Combining Column Generation and Column Elimination

Andre A. Cire, Ricardo Fukasawa, Anthony Karahalios, Matheus J. Ota and Willem-J. van Hoeve

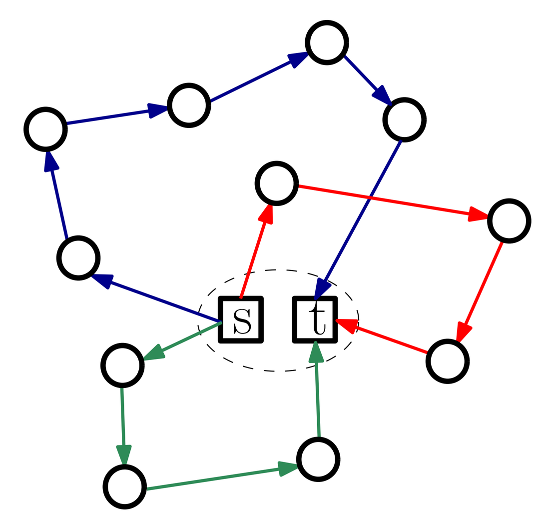
Institution: University of Waterloo, Department of Combinatorics and Optimization Email: mjota@uwaterloo.ca

UNIVERSITY OF WATERLOO

Introduction

Let $D = (V = \{s, t\} \cup V_+, A)$ be a digraph and let $\mathcal{P}$ be a set of $s - t$ paths in D that satisfy some **"complicating constraints"**. We consider routing problems that can be formulated as

$$\text{(SP}(\mathcal{P})) \quad \min \sum_{P \in \mathcal{P}} c(P) \cdot \lambda_P$$
$$\text{s.t.} \sum_{P \in \mathcal{P}} \text{COUNT}(v, P) \cdot \lambda_P = 1, \quad \forall v \in V_+,$$
$$\sum_{P \in \mathcal{P}} \lambda_P = k,$$
$$\lambda_P \in \{0, 1\}, \quad \forall P \in \mathcal{P}.$$



Motivation:

- State-of-the-art algorithms for (SP($\mathcal{P}$)) typically rely on **column generation (CG)**, where the **pricing problem** is modeled as a **resource-constrained shortest path problem (RCSPP)** and solved by a **labeling algorithm**.
- While successful in many cases, **depending on the choice of $\mathcal{P}$**, such an approach might still fail because it leads to an **explosion on the number of explored labels**.

Examples:

- **Chance-Constrained Vehicle Routing Problem (CCVRP):** Vehicle capacity is $C \in \mathbb{Q}_{++}$ and d is a random vector of customer demands. Then,
$$\mathcal{P} = \{s - t \text{ path } P : \mathbb{P}(d(P) \leq C) \geq 1 - \varepsilon\},$$
where $\varepsilon \in (0, 1)$ is a tolerance parameter.
  - If $\mathbb{P}$ **is given by scenarios**, even pricing *non-elementary paths* in $\mathcal{P}$ (i.e., paths that might visit a customer more than once) is **strongly $\mathcal{NP}$-hard** *[Dinh et al., 2018]*.
- **1-Commodity Pickup and Delivery Vehicle Routing Problem (1-PDVRP):** Vehicle capacity is $C \in \mathbb{Q}_{++}$ and each customer $v \in V_+$ has a demand $d(v)$ that can be positive or negative. Then,
$$\mathcal{P} = \{s - t \text{ path } P = (s, v_1, \ldots, v_\ell, t) : 0 \leq d((v_1, \ldots, v_j)) \leq C, \quad \forall j \in [\ell]\}.$$
  - **Non-monotone** accumulated demand prevents the use of **dominance rules**, which are crucial for the good performance of labeling algorithms.

An Initial Formulation

Let $\mathcal{Q} \supset \mathcal{P}$ be such that **pricing over $\mathcal{Q}$ is "easy"**. We can formulate (SP($\mathcal{P}$)) as follows.

$$\min \sum_{P \in \mathcal{Q}} c(P) \cdot \lambda_P$$
s.t. λ is feasible for (SP($\mathcal{Q}$)),
$$x_a = \sum_{P \in \mathcal{Q}} \text{count}(a, P) \cdot \lambda_P, \quad \forall a \in A, \quad (1)$$
$$\sum_{i=0}^{\ell} \sum_{j=i+1}^{\ell+1} x_{v_i, v_j} \leq |V(P)| - 2, \quad \forall P = (v_0 = s, v_1, \ldots, v_\ell, v_{\ell+1} = t) \in \mathcal{Q} \setminus \mathcal{P}. \quad (2)$$

Inequalities (2) are the **tournament inequalities** of Ascheuer et al. (2020). However, these inequalities are known to provide **weak LP bounds**.

Stronger Relaxations via Column Elimination

Assume that $\mathcal{Q}$ is a set of **resource constrained $s - t$ paths** in D for some set of resources R. Let $\mathbb{A}(D, R)$ be an **algorithm that solves the RCSPP** over D with resources R.

```
procedure CG+CE
  Q' ← Q (We don't store Q. This is just to describe the algorithm.)
  D' ← D
  repeat
    Solve the LP relaxation of SP(Q') using A(D', R) to solve the pricing problem.
    Let λ̄ be the obtained solution.
    for each P ∈ Q' \ P with λ̄_P > 0 do
      "Refine" D' to "eliminate" path P.
      Q' ← Q' \ {P}
  until No refinement could be made (or stop earlier for practical reasons).
```

Proposition 1: By the end of CG+CE (if we run until no refinement could be made), the LP bound of SP($\mathcal{Q}'$) is the same as the LP bound of SP($\mathcal{P}$).

Why combine?

- **Advantages to CG:** Standard **CG fixes the set of columns $\mathcal{Q}$** in advance, which might not capture well the **"complicating constraints"** in set $\mathcal{P}$. The CE approach is more flexible: it **dynamically builds a relaxation** using only the infeasible paths in $\bar{\lambda}$.
- **Advantages to CE:** The **CE method** *[Karahalios and van Hoeve, 2024]* solves SP($\mathcal{Q}'$) via shortest paths in a **state-transition graph**. But state-of-the-art **RCSPP solvers** (i.e., algorithm $\mathbb{A}$) already handle well some sources of path infeasibility (e.g., repeated customers). Our approach only refines when $\mathbb{A}$ **cannot handle the infeasibility.**
- In fact, because we use algorithm $\mathbb{A}$, **we don't even need the state-transition graph.**

Column Elimination without State-Transition Graphs

- CE is based on **state-transition graphs** *[Karahalios and van Hoeve, 2024]*, which are acyclic networks where **nodes** $\Longleftrightarrow$ **states** and **paths** $\Longleftrightarrow$ **solutions.**
- CE **"refines"** the network to **"eliminate"** certain paths.

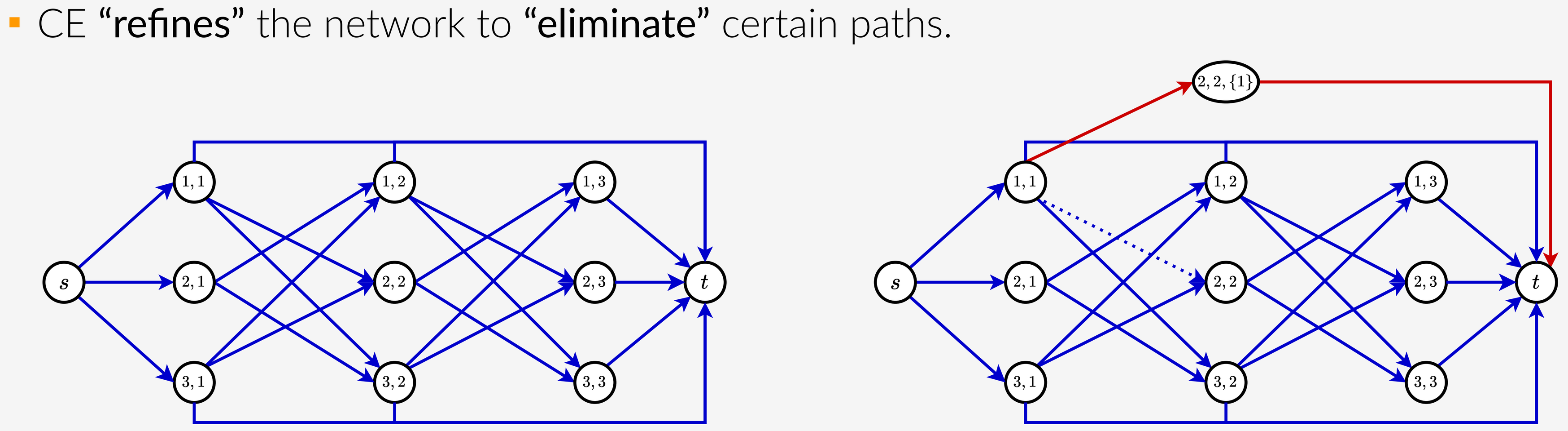


Figure 1. Example with a single resource r that has consumption 1 at every $v \in V_+$. States are in the form [*customer*, *consumption of* r]. The $s - t$ paths P in the left network are such that $r(P) \leq 3$. The network in the right has no path that maps to $(s, 1, 2, 3, t)$ (and to other infeasible paths).

Proposition 2: The same refinement can be applied to the original graph D.

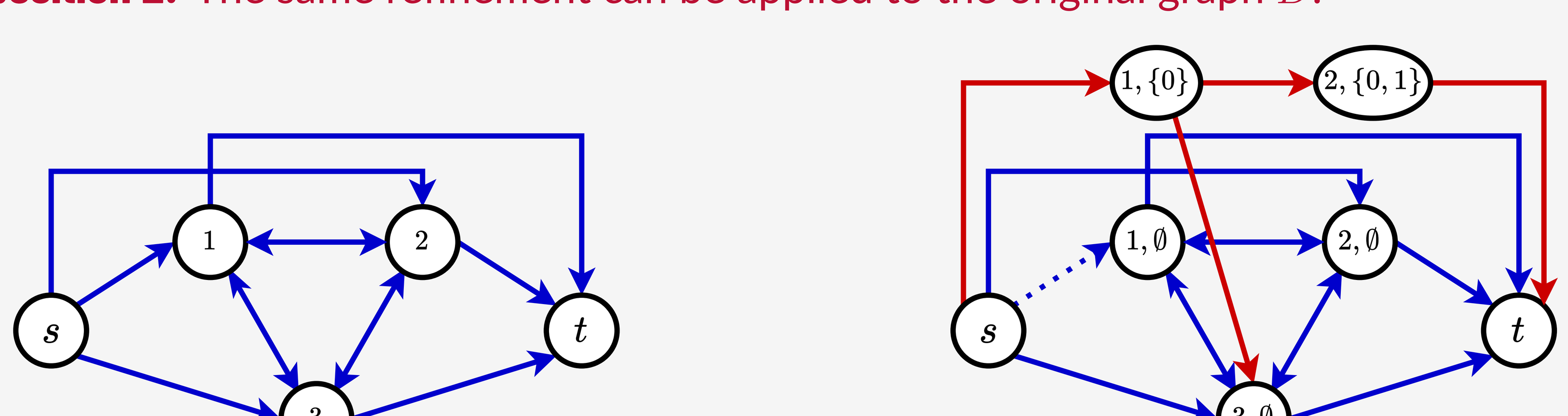


Figure 2. Applying refinement on the original graph D for the same instance as in Figure 1. Observe that the graph in the right do not contain path $(s, 1, 2, 3, t)$ (and other infeasible paths).

Does it work?

Implementation and Setup

- Implemented in C++ and used **BaPCod/VRPSolver** to solve the set partitioning model.
  - **State-of-the-art branch-cut-and-price (BCP) algorithms features** such as ng-path relaxation, labeling algorithms using bucket graphs, rank-1 cuts with limited memory, etc.
- We solve the root using CG+CE. We run for at most **20 iterations**. In each iteration, we eliminate at most **50 paths.**
- We omit results using CG+CE and the state-transition graph, since it **often fails to solve the root in 1 hour**.

Experiments for CCVRP

- Instances and **cuts** from Dinh et al. (2018). **Time limit:** 1 hour.
- **(CG)** uses only VRPSolver+cuts. **(CG+CE)** uses the proposed approach. **(CE)** is the method of *[Karahalios and van Hoeve, 2024]*.

	Dinh et al.		CG		CE			CG+CE		
Instance	LPG	T(s)	LPG	T(s)	LPG	T(s)	El. Col.	LPG	T(s)	El. Col.
A-n32-k5-L	1.30%	86	**0%**	**< 1**	**0%**	2105	685	**0%**	**< 1**	0
A-n32-k6-H	5.90%	396	3.33%	789	3.07%	-	2285	**0%**	**137**	495
A-n44-k7-L	1.50%	2909	1.27%	**21**	0.38%	-	936	**0.04%**	50	113
A-n44-k8-H	8.80%	-	8.38%	-	**4.53%**	-	3961	4.84%	-	961
P-n50-k12-L	2.78%	-	1.63%	525	**0%**	1823	423	**0%**	**45**	240
P-n50-k13-H	7.07%	-	6.29%	-	0.16%	-	1546	**0%**	**262**	739
P-n51-k12-L	3.28%	-	1.55%	1265	0.13%	-	474	**0%**	**34**	267
P-n51-k13-H	7.50%	-	11.88%	-	6.41%	-	2284	**5.99%**	-	965

Conclusion: CG+CE is generally faster than all other approaches. Moreover, it achieves bounds similar to CE with less refinements.

Experiments for 1-PDVRP

- Instances from Gunes et al. (2010), whose best exact algorithm used **constraint programming. Time limit:** 30 minutes.
- Deactivated path enumeration and rank-1 cuts.
- **(CG)** solves a pricing problem with a **non-monotone** resource. $\Longrightarrow$ **Billions of dominance checks** between labels.
- **(+Cuts)** are new cuts that we derived based on previous work for 1-PDTSP.

	Gunes et al.	CG		CG+Cuts		CG+CE+Cuts		
Instance	T(s)	LPG	T(s)	LPG	T(s)	LPG	T(s)	Ref.
$\lvert V_+\rvert = 13,\ k = 1$	< 120	**0%**	259	**0%**	**< 1**	**0%**	**< 1**	0
$\lvert V_+\rvert = 14,\ k = 1$	< 120	**0%**	24	**0%**	**< 1**	**0%**	**< 1**	0
$\lvert V_+\rvert = 15,\ k = 1$	< 120	-	-	**0%**	**< 1**	**0%**	**< 1**	0
$\lvert V_+\rvert = 16,\ k = 1$	< 120	-	-	**0%**	**< 1**	**0%**	**< 1**	0
$\lvert V_+\rvert = 18,\ k = 1$	< 120	-	-	**0%**	**< 1**	**0%**	**< 1**	0
$\lvert V_+\rvert = 30,\ k = 2$	-	-	-	2.05%	169	**0%**	**30**	52
$\lvert V_+\rvert = 60,\ k = 4$	-	-	-	0.24%	**235**	**0.12%**	469	35

Conclusion: Our cuts already do the job, but CG+CE improves the LP gap.

References

[1] Thai Dinh, Ricardo Fukasawa, and James Luedtke. Exact algorithms for the chance-constrained vehicle routing problem, *Mathematical Programming*, 2018.

[2] Canan Gunes, Willem-Jan van Hoeve, and Sridhar Tayur. Vehicle routing for food rescue programs: A comparison of different approaches, *CPAIOR*, 2010.

[3] Anthony Karahalios and Willem-Jan van Hoeve. Column elimination: An iterative approach to solving integer programs, 2024.

[4] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. A generic exact solver for vehicle routing and related problems, *Mathematical Programming*, 2020.