Carnegie Mellon University
Tepper School of Business

Column Elimination for Arc Flow Formulations

Willem-Jan van Hoeve
Carnegie Mellon University

Based on joint work with Ziye Tang and Anthony Karahalios

Agenda

- Example: Graph Coloring
- Column elimination as general framework
- Experimental evaluation
- Summary and outlook

Graph Coloring

- Assign a color to each vertex such that adjacent vertices have a different color. Minimize the number of colors.
- MIP model 1

x_{vk} : vertex v is assigned color k, for $v \in V, k \in K$

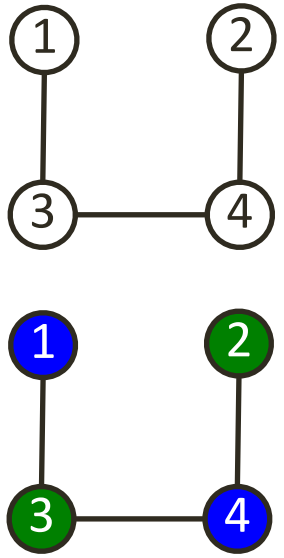
y_k : color k is used, for $k \in K$

$$\min \sum_{k \in K} y_k$$

$$\text{s.t. } x_{uk} + x_{vk} \leq y_k \qquad \text{for } (u,v) \in E, k \in K$$

$$x_{vk} \in \{0,1\}, y_k \in \{0,1\} \qquad \text{for } v \in V, k \in K$$

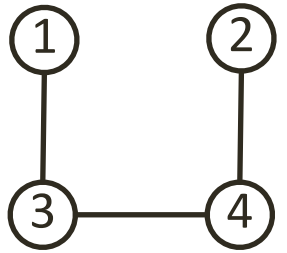
- Relatively weak LP relaxation

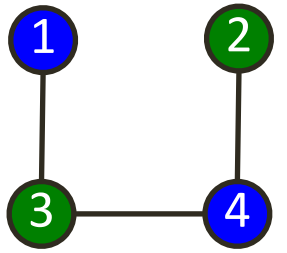


$G = (V, E)$

Graph Coloring: Dantzig-Wolfe Reformulation

- Each integer solution is a collection of *independent sets* (=subset of vertices that share no edges)
  - Independent set represents a color class
- MIP model 2: binary variable x_i for each independent set i





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

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1 :	1	1	1	0	0	0	0	0
2 :	1	0	0	1	1	0	0	0
3 :	0	0	0	0	1	1	0	0
4 :	0	0	1	0	0	0	1	0

$$\min \sum_{i \in I} x_i$$

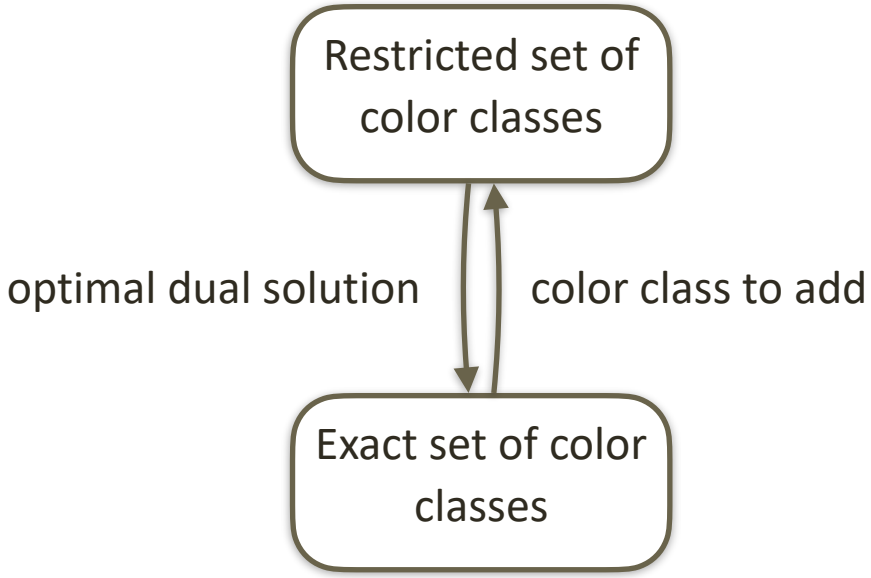
$$\text{s.t.} \sum_{i: v \in i} x_i = 1 \quad \text{for } v \in V$$

$$x_i \in \{0,1\} \quad \text{for } i \in I$$

- Stronger LP relaxation (but exponential number of variables!)

Solving Linear Program with Column Generation

Master Problem: Linear Program

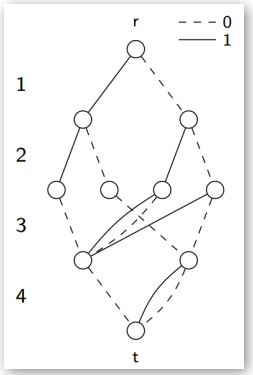


Pricing Problem: Dynamic Program*

$$\min \sum_{i \in I'} x_i$$

$$\text{s.t.} \sum_{i: v \in i} x_i = 1 \text{ for } v \in V$$

$$x_i \in [0,1] \text{ for } i \in I'$$



Integer Optimality: Embed Column Generation LPs within Branch-and-Price

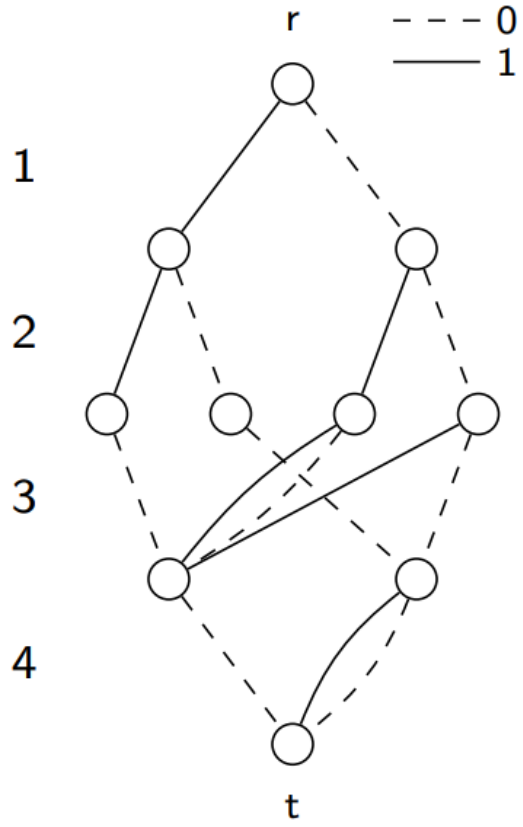
[Desrosiers et al. 1984, Barnhart et al. 1998, Mehrotra and Trick 1996]

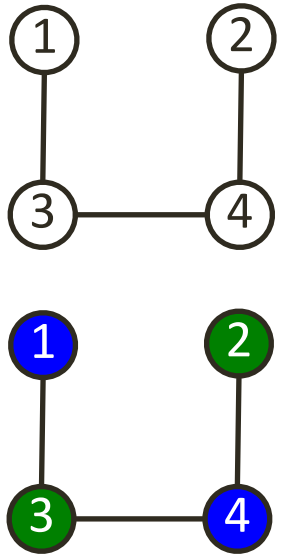
* When pricing problem is NP-hard: Use state-space relaxations or other techniques to compute a dual bound

Graph Coloring: Network Representation

- We can compactly represent all columns as *paths* in a network

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1 :	1	1	1	0	0	0	0	0
2 :	1	0	0	1	1	0	0	0
3 :	0	0	0	0	1	1	0	0
4 :	0	0	1	0	0	0	1	0





$G = (V, E)$

- This is an exact *decision diagram* representing all independent sets

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

Graph Coloring: Arc Flow Formulation

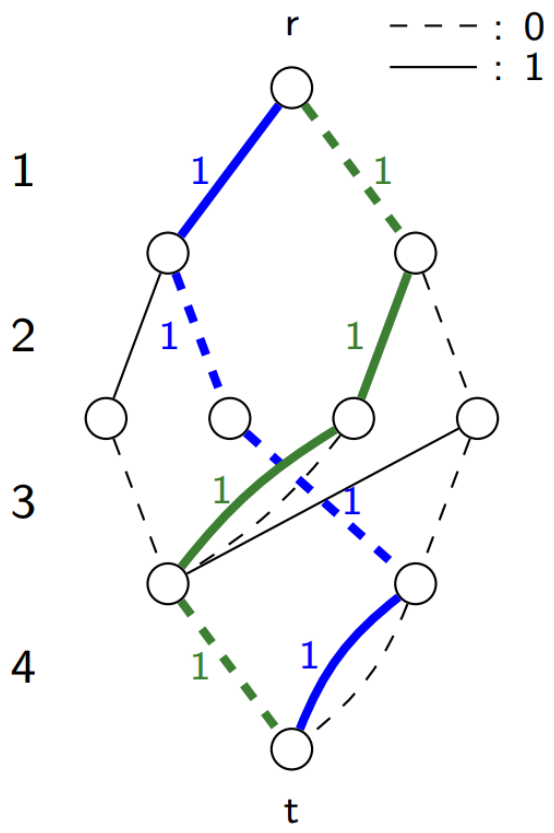
- MIP model 3: variable y_a for each arc a

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

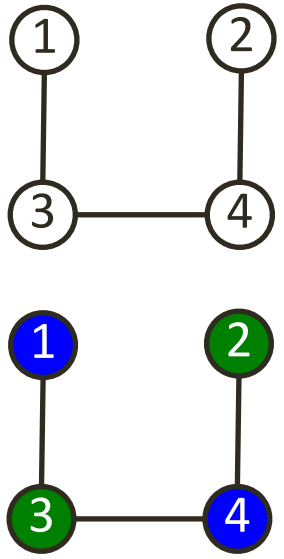
$$\text{s.t.} \sum_{a \in \delta^-(i)} y_a - \sum_{a \in \delta^+(i)} y_a = 0 \quad \text{for } i \in N$$

$$\sum_{a:\ell(a)=v} y_a = 1 \quad \text{for } v \in V$$

$$y_a \in \{0,1,\ldots,n\} \quad \text{for } a \in A$$



$D = (N, A)$



$G = (V, E)$

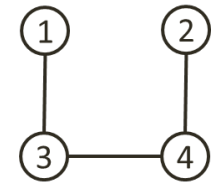
- Same LP bound as Dantzig-Wolfe

Two Main Challenges

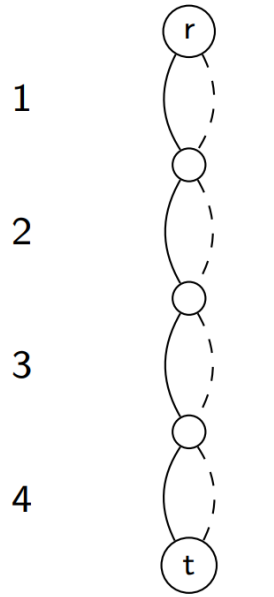
1. Exact decision diagrams can be of exponential size (in the size of the input graph)
   - Remedy: Use smaller *relaxed* decision diagrams instead
   - Provides lower bound on coloring number
2. Solving the arc flow formulation is NP-hard
   - Less relevant in practice: MIP solvers scale well
   - But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

Exact and Relaxed Decision Diagrams

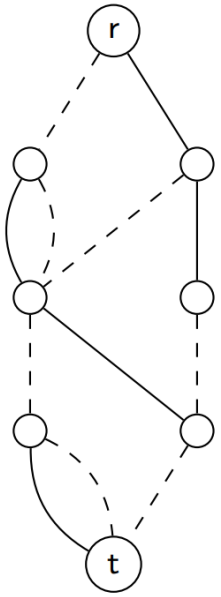
- Decision diagram *D* for problem *P* is $\begin{cases} \textit{exact} & \text{if } \mathrm{Sol}(D) = \mathrm{Sol}(P) \\ \textit{relaxed} & \text{if } \mathrm{Sol}(D) \supseteq \mathrm{Sol}(P) \end{cases}$



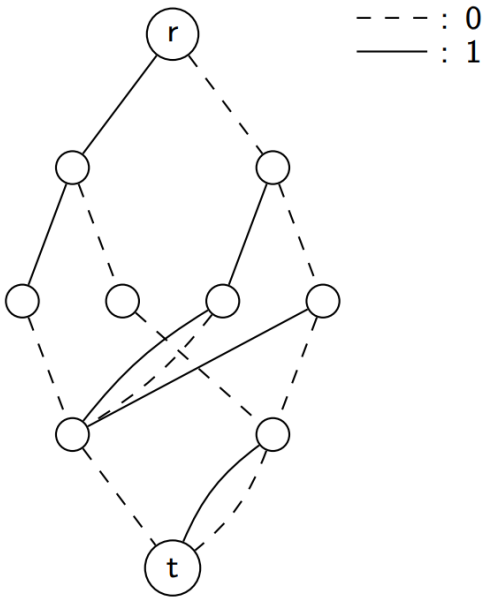
input graph



relaxed
(5 nodes)



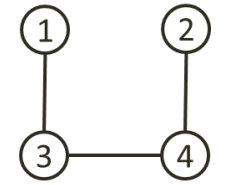
relaxed
(8 nodes)



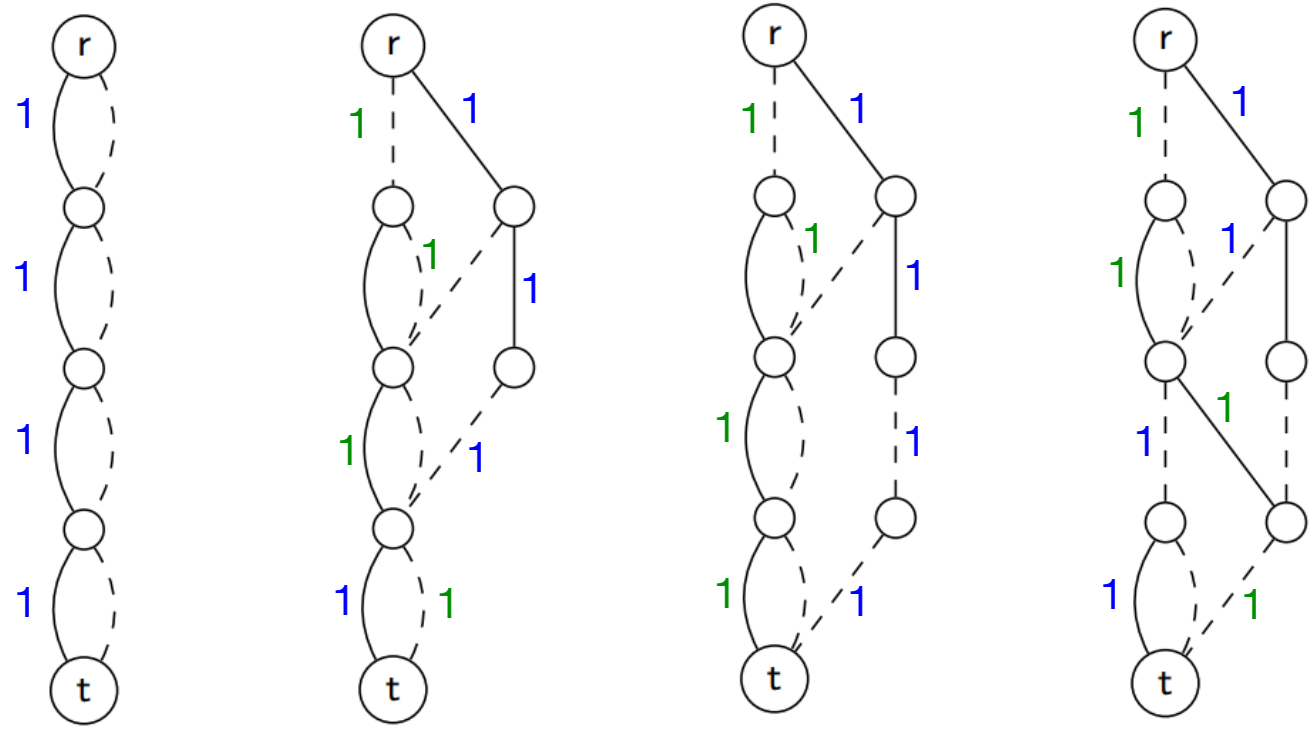
exact
(10 nodes)

[Bergman, Cire, vH, Hooker, 2016]

Column Elimination: Iterative Refinement



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

Lemma: Conflicts can be found in polynomial time (in the size of the state-transition graph) via a path decomposition of the flow

Lemma: Eliminating *k* conflicts increases the size by at most O(*kn*) nodes

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

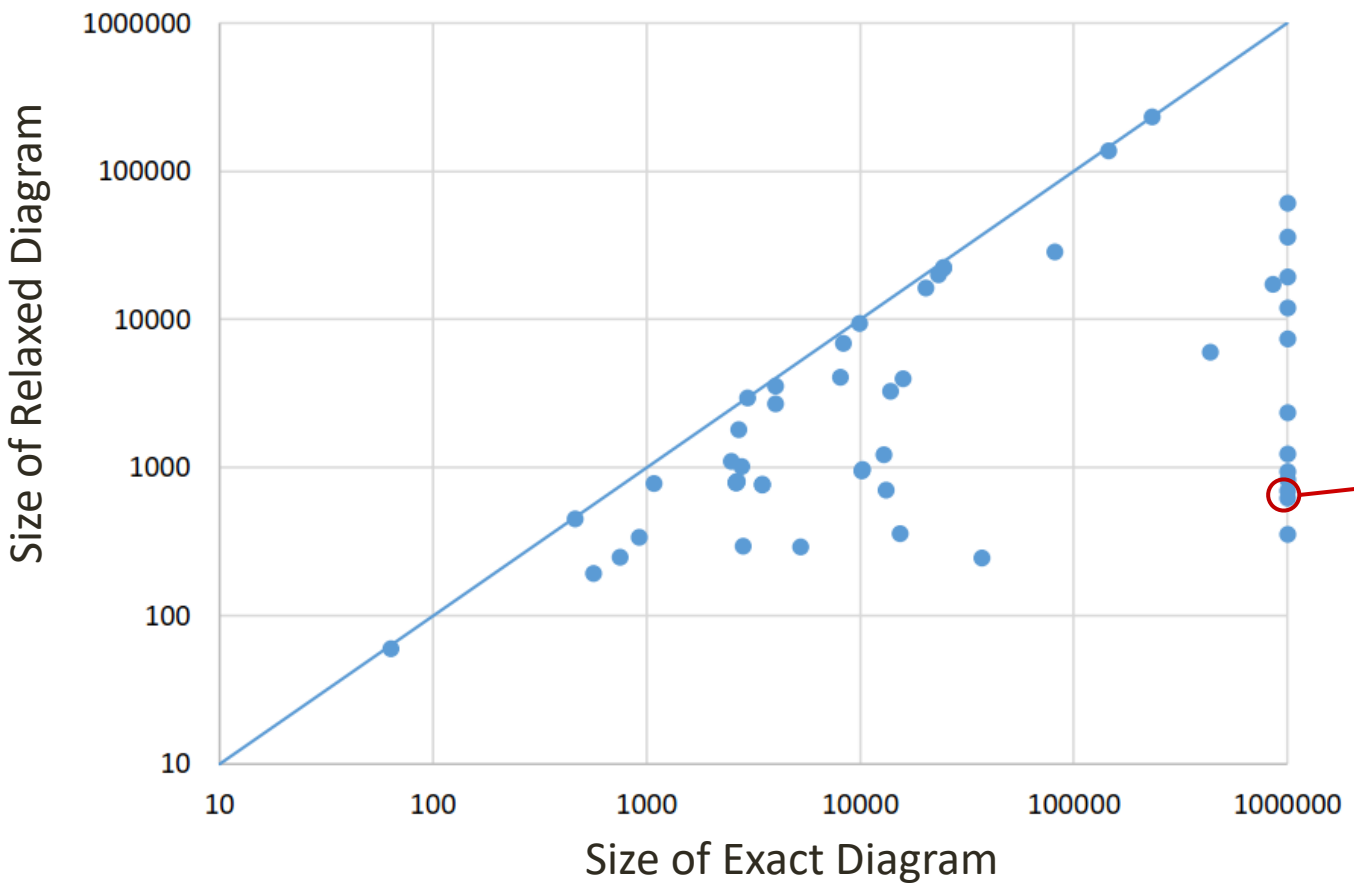
Lemma: In each iteration, compilation via conflict elimination produces a valid lower bound

Lemma: Eliminating all conflicts yields the exact dynamic program

Theorem: The algorithm terminates with an optimal LP/IP solution (if time permits)

[vH *IPCO* 2020, *Math. Prog.* 2022]

Evaluation on DIMACS benchmark instances



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

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

Agenda

- Example: Graph Coloring
- Column elimination as general framework
- Experimental evaluation
- Summary and outlook

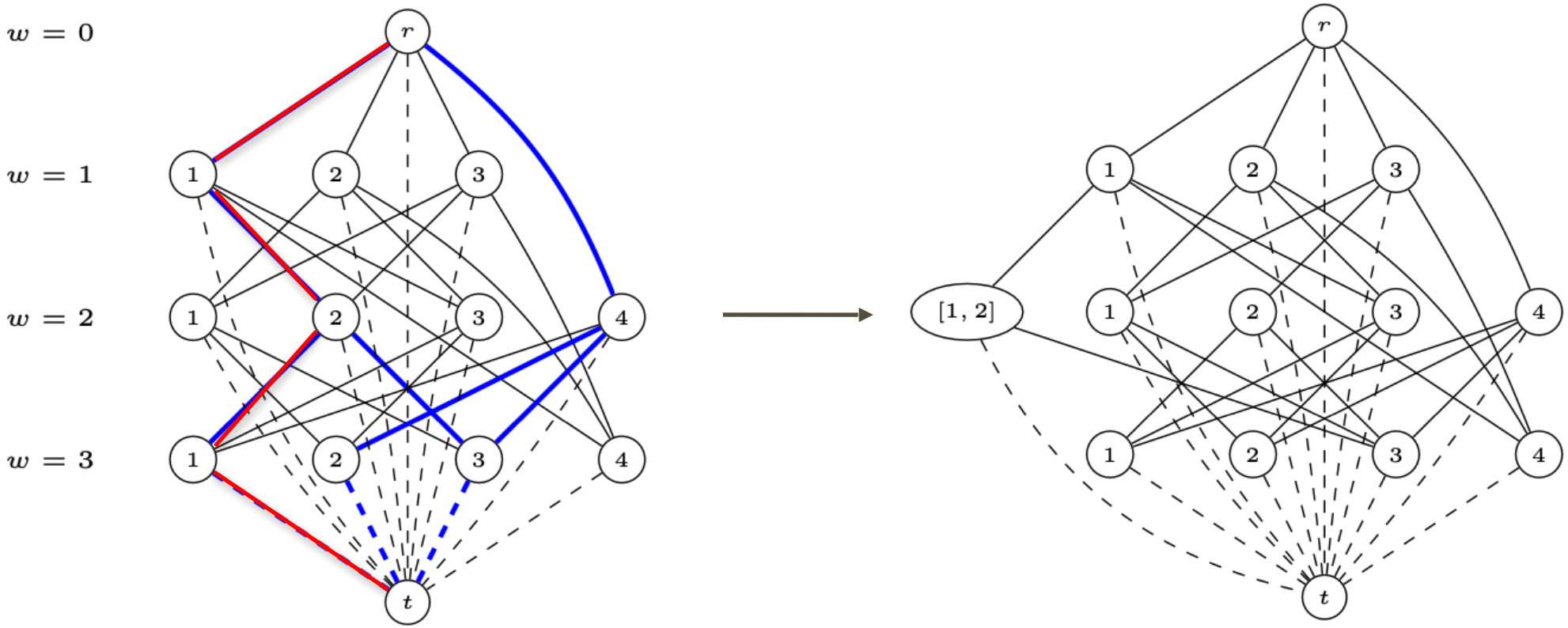
Dynamic Programming-Based Arc Flow Models

- Arc flow models can be defined over the *state-transition graph* of a dynamic program [de Lima et al. 2022]
- Exact decision diagrams can likewise be compiled using dynamic programming models
- Column elimination works with relaxations
  - But these are not the classical state-space relaxations which are often used in column generation
  - What dynamic program leads to relaxed decision diagrams?

Example: Capacitated Vehicle Routing

Example: Capacitated Vehicle Routing

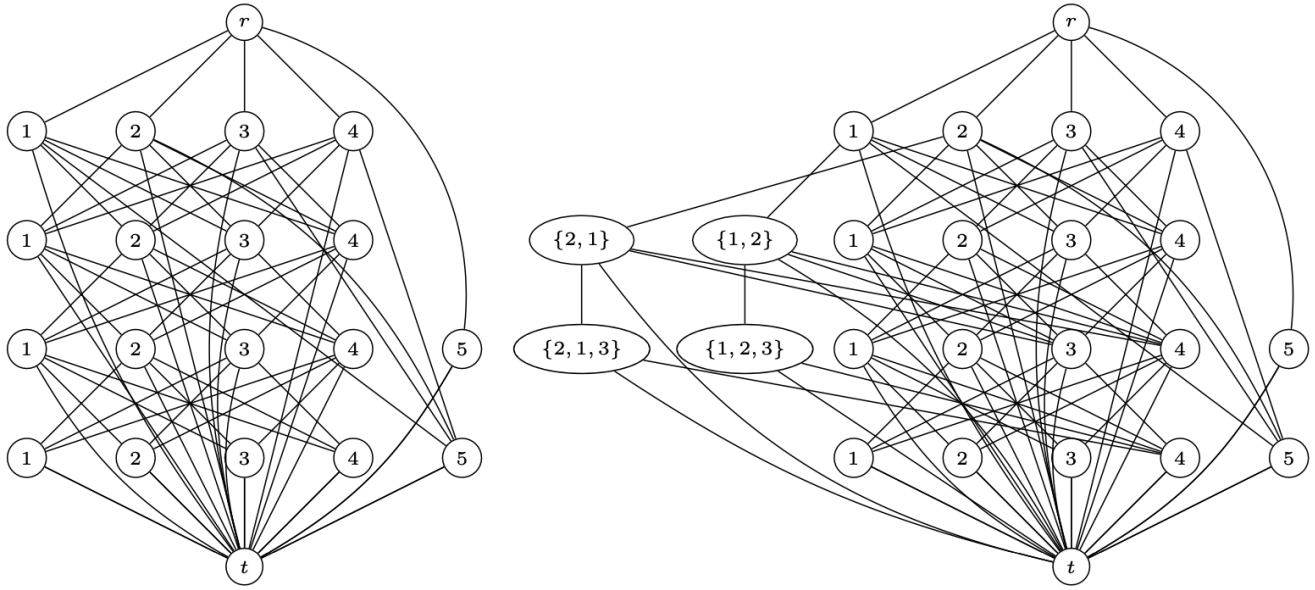
Update the DP states to ‘remember’ the repeated location



Q-route relaxation [Fukasawa et al. 2006]

Dynamic Program Relaxation

DEFINITION 1. Let P_1 and P_2 be dynamic programs with solution sets $\mathcal{X}_1$, $\mathcal{X}_2$ and solution cost functions ϕ_1, ϕ_2, respectively. P_2 is a dynamic program relaxation w.r.t. P_1 if the following holds: $\mathcal{X}_1 \subseteq \mathcal{X}_2$ and for all $x \in \mathcal{X}_1$, $\phi_1(x) \geq \phi_2(x)$.



a. State-space relaxation.

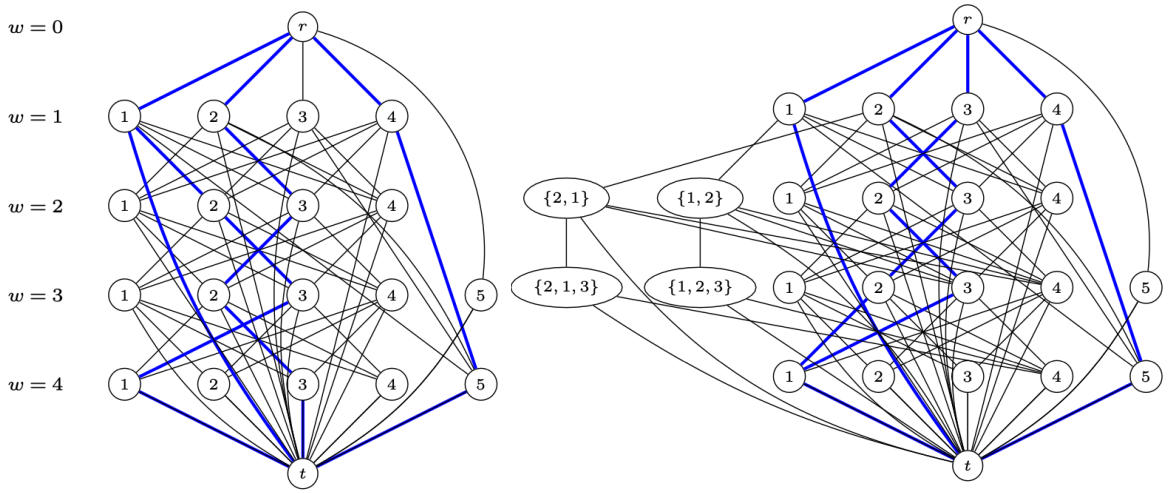
b. Dynamic program relaxation.

[Christofides et al. 1981]

Column Elimination Revisited

Column elimination solves ~~(a relaxation of)~~ the linear program relaxation.

1. Construct an initial dynamic program relaxation.
2. Solve the linear program relaxation of the arc-flow model.
3. If the optimal solution uses an “infeasible” r-t path, eliminate the r-t path from the dynamic program relaxation and go back to step 2.
4. If the optimal solution has no “infeasible” r-t paths, terminate.



Column Elimination: General Approach

- Input:
  - Dynamic Programming model (its state-transition graph defines sequences)
  - Integer Programming constraints: Arc flow formulation

$$\min \sum_{a \in \mathcal{A}} c(a) y_a$$

$$\text{s.t.} \sum_{a \in \mathcal{A}} g_j(a) y_a \geq b_j \qquad \forall\, j \in \{1, \ldots, m\}$$

← The constraints are functions of the arcs

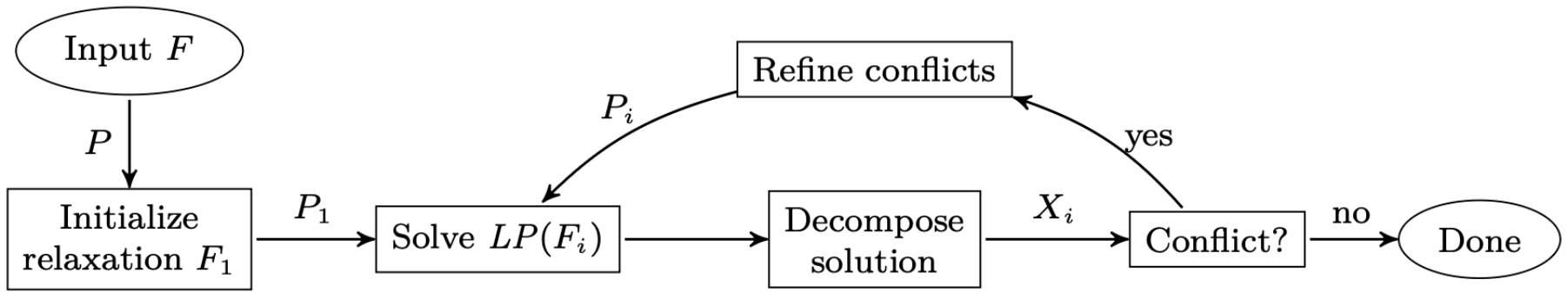
$$\sum_{a \in \delta^+(s)} y_a - \sum_{a \in \delta^-(s)} y_a = 0 \qquad \forall\, s \in \mathcal{N} \setminus \{r, t\}$$

$$y_a \in \mathbb{Z}_+ \qquad \forall\, a \in \mathcal{A}$$

Column Elimination: Linear Program

- Input:
  - Dynamic Programming model (its state-transition graph defines sequences)
  - Integer Programming constraints: Arc flow formulation

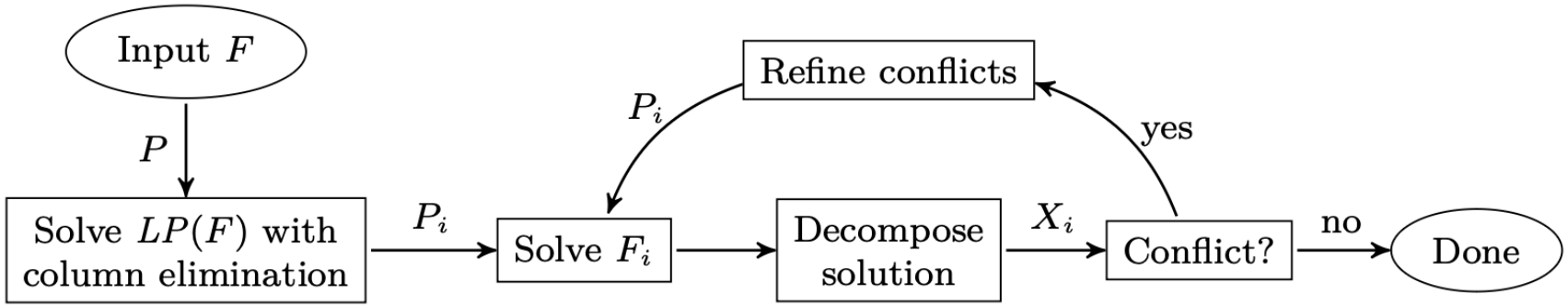
LP



Column Elimination: Integer Program

- Input:
  - Dynamic Programming model (its state-transition graph defines sequences)
  - Integer Programming constraints: Arc flow formulation

IP



Lagrangian Reformulation

Large LPs can still be difficult to solve. We consider a Lagrangian Reformulation:

$$\max_{\lambda} L(\lambda)$$

$$L(\lambda): \quad \min \sum_{a \in \mathcal{A}} c_a y_a + \sum_{j \in 1,\ldots,|\mathcal{G}|} \lambda_j (b_j - \sum_{a \in \mathcal{A}} g_j(a) y_a)$$

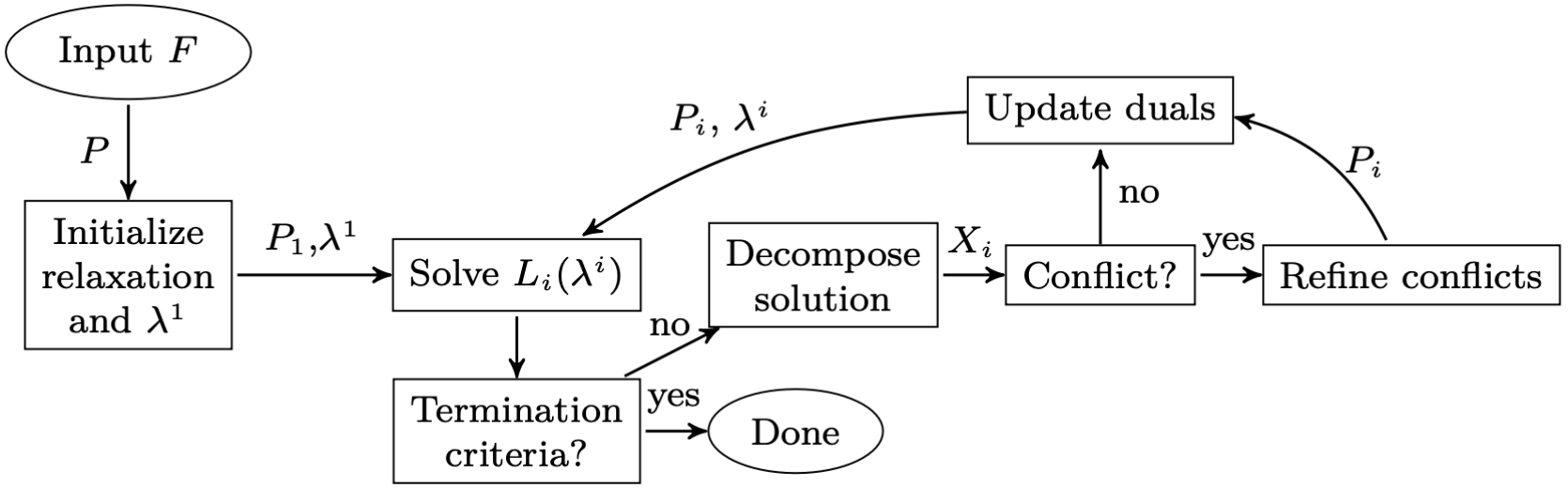
$$\text{s.t.} \sum_{a \in \delta^-(u)} y_a - \sum_{a \in \delta^+(u)} y_a = 0 \qquad \forall u \in \mathcal{N} \setminus \{r, t\}$$

$$y_a \geq 0, y_a \in \mathbb{Z} \qquad \forall a \in \mathcal{A}$$

- For fixed λ, L(λ) can be solved by a Successive Shortest Paths algorithm. [Wang et al. 2020]
- Max L(λ) can be solved using Subgradient Descent.

Column Elimination: Lagrangian Method

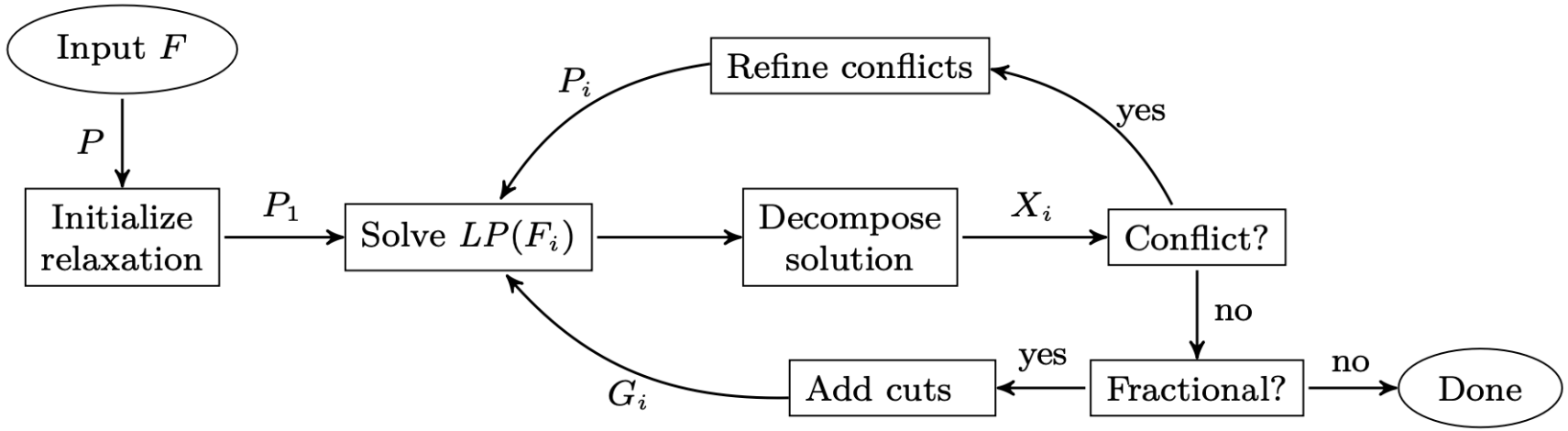
Our method combines Subgradient Descent and relaxation updates.



- Each iteration computes a dual bound, which is helpful for variable fixing.
- We are using Subgradient Descent to solve a changing problem.

Cut-and-Refine

To solve the IP, we can add cutting planes into the LP arc-flow formulation.



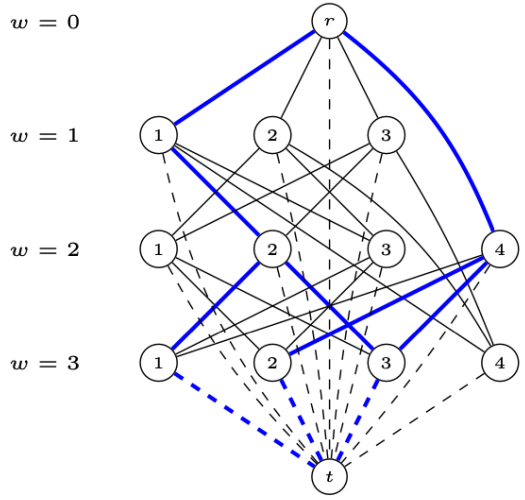
- The cuts need to be in terms of the arc variables.
- There are "robust" and "non-robust" cuts, as in column generation. [Poggi 2003]
- Effectively adding cuts is difficult for the Lagrangian method. [Lucena 2005]

Apply Arc Fixing to Improve Efficiency

Can arcs be pruned? (i.e. provably not in an optimal solution)

Reduced Cost Arc Fixing:

1. Input a feasible solution v to the dual of LP(F(D)) with associated LB, and an UB on the IP.
2. Consider an arc. If the best *reduced cost* based on v for any r-t path through the arc is such that LB + reduced cost $\geq$ UB, then the arc can be fixed to 0.



[Pecin et al. 2017]

Discussion: Column Generation / Elimination

Similarities

1. The models are equivalent when the dynamic program relaxation is the same for the arc-flow formulation and the pricing problem.
2. The dynamic program relaxation can be strengthened for both models.

Differences

1. Column generation solves a restricted master problem and column elimination solves relaxed models.
    - The method for computing lower bounds is different.
    - The dynamic program relaxations in each method can be different.

Discussion: Related Literature

Similar Methods

- Iterative Refinement of Relaxed Decision Diagrams [Hazdic et al. 2008]
- Dynamic Discretization Discovery [Boland et al. 2017]
- Iterative Aggregation and Disaggregation [Clautiaux et al. 2017]
- Decremental State Space Relaxations [Righini & Salani 2018]
- State Augmenting Algorithm [Lozano et al. 2022]

These methods also (iteratively) update and refine the underlaying discrete relaxation

Agenda

- Example: Graph Coloring
- Column elimination as general framework
- Experimental evaluation
- Summary and outlook

Experimental Setup

Experimental Setup:

- The code is written in C++
- We use CPLEX as LP/IP Solver
- We use a timeout of 3600 seconds

Experiments:

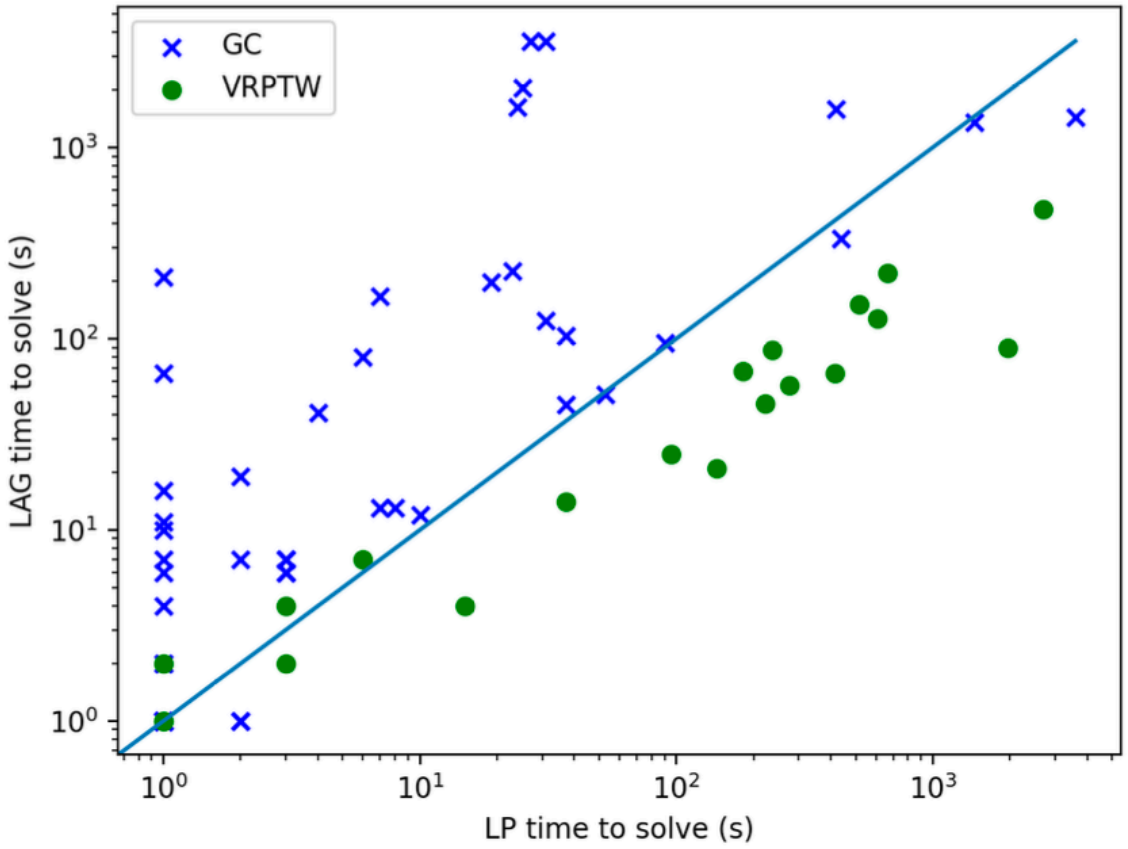
1. Evaluate the impact of LP vs LAG and variable fixing
2. Benchmark comparisons with state-of-the-art on CVRP, VRPTW, Graph Multicoloring, and PDPTW

Experiments: LP vs. LAG

We plot the runtimes of column elimination with an LP solver versus the Lagrangian method.

We evaluate graph coloring (GC) and vehicle routing problem with time windows (VRPTW)

Why is LP best for GC and LAG best for VRPTW? Primal and dual solutions change more for GC as the relaxation is refined.

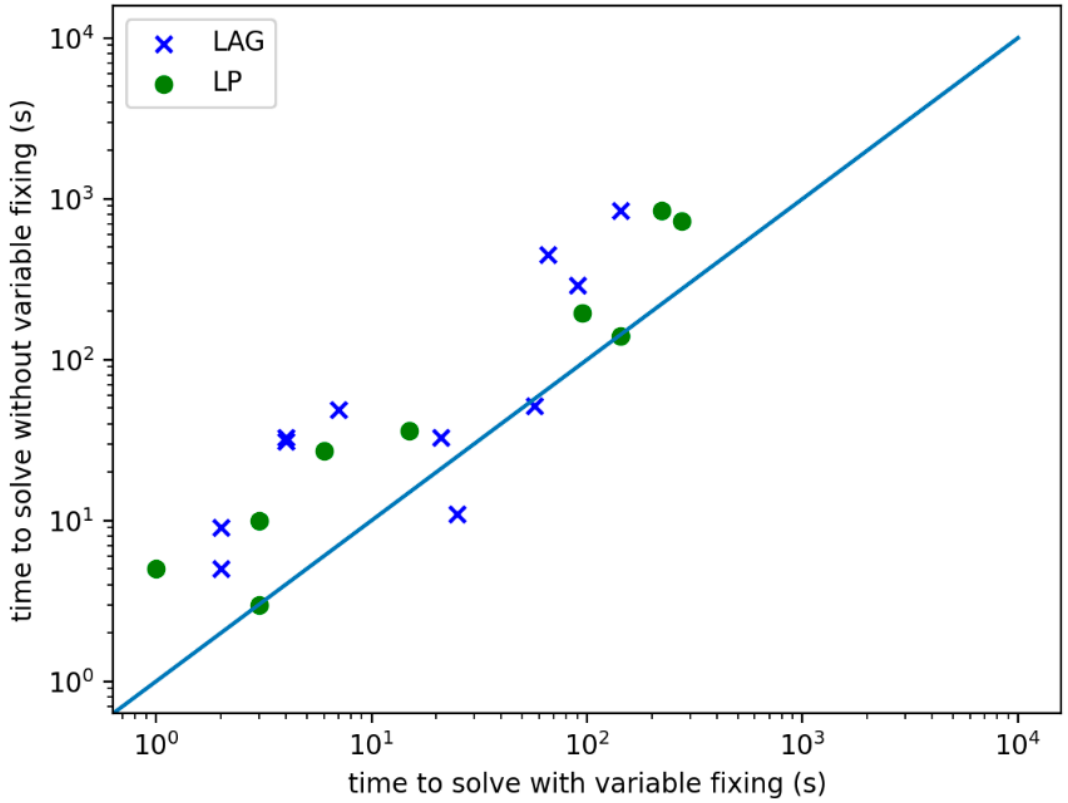


Experiments: Variable Fixing

Variable fixing uses a feasible dual solution to prove that we can remove arcs in the dynamic program. [Pecin et al. 2017]

Benefits both the LP and LAG method.

Can be very effective for large instances.



Experiments: CVRP

Initial results on the Capacitated Vehicle Routing Problem [Karahalios & vH, 2023]

Class	NP	Pecin Gap (%)	CE Gap (%)
A	22	0.36	0.66
B	20	0.14	0.61
E-M	12	0.33	2.60
F	3	0.00	16.41
P	24	0.42	0.85
X	100	0.44	2.13

Instance classes:

- NP = number of instances
- 30 to 1000+ locations

We compare the optimality gap found by CE to the optimality gap found by column generation (the root node of BCP). [Pecin et al. 2017]

Experiments: VRPTW

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CEIt	CESIt	CR	Cuts	Time (s)
C2_6_10	7123.9	6340.81	1	3600	**6437.63**	1	94	1733	0	3600
C2_6_2	7471.5	7075.15	1	3600	**7177.06**	1	94	1546	0	3600
C2_6_3	7215	4670.06	1	3600	**5953.32**	1	41	593	0	3600
C2_6_5	7553.8	7540.44	1	3600	7241.6	1	231	4427	0	3600
C2_6_6	7449.8	7400.61	1	3600	6976.78	1	168	3227	0	3600
C2_6_7	7491.3	6294.69	1	3600	**6966.47**	1	151	2871	0	3600
C2_6_8	7303.7	7223.09	1	3600	6753.56	1	140	2559	0	3600
C2_6_9	7303.2	5754.15	1	3600	**6741.86**	1	104	1834	0	3600
C2_8_1	11631.9	-	-	3600	11551.8	1	196	1177	0	3600
C2_8_10	10946	-	-	3600	9589.46	1	62	1133	0	3600
C2_8_2	11394.5	-	-	3600	10571.2	1	66	1403	0	3600
C2_8_3	11138.1	-	-	3600	7521.76	1	23	438	0	3600
C2_8_5	11395.6	-	-	3600	10829.3	1	154	3589	0	3600
C2_8_6	11316.3	-	-	3600	10462.4	1	104	2330	0	3600
C2_8_7	11332.9	-	-	3600	10403.4	1	88	1968	0	3600
C2_8_8	11133.9	-	-	3600	10059.0	1	80	1700	0	3600
C2_8_9	11140.4	-	-	3600	9941.42	1	65	1332	0	3600

UB = upper bound

LB = lower bound

Nodes = VRPSolver BCP nodes

Time = total running time in seconds

CEIt = CE iterations

CESIt = CE subgradient descent iterations

CR = number of refinements

Cuts = number of cuts added

Time = total running time in seconds

Instances: Clustered on 400 and 600 locations

VRPSolver: [Pessoa et al. 2020] Branch-cut-and-price

Experiments: VRPTW cont'd

Longer Routes

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CElt	CESlt	CR	Cuts	Time (s)
C2_6_10	7123.9	6340.81	1	3600	**6437.63**	1	94	1733	0	3600
C2_6_2	7471.5	7075.15	1	3600	**7177.06**	1	94	1546	0	3600
C2_6_3	7215	4670.06	1	3600	**5953.32**	1	41	593	0	3600
C2_6_5	7553.8	7540.44	1	3600	7241.6	1	231	4427	0	3600
C2_6_6	7449.8	7400.61	1	3600	6976.78	1	168	3227	0	3600
C2_6_7	7491.3	6294.69	1	3600	**6966.47**	1	151	2871	0	3600
C2_6_8	7303.7	7223.09	1	3600	6753.56	1	140	2559	0	3600
C2_6_9	7303.2	5754.15	1	3600	**6741.86**	1	104	1834	0	3600
C2_8_1	11631.9	-	-	3600	11551.8	1	196	1177	0	3600
C2_8_10	10946	-	-	3600	9589.46	1	62	1133	0	3600
C2_8_2	11394.5	-	-	3600	10571.2	1	66	1403	0	3600
C2_8_3	11138.1	-	-	3600	7521.76	1	23	438	0	3600
C2_8_5	11395.6	-	-	3600	10829.3	1	154	3589	0	3600
C2_8_6	11316.3	-	-	3600	10462.4	1	104	2330	0	3600
C2_8_7	11332.9	-	-	3600	10403.4	1	88	1968	0	3600
C2_8_8	11133.9	-	-	3600	10059.0	1	80	1700	0	3600
C2_8_9	11140.4	-	-	3600	9941.42	1	65	1332	0	3600

Shorter Routes

Instance		VRPSolver			Column Elimination					
Name	UB	LB	Nodes	Time (s)	LB	CElt	CESlt	CR	Cuts	Time (s)
C1_6_2	13948.3	13948.3	15	1616	13725.8	1	157	3191	0	3600
C1_6_3	13757	13702.24	1	3600	13285.1	1	58	4470	0	3600
C1_6_4	13538.6	13347.86	1	3600	13026.3	1	24	2556	0	3600
C1_6_5	14066.8	14066.8	1	393	14066.8	3	139	1038	0	1450
C1_6_6	14070.9	14070.9	1	531	14007.8	1	318	3537	0	3600
C1_6_7	14066.8	14066.8	1	476	13999.1	1	314	3249	0	3600
C1_6_8	13991.2	13967.03	3	3600	13598.3	1	220	6853	0	3600
C1_6_9	13664.5	13649.77	1	3600	13225.7	1	136	7265	0	3600
C1_8_1	25156.9	25156.9	1	761	25156.9	3	89	311	0	674
C1_8_10	24026.7	23640.28	1	3600	22962.9	1	49	4641	0	3600
C1_8_2	24974.1	24910.3	1	3600	24094.4	1	77	4205	0	3600
C1_8_3	24156.1	23865.67	1	3600	23194.8	1	26	3668	0	3600
C1_8_4	23797.3	-	-	3600	22620.8	1	9	1627	0	3600
C1_8_5	25138.6	25138.6	1	737	25071.0	1	262	2692	0	3600
C1_8_6	25133.3	25133.3	1	1056	24841.7	1	183	4648	0	3600
C1_8_7	25127.3	25127.3	7	1140	24747.7	1	180	4032	0	3600

Experiments: Three Difficult VRPTW Instances

- **January, 18 2023 -** Proven optimal VRPTW solutions by João Marcos Pereira Silva and Eduardo Uchoa using VRPSolver with a new cluster branching strategy: C1_10_1 (42444.8), C1_10_5 (42434.8), C1_10_6 (42437.0), C1_6_1 (14076.6), C1_6_2 (13948.3), C1_6_3 (13756.5), C1_6_4 (13538.6), C1_6_5 (14066.8), C1_6_6 (14070.9), C1_6_7 (14066.8), C1_6_8 (13991.2), C1_6_9 (13664.5), C1_8_1 (25156.9), C1_8_5 (25138.6), C1_8_6 (25133.3), C1_8_7 (25127.3). For instance C1_6_3, the optimal solution found was an improved one.

Table 2 The performance of column elimination on three difficult instances.

Instance		VRPSolver			Column Elimination				
Name	UB	LB	Nodes	Time (s)	LB	LPIt	LagIt	CR	Time (s)
C1_10_5	42434.8	42434.8	1	1227	42434.8	5	397	7468	6224
C1_8_5	25138.6	25138.6	1	737	25138.6	4	144	3198	1340
C2_10_1	16841.1	-	-	3600	**16841.1**	17	145	1661	10049

[Pessoa et al. 2020]

Experiments: Graph Multi-Coloring

Instance				GM			Column Elimination				
Name	n	m	ω	LB	UB	Time (s)	LB	UB	CElt	CR	Time (s)
COG-10teams	3200	124480	73	-	-	3600	71	1600	25	3713	3373
COG-air04	17808	2121648	377	377	377	1.8	377	377	2	0	6
COG-air05	14390	2527253	413	-	-	3600	1	5295	0	0	2117
COG-atlanta-ip	8124	9250	15	15	15	1844	15	15	2	17	1
COG-cap6000	11992	12103	14	14	14	304	14	14	2	0	1
COG-ds	15252	2057486	1	500	500	6.5	-	-	-	-	3600
COG-gesa2-o	192	144	12	12	13	3600	12	**12**	2	0	0
COG-misc07	410	2928	36	36	39	3600	36	**36**	141	581	139
COG-mkc	10394	154870	169	169	169	0.1	169	169	2	0	1
COG-mod011	192	336	12	12	13	3600	12	13	61	2395	3266
COG-mzzv11	19942	257012	101	101	101	0.1	101	101	2	0	5
COG-mzzv42z	18806	225687	91	91	91	0.1	91	91	2	0	4
COG-net12	3202	4835	17	17	17	1301	17	17	2	17	0
COG-nsrand-ipx	13240	69510	30	-	-	3600	30	**30**	2	0	5
COG-opt1217	1536	6528	26	-	-	3600	26	**26**	2	0	13
COG-rd-rplusc-21	904	11785	109	109	109	0	109	109	2	0	0
COG-rout	560	2940	30	30	32	3600	30	**30**	2	0	0
COG-swath	12480	958000	317	-	-	3600	-	-	-	-	3600

Graph multi-coloring:

Assign k colors to each vertex such that adjacent vertices have no colors in common.

(Each color induces an independent set)

GM:

[Gualandi & Malucelli 2012]

Constraint programming + branch-and-price method

Experiments: PDPTW (=VRPTW + precedences)

Instance		BBM			VRPSolver		Column Elimination				
Name	UB	LB	UB	Time (s)	LB	Time (s)	LB	CElt	CESlt	CR	Time (s)
LC1_2_1	2704.6	2704.6	2704.6	3.3	-	3600	2704.57	10	1	7	3600
LC1_2_10	2741.6	2741.6	2741.6	137.1	-	3600	2389.82	1	324	6034	3600
LC1_2_2	2764.6	2764.6	2764.6	21.5	-	3600	2757.11	74	126	2608	3600
LC1_2_3	2772.2	2772.2	2772.2	114.9	-	3600	2499.26	1	124	2080	3600
LC1_2_4	2661.4	2395.8	2661.4	454.2	-	3600	-	-	-	-	3600
LC1_2_5	2702.0	2702.0	2702.0	4.8	-	3600	2702.05	10	13	176	130
LC1_2_6	2701.0	2701.0	2701.0	7.4	-	3600	2701.04	10	24	337	129
LC1_2_7	2701.0	2701.0	2701.0	7.7	-	3600	2701.04	9	23	311	130
LC1_2_8	2689.8	2689.8	2689.8	16.0	-	3600	2673.18	5	1053	6399	3600
LC1_2_9	2724.2	2724.2	2724.2	55.3	-	3600	2606.42	1	638	11183	3600
LR1_2_1	4819.1	4819.1	4819.1	1.6	-	3600	4819.12	18	1	416	75
LR1_2_10	3386.3	3386.3	3386.3	1376.7	-	3600	2614.42	1	342	3346	3600
LR1_2_2	4093.1	4093.1	4093.1	20.6	-	3600	3868.42	1	176	2011	3600
LR1_2_3	3486.8	3486.8	3486.8	3690.8	-	3600	-	-	-	-	3600
LR1_2_4	2830.7	2341.8	2830.7	1809.6	-	3600	-	-	-	-	3600

200 locations

BBM = [Baldacci et al. 2012]

Instance		Column Elimination				
Name	UB	LB	CElt	CESlt	CR	Time (s)
LC1_4_1	7152.06	**7152.06**	8	15	217	50
LC1_4_2	8007.79	4980.79	1	4	433	3600
LC1_4_5	7150.0	**7150.0**	9	32	609	477
LC1_4_6	7154.02	**7154.02**	19	73	1867	3295
LC1_4_7	7149.43	7119.88	3	120	2573	3600
LC1_4_8	8305.42	6941.46	1	293	8755	3600
LC1_4_9	7451.2	5529.08	1	19	1149	3600
LC1_6_1	14095.64	14095.6	8	56	741	3600
LC1_6_5	14086.3	**14086.3**	8	91	2140	1620
LC1_6_6	14090.79	14002.8	1	168	4464	3600
LC1_6_7	14083.76	13443.7	1	44	2197	3600
LC2_2_1	1931.44	**1931.44**	37	54	494	300
LC2_2_10	1817.45	1697.61	1	496	3168	3600
LC2_2_2	1881.4	1839.51	1	231	1415	3600
LC2_2_3	1844.33	1605.18	1	90	458	3600
LC2_2_4	1767.12	1320.68	1	46	118	3600
LC2_2_5	1891.21	1852.14	5	389	2845	3600
LC2_2_6	1857.78	1794.7	2	854	6119	3600
LC2_2_7	1850.13	1804.02	1	711	4738	3600
LC2_2_8	1824.34	1740.64	1	551	3798	3600

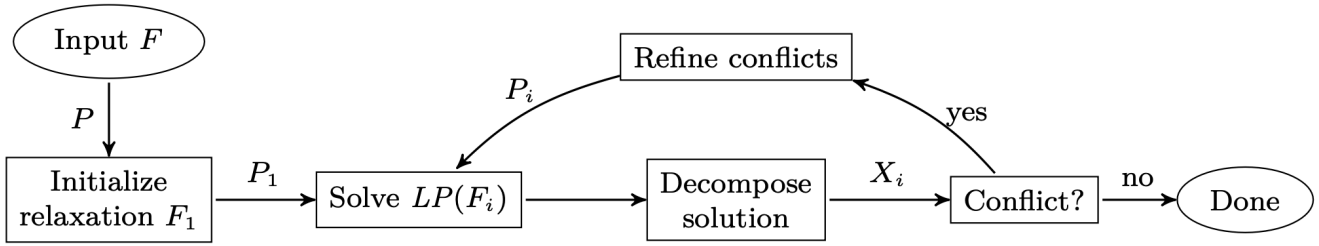
1000 locations

Instances:

[Li and Lim 2001]

Conclusion

- Column Elimination solves integer programming problems as arc flow models over dynamic programming relaxations.
- Column Elimination solves iteratively strengthened DP relaxations.



- Column Elimination has closed open instances of Vehicle Routing Problems and Graph Coloring Problems.
- Column Elimination has been integrated as generic technique in the commercial optimization solver Hexaly (to be released in July 2025)

Column Elimination References

- A. Karahalios and W.-J. van Hoeve. Column Elimination: An Iterative Approach to Solving Integer Programs. *Under Review*.
- Z. Tang and W.-J. van Hoeve. Dual Bounds from Decision Diagram-Based Route Relaxations: An Application to Truck-Drone Routing. *Transportation Science* 58(1):257-278, 2024.
- A. Karahalios and W.-J. van Hoeve. Column Elimination for Capacitated Vehicle Routing Problems. In *Proceedings of CPAIOR*, LNCS 13884:35-51. Springer, 2023.
- A. Karahalios and W.-J. van Hoeve. Variable Ordering for Decision Diagrams: A Portfolio Approach. *Constraints* 27:116-133, 2022.
- W.-J. van Hoeve. Graph Coloring with Decision Diagrams. *Mathematical Programming* 192(1):631-674, 2022.
- W.-J. van Hoeve. Graph Coloring Lower Bounds from Decision Diagrams. In *Proceedings of IPCO*, LNCS 12125:405-419. Springer, 2020.