Carnegie Mellon University
Tepper School of Business

Column elimination: An alternative to column generation via relaxed decision diagrams

Willem-Jan van Hoeve
Carnegie Mellon University

Includes joint work with Ziye Tang and Anthony Karahalios

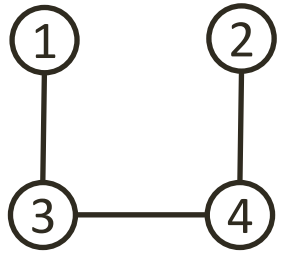
Partially supported by ONR Grants N00014-18-1-2129 and N00014-21-1-2240, and NSF Award #1918102

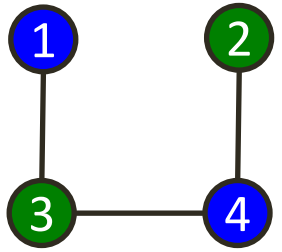
Plan

- Column generation: brief introduction
  - graph coloring
- Decision diagrams: an alternative approach
  - 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

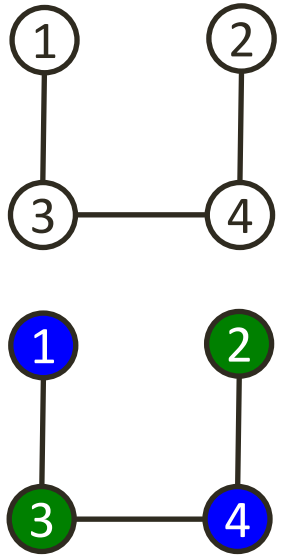




- Fundamental combinatorial optimization problem
- Many applications, e.g., rostering, scheduling, …
- Challenge for exact methods: good lower bounds

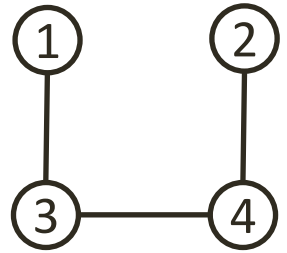
MIP formulation? Attempt 1

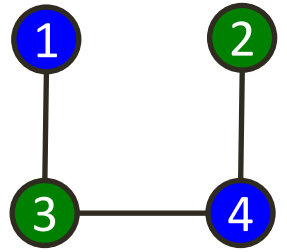
- Attempt 1
  - binary variable y_{ik}: vertex i has color k
  - $y_{ik} + y_{jk} \leq 1$ for all edges {i,j} and colors k
  - weak LP relaxation
- (Stronger formulations exist using this approach)



MIP formulation? Attempt 2

- Attempt 2
  - 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
  - stronger LP relaxation





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

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

drawback: I has exponential size

Solve LP Model via Column Generation

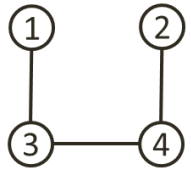
- Master Problem
  - Restricted set I of variables ('columns')
  - Initialize to ensure feasibility, e.g., {{1},{2},{3},{4}}
  - Solve LP relaxation: shadow price π_i for vertex i
- Pricing Problem
  - Find new LP variable (an independent set) with negative reduced cost: $1 - \sum_i \pi_i y_i < 0$
  - 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

$$\begin{aligned}
\min \quad & x_1 + x_2 + x_3 + x_4 \\
& x_1 = 1 \\
& x_2 = 1 \\
& x_3 = 1 \\
& x_4 = 1 \\
& x_i \geq 0 \quad (i = 1, \ldots, 4)
\end{aligned}$$

$$\text{objective} = 4$$
$$x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1$$
$$\pi_1 = 1, \pi_2 = 1, \pi_3 = 1, \pi_4 = 1$$

$$\begin{aligned}
\min \quad & 1 - y_1 - y_2 - y_3 - y_4 \\
& y_1 + y_3 \leq 1 \\
& y_2 + y_4 \leq 1 \\
& y_3 + y_4 \leq 1 \\
& y_i \in \{0, 1\} \quad (i = 1, \ldots, 4)
\end{aligned}$$

$$\text{reduced cost} = -1$$
$$y_1 = 1, y_2 = 1, y_3 = 0, y_4 = 0$$



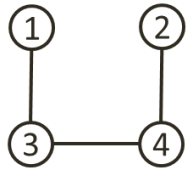
Solve LP Model via Column Generation

- 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

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 + x_4 + x_5 \\ & x_1 + x_5 = 1 \\ & x_2 + x_5 = 1 \\ & x_3 = 1 \\ & x_4 = 1 \\ & x_i \geq 0 \quad (i = 1, \ldots, 5) \end{aligned}$$

$$\text{objective} = 3$$
$$x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 1$$
$$\pi_1 = 1, \pi_2 = 0, \pi_3 = 1, \pi_4 = 1$$

$$\begin{aligned} \min \quad & 1 - y_1 - y_3 - y_4 \\ & y_1 + y_3 \leq 1 \\ & y_2 + y_4 \leq 1 \\ & y_3 + y_4 \leq 1 \\ & y_i \in \{0, 1\} \quad (i = 1, \ldots, 4) \end{aligned}$$



$$\text{reduced cost} = -1$$
$$y_1 = 1, y_2 = 0, y_3 = 0, y_4 = 1$$

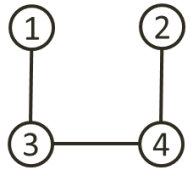
Solve LP Model via Column Generation

- 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

$$\begin{aligned}
\min \quad & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
& x_1 \qquad\qquad\quad + x_5 \quad + x_6 \quad = \quad 1 \\
& \quad x_2 \qquad\quad + x_5 \qquad\quad = \quad 1 \\
& \qquad x_3 \qquad\qquad\qquad\quad = \quad 1 \\
& \qquad\quad x_4 \qquad\quad + x_6 \quad = \quad 1 \\
& x_i \geq 0 \quad (i = 1, \ldots, 6)
\end{aligned}$$

$$\text{objective} = 3$$
$$x_1 = 0, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 1, x_6 = 0$$
$$\pi_1 = 0, \pi_2 = 1, \pi_3 = 1, \pi_4 = 1$$

$$\begin{aligned}
\min \quad & 1 - y_2 - y_3 - y_4 \\
& y_1 + y_3 \leq 1 \\
& y_2 + y_4 \leq 1 \\
& y_3 + y_4 \leq 1 \\
& y_i \in \{0, 1\} \quad (i = 1, \ldots, 4)
\end{aligned}$$



$$\text{reduced cost} = -1$$
$$y_1 = 0, y_2 = 1, y_3 = 1, y_4 = 0$$

Solve LP Model 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

$$\min \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7$$

$$\begin{aligned} x_1 \qquad\qquad + x_5 \quad + x_6 \qquad\quad &= 1 \\ x_2 \qquad\qquad + x_5 \qquad\quad + x_7 \;&= 1 \\ x_3 \qquad\qquad\qquad\quad + x_7 \;&= 1 \\ x_4 \qquad\quad + x_6 \qquad\quad &= 1 \end{aligned}$$

$$x_i \geq 0 \quad (i = 1, \ldots, 7)$$

objective $= 2$

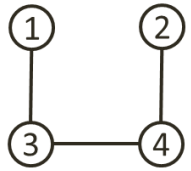
$x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0, x_5 = 0, x_6 = 1, x_7 = 1$

$\pi_1 = 1, \pi_2 = 0, \pi_3 = 1, \pi_4 = 0$

$$\min \quad 1 - y_1 - y_3$$

$$\begin{aligned} y_1 + y_3 &\leq 1 \\ y_2 + y_4 &\leq 1 \\ y_3 + y_4 &\leq 1 \end{aligned}$$

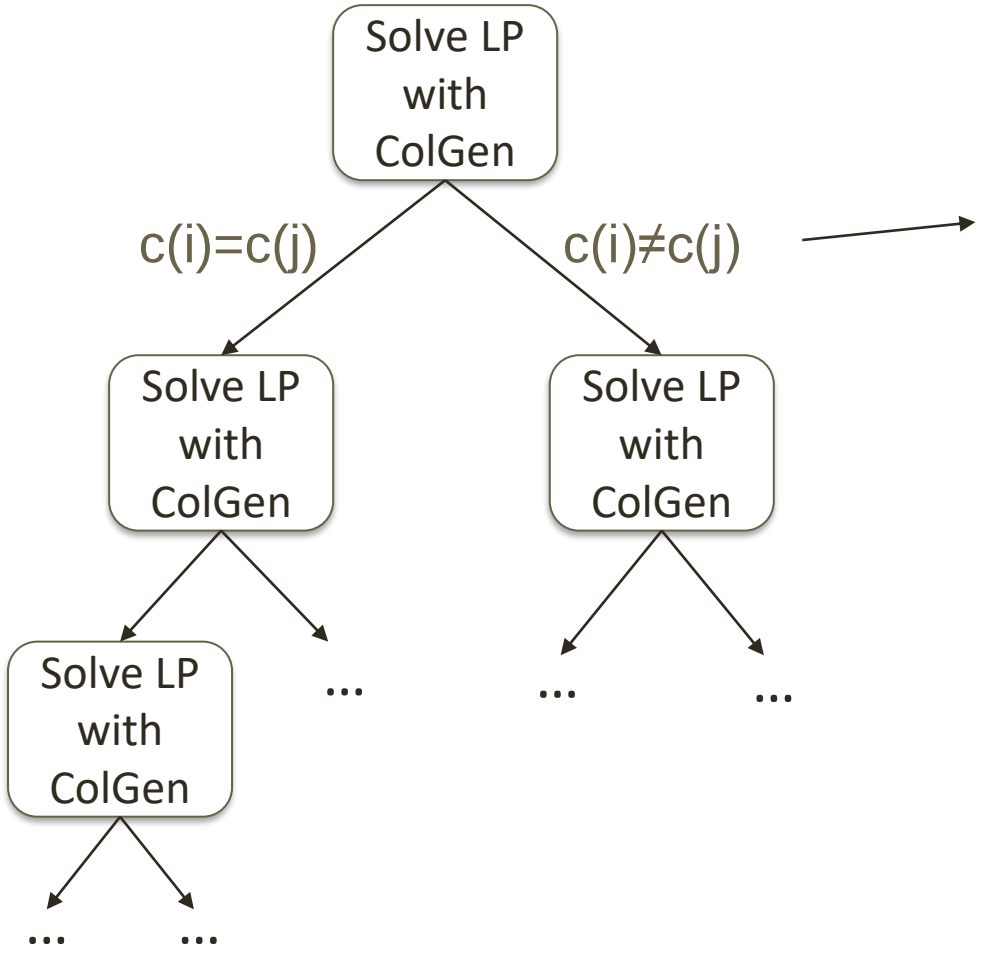
$$y_i \in \{0, 1\} \quad (i = 1, \ldots, 4)$$



reduced cost $= 0$

$y_1 = 1, y_2 = 0, y_3 = 0, y_4 = 0$

Integer Optimality: Branch-and-Price



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

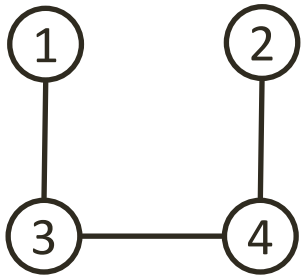
Column generation/branch-and-price for graph coloring: [Mehrotra&Trick 1996] [MMT2011,GM2012,HCS2012,MSJ2016]

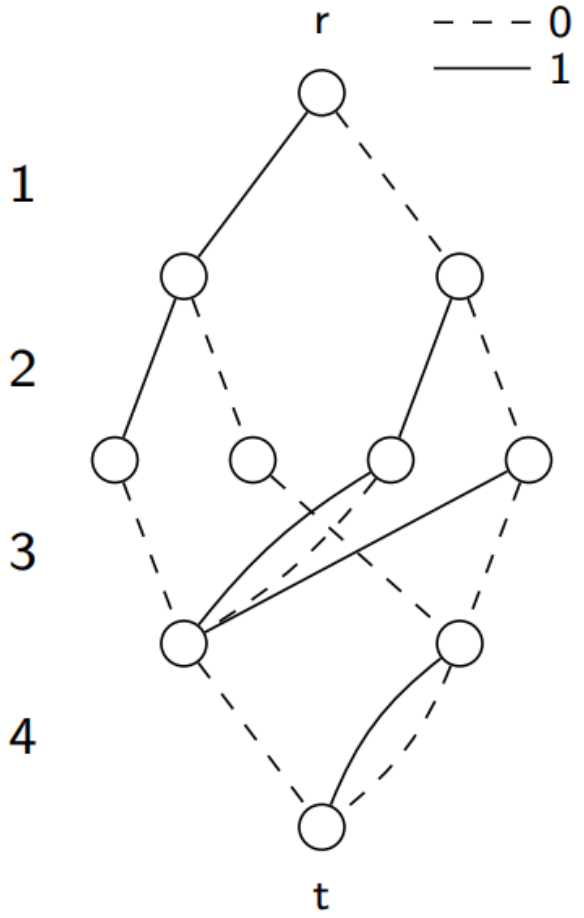
Column ‘elimination’ instead of column generation?

- 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, *Mathematical Programming*, 2021]

Representing all independent sets as decision diagram

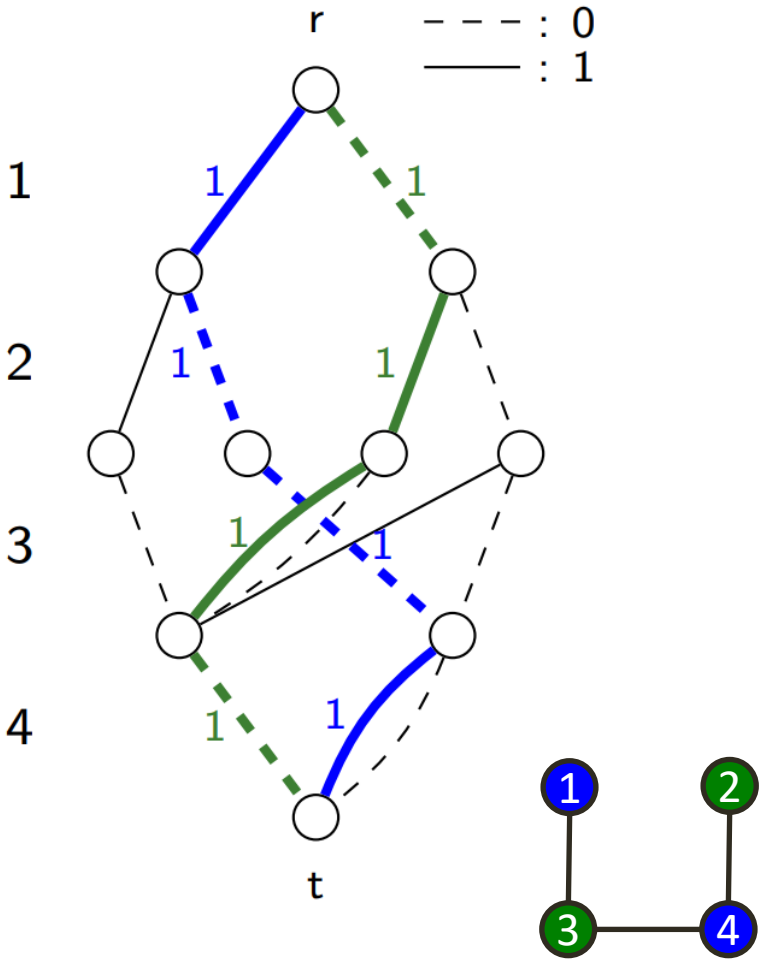




- Exact decision diagram: each r-t path corresponds to an independent set
- Prior work: compilation method that builds the unique minimum size diagram

  [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)

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

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

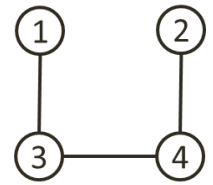
$$y_a \in \{0,1,\ldots,n\} \qquad \forall a \in A$$ — integrality

Two Main Challenges

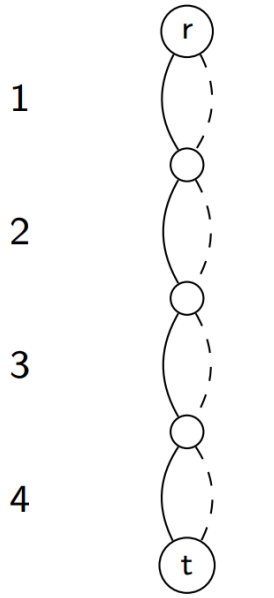
1. Exact decision diagrams can be of exponential size (in the size of the input graph)
   - Use *relaxed* decision diagrams instead
   - Provides lower bound on coloring number
2. Solving the constrained integer flow problem is NP-hard
   - Less relevant in practice: MIP solvers scale well
   - But we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

Exact and Relaxed Decision Diagrams

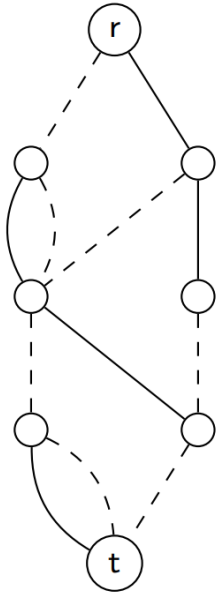
- Decision diagram *D* for problem *P* is $\begin{cases} \textit{exact} & \text{if } \mathrm{Sol}(D) = \mathrm{Sol}(P) \\ \textit{relaxed} & \text{if } \mathrm{Sol}(D) \supseteq \mathrm{Sol}(P) \end{cases}$



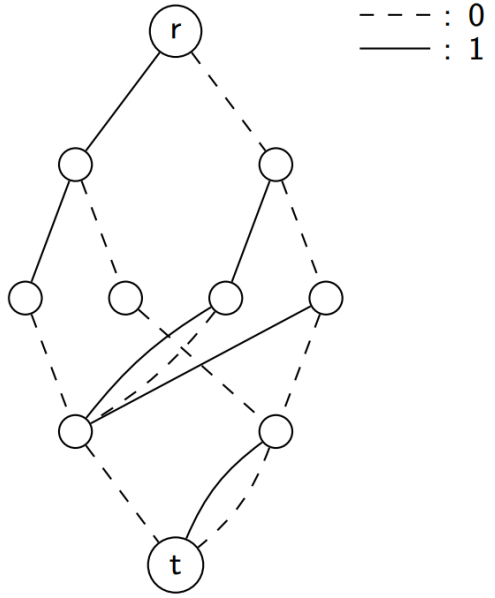
input graph



relaxed
(5 nodes)

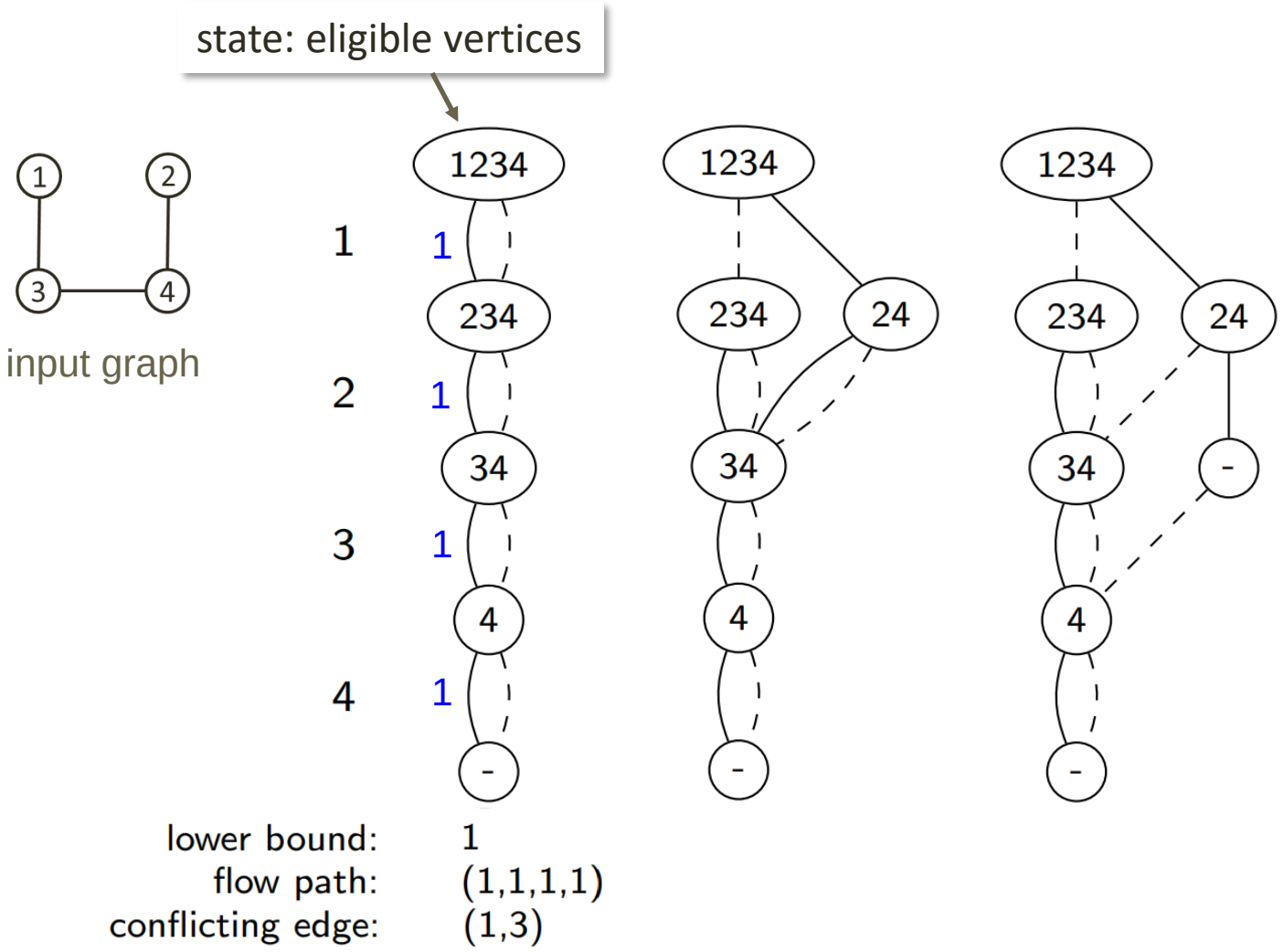


relaxed
(8 nodes)



exact
(10 nodes)

Incremental Refinement by Separating Conflicts

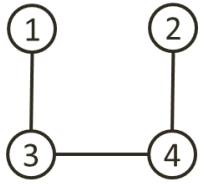


Lemma: Refinement increases each layer by at most one node

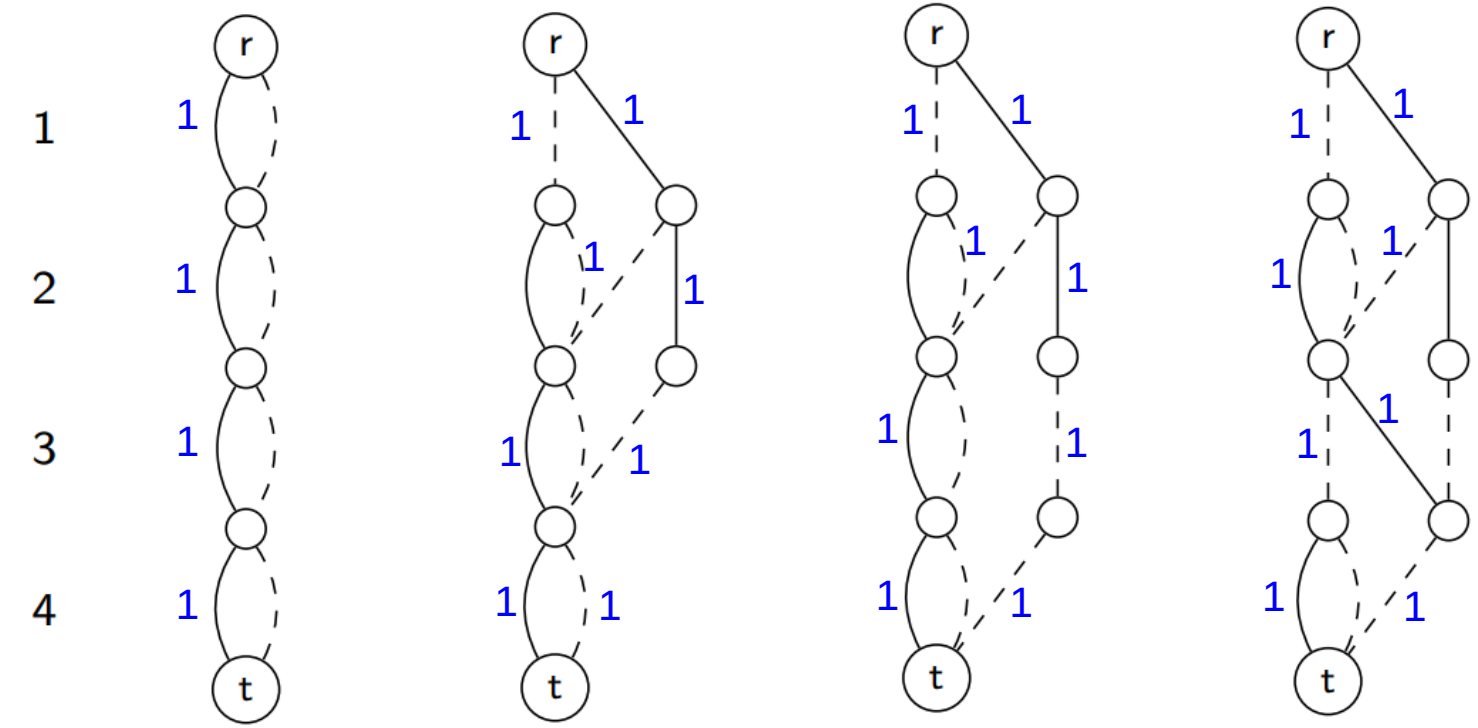
Corollary: Separating *k* conflicts yields diagram of at most O(*kn*) size

Lemma: Repeated application yields the unique exact diagram

Compilation by Iterative Conflict Separation



input graph



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)	-

Optimal!

Analysis of overall procedure

1. Conflicts can be found in polynomial time (in the size of the diagram) via a **path decomposition** of the flow
2. Separating *k* conflicts yields diagram of at most O(*kn*) size
3. In each iteration, compilation via conflict separation produces a **valid lower bound**
4. 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

Overall Algorithm with Lower and Upper Bounds

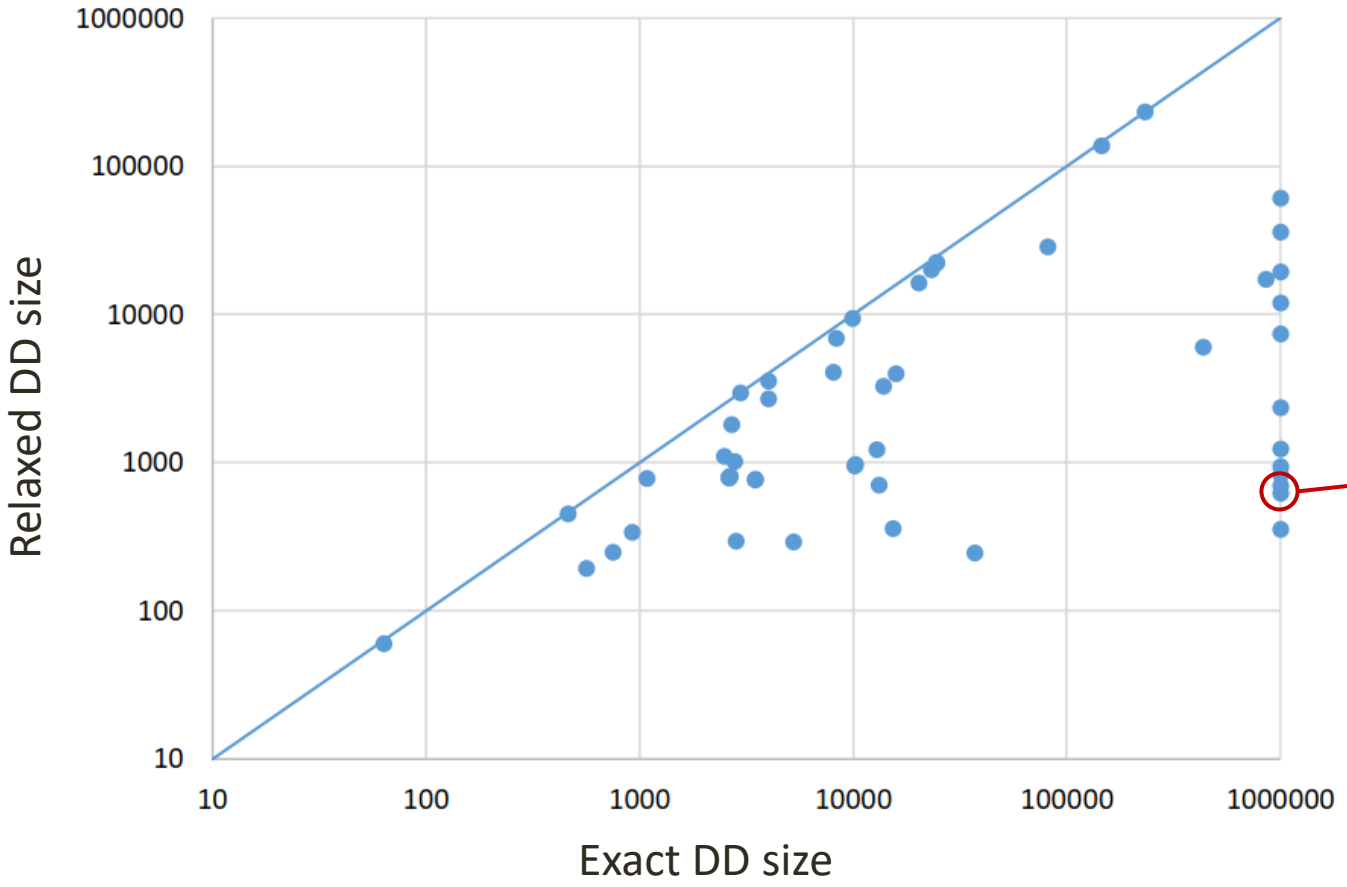
```
1  input: input graph G = (V, E)
2  output: chromatic number of G
3  begin
4      initialize width-1 decision diagram D
5      lowerBound ← 0
6      upperBound ← |V|
7      while lowerBound < upperBound do
8          solve flow model (F) with decision diagram D
9          lowerBound ← obj(F)
10         apply primal heuristic to determine coloring C
11         upperBound ← min(upperBound, |C|)
12         apply path decomposition algorithm to determine conflict (j, k) along path P
13         if no conflict is detected then upperBound ← obj(F)
14         else separate conflict (j, k) along path P in D
15     return lowerBound
```

Implementation details

- C++ implementation, using IBM ILOG CPLEX 12.9 for LP/MIP
- Design choices
  - Vertex ordering*: select most-connected vertex first (degree as tie-breaker)
  - Apply LP relaxation first, until no more conflicts found (then use MIP)
  - Separate multiple conflicts per iteration (on distinct paths)
  - Apply heuristic based on relaxed decision diagram to find upper bound (can help prove optimality earlier)
- Benchmark: 137 DIMACS Instances (time limit 3,600s)

* We also developed a *portfolio* approach using multiple orderings [Karahalios&vH, 2021]

Exact versus Relaxed Decision Diagrams



- 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

Comparison with Branch-and-Price

- Compare with code by Held, Cook, and Sewell [HCS, 2012]

	HCS	DD
Instances solved optimally	60	50
Lower bounds returned	87	137
Best known lower bounds	70	73
Best known upper bounds	82	80

	HCS better than DD	Same value	DD better than HCS
Lower bounds	21	65	51
Upper bounds	37	87	13

Results on Open DIMACS Instances

						Relaxed DD		
Instance	n	m	d	$\underline{\chi}$	$\overline{\chi}$	LB	time (s)	LB gap
C2000.9	2000	1799532	0.90	98	400	**145**	4.7 days	-0.48
DSJC500.9	500	112437	0.90	123	126	123	27.3	0
latin_square_10	900	307350	0.76	90	97	90	7.7	0
queen16_16	256	6320	0.19	16	17	16	0.0	0
wap02a	2464	111742	0.04	40	42	40	3.1	0
wap03a	4730	286722	0.03	40	47	40	6.1	0
wap04a	5231	294902	0.02	40	42	40	7.3	0
wap07a	1809	103368	0.06	40	41	40	291.2	0
wap08a	1870	104176	0.06	40	42	40	3224.2	0
wap01a	2368	110871	0.04	41	43	40	8.1	0.02
r1000.1c	1000	485090	0.97	96	98	88	2985.7	0.08
DSJC250.1	250	3218	0.10	6	8	5	0.0	0.17
1-Insertions_6	607	6337	0.03	4	7	3	0.0	0.25
3-Insertions_5	1406	9695	0.01	4	6	3	0.1	0.25
DSJC250.5	250	15668	0.50	26	28	16	594.5	0.38
DSJC1000.1	1000	49629	0.10	10	20	6	3.1	0.40
DSJC500.1	500	12458	0.10	9	12	5	0.1	0.44
DSJC500.5	500	62624	0.50	43	47	18	1317.3	0.58
DSJC1000.9	1000	449449	0.90	216	222	86	3290.7	0.60
flat1000_76_0	1000	246708	0.49	72	81	19	3052.6	0.74
DSJC1000.5	1000	249826	0.50	73	82	19	1975.0	0.74
C2000.5	2000	999836	0.50	99	145	20	823.1	0.80
C4000.5	4000	4000268	0.50	107	259	20	1640.3	0.81

- Performance is strongest when input graph has either high or low density

Summary and Next Steps

- Column elimination via decision diagrams is a promising alternative to column generation
  - Can generate strong dual bounds
- Current work 1: embed DD bounds in branch-and-bound (for graph coloring)
- Current work 2: vehicle routing applications
  - Wide use of column generation. What is the potential for DDs?
  - First application: TSP with a drone

Case Study: Truck-Drone Routing

- One truck + one drone
- Possible legs include:

  truck, drone, combined
- Example route duration =

  max{1, 0.5+0.5} +

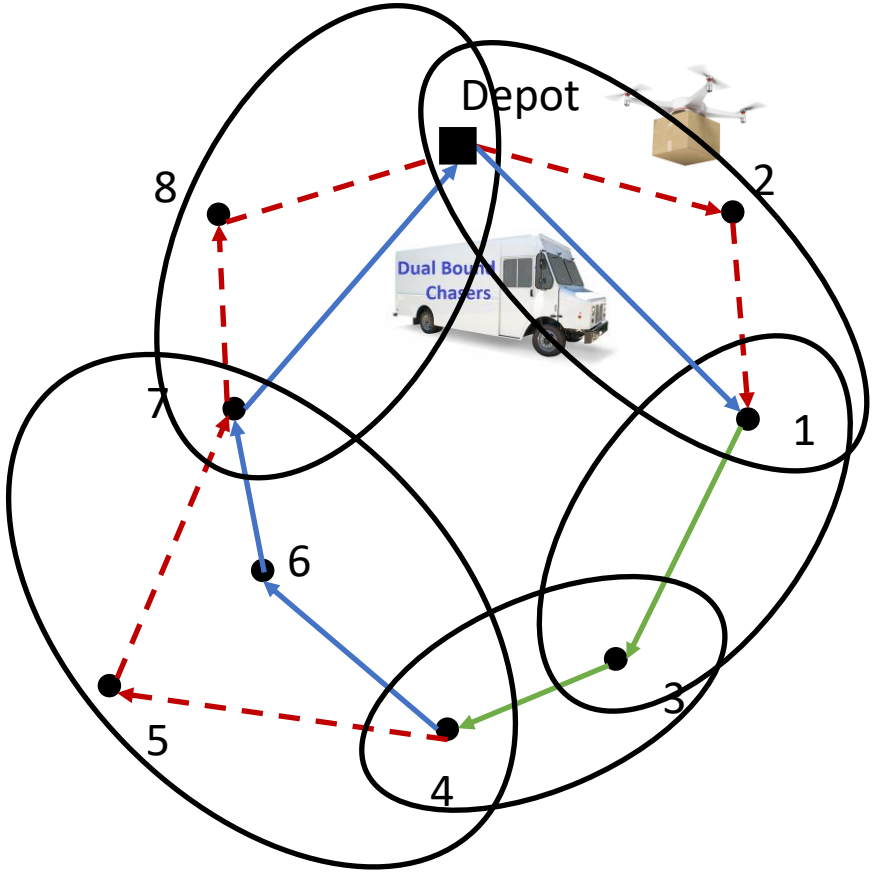
  1 +

  1 +

  max{1+1, 0.5+0.5} +

  max{1, 0.5+0.5}

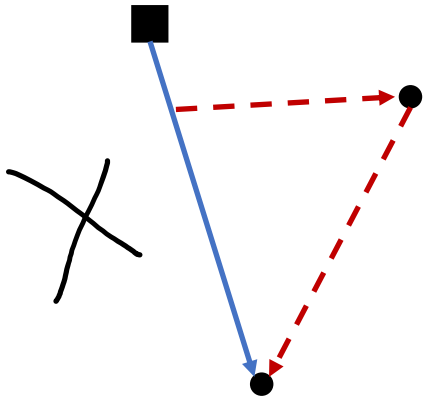
  = 6



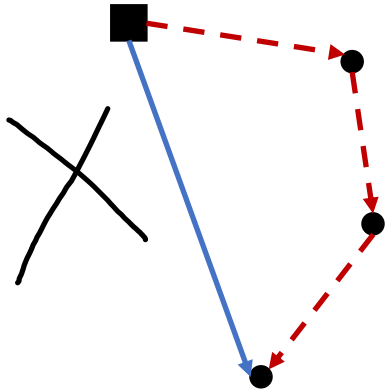
truck speed: 1 unit per edge
drone speed: 0.5 unit per edge

Definition of TSP-D

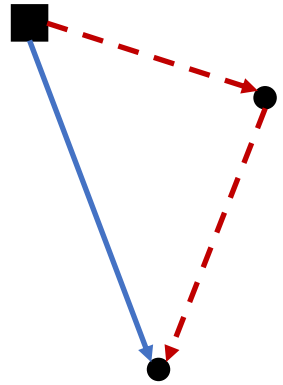
- TSP-D: Traveling Salesperson with a Drone
- Drone speed = α * truck speed (for some fixed α)
- Goal: minimize route duration
- Assumptions:



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

TSP-D: Related Work

- Initiated by Murray and Chu (2015), now very active, see surveys by Macrina et al. (2020), Chung et al. (2020)
- Heuristic Algorithms

  [Murray&Chu, TS2015] [Ponza, master 2016]
  [Agatz et al., TS2018] [Poikonen et al., IJOC 2019]
  ...
- Exact Algorithms

  [Yurek&Ozmutlu, TS2018] [Bouman et al., Networks18]
  [Tang et al., CPAIOR19] [Vasequez et al., 2021]
  [Roberti&Ruthmair, TS2021]

Column generation (Branch-and-Price)
- Master LP: set partitioning model
- Pricing: DP model (with ng-route relaxation)

Set Partitioning Formulation for TSP-D

N: set of locations

R: set of routes

z_r: indicator variable for route r

c_r = duration of route r

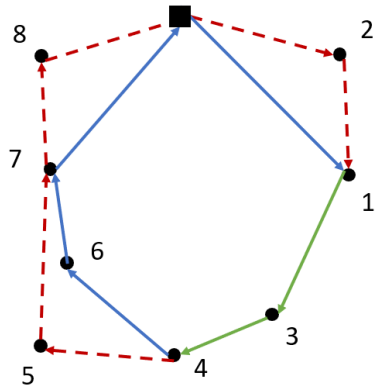
a_{ir}= #times location i is visited in route r

$$\min \sum_{r \in R} c_r z_r$$

$$\text{s.t.} \sum_{r \in R} z_r = 1 \quad \longrightarrow \text{exactly one route is selected}$$

$$\sum_{r \in R} a_{ir} z_r = 1 \quad \forall i \in N \quad \longrightarrow \text{each location is visited once}$$

$$z_r \in \{0,1\} \quad \forall r \in R$$



feasible route

$a_{ir} = 1, \forall\, i \in N$

[Roberti&Ruthmair, 2021]

Pricing Problem for TSP-D

N: set of locations

R: set of routes

z_r: indicator variable for route r

c_r = duration of route r

a_{ir}= #times location i is visited in route r

$$\min \sum_{r \in R} c_r z_r$$

$$\text{s.t.} \sum_{r \in R} z_r = 1 \quad \rightarrow u_0$$

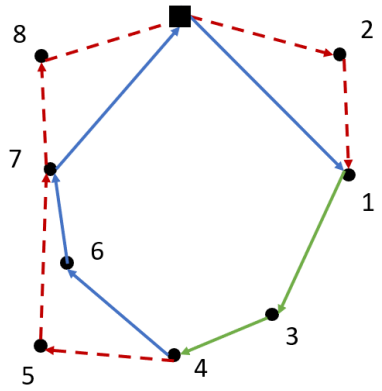
$$\sum_{r \in R} a_{ir} z_r = 1 \quad \forall i \in N \quad \rightarrow u_i$$

~~$z_r \in \{0,1\} \quad \forall r \in R$~~

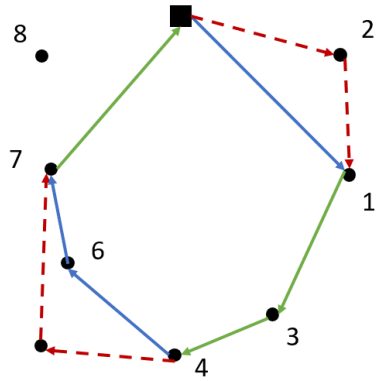
Pricing Problem

Find route with negative reduced cost, i.e.,
$c_r - u_0 - \sum_i a_{ir} u_i < 0$

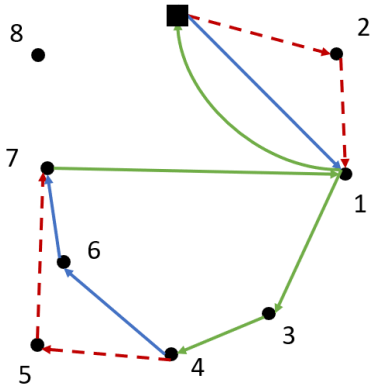
- Similar to TSP-D if only feasible routes allowed
- Allow infeasible routes: ng-route relaxation



feasible route
$a_{ir} = 1, \forall\, i \in N$



infeasible route
$a_{8r} = 0,\ a_{ir} = 1$ otherwise



infeasible route
$a_{8r} = 0, a_{1r} = 2, a_{ir} = 1$ otherwise

[Roberti&Ruthmair, 2021]

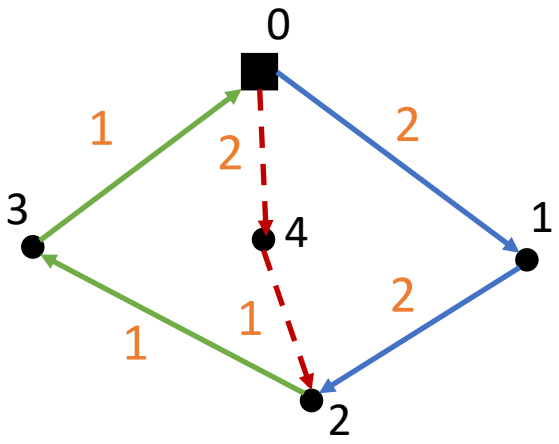
Dynamic Programming Model for TSP-D

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

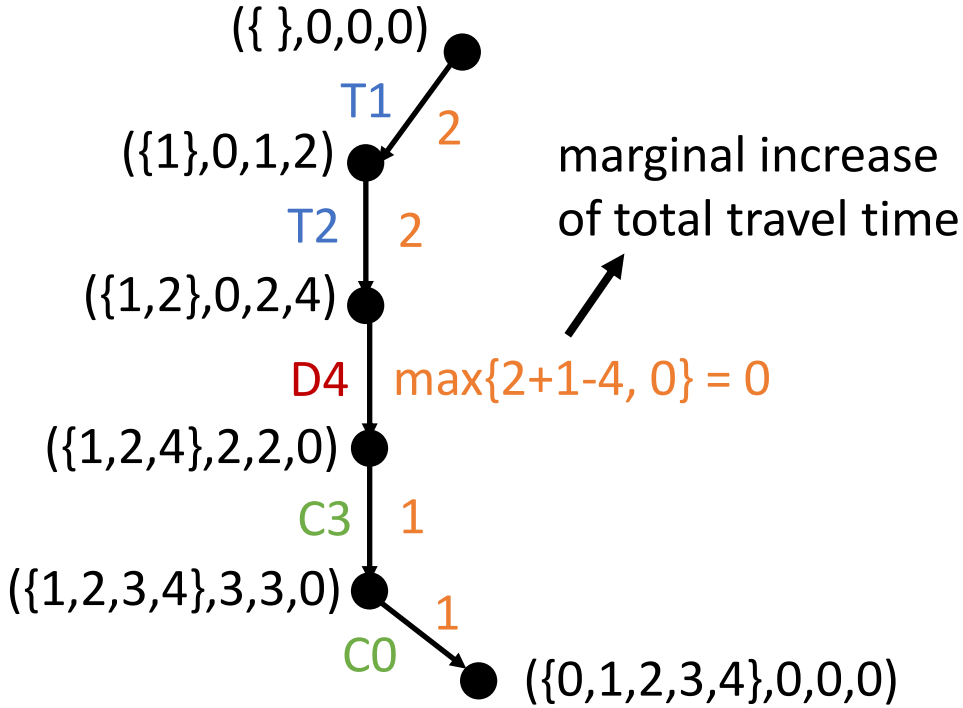
- S = customers visited so far
- LC = latest location visited by both vehicles
- LT = latest location visited by truck alone
- t = time spent by the truck traveling alone since leaving LC

Set of **controls**

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



Route: T1, T2, D4, C3, C0

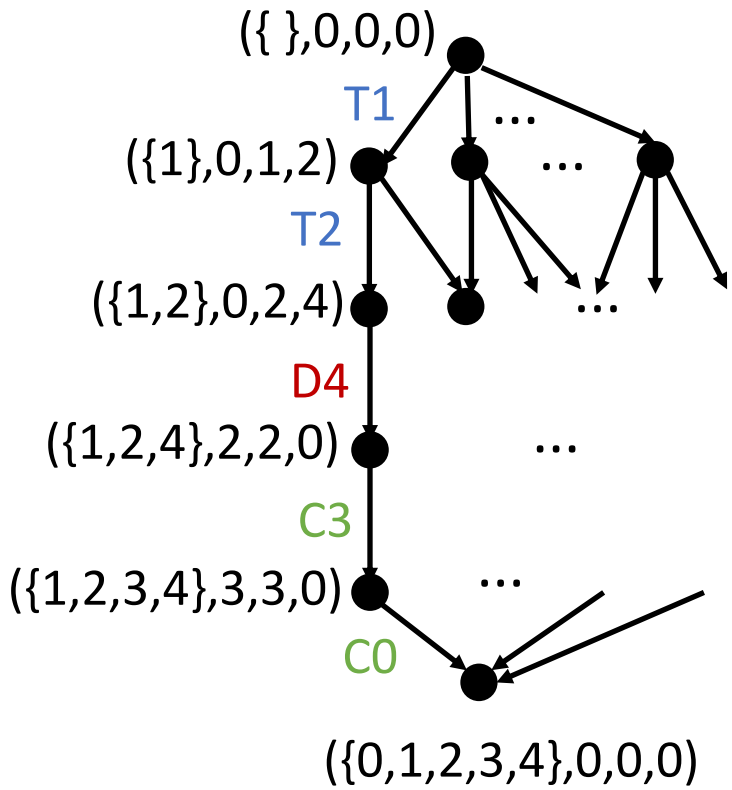


[Roberti&Ruthmair, 2021]

Decision Diagram Compilation for TSP-D

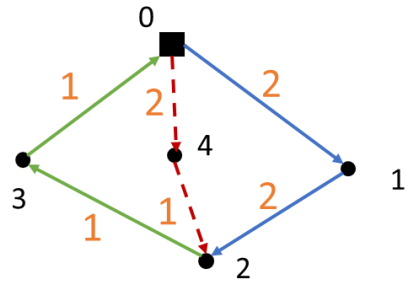
- Top-down DD compilation can be defined by state transition function of DP model

  [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 based on merging rules

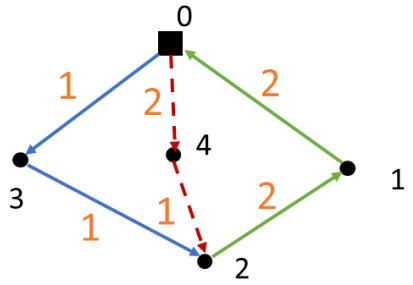


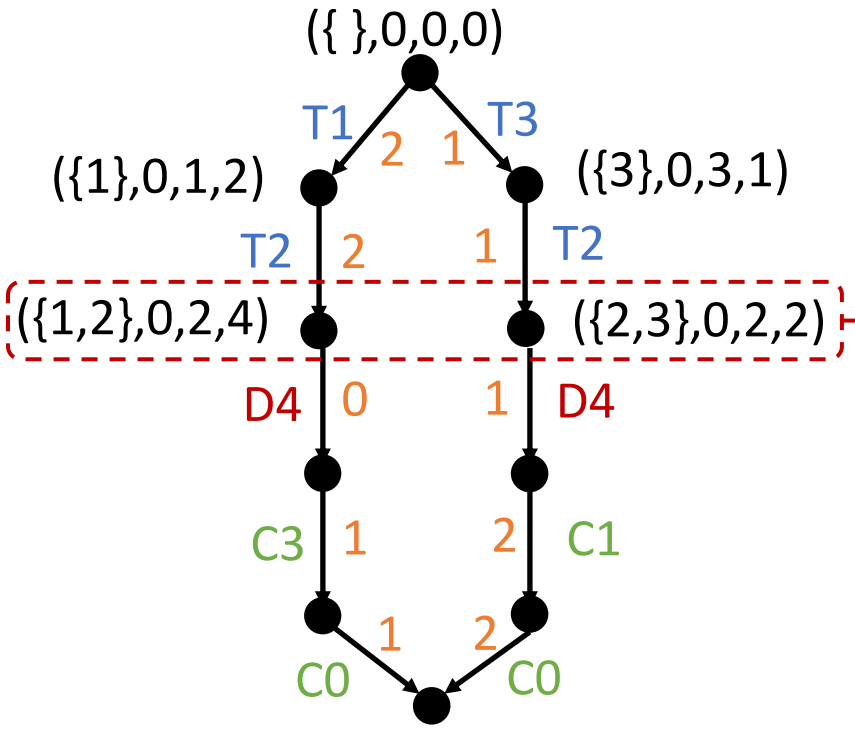
Node Merging: Example

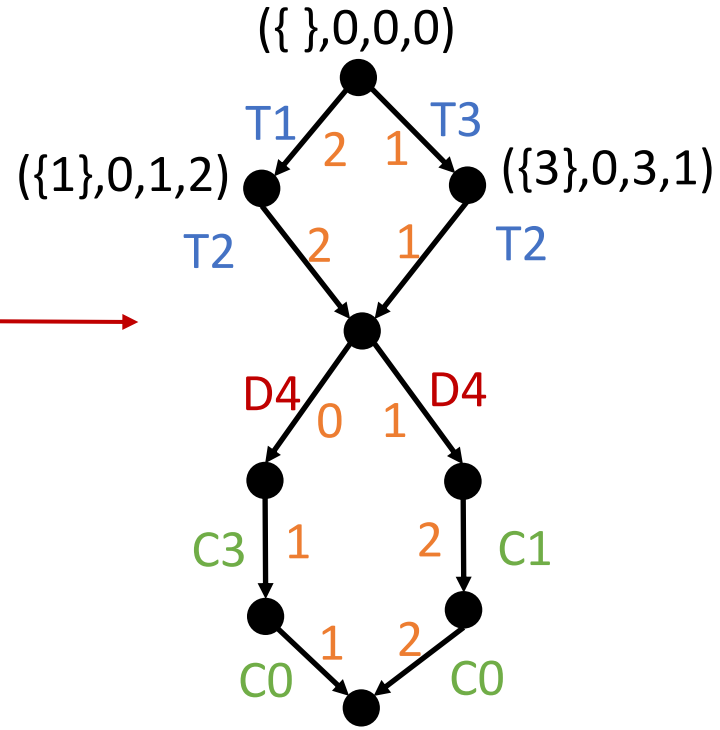
Route 1: T1, T2, D4, C3, C0



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





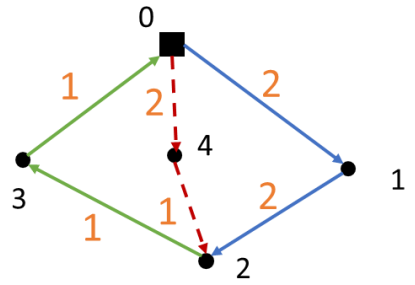


Only merge states with equal (LC,LT)!

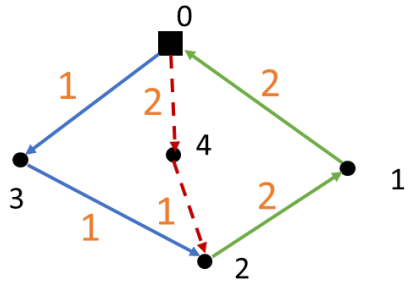
State (S, LC, LT, t): ({visited}, latest combined, latest truck, time spent by truck alone)

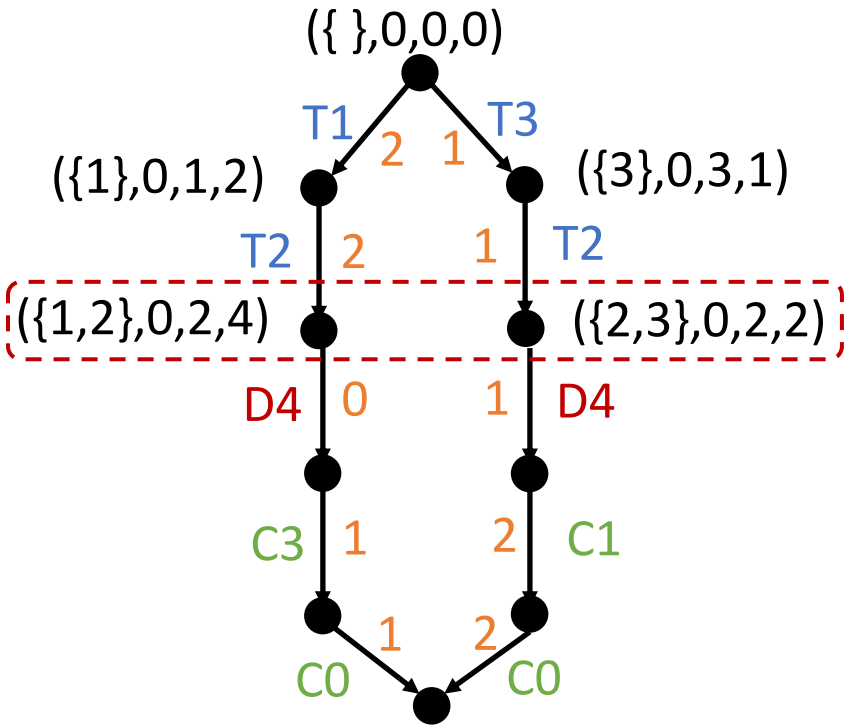
Node Merging: Example

Route 1: T1, T2, D4, C3, C0



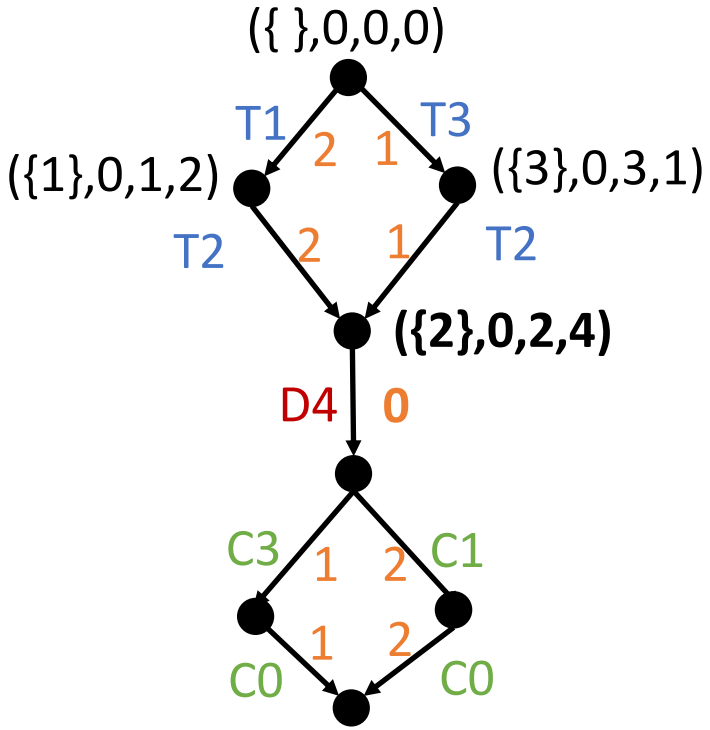
Route 2: T3, T2, D4, C1, C0





Only merge states with equal (LC,LT)!

State (S, LC, LT, t): ({visited}, latest combined, latest truck, time spent by truck alone)

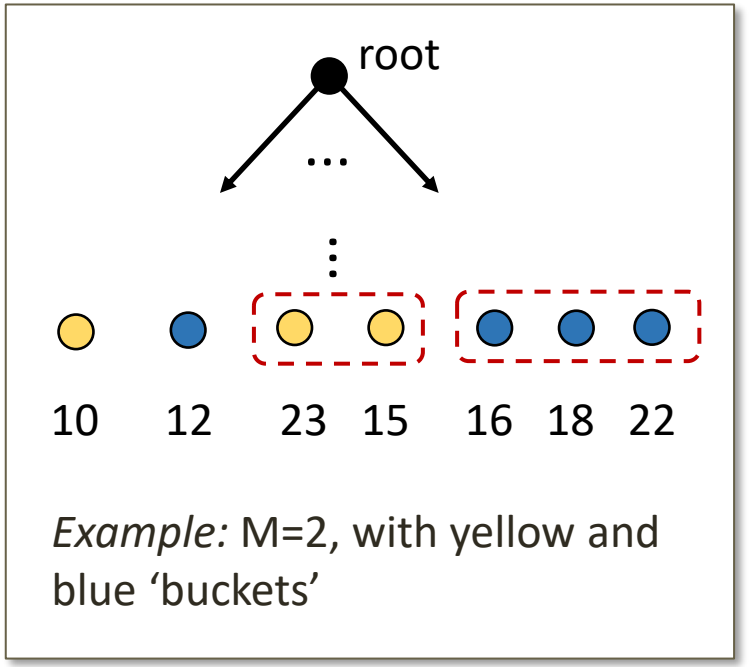


Merging rule $S = S_1 \cap S_2$

Merging rule $t = \max(t_1, t_2)$

Top-Down Compilation with Node Merging

- Define ‘bucket size’ M
- For each layer
  - Partition nodes by (LC, LT) pair into ‘buckets’
  - Compute shortest path length from the root to each node
  - Merge those with longer path lengths until each bucket has size ≤ M
- Relaxed DD has size $O(Mn^3)$



Derive Bound From Constrained Network Flow

Constrained integer network flow model (NP-hard):

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

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

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

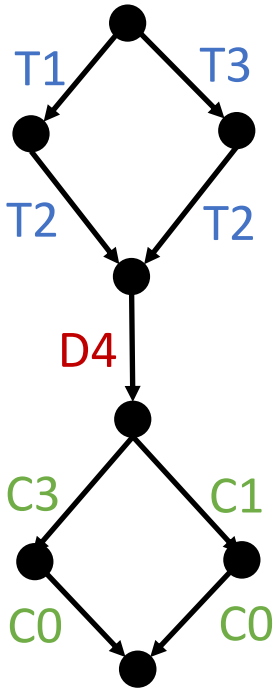
$$\sum_{a \in \delta^-(t)} y_a = 1$$

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

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

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

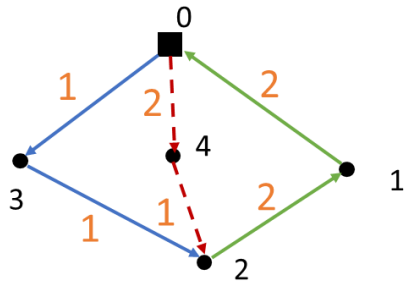
Theorem: SPLP(R), CFLP(D_R), and LR(D_R) have the same optimal objective value

Going Beyond the Set Partitioning Bound

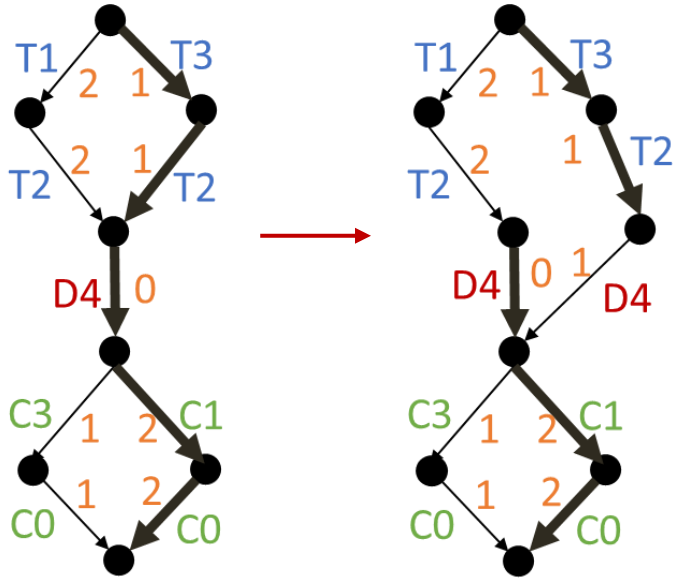
- Resolve conflicts along solution paths by refining the DD

Type 1: objective function

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

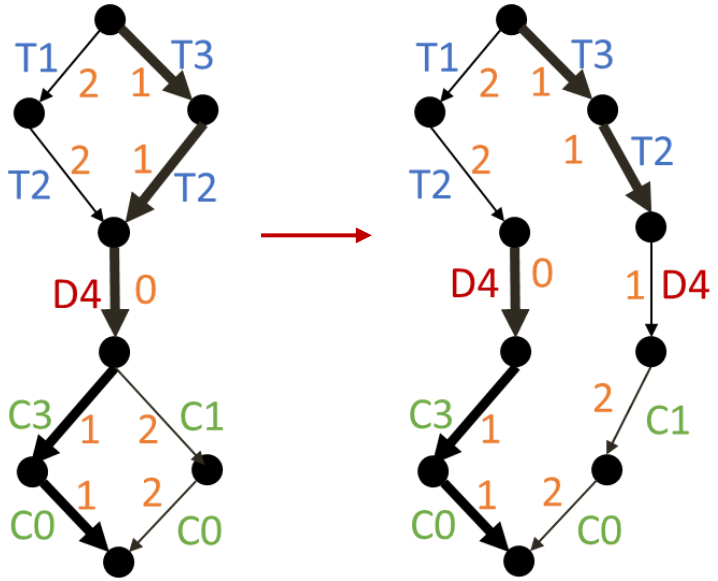


Duration = 7



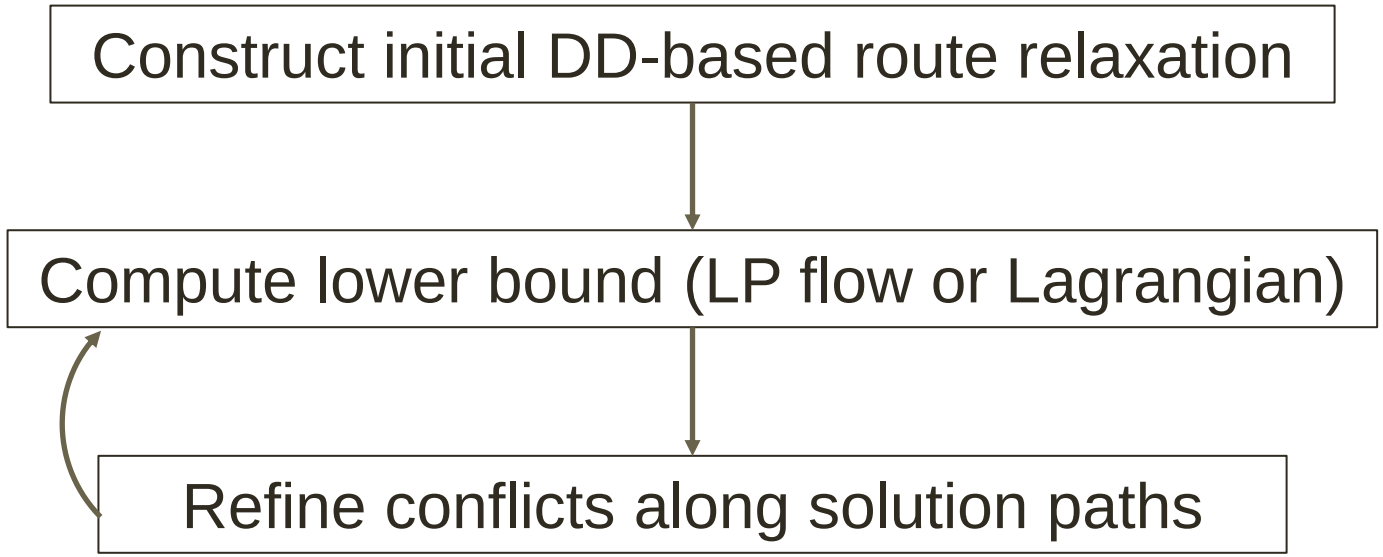
Path length = 6

Type 2: repeated visits



Customer 3 repeated

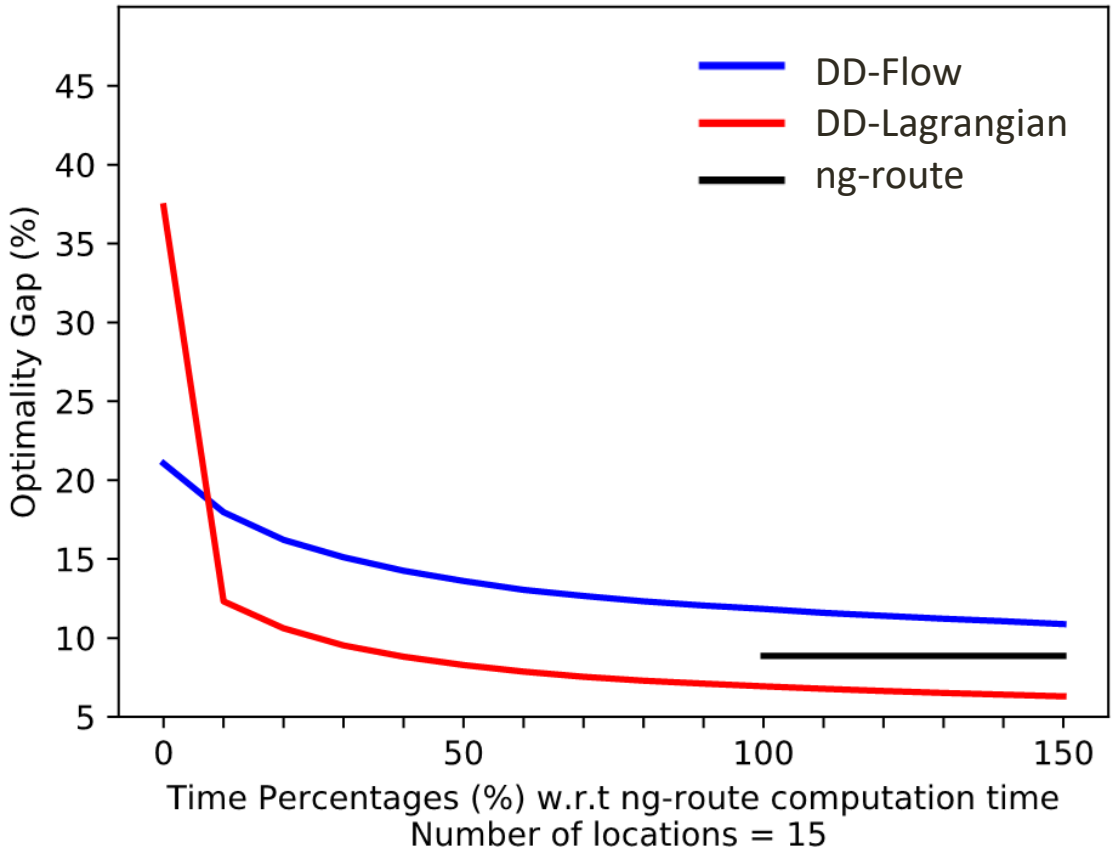
Overall Framework

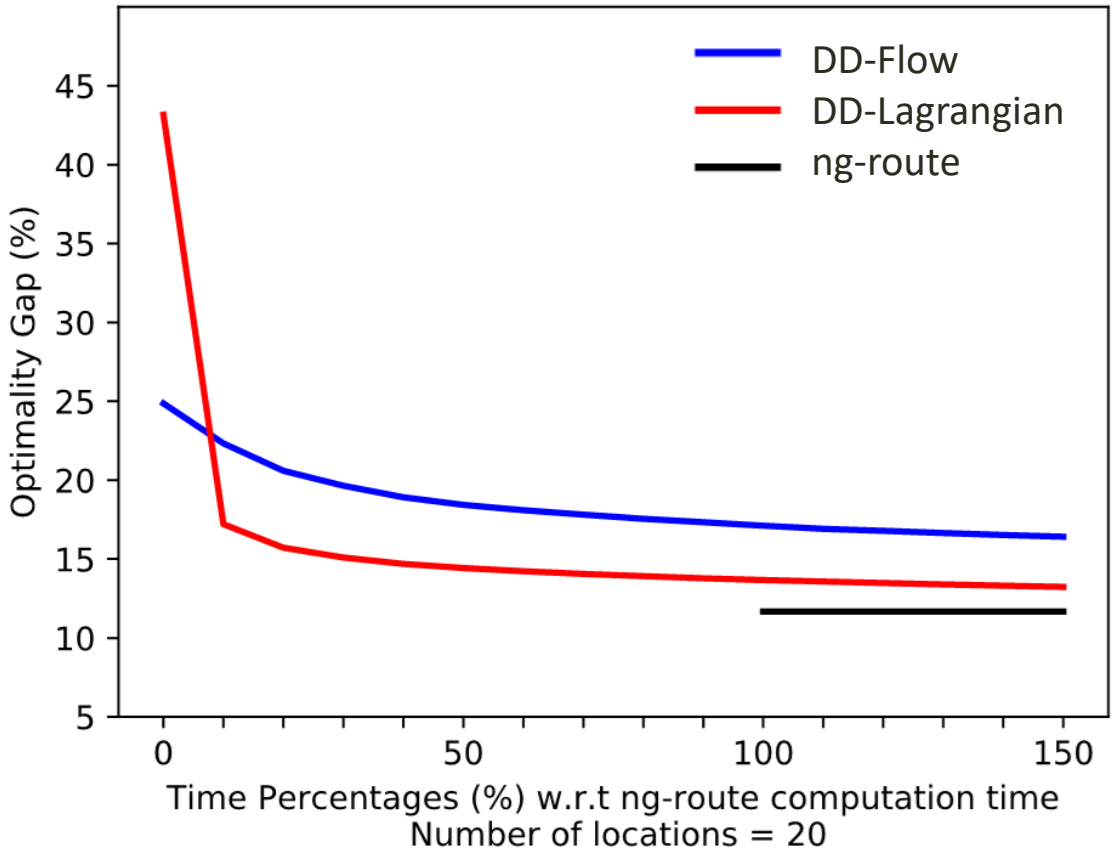


Experimental Evaluation on TSP-D

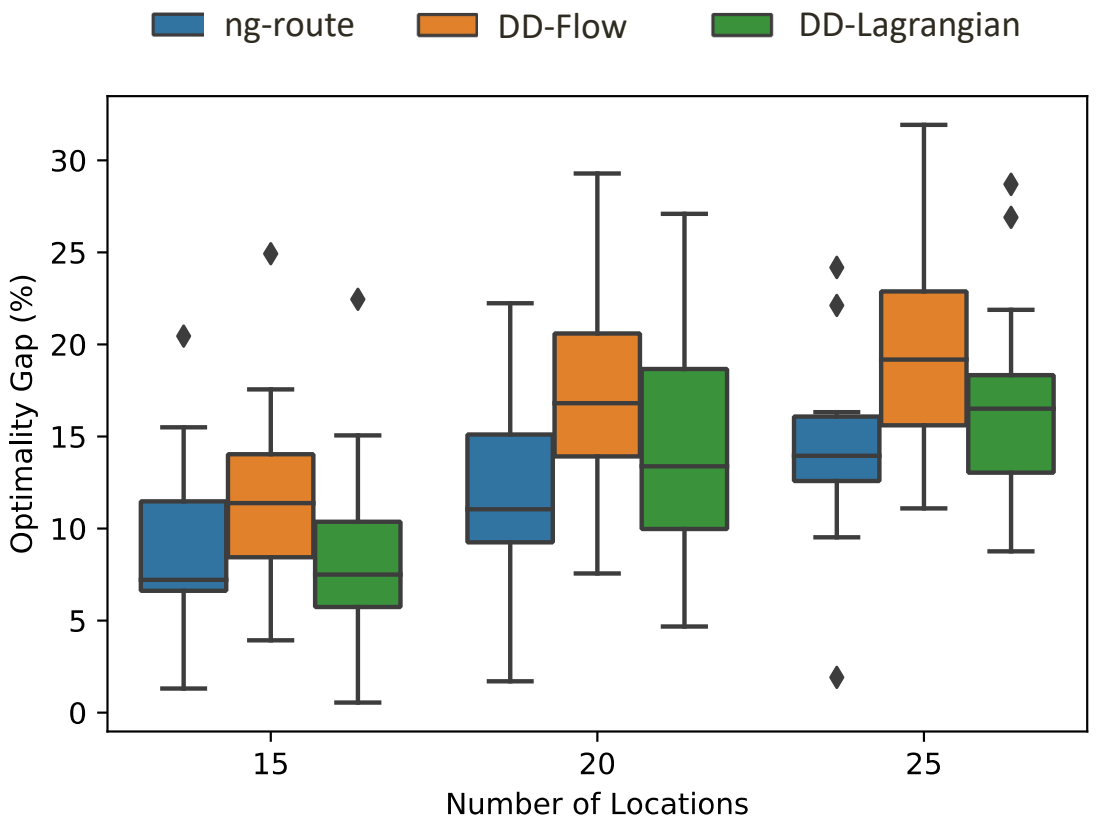
- Evaluate two variants
  - DD-Flow: lower bound from constrained network flow LP
  - DD-Lagrangian: lower bound from Lagrangian
  - both apply iterative refinement based on conflicts
- Comparison with state-of-the-art bound for TSP-D
  - column generation model from [Roberti&Ruthmair, TS2021]
  - set 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



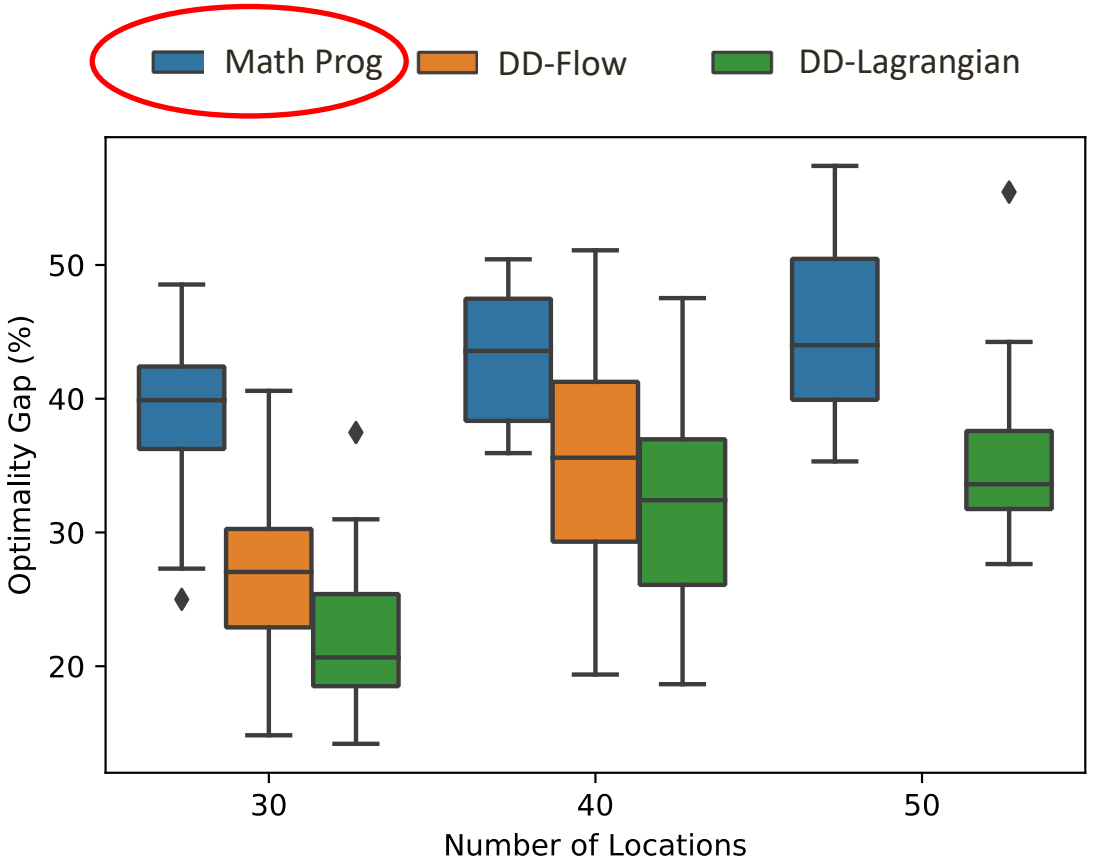
(Time limit for DD methods is the ng-route solving time)

Decision Diagram/Dynamic Program size comparison

#Locations	15	20	25
initial DD	9,736	25,086	51,485
DD-Flow	10,725	26,538	51,967
DD-Lagrangian	15,567	37,197	59,204
ng-route*	24,114	73,554	174,725

* size for ng-route is #labels after dominance rules are applied

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 for Routing Application

- Relaxed Decision Diagrams can be used as an alternative for column generation to compute lower bounds
  - When defined on existing route relaxation (dynamic program for pricing problem) it produces the same LP bound
- We introduced a new DD-based route relaxation
  - Conflict refinement can further improve the lower bound
- Experiments of TSP with Drone show competitive performance
  - Especially for larger instances when column generation cannot be applied