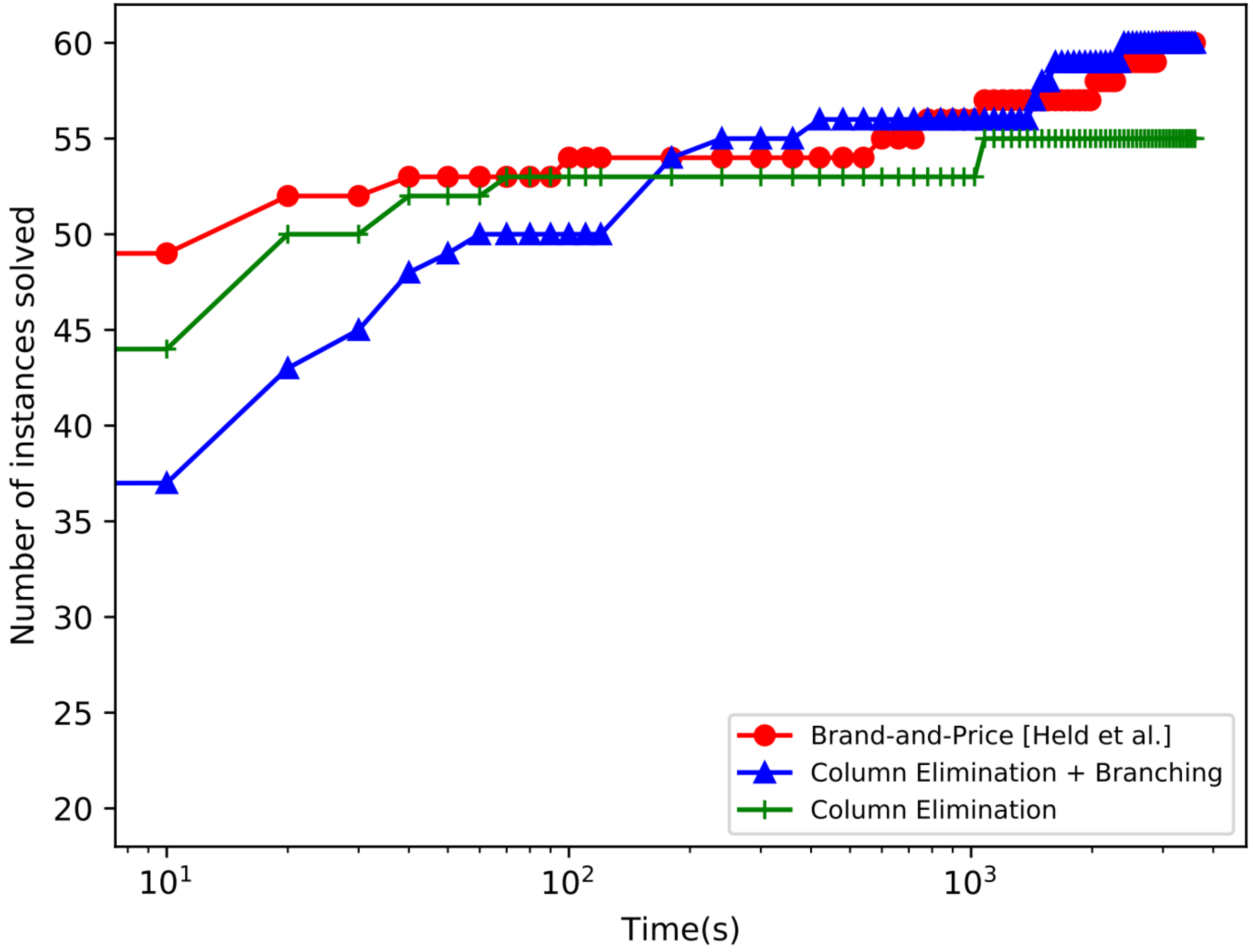
Branch-and-Bound with Column Elimination



Design choices:

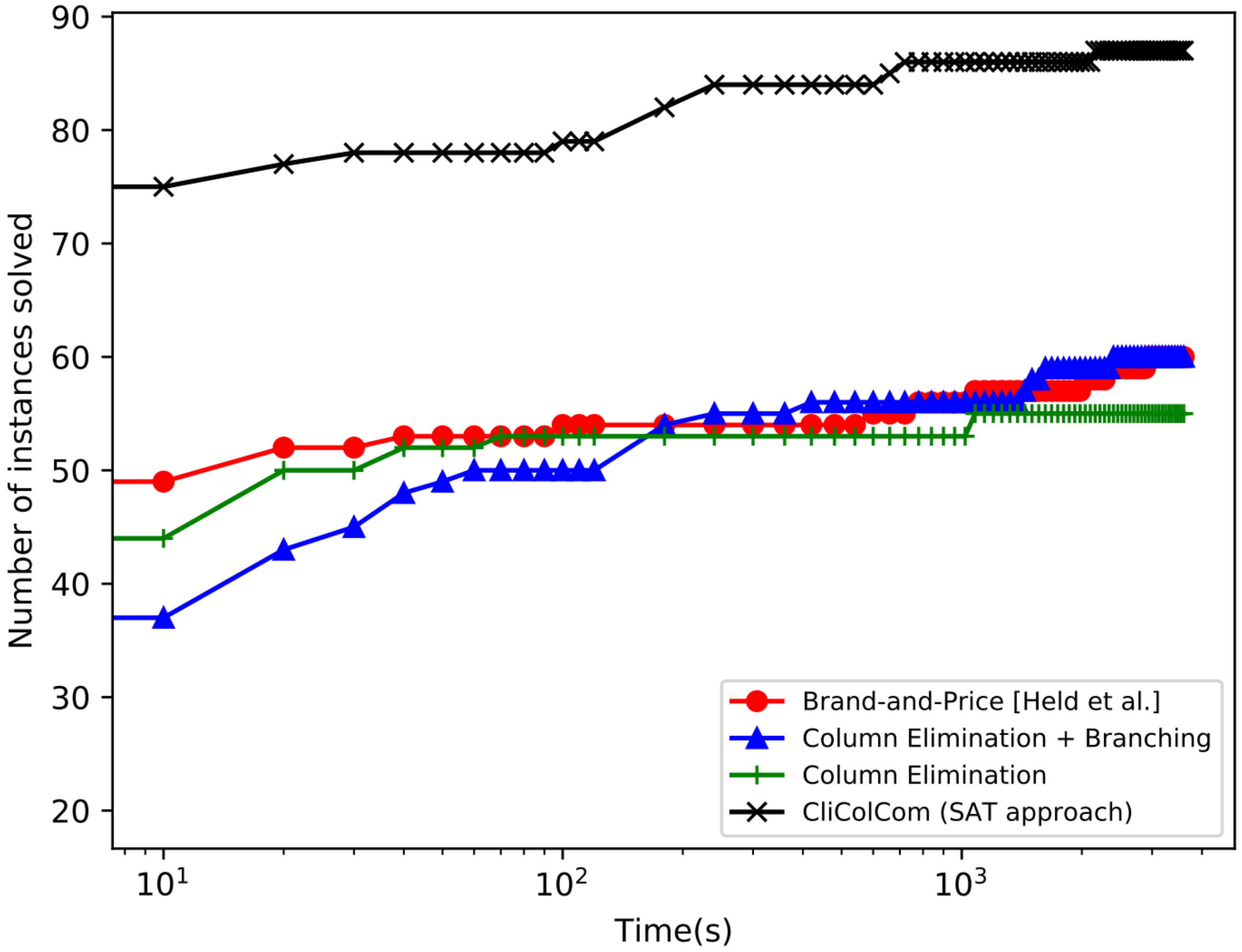
- Zykov branching (or Ryan/Foster) on two vertices that do not share an edge with highest sum of degrees
- Best-bound node processing order
- Branch after 20s of not improving neither lower nor upper bound

Comparison with Branch-and-Price



- Benchmark: DIMACS Coloring instances
- Branch-and-Price: [Held, Cook, & Sewell, 2012]
- Column Elimination: Uses portfolio of orderings [Karahalios & vH, 2022]

Comparison with State of the Art



- Benchmark: DIMACS Coloring instances
- Branch-and-Price: [Held, Cook, & Sewell, 2012]
- Column Elimination: Uses portfolio of orderings [Karahalios & vH, 2022]
- CliColCom: [Heule, Karahalios, & vH, CP2022]

Generalization

- Column elimination via decision diagrams is a promising alternative to column generation
- Q: What is needed to apply this to other problems?
- A: Dynamic programming formulation of ‘pricing problem’
  - Provides the transition rules to compile the decision diagram
  - Instead of solving for one column, we explicitly represent all columns
  - Solve the LP (or IP) over the entire set of columns! No need to price.
- Next application: Vehicle Routing

Case Study: Truck-Drone Routing

- One truck + one drone
- Possible legs include:

  truck, drone, combined

- Example route duration =

  max{1, 0.5+0.5} +

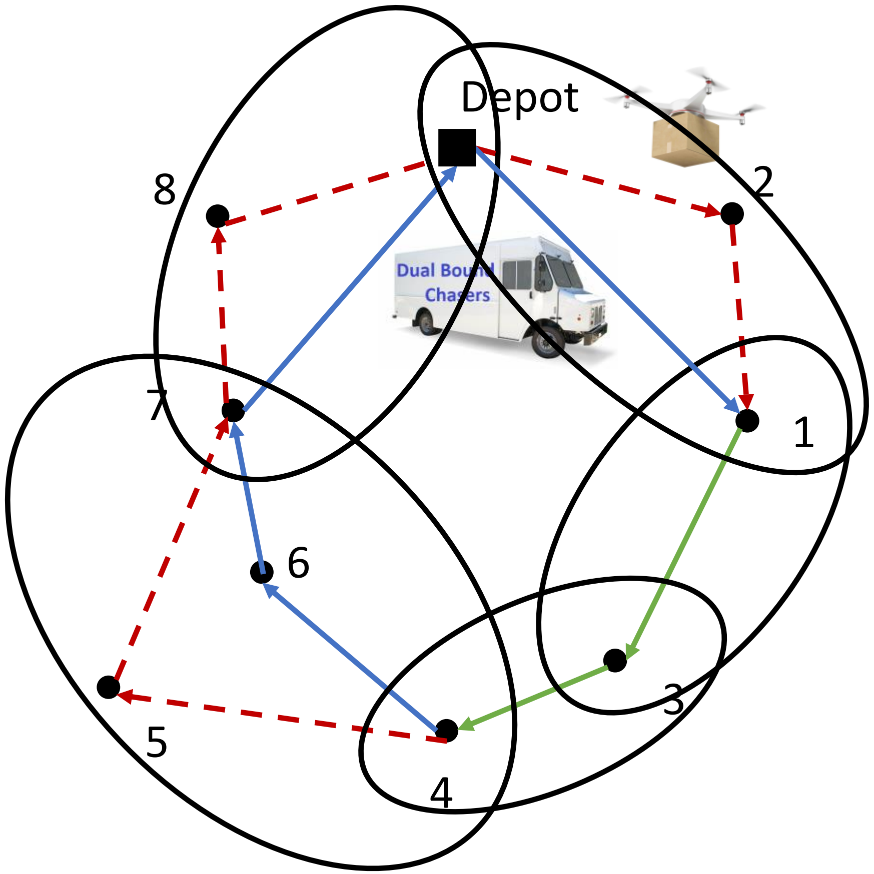
  1 +

  1 +

  max{1+1, 0.5+0.5} +

  max{1, 0.5+0.5}

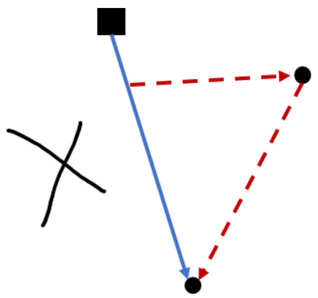
  = 6



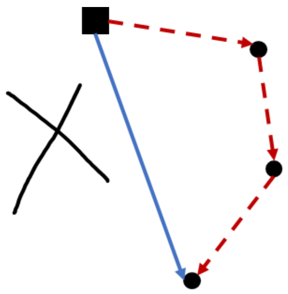
truck speed: 1 unit per edge
drone speed: 0.5 unit per edge

Definition of TSP-D

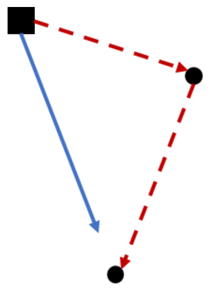
- TSP-D: Traveling Salesperson with a Drone
- Drone speed = α * truck speed (for some fixed α)
- Goal: minimize route duration
- Assumptions:



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]

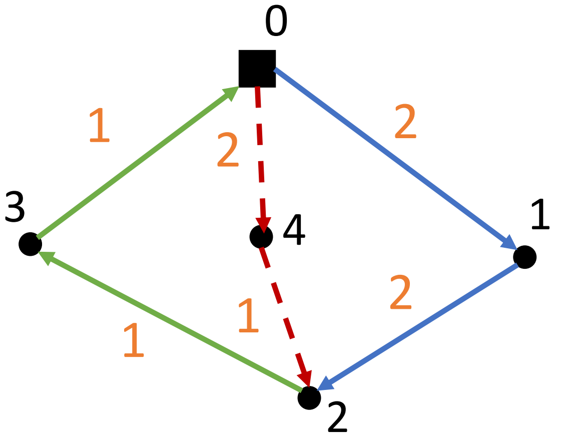
Dynamic Programming Model for TSP-D

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

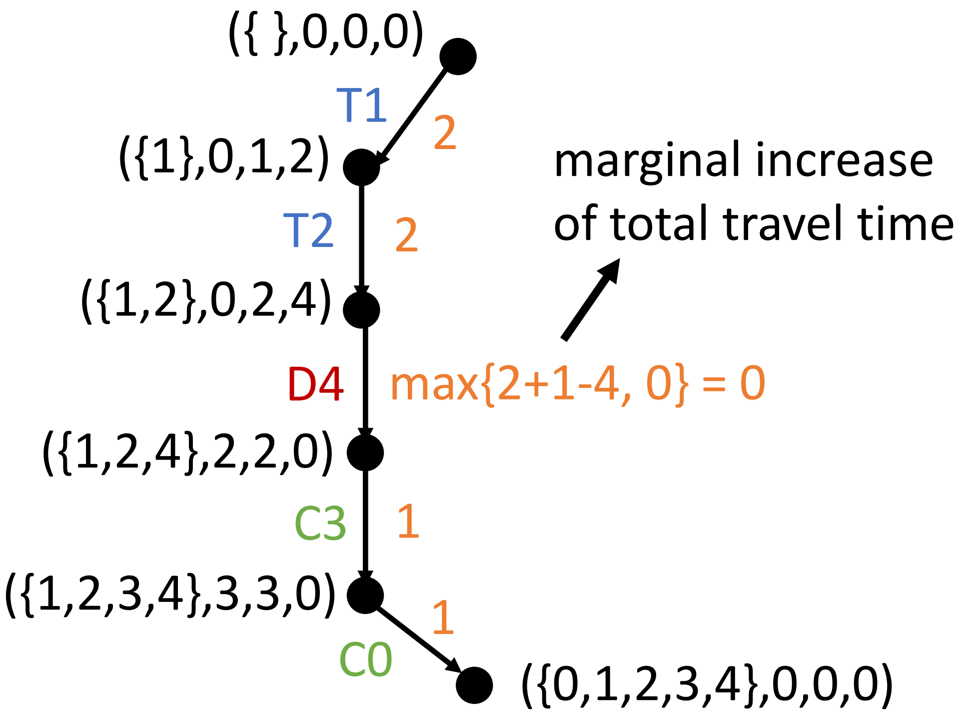
- S = customers visited so far
- LC = latest location visited by both vehicles
- LT = latest location visited by truck alone
- t = time spent by the truck traveling alone since leaving LC

Set of **controls**

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



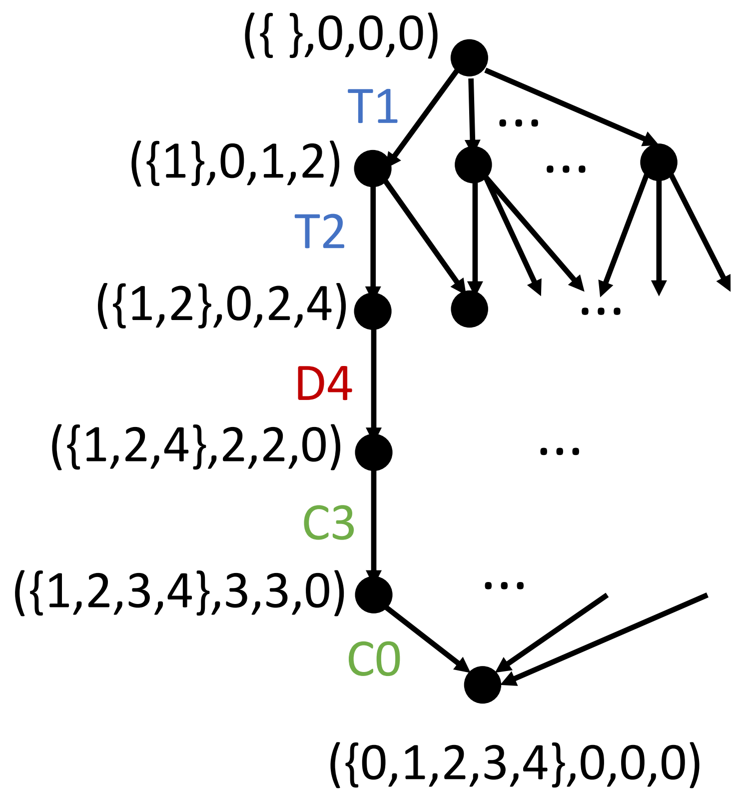
Route: T1, T2, D4, C3, C0



[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):

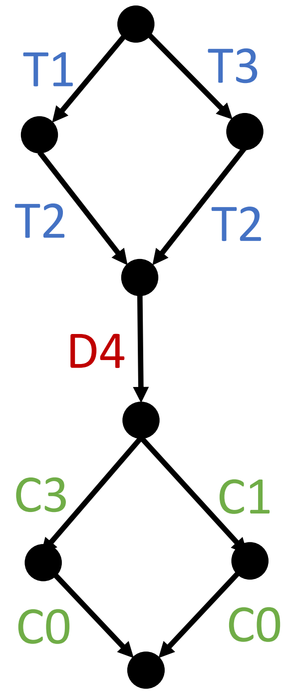
$$
\begin{aligned}
\min \quad & \sum_{a \in A_D} \gamma_a y_a \\
\text{s.t.} \quad & \sum_{a \in \delta^+(u)} y_a = \sum_{a \in \delta^-(u)} y_a, \quad \forall u \in V_D, u \neq r, t \\
& \sum_{a \in \delta^+(r)} y_a = 1 \\
& \sum_{a \in \delta^-(t)} y_a = 1 \\
& \sum_{l(a) \text{ is a visit to customer i}} y_a = 1, \quad \forall i \in N \\
& y_a \in \{0, 1\}, \quad \forall a \in A_D
\end{aligned}
$$

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

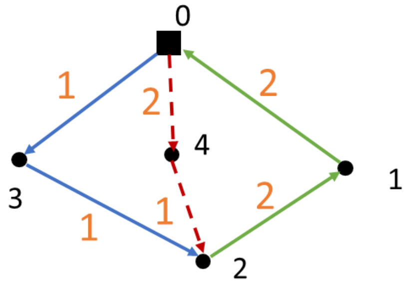
Theorem: SPLP(R), CFLP(D_R), and LR(D_R) have the same optimal objective value

Going Beyond the ng-Route Bound

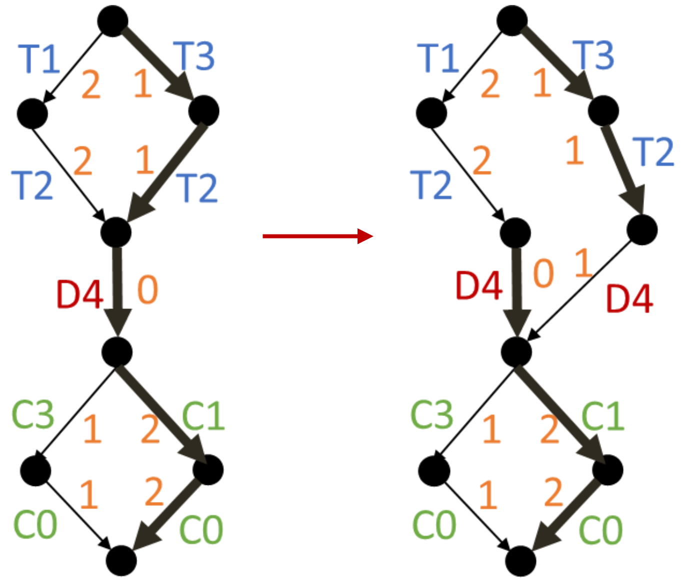
- Resolve conflicts along solution paths by refining the DD

Type 1: objective function

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

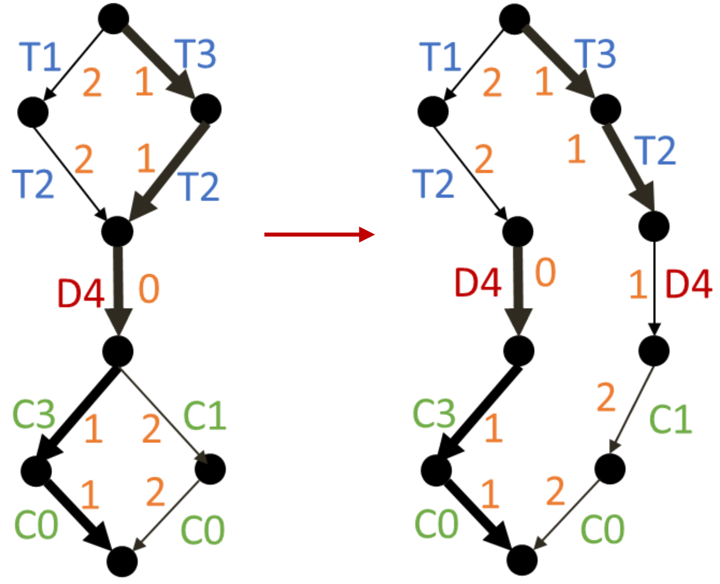


Duration = 7



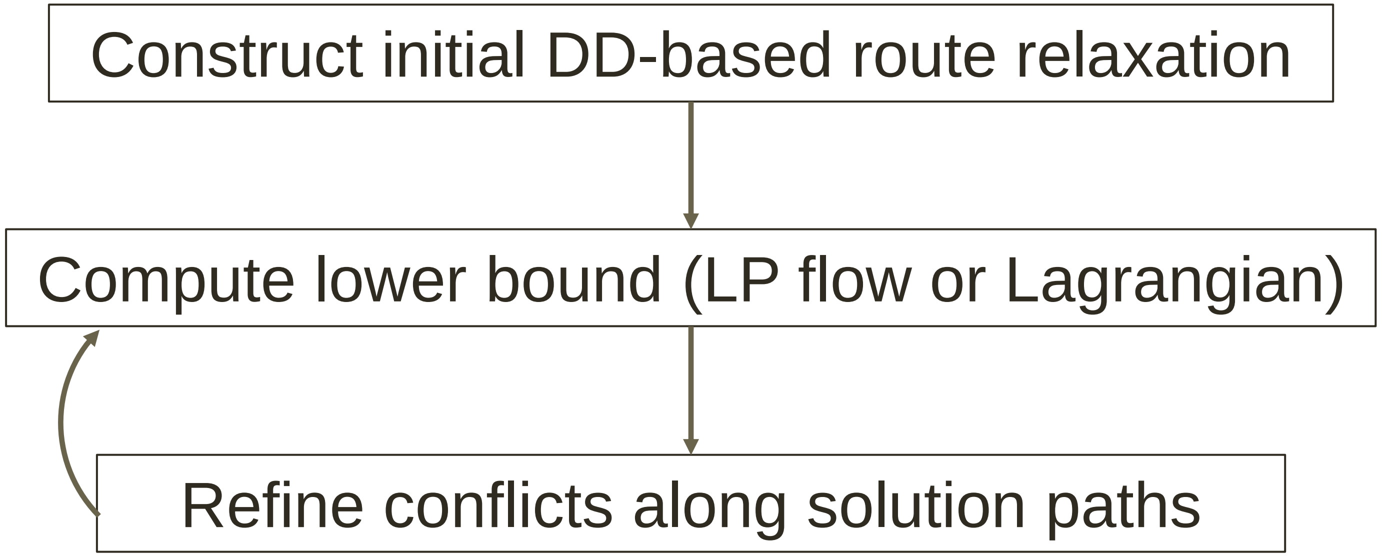
Path length = 6

Type 2: repeated visits



Customer 3 repeated

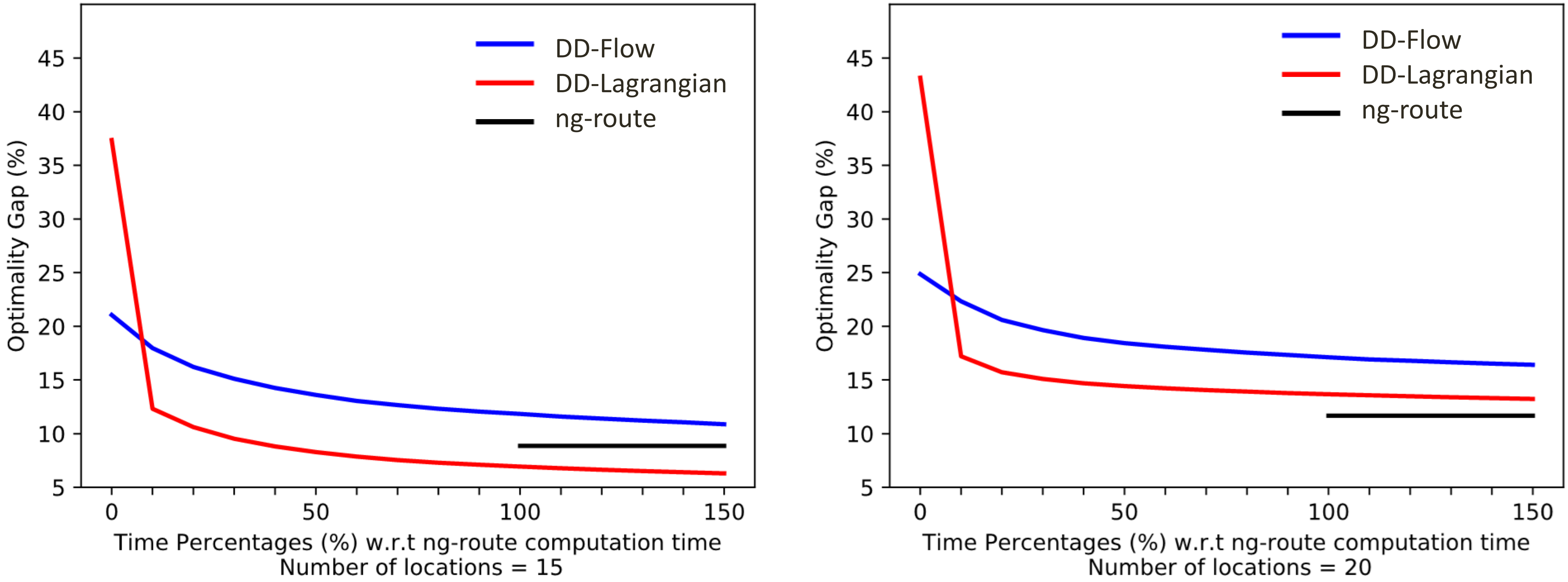
Overall Framework



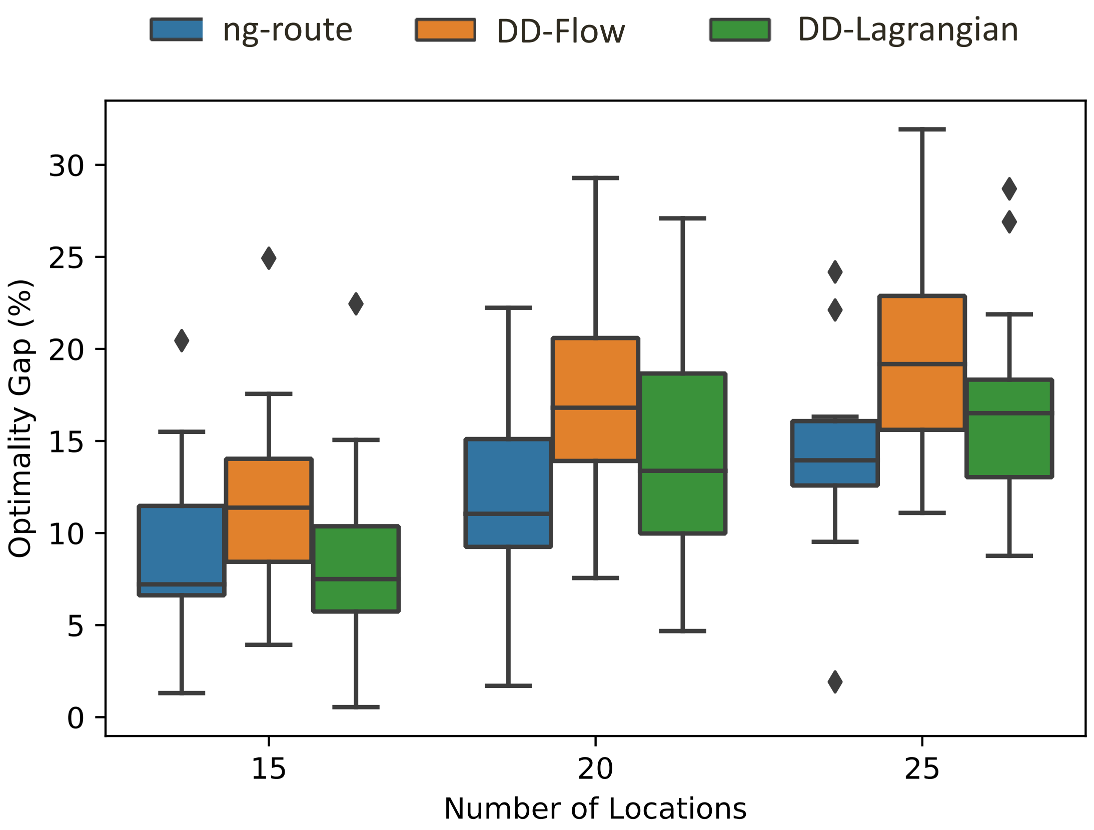
Experimental Evaluation on TSP-D

- 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, CPAIOR19]

Optimality gap improvement over time

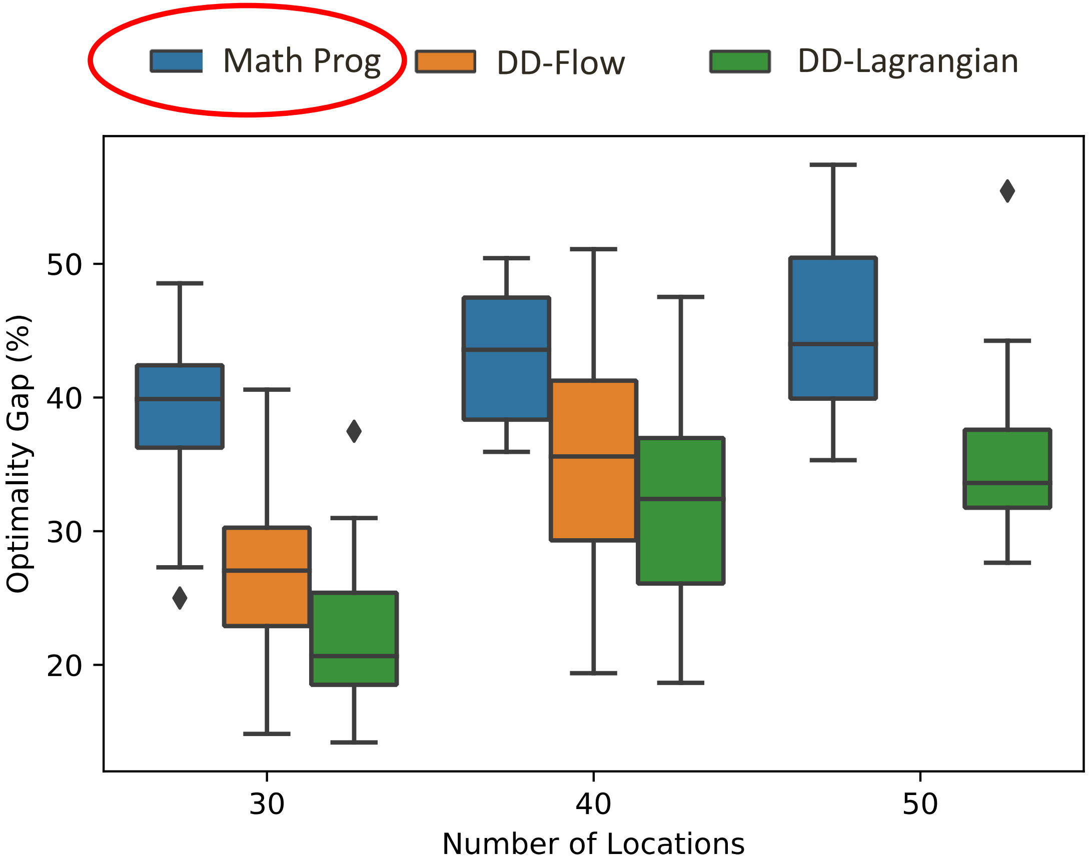


Optimality gap for varying problem sizes



(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]

Conclusion

- Column Elimination with relaxed decision diagrams can be used as an alternative for column generation/branch-and-price
  - Replaces pricing problem with incremental refinement by eliminating conflicts
  - Provides a lower bound at each iteration. Can solve as LP or MIP.
  - Avoids LP degeneracy and related convergence and stability issues
  - When defined on the dynamic program for pricing problem it produces the same set partitioning LP bound
- Competitive results on graph coloring and TSP+drone routing