INDUSTRIAL AND SYSTEMS ENGINEERING

ISE

Dantzig-Wolfe and Arc-Flow Reformulations: A Systematic Comparison

Daniel Yamín and Willem-Jan van Hoeve

Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA USA

Ted K. Ralphs

Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA USA

COR@L Technical Report 26T-004

LEHIGH UNIVERSITY

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Dantzig-Wolfe and Arc-Flow Reformulations: A Systematic Comparison

Daniel Yamín*[1], Willem-Jan van Hoeve†[1], and Ted K. Ralphs‡[2]

[1]Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA USA
[2]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA USA

February 24, 2026

Abstract

Dantzig-Wolfe reformulation is a widely used technique for obtaining stronger relaxations in the context of branch-and-bound methods for solving integer optimization problems. Arc-Flow reformulations are a lesser known technique related to dynamic programming that has experienced a resurgence as result of the recent popularization of decision diagrams as a tool for solving integer programs. Although these two reformulation techniques arose independently, the recently proposed solution paradigm known as *column elimination* has revealed that they are in fact closely connected. Building on a unified formulation and notation, this study clarifies the theoretical connections and computational trade-offs between these two reformulations.

We first revisit the known fact that the LP relaxations of these two reformulations yield the same dual bound. We then dig deeper, establishing conditions under which valid inequalities in the original, Dantzig-Wolfe, or Arc-Flow spaces can be translated across reformulations without loss of strength, and reinterpreting iterative strengthening methods, such as decremental state-space relaxation and column elimination, through the lens of cutting planes. To assess the potential impact of these insights empirically, we benchmark both reformulations under identical conditions on the vehicle routing problem with time windows using state-of-the-art column- and cut-generation techniques. The results identify clear contrasts: the Arc-Flow reformulation benefits from faster convergence and performs best when subproblems are highly relaxed or low-dimensional, whereas the Dantzig-Wolfe reformulation is more efficient when the master problem remains compact. Overall, our study provides a unified perspective and practical guidelines for choosing between Dantzig-Wolfe and Arc-Flow reformulations in large-scale integer optimization.

1 Introduction

Many real-world problems, such as vehicle routing, bin packing, and crew scheduling, have an underlying combinatorial structure and can be effectively solved by formulating them as integer

*dyamin@andrew.cmu.edu
†vanhoeve@andrew.cmu.edu
‡ted@lehigh.edu

programs (IPs). Although these IPs may have compact formulations, these formulations often suffer from significant solution symmetry and weak linear optimization problem (LP) relaxations, making them challenging for general-purpose solvers.

To overcome these issues, a common strategy is the classical Dantzig-Wolfe (DW) reformulation, which reduces symmetry and strengthens the dual bounds obtained by solving the LP relaxation at the expense of introducing exponentially many variables (Vanderbeck 2000), each of which corresponds to one solution of a chosen relaxation. To make this approach computationally tractable, several techniques have been developed, including column generation, problem-specific relaxations and pricing algorithms, and dynamic cut generation (see, e.g., Desrosiers et al. (2024)). DW combined with column and cut generation in the context of a branch-and-bound algorithm is one of the most effective methods for solving certain classes of large-scale IPs, despite computational challenges such as convergence issues (Pessoa et al. 2018).

A related line of research involves Arc-Flow (AF) formulations, which represent the feasible solutions of a bounded IP as paths from a root to a terminal node in a directed acyclic graph (DAG). In AF formulations derived from dynamic programming, the DAG is the unfolded state-transition graph: nodes represent states and arcs represent transitions (typically extending or modifying a partial solution by changing or fixing the value of a variable) (Martin et al. 1990). The origins of AF models trace back to Ford and Fulkerson (1958), who studied maximal dynamic flows on time-expanded networks; Shapiro (1968), who modeled the knapsack problem as a pseudopolynomial network flow; and Wolsey (1973), who introduced a general algorithm for solving integer optimization problems by recasting them as longest-route problems.

An AF reformulation can also be derived from a decision diagram, which in the context of optimization is a DAG encoding feasible solutions as paths (Bergman et al. 2016). In fact, the state-transition graph of a dynamic program can be interpreted as a decision diagram under mild conditions. The main difference between AF formulations based on dynamic programming and those based on decision diagrams lies in how the DAG is constructed and manipulated; see de Lima et al. (2022), Castro et al. (2022), and van Hoeve (2024) for reviews.

In contrast to DW, where the solutions to a given relaxation are explicitly enumerated by solving a subproblem, an AF *reformulation* represents those solutions implicitly as paths in a DAG, preserving the improved dual bound while replacing the exponentially many variables associated with individual elements of the given relaxation with a potentially more compact set of arc variables (de Lima et al. 2022). These arcs represent *disaggregations* of the columns in DW, which enable their *recombination* into new feasible solutions during column generation, accelerating convergence (Sadykov and Vanderbeck 2013). Despite being studied for decades, AF has gained renewed attention due to advances in solvers, connections to decision diagrams, and strong results on combinatorial problems, such as cutting and packing (de Lima et al. 2023), machine scheduling (Sadykov and Vanderbeck 2013), and graph coloring (van Hoeve 2022).

While DW and AF are known to be related, formal results on their connection are limited, particularly in the context of advanced algorithms that combine solution of subproblem relaxations with methods for dynamically generating cutting planes, two core techniques in modern exact methods. Establishing these connections will inform the design of more efficient computational methods. Our study aims to establish such connections via a theoretical and computational analysis. First, we revisit the foundations of DW and AF in a unified framework, highlighting their relationships and trade-offs. Building on these foundations, we extend the analysis to settings with cutting planes and subproblem relaxations. For cutting planes, we establish the conditions under

which valid inequalities in the original, DW, or AF spaces can be translated across reformulations, showing that these cuts preserve their strength and analyzing their computational impact. For subproblem relaxations, we reinterpret iterative strengthening methods, such as *decremental state-space relaxation* and *column elimination*, through the lens of cutting planes, clarifying their computational role.

To empirically test our insights, we benchmark DW and AF under identical conditions on the vehicle routing problem with time windows (VRPTW), employing state-of-the-art techniques. The results reveal contrasts between the approaches across instance and methodological features, e.g., when time windows are tight or loose, or when the relaxation is weak or strong. Overall, our study provides new insights into the relative theoretical and computational strengths of the two reformulations.

The remainder of this paper is organized as follows. Section 2 reviews the theoretical and computational foundations of DW and AF. Section 3 analyzes their relationship in the presence of cutting planes, and Section 4 examines iterative strengthening methods for subproblem relaxations. Section 5 illustrates our comparative study on the VRPTW, and Section 6 reports numerical results. Finally, Section 7 outlines guidelines for reformulation choice and future research directions.

2 Theoretical and Computational Foundations

We consider an integer program of the form (Vanderbeck 2000):

$$
\begin{aligned}
z := \max \quad & \mathbf{cx} && \text{(IP-a)}\\
\text{s.t.} \quad & \mathbf{Ax} \le \mathbf{b} && \text{(IP-b)}\\
& \mathbf{Dx} \le \mathbf{d} && \text{(IP-c)}\\
& \mathbf{x} \in \mathbb{Z}^n_+, && \text{(IP-d)}
\end{aligned}
$$

where $\mathbf{A} \in \mathbb{Q}^{m\times n}_+$ and $\mathbf{D} \in \mathbb{Q}^{l\times n}_+$ are nonnegative rational matrices and $\mathbf{c} \in \mathbb{Q}^n_+, \mathbf{b} \in \mathbb{Q}^m_+$, and $\mathbf{d} \in \mathbb{Q}^l_+$ are nonnegative rational vectors. The reason for separating equations (IP-b) from (IP-c) is to define the *subproblem*, a relaxation with the feasible region $X := \{\mathbf{x} \in \mathbb{Z}^n_+ : \mathbf{Dx} \le \mathbf{d}\}$, which we assume has some special structure that makes solution of it more tractable than the original problem. In many applications, for example, the subproblems can be solved effectively using a dynamic programming reformulation. Both X and the feasible region of (IP) are assumed to be bounded and feasible.

2.1 Dantzig-Wolfe Reformulation

Since X is bounded, it contains a finite number of integer points leading to the following conceptual redefinition:

$$
X = \left\{ \mathbf{x} \in \mathbb{R}^n_+ : \mathbf{x} = \sum_{q\in Q} \mathbf{x}^q \lambda_q,\ \sum_{q\in Q} \lambda_q = 1,\ \boldsymbol{\lambda} \in \{0,1\}^{|Q|} \right\} = \{\mathbf{x}^q\}_{q\in Q}, \tag{1}
$$

where Q indexes the solutions of X. The reformulation of X yields the DW reformulation of (IP):

$$\max \quad \sum_{q \in Q} (\mathbf{c}\mathbf{x}^q)\lambda_q \tag{DW-a}$$

$$\text{s.t.} \quad \sum_{q \in Q} (\mathbf{A}\mathbf{x}^q)\lambda_q \leq \mathbf{b} \tag{DW-b}$$

$$\sum_{q \in Q} \lambda_q = 1 \tag{DW-c}$$

$$\boldsymbol{\lambda} \in \{0,1\}^{|Q|}. \tag{DW-d}$$

Although the redefinition of X in (1) introduces exponentially many variables, dropping integrality restriction on $\boldsymbol{\lambda}$ yields $\text{conv}(X)$, which means that solving the LP relaxation of (DW) yields

$$z_{DW}^{LP} = \max \left\{ \sum_{q \in Q} (\mathbf{c}\mathbf{x}^q)\lambda_q \ : \ \sum_{q \in Q} (\mathbf{A}\mathbf{x}^q)\lambda_q \leq \mathbf{b}, \sum_{q \in Q} \lambda_q = 1, \boldsymbol{\lambda} \geq \mathbf{0} \right\}, \tag{2}$$

which is a potentially better dual bound than that yielded by solving the LP relaxation of (IP). In fact, it is not hard to observe that solving the LP relaxation of (DW) yields the so-called *decomposition bound*

$$z_{DW}^{LP} = z_D := \max \{\mathbf{c}\mathbf{x} \ : \ \mathbf{A}\mathbf{x} \leq \mathbf{b}, \ \mathbf{x} \in \text{conv}(X)\} \ \leq \ \max \left\{\mathbf{c}\mathbf{x} \ : \ \mathbf{A}\mathbf{x} \leq \mathbf{b}, \ \mathbf{D}\mathbf{x} \leq \mathbf{d}, \mathbf{x} \in \mathbb{R}^n_+\right\} =: z^{LP} \tag{3}$$

In a branch-and-price algorithm, the LP relaxation of (DW) is solved to obtain an improved dual bound on the optimal solution value. This LP relaxation is referred to as the *Dantzig-Wolfe master problem* (DW–MP) and can be solved by column generation. In each iteration, a subset $Q' \subseteq Q$ of variables is considered, resulting in a restricted DW–MP (DW–RMP). To determine whether the current solution is optimal, we must determine whether any of the columns not included in the current DW–RMP have a positive reduced cost. If not, this is a proof that the solution to the current DW–RMP is also optimal for the DW–MP. The determination of whether the current solution is optimal is done by solving the following DW pricing problem:

$$\overline{c}^\star := -\pi_0 + \max_{\mathbf{x}^q \in X} \left\{(\mathbf{c}\mathbf{x}^q) - \boldsymbol{\pi}(\mathbf{A}\mathbf{x}^q)\right\}, \tag{4}$$

where $\boldsymbol{\pi} \in \mathbb{R}^m_+$ and $\pi_0 \in \mathbb{R}$ are the dual variables associated with constraints (DW-b) and (DW-c). If $\overline{c}^\star \leq 0$, no positive reduced cost column exists, and the current DW–RMP solution yields an optimal DW–MP solution by setting to zero all nongenerated variables. Otherwise, at least one positive reduced cost column $[(\mathbf{c}\mathbf{x}^q), (\mathbf{A}\mathbf{x}^q), 1]$ is added to DW–RMP and the procedure is repeated.

2.2 Arc-Flow Reformulation

AF provides an alternative reformulation with similar properties. The difference is that the members of X are represented as paths in a DAG rather than as discrete points in a polyhedron. The reformulation of (IP) using AF requires two elements: a DAG $\mathcal{D}$ with root node r and terminal node t in which each path is associated with a unique element of X; and a *projection matrix* $\mathbf{T}$ that maps a unit flow from r to t in $\mathcal{D}$ (which denotes a single r-t path) to a unique element of X. Such a DAG and an associated projection matrix always exists, as discussed in further detail below.

Unlike DW, AF seems to lack an obvious unified theoretical framework, since multiple techniques can be used to construct the underlying DAG, which is a necessary ingredient to any formulation. Building on the extended formulations proposed by Sadykov and Vanderbeck (2013), we introduce a generic AF characterization that links the DAG to the subproblem X through the linear projection $\mathbf{T}$, explicitly relating AF to (IP) and (DW).

Definition 1. *A DAG $\mathcal{D} = (\mathcal{N}, \mathcal{E})$, along with a projection matrix $\mathbf{T} \in \{0,1\}^{n \times |\mathcal{E}|}$, a root node $r \in \mathcal{N}$, and a terminal node $t \in \mathcal{N}$, defines an* arc-flow representation *of X if:*

(i) $conv(X) = proj_{\mathbf{x}}(F)$ where F is the flow polytope defined over $\mathcal{D}$, namely,

$$F := \left\{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}_+^{n+|\mathcal{E}|} \,:\, \mathbf{x} = \mathbf{T}\mathbf{y},\ \textstyle\sum_{e \in \delta^+(r)} y_e = 1,\ \sum_{e \in \delta^+(u)} y_e - \sum_{e \in \delta^-(u)} y_e = 0 \ \ \forall u \in \mathcal{N} \setminus \{r,t\}\right\},$$

where $\delta^+(u) := \{e = (u,v) \in \mathcal{E}\}$ and $\delta^-(u) := \{e = (v,u) \in \mathcal{E}\}$; and

(ii) $X = proj_{\mathbf{x}}(Y)$, where $Y := \left\{(\mathbf{x}, \mathbf{y}) \in F \,:\, \mathbf{y} \in \{0,1\}^{|\mathcal{E}|}\right\}$. ■

One obvious way of obtaining a DAG satisfying this definition is by expressing the problem of optimizing over X as a dynamic program, as described in Nemhauser and Wolsey (1999, p. 308). In this construction (which assumes that X is an independence system), the root node represents the zero solution and traversing an arc corresponds to changing the value of a variable. Whenever the arcs in a given DAG correspond to changing (or setting) the value of a single variable, then a projection matrix $\mathbf{T}$ exists and we have an arc-flow representation satisfying Definition 1.

Just as with the DW reformulation, even though the representation of X as paths in a DAG may increase the number of variables substantially, it nevertheless provides a foundation for deriving exact extended formulations, the solution of whose LP relaxations yield improved dual bounds.

Given $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ and its associated projection matrix $\mathbf{T}$ satisfying Definition 1, the associated *AF reformulation* of (IP) is:

$$\begin{aligned}
\max \quad & (\mathbf{cT})\mathbf{y} && && \text{(AF-a)}\\
\text{s.t.} \quad & (\mathbf{AT})\mathbf{y} \le \mathbf{b} && && \text{(AF-b)}\\
& \sum_{e \in \delta^+(r)} y_e = 1 && && \text{(AF-c)}\\
& \sum_{e \in \delta^+(u)} y_e - \sum_{e \in \delta^-(u)} y_e = 0 && u \in \mathcal{N} \setminus \{r,t\} && \text{(AF-d)}\\
& \mathbf{y} \in \{0,1\}^{|\mathcal{E}|}, && && \text{(AF-e)}
\end{aligned}$$

where $\mathbf{y} \in \{0,1\}^{|\mathcal{E}|}$ implies $\mathbf{x} = \mathbf{T}\mathbf{y} \in \mathbb{Z}_+^n$. We denote the *flow constraints* (AF-c) and (AF-d) as $\mathbf{F}\mathbf{y} = \mathbf{f}$. As with DW, the LP relaxation of (AF) yields an improved dual bound

$$z_{AF}^{LP} = \max\{(\mathbf{cT})\mathbf{y} \,:\, (\mathbf{AT})\mathbf{y} \le \mathbf{b},\ \mathbf{F}\mathbf{y} = \mathbf{f},\ \mathbf{y} \ge \mathbf{0}\} \tag{5}$$

This bound can be computed by column-and-row (or, arc-and-node) generation (de Lima et al. 2023). In each iteration, a subset of arcs $\mathcal{E}' \subseteq \mathcal{E}$ and nodes $\mathcal{N}' \subseteq \mathcal{N}$ with $r, t \in \mathcal{N}'$ is considered in a restricted MP (AF–RMP). Given an AF–RMP solution, the pricing step solves:

$$\overline{c}^{\star} := -\rho_r + \max_{\mathbf{y}^p \in Y} \left\{(\mathbf{cT})\mathbf{y}^p - \boldsymbol{\pi}(\mathbf{AT})\mathbf{y}^p\right\}, \tag{6}$$

where P indexes the solutions in Y, i.e., $Y = \{\mathbf{y}^p\}_{p\in P}$; $\boldsymbol{\pi} \in \mathbb{R}^m_+$ are the AF–RMP dual variables associated with constraints (AF-b); and $\boldsymbol{\rho} \in \mathbb{R}^{|\mathcal{N}'|}$ are the AF–RMP dual variables associated with constraints (AF-c) and (AF-d), with $\rho_t := 0$ for notational conciseness. Due to flow constraints $\mathbf{Fy} = \mathbf{f}$, (6) is a longest path problem over $\mathcal{D}$ with modified arc costs $\bar{\mathbf{c}} := (\mathbf{cT}) - \boldsymbol{\pi}(\mathbf{AT})$. If $\bar{c}^\star \le 0$, the current AF–RMP solution yields an optimal AF–MP solution by setting to zero all nongenerated variables. Otherwise, there exists at least one path $\mathbf{y}^p \in Y$ with $\bar{\mathbf{c}}\mathbf{y}^p > \rho_r$, and therefore the arcs and nodes used by $\mathbf{y}^p$ are added to AF–RMP (if not already in $\mathcal{E}'$ and $\mathcal{N}'$). Thus, the flow conservation constraints (AF-d) in AF–RMP are generated *on demand*, with the constraint for node $u \in \mathcal{N}$ added when u first appears as the head or tail of an AF–RMP arc variable. The exactness of this procedure is proved in de Lima et al. (2023). The discussion above extends naturally to (IP) with a block-diagonal structure, as detailed in Appendix A.

2.3 Connections Between Reformulations

In this section, we revisit the key theoretical relationships between DW and AF. First discussed in Sadykov and Vanderbeck (2013) as formal claims or (informal) observations within the broader context of extended formulations, we derive tailored proofs that establish a deeper connection between DW and AF not previously made explicit, also enabling the analysis of DW and AF under cutting planes and subproblem relaxations. We first introduce notation: the feasible set of the LP relaxation of (DW) is defined as $\mathrm{DW}_{LP} := \big\{(\mathbf{x}, \boldsymbol{\lambda}) \in \mathbb{R}^{n+|Q|}_+ : \mathbf{x} = \sum_{q\in Q} \mathbf{x}^q\lambda_q,\ \sum_{q\in Q}(\mathbf{Ax}^q)\lambda_q \le \mathbf{b},\ \sum_{q\in Q}\lambda_q = 1\big\}$, and the feasible set of the LP relaxation of (AF) is defined as $\mathrm{AF}_{LP} := \big\{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{n+|\mathcal{E}|}_+ : \mathbf{x} = \mathbf{Ty},\ (\mathbf{AT})\mathbf{y} \le \mathbf{b},\ \mathbf{Fy} = \mathbf{f}\big\}$. Proposition 1 compares the strength of the two reformulations.

Proposition 1 (**Observation 1 in Sadykov and Vanderbeck (2013)**)**.**

$$z_D \;=\; z^{LP}_{DW} \;=\; z^{LP}_{AF} \;\le\; z^{LP}.$$

Proof. By definition of DW_{LP}, we have $\mathrm{proj}_{\mathbf{x}}(\mathrm{DW}_{LP}) = \big\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b}\big\} \cap \big\{\mathbf{x} = \sum_{q\in Q}\mathbf{x}^q\lambda_q : \sum_{q\in Q}\lambda_q = 1,\ \boldsymbol{\lambda} \ge \mathbf{0}\big\} = \big\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b}\big\} \cap \mathrm{conv}(X)$. Similarly, by definition of AF_{LP} and Definition 1(i), $\mathrm{proj}_{\mathbf{x}}(\mathrm{AF}_{LP}) = \big\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b}\big\} \cap \{\mathbf{x} = \mathbf{Ty} : \mathbf{Fy} = \mathbf{f}, \mathbf{y} \ge \mathbf{0}\} = \big\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b}\big\} \cap \mathrm{conv}(X)$. Both LP relaxations maximize $\mathbf{cx} = \mathbf{c}\sum_{q\in Q}\mathbf{x}^q\lambda_q = \mathbf{c}(\mathbf{Ty})$ over the same convex set, hence $z^{LP}_{DW} = z^{LP}_{AF} = z_D$. Finally, since $\mathrm{conv}(X) \subseteq \{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Dx} \le \mathbf{d}\}$, it follows that $z_D \le z^{LP}$. □

Proposition 2 relates the number of columns in DW to the number of paths in AF. Based on this, Remark 1 highlights a symmetry drawback in AF.

Proposition 2. *The number of root-to-terminal paths in the DAG of AF is at least as large as the number of DW columns.*

Proof. By Definition 1(ii), each $\mathbf{y}^p \in Y$ projects to exactly one $\mathbf{x}^q \in X$ via $\mathbf{Ty}^p = \mathbf{x}^q$. Therefore, we can define the index map $\varphi : P \longrightarrow Q,\ \varphi(p) := q$ with $\mathbf{Ty}^p = \mathbf{x}^q$. Note that φ is surjective but generally not injective, because two different paths $\mathbf{y}^p, \mathbf{y}^{p'}$ can project to the same $\mathbf{x}^q$, i.e., $\varphi(p) = \varphi(p')$. Thus, the inverse image $\phi := \varphi^{-1} : Q \longrightarrow 2^P \setminus \{\emptyset\},\ q \longmapsto \{p \in P : \varphi(p) = q\}$, is a nonempty set-valued map. Henceforth, we fix a selection $\sigma : Q \to P$ with $\sigma(q) \in \phi(q)$ for all $q \in Q$. The surjection of φ implies $|Q| \le |P|$. □

Remark 1. *AF has a symmetry drawback relative to DW, stemming from the non-uniqueness of lifted solutions, i.e., $\phi(\cdot)$ is not necessarily a singleton. This may result in redundant paths encoded in the DAG and increased computational overhead.*

Remark 2 analyzes the computational complexity of lifting solutions from the original to the AF space. Building on this, Proposition 3 establishes a constructive correspondence between DW and AF solutions, while Remark 3 relates their pricing problems.

Remark 2 (**Observation 4 in Sadykov and Vanderbeck (2013)**). *Given $\mathbf{x}^q \in X$, a lifting procedure outputs $\mathbf{y}^p \in Y$ such that $\mathbf{x}^q = \mathbf{T}\mathbf{y}^p$. One generic lifting procedure is solving $\max\{0 : \mathbf{T}\mathbf{y} = \mathbf{x}^q, \mathbf{F}\mathbf{y} = \mathbf{f}, \mathbf{y} \in \{0,1\}^{|\mathcal{E}|}\}$, which is NP-hard. Dedicated lifting procedures can often recover the corresponding path through a linear scan of the DAG. For example, Martin et al. (1990) propose a greedy lifting procedure for AF models based on dynamic programming that runs in polynomial time in the size of $\mathcal{D}$.*

Proposition 3. *For every $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}) \in \mathrm{DW}_{LP}$, there exists $\bar{\mathbf{y}}$ with $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathrm{AF}_{LP}$. Similarly, for every $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathrm{AF}_{LP}$, there exists $\bar{\boldsymbol{\lambda}}$ with $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}) \in \mathrm{DW}_{LP}$.*

Proof. Let $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}) \in \mathrm{DW}_{LP}$. Define $\bar{\mathbf{y}} := \sum_{q \in Q} \mathbf{y}^{\sigma(q)} \bar{\lambda}_q$ with $\mathbf{1}^\top \bar{\boldsymbol{\lambda}} = 1$ and $\bar{\boldsymbol{\lambda}} \geq \mathbf{0}$. The existence of $\mathbf{y}^{\sigma(q)}$ for every $q \in Q$ is guaranteed by Definition 1(ii). It follows that $\mathbf{T}\bar{\mathbf{y}} = \mathbf{T}\left(\sum_{q \in Q} \mathbf{y}^{\sigma(q)} \bar{\lambda}_q\right) = \sum_{q \in Q} \left(\mathbf{T}\mathbf{y}^{\sigma(q)}\right)\bar{\lambda}_q = \sum_{q \in Q} \mathbf{x}^q \bar{\lambda}_q = \bar{\mathbf{x}}$, $\mathbf{F}\bar{\mathbf{y}} = \mathbf{F}\left(\sum_{q \in Q} \mathbf{y}^{\sigma(q)} \bar{\lambda}_q\right) = \sum_{q \in Q} \left(\mathbf{F}\mathbf{y}^{\sigma(q)}\right)\bar{\lambda}_q = \sum_{q \in Q} \mathbf{f}\bar{\lambda}_q = \mathbf{f}$, , and $(\mathbf{A}\mathbf{T})\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{x}} = \sum_{q \in Q} (\mathbf{A}\mathbf{x}^q)\bar{\lambda}_q \leq \mathbf{b}$. Thus, $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathrm{AF}_{LP}$. For the converse, let $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \in \mathrm{AF}_{LP}$. Since $\mathcal{D}$ is acyclic, the flow decomposition theorem (Ahuja et al. 1993, p. 33) guarantees that nonnegative arc flow $\bar{\mathbf{y}}$ can be decomposed into a (not necessarily unique) path flow, that is, $\bar{\mathbf{y}} = \sum_{p \in P} \mathbf{y}^p \theta_p$ with $\mathbf{1}^\top \boldsymbol{\theta} = 1$ and $\boldsymbol{\theta} \geq \mathbf{0}$, where θ_p is the flow through path $\mathbf{y}^p$. For every $q \in Q$, set $\bar{\lambda}_q := \sum_{p \in \phi(q)} \theta_p$. Consequently, $\mathbf{1}^\top \bar{\boldsymbol{\lambda}} = 1$ and $\bar{\boldsymbol{\lambda}} \geq \mathbf{0}$. Since $\bar{\mathbf{x}} = \mathbf{T}\bar{\mathbf{y}} = \mathbf{T}\left(\sum_{p \in P} \mathbf{y}^p \theta_p\right) = \sum_{p \in P} \left(\mathbf{T}\mathbf{y}^p\right)\theta_p = \sum_{p \in P} \mathbf{x}^{\varphi(p)} \theta_p = \sum_{q \in Q} \mathbf{x}^q \bar{\lambda}_q$ and $\sum_{q \in Q} \left(\mathbf{A}\mathbf{x}^q\right)\bar{\lambda}_q = \mathbf{A}\bar{\mathbf{x}} = (\mathbf{A}\mathbf{T})\bar{\mathbf{y}} \leq \mathbf{b}$, then $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}}) \in \mathrm{DW}_{LP}$. □

Remark 3 (**Remark 3 in Sadykov and Vanderbeck (2013)**). *For any $\boldsymbol{\pi} \in \mathbb{R}^m_+$, we have that $\max_{\mathbf{x}^q \in X}\{(\mathbf{c}\mathbf{x}^q) - \boldsymbol{\pi}(\mathbf{A}\mathbf{x}^q)\} = \max_{\mathbf{y}^p \in Y}\{(\mathbf{c}\mathbf{T})\mathbf{y}^p - \boldsymbol{\pi}(\mathbf{A}\mathbf{T})\mathbf{y}^p\}$. As a result, in AF, one can solve* (4) *and* lift *the solution, whereas in DW, one can solve* (6) *and* project *the solution.*

2.4 Computational Implications in Practice

In this section, we discuss the computational implications of these connections when solving the reformulations. From the subproblem perspective, pricing is often performed in the AF space even under a DW reformulation, enabling the use of problem-specific techniques unavailable in the original space. Pricing in the AF space may rely on either an implicit or explicit DAG: DW-based methods typically use implicit representations via labeling algorithms (see, e.g., Desrosiers et al. (2024)), while AF approaches employ explicit DAGs and shortest-path algorithms (see, e.g., Karahalios (2025)). The implicit approach supports larger DAGs, and the explicit one enables fine-grained manipulation.

From the master problem perspective, DW and AF differ in how they represent subproblem solutions: DW uses compact (aggregated) columns, while AF employs arc-based (disaggregated)

ones. Aggregated columns yield smaller RMPs and faster intermediate LP solves, whereas disaggregated columns introduce multiple variables and constraints per path, enlarging the RMP and increasing LP solution time. Consequently, the compact DW representation is more advantageous as the AF space grows. Conversely, disaggregated columns in AF enable path *recombination* into new subproblem solutions, accelerating convergence (Sadykov and Vanderbeck 2013). Overall, AF is expected to perform better when recombination is sufficiently effective to offset its higher per-iteration cost; otherwise, DW is likely to perform better due to its lower iteration cost.

3 Cutting Planes

Since integrality constraints are imposed on the $\mathbf{x}$-, $\boldsymbol{\lambda}$-, and $\mathbf{y}$-variables in (IP), (DW), and (AF), valid inequalities can be derived in any of these spaces. In this section, we examine the relationships between DW and AF when introducing three classes of cuts, namely those derived in the original space (§3.1), in the DW space (§3.2), and in the AF space (§3.3). Cuts in the original and DW spaces have been extensively studied (Ralphs and Galati 2006, Desaulniers et al. 2011). Despite recent progress by Karahalios (2025), no work has systematically addressed the translation of such cuts into the AF space.

3.1 Cuts in the Original Space

A pair $(\boldsymbol{\alpha}, \alpha_0) \in \mathbb{R}^{n+1}$ is a valid inequality for (IP) if $\boldsymbol{\alpha}\mathbf{x} \le \alpha_0$ holds for all $\mathbf{x} \in \mathbb{Z}^n_+$ satisfying (IP-b)–(IP-c). In (IP), the inequality may be appended to the constraints $\mathbf{Ax} \le \mathbf{b}$ or to the constraints $\mathbf{Dx} \le \mathbf{d}$. In the former case, by (1), the inequality $\sum_{q\in Q}(\boldsymbol{\alpha}\mathbf{x}^q)\lambda_q \le \alpha_0$ is added to (DW). In DW–RMP, an additional dual variable $\mu \ge 0$ is introduced and the pricing objective (4) subtracts $\mu(\boldsymbol{\alpha}\mathbf{x}^q)$. Similarly, by Definition 1, the constraint $(\boldsymbol{\alpha}\mathbf{T})\mathbf{y} \le \alpha_0$ is added to (AF), and the pricing objective (6) subtracts $\mu(\boldsymbol{\alpha}\mathbf{T})\mathbf{y}^p$. Note that the cut $(\boldsymbol{\alpha}, \alpha_0)$ has the same computational impact in DW and AF, adding one constraint to the MP. With such cuts, both reformulations yield the same bound.

Proposition 4. *Let* $\mathrm{DW}^{cut}_{LP} := \mathrm{DW}_{LP} \cap \left\{\boldsymbol{\lambda} \ge \mathbf{0} : \sum_{q\in Q}(\boldsymbol{\alpha}\mathbf{x}^q)\lambda_q \le \alpha_0\right\}$ *and* $\mathrm{AF}^{cut}_{LP} := \mathrm{AF}_{LP} \cap \left\{\mathbf{y} \ge \mathbf{0} : (\boldsymbol{\alpha}\mathbf{T})\mathbf{y} \le \alpha_0\right\}$. *Then,* $proj_{\mathbf{x}}\left(\mathrm{DW}^{cut}_{LP}\right) = proj_{\mathbf{x}}\left(\mathrm{AF}^{cut}_{LP}\right)$.

Proof. Using (1) and Definition 1(i),

$$\begin{aligned}
\mathrm{proj}_{\mathbf{x}}\left(\mathrm{DW}^{\mathrm{cut}}_{LP}\right) &= \left\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b},\ \boldsymbol{\alpha}\mathbf{x} \le \alpha_0\right\} \cap \left\{\mathbf{x} = \textstyle\sum_{q\in Q} \mathbf{x}^q\lambda_q : \mathbf{1}^\top\boldsymbol{\lambda} = 1,\ \boldsymbol{\lambda} \ge 0\right\} \\
&= \left\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b},\ \boldsymbol{\alpha}\mathbf{x} \le \alpha_0\right\} \cap \mathrm{conv}(X) \\
&= \left\{\mathbf{x} \in \mathbb{R}^n_+ : \mathbf{Ax} \le \mathbf{b},\ \boldsymbol{\alpha}\mathbf{x} \le \alpha_0\right\} \cap \left\{\mathbf{x} = \mathbf{Ty} : \mathbf{Fy} = \mathbf{f},\ \mathbf{y} \ge \mathbf{0}\right\} \\
&= \mathrm{proj}_{\mathbf{x}}\left(\mathrm{AF}^{\mathrm{cut}}_{LP}\right).
\end{aligned}$$

□

A second case arises when a valid inequality $(\boldsymbol{\delta}, \delta_0) \in \mathbb{R}^{n+1}$ for (IP) is appended to $\mathbf{Dx} \le \mathbf{d}$. In this case, the subproblem is modified to $\widehat{X} := \{\mathbf{x} \in X : \boldsymbol{\delta}\mathbf{x} \le \delta_0\}$. Consequently, the DW pricing problem (4) is now defined over $\widehat{X}$, but its objective remains unchanged. In DW–MP, the set Q is

redefined as $\widehat{Q}$, where $\widehat{X} = \{\mathbf{x}^q\}_{q\in\widehat{Q}}$. In dynamic cut generation, if a variable $q \in Q \setminus \widehat{Q}$ has been added to DW–RMP, it must be removed from Q'.

In AF, a new DAG $\widehat{\mathcal{D}} = (\widehat{\mathcal{N}}, \widehat{\mathcal{E}})$, satisfying Definition 1(i)-(ii) for $\widehat{X}$ must be constructed. Consequently, the AF pricing problem (6) is defined over $\widehat{Y}$, where $\widehat{X} = \text{proj}_{\mathbf{x}}(\widehat{Y})$. In AF–MP, the arc set $\mathcal{E}$ and node set $\mathcal{N}$ are redefined as $\widehat{\mathcal{E}}$ and $\widehat{\mathcal{N}}$, so generated arcs $e \in \mathcal{E}\setminus\widehat{\mathcal{E}}$ and nodes $u \in \mathcal{N}\setminus\widehat{\mathcal{N}}$ for AF–RMP must be removed from $\mathcal{E}'$ and $\mathcal{N}'$.

3.2 Cuts in the Dantzig-Wolfe Space

A pair $(\boldsymbol{\gamma}, \gamma_0) \in \mathbb{R}^{|Q|+1}$ is a valid inequality for (DW) if $\boldsymbol{\gamma}\boldsymbol{\lambda} \le \gamma_0$ holds for all $\boldsymbol{\lambda} \in \{0,1\}^{|Q|}$ satisfying (DW-b)–(DW-c). As previously discussed, every valid inequality $(\boldsymbol{\alpha}, \alpha_0) \in \mathbb{R}^{n+1}$ in the original space induces a valid inequality in the DW space via $\gamma_q = \boldsymbol{\alpha}\mathbf{x}^q$ for all $q \in Q$ and $\gamma_0 = \alpha_0$. In some cases, however, valid inequalities can also be generated directly in the extended DW space. Such cuts typically require imposing additional constraints in the subproblem that cannot be readily expressed as linear inequalities in the original space. Nevertheless, they have proven effective in certain problem classes (Desaulniers et al. 2011). We discuss these cuts in further detail below.

In a similar spirit to cut-generating functions (Balas 1971), the cut coefficients are defined as $\gamma_q = g(\mathbf{A}\mathbf{x}^q)$ for $q \in Q$, where g is a nonlinear function, e.g., the floor operator in Chvátal-Gomory (CG) rank-1 cuts (Chvátal 1973). If g admits a linearization in the original space by introducing additional variables and constraints, then there exists an augmented (IP) formulation (Villeneuve et al. 2005): $\max\{\mathbf{c}\mathbf{x} : \mathbf{A}\mathbf{x} \le \mathbf{b},\ \gamma \le \gamma_0,\ \mathbf{D}\mathbf{x} \le \mathbf{d},\ \gamma = g(\mathbf{A}\mathbf{x}),\ (\mathbf{x},\gamma) \in \mathbb{Z}^{n+1}_+\}$. In this lifted space, the variable $\gamma \in \mathbb{Z}_+$ captures the value of the nonlinear function $g(\mathbf{A}\mathbf{x})$, and the cut is enforced linearly as $\gamma \le \gamma_0$.

Performing a DW reformulation while keeping $\gamma \le \gamma_0$ at the master level and $\gamma = g(\mathbf{A}\mathbf{x})$ at the subproblem level results in $\widehat{X} := \{(\mathbf{x},\gamma) \in \mathbb{Z}^{n+1}_+ : \mathbf{D}\mathbf{x} \le \mathbf{d},\ \gamma = g(\mathbf{A}\mathbf{x})\}$. Reformulating $\widehat{X} = \{(\mathbf{x}^q, \gamma_q)\}_{q\in\widehat{Q}}$ as in (1) gives $\mathbf{x} = \sum_{q\in\widehat{Q}} \mathbf{x}^q\lambda_q$ and $\gamma = \sum_{q\in\widehat{Q}} \gamma_q\lambda_q$. Thus, DW–MP is exactly (DW) defined over $\widehat{Q}$, with the additional inequality $\sum_{q\in\widehat{Q}} \gamma_q\lambda_q \le \gamma_0$. The DW pricing problem becomes $\overline{c}^\star := -\pi_0 + \max_{(\mathbf{x}^q,\gamma_q)\in\widehat{X}}\{(\mathbf{c}\mathbf{x}^q) - \boldsymbol{\pi}(\mathbf{A}\mathbf{x}^q) - \mu\gamma_q\}$. Introducing the variable γ and the nonlinear function g increases the complexity of the subproblem.

To translate the cut $(\boldsymbol{\gamma}, \gamma_0)$ into (AF), note that $\widehat{X}$ is a finite integer set and therefore a DAG $\widehat{\mathcal{D}} = (\widehat{\mathcal{N}}, \widehat{\mathcal{E}})$ satisfying Definition 1(i)-(ii) for $\widehat{X}$ exists. Associated with $\widehat{\mathcal{D}}$, we have a projection matrix $\widehat{\mathbf{T}} := [\widehat{\mathbf{T}}_{\mathbf{x}}\,;\,\boldsymbol{\beta}] \in \{0,1\}^{(n+1)\times|\widehat{\mathcal{E}}|}$ such that $\mathbf{x} = \widehat{\mathbf{T}}_{\mathbf{x}}\mathbf{y}$ and $\gamma = \boldsymbol{\beta}\mathbf{y}$. As a result, in AF–MP, we have the following constraints: $(\mathbf{A}\widehat{\mathbf{T}}_{\mathbf{x}})\mathbf{y} \le \mathbf{b}$ and $\boldsymbol{\beta}\mathbf{y} \le \gamma_0$, the latter being the new cut. Accordingly, the AF pricing problem becomes $\overline{c}^\star := -\rho_r + \max_{\mathbf{y}^p\in\widehat{Y}}\{(\mathbf{c}\widehat{\mathbf{T}}_{\mathbf{x}})\mathbf{y}^p - \boldsymbol{\pi}(\mathbf{A}\widehat{\mathbf{T}}_{\mathbf{x}})\mathbf{y}^p - \mu(\boldsymbol{\beta}\mathbf{y}^p)\}$, where $\{\mathbf{y}^p\}_{p\in\widehat{P}}$ is the enumerated set of solutions of $\widehat{Y}$. Computationally, this cut can affect AF more severely than DW at the master level, as encoding $\gamma = g(\mathbf{A}\mathbf{x})$ may require a substantial number of additional arcs and nodes in the DAG, enlarging AF–MP and reducing recombination. Still, both reformulations yield the same bound.

Proposition 5. *Let* $\text{DW}^{cut}_{LP} := \text{DW}_{LP} \cap \{\boldsymbol{\lambda} \ge \mathbf{0} : \sum_{q\in\widehat{Q}} \gamma_q\lambda_q \le \gamma_0\}$ *and* $\text{AF}^{cut}_{LP} := \text{AF}_{LP} \cap \{\mathbf{y} \ge \mathbf{0} : \boldsymbol{\beta}\mathbf{y} \le \gamma_0\}$. *Then,* $proj_{(\mathbf{x},\gamma)}(\text{DW}^{cut}_{LP}) = proj_{(\mathbf{x},\gamma)}(\text{AF}^{cut}_{LP})$.

Proof. From the previous discussion,

$$\begin{aligned}
\mathrm{proj}_{(\mathbf{x},\gamma)}\big(\mathrm{DW}^{\mathrm{cut}}_{LP}\big) &= \Big\{(\mathbf{x},\gamma) \in \mathbb{R}^{n+1}_{+} \,:\, \mathbf{A}\mathbf{x} \le \mathbf{b},\, \gamma \le \gamma_0\Big\} \cap \Big\{(\mathbf{x},\gamma) = \textstyle\sum_{q\in\widehat{Q}}(\mathbf{x}^q,\gamma_q)\lambda_q \,:\, \mathbf{1}^\top\boldsymbol{\lambda} = 1,\ \boldsymbol{\lambda} \ge \mathbf{0}\Big\} \\
&= \Big\{(\mathbf{x},\gamma) \in \mathbb{R}^{n+1}_{+} \,:\, \mathbf{A}\mathbf{x} \le \mathbf{b},\, \gamma \le \gamma_0\Big\} \cap \mathrm{conv}(\widehat{X}) \\
&= \Big\{(\mathbf{x},\gamma) \in \mathbb{R}^{n+1}_{+} \,:\, \mathbf{A}\mathbf{x} \le \mathbf{b},\, \gamma \le \gamma_0\Big\} \cap \Big\{(\mathbf{x},\gamma) = \big[\widehat{\mathbf{T}}_{\mathbf{x}}\ \boldsymbol{\beta}\big]^\top\mathbf{y} \,:\, \widehat{\mathbf{F}}\mathbf{y} = \widehat{\mathbf{f}},\, \mathbf{y} \ge \mathbf{0}\Big\} \\
&= \mathrm{proj}_{(\mathbf{x},\gamma)}\big(\mathrm{AF}^{\mathrm{cut}}_{LP}\big).
\end{aligned}$$

□

3.3 Cuts in the Arc-Flow Space

A pair $(\boldsymbol{\beta},\beta_0) \in \mathbb{R}^{|\mathcal{E}|+1}$ is a valid inequality for (AF) if $\boldsymbol{\beta}\mathbf{y} \le \beta_0$ holds for all $\mathbf{y} \in \{0,1\}^{|\mathcal{E}|}$ satisfying (AF-b)–(AF-d). The cut is linear in the original space if there exists $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\boldsymbol{\beta} = \boldsymbol{\alpha}\mathbf{T}$, i.e., $\boldsymbol{\beta}$ lies in the row space of $\mathbf{T}$. In that case, $\boldsymbol{\beta}\mathbf{y} = \boldsymbol{\alpha}\mathbf{x} \le \beta_0$, recovering the case in §3.1. Otherwise, it cannot be represented as a single linear constraint in (IP). Just as in the DW case, valid inequalities may also be generated directly in the extended AF space.

To translate the cut $(\boldsymbol{\beta},\beta_0)$ into (DW), one might write $\sum_{q\in Q}\gamma_q\lambda_q \le \beta_0$ with $\gamma_q := \boldsymbol{\beta}\mathbf{y}^{\phi(q)}$. This, however, raises two issues. First, if $\phi(q)$ contains multiple paths mapping to $\mathbf{x}^q$ and $\boldsymbol{\beta}\mathbf{y}^p \ne \boldsymbol{\beta}\mathbf{y}^{p'}$ for some $p,p' \in \phi(q)$, it is unclear which coefficient to assign to λ_q (see Remark 1). Without these symmetry issues, the cut remains valid in DW and preserves its strength.

Proposition 6. *Let $(\boldsymbol{\beta},\beta_0) \in \mathbb{R}^{|\mathcal{E}|+1}$ be a valid inequality for* (AF). *If $|Q| = |P|$, then $(\boldsymbol{\gamma},\beta_0) \in \mathbb{R}^{|Q|+1}$ with $\gamma_q := \boldsymbol{\beta}\mathbf{y}^{\phi(q)}$ is a valid inequality for* (DW). *Moreover, $\mathrm{proj}_{\mathbf{x}}\big(\mathrm{DW}^{cut}_{LP}\big) = \mathrm{proj}_{\mathbf{x}}\big(\mathrm{AF}^{cut}_{LP}\big)$, where $\mathrm{DW}^{cut}_{LP} := \mathrm{DW}_{LP} \cap \big\{\boldsymbol{\lambda} \ge \mathbf{0} \,:\, \sum_{q\in Q}\gamma_q\lambda_q \le \beta_0\big\}$ and $\mathrm{AF}^{cut}_{LP} := \mathrm{AF}_{LP} \cap \big\{\mathbf{y} \ge \mathbf{0} \,:\, \boldsymbol{\beta}\mathbf{y} \le \beta_0\big\}$.*

Proof. Take any $(\mathbf{x},\boldsymbol{\lambda})$ feasible for (DW). By the assumption, there is a unique $\bar{q} \in Q$ with $\lambda_{\bar{q}} = 1$ and $\lambda_q = 0$ for $q \in Q \setminus \{\bar{q}\}$. Because φ is bijective, $\mathbf{x}^{\bar{q}} = \mathbf{T}\mathbf{y}^{\phi(\bar{q})}$ for a single path $\mathbf{y}^{\phi(\bar{q})} \in Y$ and $(\mathbf{A}\mathbf{T})\mathbf{y}^{\phi(\bar{q})} = \mathbf{A}\mathbf{x}^{\bar{q}} \le \mathbf{b}$. Thus, $\mathbf{y}^{\phi(\bar{q})}$ is feasible for (AF) and $\sum_{q\in Q}\gamma_q\lambda_q = \sum_{q\in Q}(\boldsymbol{\beta}\mathbf{y}^{\phi(q)})\lambda_q = \boldsymbol{\beta}\mathbf{y}^{\phi(\bar{q})} \le \beta_0$, so the cut is valid for all feasible $\boldsymbol{\lambda}$ in (DW).

Next, we show that the cut preserves its strength. Take any $\bar{\mathbf{x}} \in \mathrm{proj}_{\mathbf{x}}(\mathrm{DW}^{\mathrm{cut}}_{LP})$, so there exists $\bar{\boldsymbol{\lambda}} \ge \mathbf{0}$ with $(\bar{\mathbf{x}},\bar{\boldsymbol{\lambda}}) \in \mathrm{DW}^{\mathrm{cut}}_{LP}$. Define $\bar{\mathbf{y}} := \sum_{q\in Q}\mathbf{y}^{\phi(q)}\bar{\lambda}_q$. Since $\mathbf{1}^\top\bar{\boldsymbol{\lambda}} = 1$, we have $\bar{\mathbf{x}} = \mathbf{T}\bar{\mathbf{y}}$, $\mathbf{F}\bar{\mathbf{y}} = \mathbf{f}$, and $(\mathbf{A}\mathbf{T})\bar{\mathbf{y}} = \mathbf{A}\bar{\mathbf{x}} \le \mathbf{b}$. Thus, $(\bar{\mathbf{x}},\bar{\mathbf{y}}) \in \mathrm{AF}_{LP}$. Moreover, $\boldsymbol{\beta}\bar{\mathbf{y}} = \boldsymbol{\beta}\big(\sum_{q\in Q}\mathbf{y}^{\phi(q)}\bar{\lambda}_q\big) = \sum_{q\in Q}(\boldsymbol{\beta}\mathbf{y}^{\phi(q)})\bar{\lambda}_q = \sum_{q\in Q}\gamma_q\bar{\lambda}_q \le \beta_0$. Hence, $(\bar{\mathbf{x}},\bar{\mathbf{y}}) \in \mathrm{AF}^{\mathrm{cut}}_{LP}$ and $\bar{\mathbf{x}} \in \mathrm{proj}_{\mathbf{x}}(\mathrm{AF}^{\mathrm{cut}}_{LP})$. Conversely, take any $\bar{\mathbf{x}} \in \mathrm{proj}_{\mathbf{x}}(\mathrm{AF}^{\mathrm{cut}}_{LP})$, so there exists $\bar{\mathbf{y}} \ge \mathbf{0}$ with $(\bar{\mathbf{x}},\bar{\mathbf{y}}) \in \mathrm{AF}^{\mathrm{cut}}_{LP}$. By the flow decomposition theorem, $\bar{\mathbf{y}} = \sum_{p\in P}\mathbf{y}^p\theta_p$ for some $\mathbf{1}^\top\boldsymbol{\theta} = 1$ and $\boldsymbol{\theta} \ge \mathbf{0}$. For every $q \in Q$, define $\bar{\lambda}_q := \theta_{\phi(q)}$, which is possible because φ is bijective. Then, $\mathbf{1}^\top\bar{\boldsymbol{\lambda}} = 1$ and $\bar{\boldsymbol{\lambda}} \ge \mathbf{0}$. Moreover, $\bar{\mathbf{x}} = \mathbf{T}\bar{\mathbf{y}} = \mathbf{T}\big(\sum_{p\in P}\mathbf{y}^p\theta_p\big) = \sum_{p\in P}(\mathbf{T}\mathbf{y}^p)\theta_p = \sum_{p\in P}\mathbf{x}^{\varphi(p)}\theta_p = \sum_{q\in Q}\mathbf{x}^q\bar{\lambda}_q$ and $\sum_{q\in Q}(\mathbf{A}\mathbf{x}^q)\bar{\lambda}_q = \mathbf{A}\bar{\mathbf{x}} \le \mathbf{b}$. Thus, $(\bar{\mathbf{x}},\bar{\boldsymbol{\lambda}}) \in \mathrm{DW}_{LP}$. Finally, $\sum_{q\in Q}\gamma_q\bar{\lambda}_q = \sum_{q\in Q}\big(\boldsymbol{\beta}\mathbf{y}^{\phi(q)}\big)\bar{\lambda}_q = \sum_{p\in P}\boldsymbol{\beta}(\mathbf{y}^p\theta_p) = \boldsymbol{\beta}\bar{\mathbf{y}} \le \beta_0$, because $(\bar{\mathbf{x}},\bar{\mathbf{y}}) \in \mathrm{AF}^{\mathrm{cut}}_{LP}$. Altogether, $(\bar{\mathbf{x}},\bar{\boldsymbol{\lambda}}) \in \mathrm{DW}^{\mathrm{cut}}_{LP}$. □

The result also holds under milder conditions, namely when $|Q| \le |P|$ and $\boldsymbol{\beta}\mathbf{y}^p = \boldsymbol{\beta}\mathbf{y}^{p'}$ for all $p,p' \in \phi(q)$ and every $q \in Q$. Otherwise, the cut cannot be represented in DW by a single linear inequality, since AF can distinguish between p and p', whereas DW cannot, as both $\mathbf{y}^p$ and $\mathbf{y}^{p'}$ project onto the same point $\mathbf{x}^q \in X$. The second issue occurs in the DW pricing problem. The term $\mu\gamma_q$ must be subtracted from the objective, where $\gamma_q = \boldsymbol{\beta}\mathbf{y}^{\phi(q)}$. As noted earlier, $\boldsymbol{\beta}\mathbf{y}^{\phi(q)}$ need not be linear in the original space, in which case it must be enforced by modifying the subproblem

structure as in §3.2. In practice, these two issues can be resolved by defining the DW reformulation over Y rather than X. In this case, $\mathbf{x} = \sum_{p \in P} \mathbf{x}^p \lambda_p$ with $\sum_{p \in P} \lambda_p = 1$, $\boldsymbol{\lambda} \geq 0$, and $\mathbf{x}^p = \mathbf{T}\mathbf{y}^p$ for all $p \in P$. This reformulation performs pricing in the AF space while preserving the aggregated DW column representation at the master level.

4 Iterative Subproblem Strengthening

The reformulation of the original problem is computationally motivated by the special structure of the subproblem X, which is assumed to make its solution more tractable than the original problem. In some cases, however, even this structured subproblem may still not be sufficiently tractable for practical instances. Like in a cutting-plane algorithm, the subproblem can then be relaxed and iteratively strengthened as needed. This idea underlies well-established techniques such as decremental state-space relaxation (Boland et al. 2006), dynamic discretization discovery (Boland et al. 2017), and column elimination (van Hoeve 2022), which we formalize and reinterpret through the lens of cutting planes.

Recall the definition of the subproblem in the original space as $X := \{\mathbf{x} \in \mathbb{Z}^n_+ : \mathbf{D}\mathbf{x} \leq \mathbf{d}\}$. We define a *subproblem relaxation* as any superset $\widehat{X} \supseteq X$. We denote the index set of the solutions of $\widehat{X}$ by $\widehat{Q}$, which parametrizes the DW reformulation (DW) over this relaxed subproblem. Likewise, we use Definition 1 to define an associated AF reformulation (AF) over a DAG $\widehat{\mathcal{D}}$ with projection matrix $\widehat{\mathbf{T}}$ (both exist by definition). Proposition 1 yields the following corollary.

Corollary 1. *Under the same subproblem relaxation, DW and AF yield the same LP bound.*

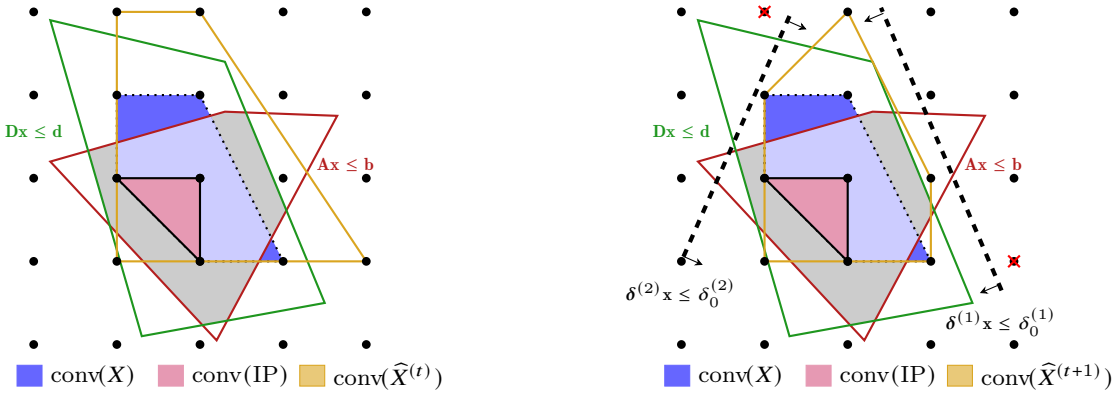
Given an initial relaxation $\widehat{X}^{(0)} \supseteq X$, an iterative subproblem strengthening method proceeds as follows. At iteration t, we solve (DW) or (AF) with the current relaxation $\widehat{X}^{(t)}$, obtaining the bound $\max\{\mathbf{c}\mathbf{x} : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \operatorname{conv}(\widehat{X}^{(t)})\} \geq z_D$. One can verify whether the solution lies in $\operatorname{conv}(X)$ by checking whether every element in the support of the decomposition (i.e., columns in DW and paths in AF) corresponds to a point in X. If so, the decomposition bound z_D has been attained. Otherwise, as described in Definition 2, as stronger relaxation $\widehat{X}^{(t+1)}$ is obtained from $\widehat{X}^{(t)}$ by removing infeasible points with respect to X, so that $\widehat{X}^{(t)} \supset \widehat{X}^{(t+1)} \supseteq X$. Repeating this process results in an iterative strengthening method, formalized in Definition 3. In practice, the procedure may be terminated early (e.g., after a fixed number of iterations or upon reaching a target bound), and the pricing solution may guide the strengthening step.

Definition 2. *Given a subproblem relaxation $\widehat{X}^{(t)}$, a* strengthening *defines $\widehat{X}^{(t+1)} = \widehat{X}^{(t)} \setminus \widetilde{X}^{(t)}$, where $\emptyset \neq \widetilde{X}^{(t)} \subset \widehat{X}^{(t)} \setminus X$.*

Definition 3. *Given an initial subproblem relaxation $\widehat{X}^{(0)} \supset X$, an* iterative strengthening method *defines a finite sequence of relaxations $\widehat{X}^{(0)} \supset \widehat{X}^{(1)} \supset \cdots \supset \widehat{X}^{(T)} \supseteq X$ via strengthenings.*

Observe that, by definition, each strengthening removes at least one infeasible point from the subproblem relaxation. As a consequence, for bounded and feasible integer programs, iterative strengthening will converge to the decomposition bound in a finite number of iterations.

Iterative strengthening methods can be classified by their strengthening step being *global* or *local*. Global methods, common in DW-based approaches, modify the transition function of the underlying dynamic program. An example is decremental state-space relaxation (Boland et al. 2006, Righini and Salani 2008), which reintroduces relaxed constraints by expanding the state



(a) Current relaxation.

(b) Strengthened relaxation.

Figure 1: Iterative strengthening methods from a cutting-plane perspective.

space. Local methods, more common in AF-based approaches, refine an explicit DAG by removing infeasible paths. For example, column elimination (van Hoeve 2022, Karahalios 2025) detects infeasible paths in the flow decomposition of the solution and removes them along with all paths sharing the same infeasible subpath. Proposition 7 characterizes the bound improvement achieved by the strengthening step from a cutting-plane perspective.

Proposition 7. *The bound improvement of a strengthening step can be equivalently obtained by introducing a finite number of subproblem-level cuts.*

Proof. Let $\widehat{X}^{(t)}$ and $\widehat{X}^{(t+1)} \subset \widehat{X}^{(t)}$ be consecutive subproblem relaxations. After the strengthening step, the subproblem bound is that of $\text{conv}(\widehat{X}^{(t+1)})$. By the separation theorem, for every extreme point $\bar{\mathbf{x}} \in \text{ext}\big(\text{conv}(\widehat{X}^{(t)})\big) \setminus \text{conv}(\widehat{X}^{(t+1)})$, there exists an inequality $(\boldsymbol{\delta}, \delta_0) \in \mathbb{R}^{n+1}$ valid for $\text{conv}(\widehat{X}^{(t+1)})$ that is violated by $\bar{\mathbf{x}}$. Because $\text{conv}(\widehat{X}^{(t)})$ is a polytope with finitely many extreme points, a finite set of inequalities $\{(\boldsymbol{\delta}^k, \delta_0^k)\}_{k \in K}$ suffices. Imposing these inequalities at the subproblem level yields $\text{conv}(\widehat{X}^{(t+1)}) = \text{conv}\big(\{\mathbf{x} \in \widehat{X}^{(t)} \ : \ \boldsymbol{\delta}^k \mathbf{x} \le \delta_0^k \ \forall k\}\big)$, because $\text{conv}(\widehat{X}^{(t+1)}) \subseteq \text{conv}(\widehat{X}^{(t)})$ and the inequalities cut off the extreme points of $\text{conv}(\widehat{X}^{(t)})$ that lie outside $\text{conv}(\widehat{X}^{(t+1)})$. □

An illustration of strengthening by cutting planes is given in Figure 1. Proposition 7 provides a polyhedral characterization of iterative strengthening methods with several practical implications. First, the effect of a strengthening step on DW and AF mirrors that of subproblem-level cuts discussed in §3.1. Second, global strengthenings are stronger–since they implicitly introduce more cuts–but also more computationally demanding, as the DAG size grows exponentially with the number of constraints (Vanderbeck 2000). Finally, this polyhedral view offers an alternative proof of convergence for these techniques.

5 Case Study: The Vehicle Routing Problem with Time Windows

We illustrate our comparative analysis on the VRPTW. The problem is defined on a directed graph $G = (V, A)$, where N is the customer set, $V := N \cup \{\underline{0}, \overline{0}\}$ is the vertex set, and $A \subset V \times V$ is the arc set. The objective is to determine a set of minimum-cost routes for K identical vehicles, each departing from a single depot represented by two vertices: the source $\underline{0}$ and the sink $\overline{0}$. Each customer $i \in N$

must be visited exactly once within its time window $[e_i, l_i]$ to satisfy its demand $d_i \in \mathbb{Z}_+$, while the vehicle load must not exceed capacity $C \in \mathbb{Z}_+$. We set $d_{\underline{0}} = d_{\overline{0}} = 0$ and $[e_{\underline{0}}, l_{\underline{0}}] = [e_{\overline{0}}, l_{\overline{0}}] = [E, L]$. Each arc $(i, j) \in A$ has cost $c_{ij} \geq 0$ and travel time $\tau_{ij} \geq 0$, including any service time at vertex i. Let $x_{ij}^k \in \{0, 1\}$ indicate whether vehicle k traverses arc (i, j), $z_i^k \in \{0, 1\}$ whether it visits customer i, and $t_i^k \geq 0$ the service start time at vertex i. The IP formulation is given by (7), where $M_{ij} := \max\{0, l_i + \tau_{ij} - e_j\}$ for $(i, j) \in A$. The objective (7a) minimizes total routing cost. Constraints (7b) ensure that each customer is visited once. Constraints (7c)–(7g) define feasible routes. Constraints (7h) link $\mathbf{x}^k$ and $\mathbf{z}^k$, where the $\mathbf{z}^k$ variables could be *substituted out* but are kept for clarity. Finally, Constraints (7i) define the variable domains.

$$\min \quad \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{7a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} z_i^k = 1 \qquad i \in N \tag{7b}$$

$$\sum_{j\in\delta^+(i)} x_{ij}^k - \sum_{j\in\delta^-(i)} x_{ji}^k = \begin{cases} 1, & i = \underline{0} \\ 0, & i \in N \\ -1, & i = \overline{0} \end{cases} \qquad i \in V,\ k = 1, \ldots, K \tag{7c}$$

$$\sum_{i\in N} d_i\, z_i^k \leq C \qquad k = 1, \ldots, K \tag{7d}$$

$$e_i\, z_i^k \leq t_i^k \leq l_i\, z_i^k \qquad i \in N,\ k = 1, \ldots, K \tag{7e}$$

$$E \leq t_i^k \leq L \qquad i \in \{\underline{0}, \overline{0}\},\ k = 1, \ldots, K \tag{7f}$$

$$t_i^k + \tau_{ij} - M_{ij}(1 - x_{ij}^k) \leq t_j^k \qquad (i, j) \in A,\ k = 1, \ldots, K \tag{7g}$$

$$z_i^k = \sum_{j\in\delta^+(i)} x_{ij}^k \qquad i \in N,\ k = 1, \ldots, K \tag{7h}$$

$$\mathbf{x}^k \in \{0,1\}^{|A|},\ \ \mathbf{z}^k \in \{0,1\}^{|N|},\ \ \mathbf{t}^k \in \mathbb{R}_+^{|V|} \qquad k = 1, \ldots, K, \tag{7i}$$

5.1 Dantzig-Wolfe Reformulation

Selecting (7b) as the *complicating* constraints and (7c)–(7h) as the *nice* ones (separable and identical for each vehicle $k = 1, \ldots, K$) yields $X := \{(\mathbf{x}, \mathbf{z}) \in \{0,1\}^{|A|+|N|} : \exists\, \mathbf{t} \in \mathbb{R}_+^{|V|} \text{ s.t. } (7c) - (7h)\} = \{(\mathbf{x}^q, \mathbf{z}^q)\}_{q\in Q}$. Because of (7h), we write $\mathbf{x}^q \in X$. DW reformulation follows:

$$\min \quad \sum_{k=1}^{K} \sum_{q\in Q} c_q \lambda_q^k \tag{8a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{q\in Q} z_i^q \lambda_q^k = 1 \qquad i \in N \tag{8b}$$

$$\sum_{q\in Q} \lambda_q^k = 1 \qquad k = 1, \ldots, K \tag{8c}$$

$$\lambda_q^k \in \{0, 1\} \qquad k = 1, \ldots, K,\ q \in Q, \tag{8d}$$

where $c_q := \sum_{(i,j)\in A} c_{ij} x_{ij}^q$ for every $q \in Q$. As discussed in Appendix A, an aggregation can be performed to remove index k, but we keep it for the cutting planes discussion.

In DW–RMP, we have dual variables $\boldsymbol{\pi} \in \mathbb{R}^{|N|}$ and $\pi_0 \in \mathbb{R}$ associated with constraints (8b) and (8c), and a modified cost $\overline{c}_{ij} = c_{ij} - \pi_i$ can be associated with each arc $(i,j) \in A$, where $\pi_{\underline{0}} := \pi_0$. The subproblem X is an Elementary Shortest Path Problem with Resource Constraints (ESPPRC), which can be more efficiently solved as a dynamic program in which states are represented by labels encoding subpaths in G (see Desrosiers et al. (2024)). Each label is a tuple $\mathscr{L} = (v, c, d, t, \mathscr{E})$, where $v \in V$ is the current vertex, $c \in \mathbb{R}$ is the cumulative reduced cost, $d \in \mathbb{Z}_+$ is the cumulative load, $t \in [E, L]$ is the earliest service start time, and $\mathscr{E} \subseteq N$ is the set of visited customers. These tuple components are commonly referred to as *resources*.

The initial label at the source depot $\underline{0}$ is $\mathscr{L} = (\underline{0}, 0, 0, E, \emptyset)$ and labels are extended through outgoing arcs until reaching the sink depot $\overline{0}$. Given a label $\mathscr{L}$ with $v(\mathscr{L}) = i$, an extension along arc $(i,j) \in A$ (i.e., a state-transition), such that $j \notin \mathscr{E}(\mathscr{L})$, produces a new label $\mathscr{L}'$, where:

$$\begin{aligned} &v(\mathscr{L}') = j, \quad c(\mathscr{L}') = c(\mathscr{L}) + \overline{c}_{ij}, \quad d(\mathscr{L}') = d(\mathscr{L}) + d_j, \\ &t(\mathscr{L}') = \max\{t(\mathscr{L}) + \tau_{ij}, e_j\}, \quad \mathscr{E}(\mathscr{L}') = \mathscr{E}(\mathscr{L}) \cup \{j\}. \end{aligned} \tag{9}$$

The resulting label $\mathscr{L}'$ is feasible if $d(\mathscr{L}') \le C$ and $t(\mathscr{L}') \le l_j$. To avoid enumerating all feasible solutions, a *labeling algorithm* discards non-useful subpaths by verifying a dominance rule between labels. Given two label $\mathscr{L}_1$ and $\mathscr{L}_2$ with $v(\mathscr{L}_1) = v(\mathscr{L}_2)$, $\mathscr{L}_1$ dominates $\mathscr{L}_2$ if:

$$c(\mathscr{L}_1) \le c(\mathscr{L}_2) \ \wedge \ d(\mathscr{L}_1) \le d(\mathscr{L}_2) \ \wedge \ t(\mathscr{L}_1) \le t(\mathscr{L}_2) \ \wedge \ \mathscr{E}(\mathscr{L}_1) \subseteq \mathscr{E}(\mathscr{L}_2). \tag{10}$$

Conditions (10) guarantee that any feasible extension of the dominated label $\mathscr{L}_2$ is also a feasible extension of the dominant label $\mathscr{L}_1$ with an equal or better reduced cost.

Solving the ESPPRC is strongly NP-hard, whereas the SPPRC can be solved in pseudopolynomial time (Desrochers et al. 1992). For this reason, several elementarity-based relaxations have been proposed, with the *ng*-SPPRC being the most widely used (Baldacci et al. 2011). In this relaxation, each customer $i \in N$ is associated with an *ng*-set $M_i \subseteq N$ containing the $\Delta \in \mathbb{Z}_+$ closest customers to i and customer i itself. An *ng*-route allows a cycle starting and ending at customer $j \in N$ if and only if there exists a customer $i \in N$ in the cycle for which $j \notin M_i$.

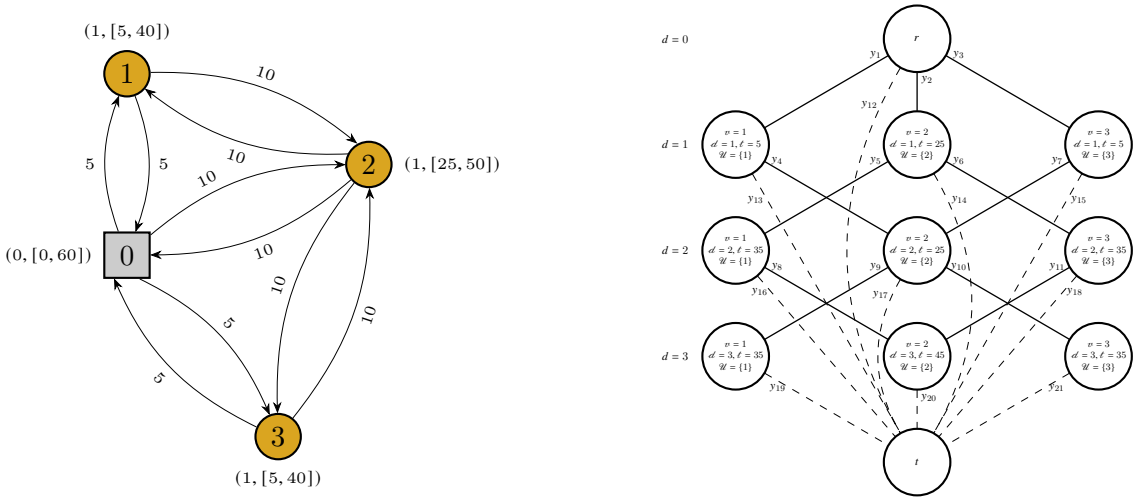
In the labeling algorithm, the set $\mathscr{E}(\mathscr{L})$ is replaced by $\mathscr{U}(\mathscr{L})$, the set of customers whose visit would violate the *ng*-route cycling restrictions. Unlike $\mathscr{E}(\mathscr{L})$, in the set $\mathscr{U}(\mathscr{L})$, customers may be forgotten and subsequently revisited. When a new label $\mathscr{L}'$ is created, then $\mathscr{U}(\mathscr{L}') = \big(\mathscr{U}(\mathscr{L}) \cap M_j\big) \cup \{j\}$. In the dominance rule, $\mathscr{E}(\mathscr{L}_1) \subseteq \mathscr{E}(\mathscr{L}_2)$ is replaced by $\mathscr{U}(\mathscr{L}_1) \subseteq \mathscr{U}(\mathscr{L}_2)$. Since $|\mathscr{U}(\mathscr{L})| \le \Delta$, this dominance criterion is less restrictive, potentially allowing more labels to be discarded. The bound quality produced by the *ng*-SPPRC depends on parameter Δ. When $\Delta = |N| - 1$, the *ng*-SPPRC reduces to the ESPPRC; when $\Delta = 0$, it becomes the SPPRC. In the remainder, X denotes the *ng*-SPPRC.

5.2 Arc-Flow Reformulation

A DAG $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ satisfying Definition 1(i)-(ii) for X can be derived from the unfolded dynamic program associated with the *ng*-SPPRC. Each node $u \in \mathcal{N}$ represents a state defined by a tuple $(v, d, t, \mathscr{U})$, and each arc $e \in \mathcal{E}$ corresponds to a state transition, as defined in (9). Moreover, the root node $r \in \mathcal{N}$ encodes the initial state $(\underline{0}, 0, E, \emptyset)$. Figure 2 illustrates this construction for a small instance with $K = 2$, $C = 3$, $c_{ij} = \tau_{ij}$ for all $(i,j) \in A$, and vertex 0 represents both $\underline{0}$ and $\overline{0}$.

Due to state-transitions (9), the column associated with arc $e = (u, v) \in \mathcal{E}$ in the projection matrix $\mathbf{T} \in \{0,1\}^{|A|\times|\mathcal{E}|}$ has an entry equal to one in the row corresponding to arc $\big(\upsilon(u), \upsilon(v)\big) \in A$. Explicitly, this projection is $x_{ij}^k = \sum_{e=(u,v)\in\mathcal{E}\,:\,\upsilon(u)=i,\upsilon(v)=j} y_e^k$ for $(i,j) \in A$ and $k = 1,\dots,K$. The cost coefficient of arc $e \in \mathcal{E}$ is $c_e = (\mathbf{cT})_e = c_{\upsilon(u),\upsilon(v)}$. The coefficient of a constraint (7b) associated with customer $i \in N$ for an arc $e = (u, v) \in \mathcal{E}$ is 1 if $\upsilon(u) = i$, and 0 otherwise. Appendix B.1 shows these computations explicitly for the example in Figure 2.

The AF reformulation of (7) is given in (11). As in (8), we keep index k for the cutting planes discussion. In AF–RMP, we have dual variables $\boldsymbol{\pi} \in \mathbb{R}^{|N|}$ and $\boldsymbol{\rho} \in \mathbb{R}^{|\mathcal{N}'|}$ associated with constraints (11b) and (11c)–(11d), and a modified cost $\overline{c}_e = c_e - \pi_{\upsilon(u)}$ can be associated with each arc $e = (u, v) \in \mathcal{E}$, where $\pi_{\underline{0}} := \rho_r$.



(a) Small-size instance.

(b) DAG $\mathcal{D} = (\mathcal{N}, \mathcal{E})$ with $\Delta = 0$.

Figure 2: Illustrative example.

$$\min \quad \sum_{k=1}^{K} \sum_{e\in\mathcal{E}} c_e y_e^k \tag{11a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \sum_{\substack{e=(u,v)\in\mathcal{E}:\\ \upsilon(u)=i}} y_e^k = 1 \qquad i \in N \tag{11b}$$

$$\sum_{e\in\delta^+(r)} y_e^k = 1 \qquad k = 1,\dots,K \tag{11c}$$

$$\sum_{e\in\delta^+(u)} y_e^k - \sum_{e\in\delta^-(u)} y_e^k = 0 \qquad u \in \mathcal{N}\setminus\{r,t\},\ k = 1,\dots,K \tag{11d}$$

$$y_e^k \in \{0,1\} \qquad e \in \mathcal{E}, k = 1,\dots,K. \tag{11e}$$

5.3 Cutting Planes

Cuts in the Original Space Several families of valid inequalities for the VRP have been derived over the original space, with generalized subtour elimination constraints (GSECs) being among the

most popular ones (Laporte 1986). Given a subset $S \subseteq N$ of customers and a lower bound κ_S on the number of vehicles required to serve all customers in S, a GSEC for (7) is written as:

$$\sum_{k=1}^{K} \sum_{(i,j)\in\delta^-(S)} x_{ij}^k \geq \kappa_S, \tag{12}$$

where $\delta^-(S) := \{(i,j) \in A : i \in V \setminus S,\ j \in S\}$ is the set of arcs entering S. In the reformulations, a cut of the form (12) is kept at the master level, as it involves all vehicles. In DW (8), this cut is: $\sum_{k=1}^{K} \sum_{q\in Q} \left(\sum_{(i,j)\in\delta^-(S)} x_{ij}^q\right) \lambda_q^k \geq \kappa_S$, where $\mu \geq 0$ is the associated dual variable in DW–RMP. For each arc $(i,j) \in \delta^-(S)$, the modified cost becomes $\overline{c}_{ij} = c_{ij} - \pi_i - \mu$. In AF (11), the lifted cut is:

$$\sum_{k=1}^{K} \sum_{\substack{e=(u,v)\in\mathcal{E}:\\ v(u)\notin S,\ v(v)\in S}} y_e^k \ \geq\ \kappa_S, \tag{13}$$

and the modified arc cost becomes $\overline{c}_e = c_e - \pi_{v(u)} - \mu$ for each arc $e = (u,v) \in \mathcal{E}$ such that $v(u) \notin S$ and $v(v) \in S$. In Figure 2(a) and 2(b), a GSEC (12) for $S = \{1,3\}$ and its lifted counterpart (13) are: $\sum_{k=1}^{K}(x_{\underline{0},1}^k + x_{\underline{0},3}^k + x_{2,1}^k + x_{2,3}^k) \geq \kappa_S$ and $\sum_{k=1}^{K}(y_1^k + y_3^k + y_5^k + y_6^k + y_9^k + y_{10}^k) \geq \kappa_S$.

Cuts in the Dantzig-Wolfe Space General cuts, such as the CG rank-1 cuts, have been used for DW (8). A CG rank-1 cut for (8) on constraints (8b) is given by:

$$\sum_{k=1}^{K} \sum_{q\in Q} \left\lfloor \sum_{i\in N} u_i z_i^q \right\rfloor \lambda_q^k \leq \left\lfloor \sum_{i\in N} u_i \right\rfloor, \tag{14}$$

where $\mathbf{u} \in [0,1)^{|N|} \cap \mathbb{Q}_+^{|N|}$ and $g(\mathbf{A}\mathbf{x}^q) := \left\lfloor \sum_{i\in N} u_i z_i^q \right\rfloor$ is the function computing the cut coefficient for the λ_q^k-variable. As outlined by Petersen et al. (2008), function g can be linearized in the original space by adding integer variables γ^k for $k = 1,\ldots,K$ and the following constraints to IP (7):

$$\sum_{k=1}^{K} \gamma^k \leq \left\lfloor \sum_{i\in N} u_i \right\rfloor, \quad \sum_{i\in N} u_i z_i^k - (1-\epsilon) \leq \gamma^k \leq \sum_{i\in N} u_i z_i^k \quad \forall k, \quad \text{and} \quad \gamma^k \in \mathbb{Z}_+ \quad \forall k,$$

where a small $\epsilon > 0$ is used to model strict inequality. Keeping the first constraint at the master level and the remaining ones at the subproblem level yields $\widehat{X} := \left\{(\mathbf{x},\gamma) \in \{0,1\}^{|A|} \times \mathbb{Z}_+ : \mathbf{x} \in X, \gamma = g(\mathbf{A}\mathbf{x})\right\}$. In the dynamic program, variable γ expands the state-space by introducing a new resource $\mathscr{g}(\mathscr{L}) \in [0,1)$, capturing the fractional part of the cut coefficient. When extending a label $\mathscr{L}$ along arc $(i,j) \in A$ with $j \in N$, this resource is updated as $\mathscr{g}(\mathscr{L}') = \left(\mathscr{g}(\mathscr{L}) + u_j\right) \mod 1$. Whenever $\mathscr{g}(\mathscr{L}) > \mathscr{g}(\mathscr{L}')$, the cut coefficient increases by one. So, the cumulative reduced cost is updated as $\mathscr{c}(\mathscr{L}') = \mathscr{c}(\mathscr{L}) + \overline{c}_{ij} - \mu \cdot [\mathscr{g}(\mathscr{L}) > \mathscr{g}(\mathscr{L}')]$, where $\mu \leq 0$ is the dual variable associated with cut (14) in DW–RMP and $[\mathbb{P}] = 1$ if $\mathbb{P}$ is true, and $[\mathbb{P}] = 0$ otherwise. Finally, the dominance rule (10) must account for these new resources (one per cut), as detailed in Jepsen et al. (2008).

A DAG $\widehat{\mathcal{D}} = (\widehat{\mathcal{N}}, \widehat{\mathcal{E}})$ satisfying Definition 1(i)-(ii) for $\widehat{X}$ can be constructed from this expanded dynamic program. The projection is given by $\gamma^k = \sum_{e=(u,v)\in\widehat{\mathcal{E}}:\, \mathscr{g}(u)>\mathscr{g}(v)} y_e^k$, leading to the following lifted cut in AF (11):

$$\sum_{k=1}^{K} \sum_{\substack{e=(u,v)\in\widehat{\mathcal{E}}:\\ \mathscr{g}(u)>\mathscr{g}(v)}} y_e^k \leq \left\lfloor \sum_{i\in N} u_i \right\rfloor,$$

and the modified arc cost becomes $\overline{c}_e = c_e - \pi_{v(u)} - \mu$ for $e = (u, v) \in \widehat{\mathcal{E}}$ such that $\mathcal{g}(\mathscr{L}) > \mathcal{g}(\mathscr{L}')$. Appendix B.2 illustrates a CG rank-1 cut in the DW and AF spaces for the example in Figure 2.

Iterative Subproblem Strengthening We apply two strengthening methods for the VRPTW: decremental state-space relaxation and column elimination. Appendix B.3 illustrates both methods on the example in Figure 2.

6 Empirical Comparison of Dantzig-Wolfe and Arc-Flow Reformulations

6.1 Experimental Setup

To test our computational insights, we benchmark DW and AF on the linear relaxation (root node) of the VRPTW. We use the benchmark of Solomon (1987), which follows a four-parameter naming convention `DTm-n`. Parameter `D` is the customer geographical distribution: R for random, C for clustered, or RC for a mix of random and clustered. Parameter `T` is the tightness of the time windows, where instances of type 1 have tighter time windows than instances of type 2. Parameter `m` is the two-digit instance number, and parameter `n` is the number of customers.

We solve both DW and AF using column generation. The subproblem relaxation is defined as an *ng*-SPPRC and is solved by the labeling algorithm described before. The (exact) labeling algorithm is invoked only when a heuristic algorithm fails to find a negative reduced cost route. This heuristic algorithm is derived from the exact one by disregarding the condition $\mathcal{U}(\mathscr{L}_1) \subseteq \mathcal{U}(\mathscr{L}_2)$ in the dominance check (10). Since the vehicle capacity is not very constraining in these instances, we add a single rounded capacity inequality (12) involving all customers a priori. We employ the α-schedule dual-price smoothing technique, which helps mitigate common convergence issues in column generation (Pessoa et al. 2018). Based on preliminary experiments, we set the initial smoothing factor to 0.65 for DW and 0.50 for AF. For aggregated results, we report the geometric mean, a more robust and conservative measure than the arithmetic mean, as it is less sensitive to large values. Note that we benchmark DW and AF under identical conditions, without tailored enhancements.

All algorithms are implemented in Java using the Branch-and-Price framework of the Java OR library (`jORLib`) and the Java graph theory library (`JGraphT`) by Michail et al. (2020). The LPs are solved using IBM CPLEX solver, version 22.1. All experiments are run on a machine with an Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz with 32GB of RAM and a two-hour time limit.

6.2 Assessment of Column Generation and Reformulation Sizes

We first evaluate the computational benefits of using column generation to solve DW and AF, i.e., solving DW–RMP and AF–RMP, against solving DW–MP and AF–MP statically, after full enumeration. We also analyze reformulation sizes to gain insight into the dimension of the DW and AF spaces. Because enumeration is computationally expensive, these experiments are limited to the `R1` 15-customer instances with *ng*-sets of size $\Delta = 6$. We hypothesize that optimal RMPs are orders of magnitude smaller than fully enumerated MPs. By construction, the number of columns in DW–MP equals the number of paths in AF–MP; we expect a similar correspondence between RMPs.

	Dantzig-Wolfe						Arc-Flow							
	MP		RMP		MP-to-RMP ratios		MP		RMP		MP-to-RMP ratios			
Instance	Columns	Time (s)	Columns	Time (s)	Columns	Time	Paths	Time (s)	Paths	Time (s)	Paths	Time	Arcs	Nodes
R101-15	204	0.01	39	0.05	5.2	0.20	204	0.02	55	0.07	3.7	0.29	3.6	3.5
R102-15	27,739	0.15	110	0.07	252.2	2.14	27,739	0.4	109	0.1	254.5	4.00	152.7	158.2
R103-15	36,707	0.22	125	0.09	293.7	2.44	36,707	0.63	128	0.13	286.8	4.85	160.9	170.6
R104-15	164,120	0.93	205	0.14	800.6	6.64	164,120	7.78	182	0.13	901.8	59.85	455.8	510.8
R105-15	680	0.01	78	0.06	8.7	0.01	680	0.02	98	0.11	6.9	0.18	5.3	5.8
R106-15	53,679	0.30	129	0.09	416.1	3.33	53,679	1.71	134	0.11	400.6	15.55	234.1	256.1
R107-15	66,629	0.36	144	0.10	462.7	3.60	66,629	2.30	144	0.12	462.7	19.17	244.2	288.8
R108-15	236,941	1.32	226	0.12	1,048.4	11.00	236,941	26.32	221	0.18	1,072.1	146.22	507.6	638.4
R109-15	4,315	0.04	103	0.07	41.9	0.57	4,315	0.07	117	0.12	36.9	0.58	26.8	28.6
R110-15	17,392	0.11	189	0.13	92.0	0.85	17,392	0.35	173	0.13	100.5	2.69	61.6	69.4
R111-15	53,407	0.28	156	0.10	342.4	2.80	53,407	2.17	171	0.19	312.3	11.42	177.4	206.9
R112-15	83,275	0.43	277	0.17	300.6	2.53	83,275	7.32	249	0.19	334.4	38.53	171.7	199.0
Geo. Mean	**20,799.7**	**0.16**	**132.9**	**0.09**	**156.5**	**1.33**	**20,799.7**	**0.75**	**138.6**	**0.13**	**150.1**	**5.91**	**90.8**	**100.7**

Table 1: Comparison of DW and AF full MPs and optimal RMPs on R1 15-customer instances.

Table 1 compares the full MPs and optimal RMPs of DW and AF. Columns 2–5 report the number of columns and CPU time for solving DW–MP and DW–RMP, while Columns 6–7 present the corresponding ratios. Columns 8–11 show the number of paths and CPU time for solving AF–MP and AF–RMP, and Columns 12–15 provide the associated ratios, including the number of arcs and nodes in AF–MP relative to AF–RMP. CPU times for the full MPs exclude variable enumeration.

For DW, the MP includes about 155 times more columns than the RMP. While the static approach is slightly faster on a few small instances (e.g., `R101-15`), the dynamic approach is on average 1.3 times faster. For AF, the DAG in the MP contains roughly 150 times more paths than that of the RMP, resulting in AF–MP having about 90 times more arcs (variables) and 100 times more nodes (constraints). Consequently, the dynamic approach to AF is, on average, approximately six times faster than static AF. Finally, the number of DW–MP columns and AF–MP paths are equivalent, while their RMP counterparts are similar, averaging 133 columns for DW and 139 paths for AF.

6.3 Benchmark of Dantzig-Wolfe and Arc-Flow

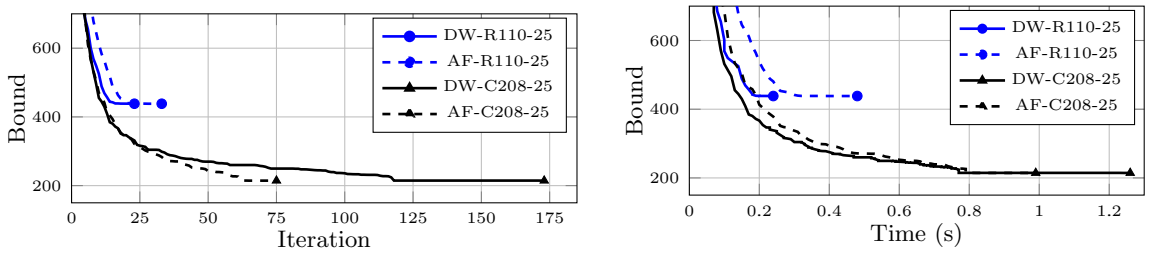
We next compare the performance of DW and AF on 25- and 50-customer instances, yielding a testbed of 112 instances with varying characteristics. We evaluate two different *ng*-relaxations: A weaker relaxation with $\Delta = 6$ and a stronger one with $\Delta = 12$. Based on the discussion in §2.3, we hypothesize that AF will perform better under weaker relaxations or clustered instances, where path recombination is more likely; otherwise, DW is expected to outperform AF.

Table 2 compares DW and AF across instance features and relaxation strengths, considering only instances solved within the time limit. For DW, Columns 2–5 present the lower bound, the time spent on the RMP, the time spent on the pricing problem, and the total CPU time. Columns 6–10 present the corresponding AF metrics, with Column 10 additionally reporting the ratio of DAG paths to priced routes as a measure of path recombination. Finally, Columns 11–13 report DW-to-AF per-instance ratios for the number of variables, pricing iterations, and total CPU time. Detailed results by instance group can be found in Appendix C.1.

For the 25-customer instances (first row), both methods take about 1.3 seconds, with DW being marginally faster overall. For the 50-customer instances (second row), AF is about 1.2 times faster

		Dantzig–Wolfe				Arc–Flow					DW-to-AF Ratios		
Feature	Value	LB	RMP (s)	PP (s)	Time (s)	LB	RMP (s)	PP (s)	Time (s)	Recomb.	Variables	Iterations	Time
Customer number	25	300.95	0.07	1.06	1.21	300.95	0.23	0.90	1.29	1.49	0.26	1.21	0.94
	50	528.65	0.52	21.30	22.52	528.65	2.61	14.43	18.22	2.27	0.23	1.45	1.23
Customer distribution	R	527.21	0.12	3.00	3.31	527.21	0.57	3.02	3.94	1.11	0.21	0.97	0.84
	C	269.57	0.33	6.37	7.02	269.57	1.02	3.29	4.77	3.46	0.29	1.96	1.47
	RC	446.51	0.19	5.61	6.15	446.51	0.79	4.74	6.08	1.62	0.24	1.22	1.01
Time windows	Tight	417.39	0.09	1.49	1.69	417.39	0.28	1.36	1.84	1.21	0.25	1.11	0.91
	Loose	381.17	0.41	15.14	16.20	381.17	2.10	9.60	12.81	2.81	0.24	1.58	1.27
Relaxation	$\Delta = 6$	398.87	0.19	4.75	5.23	398.87	0.77	3.61	4.85	1.84	0.25	1.32	1.08
	$\Delta = 12$	402.92	0.16	3.88	4.27	402.92	0.70	3.57	4.81	1.37	0.22	1.14	0.89

Table 2: Comparison of DW and AF by instance features and relaxation strength (Base case: $\Delta = 6$)



(a) Bound vs. iteration.

(b) Bound vs. time.

Figure 3: Bound progression over iterations and time for `R110-25` and `C208-25` with $\Delta = 6$.

than DW. The main difference is the distribution of computational effort. In the 50-customer cases, DW spends about half-a-second on the RMP and just over 21 seconds on pricing, whereas AF spends roughly 3 seconds on the RMP and 14 seconds on pricing. AF's higher RMP overhead is due to introducing about four times more variables, whereas its lower pricing time is due to faster convergence. Through recombination, paths generate twice as many routes as those explicitly priced, and AF performs 30% fewer iterations on average.

With clustered customers and loose time windows, AF is about 1.4 and 1.3 times faster than DW, with recombination values of approximately 3.5 and 2.8, respectively. In these settings, high recombination gives AF a computational advantage. Conversely, with randomly distributed customers and tighter time windows, DW is 1.2 and 1.1 times faster, as the AF–RMP contains about four times more variables, but recombination is low, favoring DW's compact column representation. Under the stronger relaxation ($\Delta = 12$), AF's advantage diminishes as the state space expands and recombination decreases (from 1.8 to 1.4). Consequently, all DW-to-AF ratios decrease: variables from 0.25 to 0.22, iterations from 1.32 to 1.14, and time from 1.08 to 0.89.

To illustrate this behavior, Figure 3 shows bound progression over iterations and time for instances `R110-25`, where DW performs better, and `C208-25`, where AF does. In the former, both follow similar trends, with AF performing more iterations (33 vs. 23); since DW iterations are cheaper, it finishes sooner. In the latter, AF converges in far fewer iterations (75 vs. 173), while DW stalls between iterations 118–171. Here, AF's higher per-iteration cost is offset by faster convergence.

6.4 Benchmark of Dantzig-Wolfe and Arc-Flow with Cuts

The following set of experiments compares DW and AF when subset-row cuts (SRCs) of size three are added (Jepsen et al. 2008). These rank-1 CG cuts (14) use three nonzero multipliers, each equal to 1/2. Separation is performed by enumeration and, since they increase subproblem complexity, we follow the rules used in Yamín et al. (2025): (i) add at most 30 cuts per iteration, prioritizing the most violated; (ii) allow each customer to appear in at most five triplets; and (iii) add cuts only if the strongest violation exceeds 0.1, with a total limit of 100 cuts. Prior DW research shows that these cuts effectively strengthen the LP bound, and by Proposition 5, AF should obtain the same improvement. However, as discussed in §3.2, DW absorbs this complexity in the subproblem, while AF is also affected at the RMP, where the enlarged state space increases variables and constraints and reduces recombination. We therefore expect these cuts to impact AF more severely than DW.

Table 3 reports the same information as Table 2 with SRCs. Columns 4 and 9 additionally report the average number of cuts added to DW and AF. We report the arithmetic mean in these columns since some values are zero. Detailed results by instance group can be found in Appendix C.2.

DW and AF do not always reach the same bound due to (i) solution symmetry, which can lead to different cuts being separated, and (ii) the cut limit being reached. When cuts are fully separated, both yield the same bound, consistent with Proposition 5. Compared to Table 2, AF still performs best on clustered instances, where few or no cuts are separated, while DW performs better elsewhere, including cases with loose time windows where AF was previously superior. For $\Delta = 6$, the iteration ratio decreases from 1.32 to 1.28, the time ratio from 1.08 to 0.92, and AF recombination from 1.84 to 1.74. Results for $\Delta = 12$ confirm the trend: stronger relaxations combined with SRCs further weaken AF's relative performance, with DW being approximately 1.2 times faster overall.

6.5 Assessment of Dantzig-Wolfe and Arc-Flow with Iterative Subproblem Strengthening

In this set of experiments, we assess DW and AF under decremental state-space relaxation (DSSR) and column elimination (CE). In DSSR, we start with $\Delta = 0$, solve the relaxation, and progressively increase the *ng*-set size whenever cycles appear, until the elementary bound is reached. In CE, we also start with $\Delta = 0$, solve the relaxation, and iteratively remove *conflicting paths* (one cycle per path) from the DAG until elementarity is achieved. Both methods rely on an explicit DAG and a standard shortest-path algorithm, enabling the analysis of subproblem size and avoiding

		Dantzig–Wolfe					Arc–Flow						DW-to-AF Ratios		
Feature	Value	LB	Cuts	RMP (s)	PP (s)	Time (s)	LB	Cuts	RMP (s)	PP (s)	Time (s)	Recomb.	Variables	Iterations	Time
Customer	25	305.76	10.1	0.08	1.41	1.59	305.76	10.6	0.35	1.33	1.90	1.43	0.26	1.22	0.84
number	50	549.97	38.3	0.85	35.23	37.36	550.01	38.0	6.84	27.71	37.32	2.13	0.23	1.34	1.00
Customer	R	540.09	37.0	0.19	6.03	6.68	540.09	38.6	2.04	8.50	11.78	1.06	0.20	0.89	0.57
distribution	C	269.62	0.3	0.34	6.65	7.29	269.62	0.3	0.99	3.56	5.00	3.36	0.29	1.96	1.46
	RC	473.56	35.3	0.27	8.70	9.44	473.61	34.0	1.85	7.36	10.12	1.49	0.25	1.20	0.93
Time	Tight	424.94	25.7	0.14	2.57	2.91	424.97	24.8	0.61	2.41	3.40	1.20	0.25	1.08	0.86
windows	Loose	395.73	22.7	0.49	19.28	20.50	395.73	23.8	3.95	15.26	20.86	2.54	0.24	1.52	0.98
Relaxation	$\Delta = 6$	410.08	24.2	0.26	7.04	7.72	410.09	24.3	1.55	6.07	8.42	1.74	0.24	1.28	0.92
	$\Delta = 12$	408.25	19.1	0.22	5.98	6.53	408.26	18.8	1.26	5.66	7.79	1.36	0.21	1.13	0.84

Table 3: Comparison of DW and AF with cuts by instance features and relaxation strength (Base case: $\Delta = 6$)

		DW-DSSR			AF-DSSR				DW-CE			AF-CE			
Feature	Value	Strength. Iter.	Elim. Routes	Time (s)	Strength. Iter.	Elim. Arcs	Time (s)	Change in DAG	Strength. Iter.	Elim. Routes	Time (s)	Strength. Iter.	Elim. Arcs	Time (s)	Change in DAG
Customer number	25	2.2	369.2	8.78	2.2	1,310.7	9.25	1,221,596.3	10.8	231.5	5.90	10.2	51.5	6.48	982.7
	50	3.5	1,046.9	61.42	3.2	4,622.8	68.11	4,944,384.5	169.1	857.3	44.97	171.1	431.2	45.20	9,696.1
Customer distribution	R	2.8	335.8	11.48	2.5	1,099.4	12.61	1,003,720.2	14.6	236.2	6.57	14.1	81.3	8.25	1,511.4
	C	0.5	321.8	25.88	0.5	1,178.4	25.80	4,199,578.6	3.1	71.5	20.77	2.9	8.1	18.71	252.9
	RC	5.0	1,388.9	29.85	5.0	6,243.1	34.18	3,421,434.7	232.3	1,256.8	22.41	234.9	586.3	23.13	13,144.2
Time windows	Tight	2.7	661.5	20.32	2.6	2,739.5	21.88	2,827,505.0	79.1	501.5	14.17	79.6	215.3	14.98	4,741.4

Table 4: Comparison of DW and AF with DSSR and CE by instance features.

reliance on the dominance rule (10), which is not readily applicable to CE. Although more efficient implementations exist, our purpose is to show that both DW and AF can employ these methods to reach the same bound, each exhibiting different trade-offs. We consider $\Delta = 0$ and only type 1 instances (i.e., with tight time windows), as the explicit DAG for several type 2 instances did not fit in memory. For reference, Appendix C.3 compares the explicit and implicit (labeling) approaches.

Table 4 compares DW and AF with DSSR and CE by instance features. For DW with DSSR, Columns 3–5 report the average number of strengthening iterations, routes eliminated from the RMP due to strengthenings, and CPU time, excluding the compilation time of the initial DAG. Columns 6–8 present the same information for AF with DSSR, with Column 9 additionally reporting the overall change in DAG dimension ($= |\mathcal{E}| + |\mathcal{N}|$), measured as the difference between the final and initial DAGs. This metric is omitted for DW, as it closely follows that of AF. Columns 10–16 report the same information for DW and AF under CE. For the number of strengthenings, eliminated routes, and the change in DAG dimension, we report the arithmetic mean (rather than the geometric mean) due to the presence of negative or zero values (e.g., DSSR can decrease the DAG dimension in specific cases due to forbidden extensions and resource constraints). The results include only instances solved by all approaches. Appendix C.4 provides detailed results by instance group, showing that under DSSR seven instances were not solved, and under CE four, either because the time limit was exceeded or memory was exhausted.

From Table 4, DW and AF exhibit similar performance under both DSSR and CE. The number of strengthenings differs slightly because DW and AF may yield symmetric solutions. For both reformulations, DSSR requires fewer than three strengthenings on average and eliminates over 650 routes from the DW–RMP and 2,700 arcs from the AF–RMP, whereas CE requires about 80 iterations and removes roughly 500 routes from the DW–RMP and 215 arcs from the AF–RMP. Thus, DSSR involves fewer strengthenings but produces a state space about three orders of magnitude larger than that of CE, which maintains a smaller, easier-to-optimize DAG. This is reflected in the running times: DW-DSSR and AF-DSSR take about 21 seconds on average, compared to roughly 14 seconds for DW-CE and AF-CE. Note, however, that the explicit DAG approach affects DSSR more severely due to its global strengthening step, which requires recompiling the DAG at each iteration (explaining why it is typically used with an implicit approach), whereas CE involves only local DAG refinements.

Although general conclusions are difficult to draw, as results are sensitive to implementation choices, CE's weaker strengthenings can be advantageous in memory-constrained settings (e.g., large instances), where DSSR may quickly exhaust memory, as shown in Appendix C.4. Conversely, when the gap between the target bound and the initial relaxation is large, CE's local refinements may require substantially more iterations, whereas DSSR often converges in only a few, as illustrated in

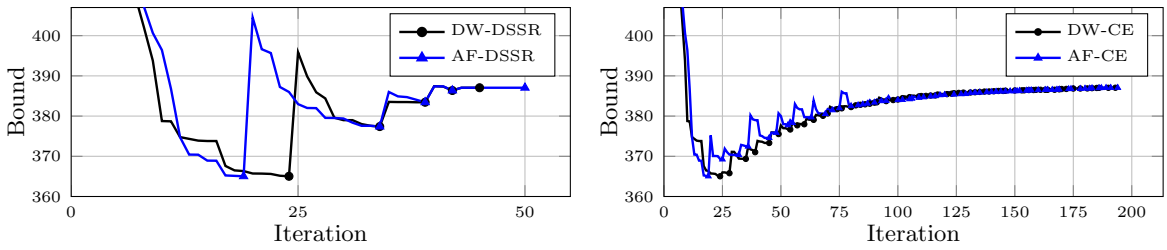


Figure 4: Bound progression over iterations for DW and AF with DSSR and CE on `R112-25`.

Figure 4. In any case, both DW and AF can employ the same strengthening methods.

7 Conclusion

This paper presented a unified framework for comparing Dantzig-Wolfe and Arc-Flow reformulations of integer programs. We formalized several relationships that had previously appeared in the literature, either as formal claims or (informal) observations, establishing precise correspondences between variables, feasible sets, and relaxations in both formulations. We showed that Dantzig-Wolfe and Arc-Flow models yield identical LP bounds, and that valid inequalities, whether defined in the original, DW, or AF spaces, preserve their strength when appropriately translated. We also demonstrated that iterative subproblem-strengthening methods–such as decremental state-space relaxation and column elimination–can be viewed as sequences of subproblem-level cutting planes, clarifying their theoretical foundation.

Computational experiments on the vehicle routing problem with time windows provide empirical support for the theoretical insights developed in this work. In this case study, both reformulations achieve the same bounds, but their computational behavior differs in a structured way. AF tends to perform better when subproblems are highly relaxed or low-dimensional, where path recombination can accelerate convergence. In contrast, DW shows advantages when the master problem remains compact, benefiting from smaller LPs and faster iterations.

Although these empirical results are specific to the VRPTW instances considered, they illustrate how structural properties (e.g., relaxation strength and state-space size) may influence the relative performance of the two reformulations. Overall, the theoretical results provide a formal basis for comparing these reformulations, while the computational study offers problem-specific guidance for selecting between them, potentially extending to closely related decomposition-based integer optimization settings. Further computational experiments across different problem classes are warranted to assess the broader applicability of these findings.

References

Ravindra K Ahuja, Thomas L Magnanti, James B Orlin, et al. *Network flows: Theory, algorithms, and applications*, volume 1. Prentice hall Englewood Cliffs, NJ, 1993.

Egon Balas. Intersection cuts—a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

Roberto Baldacci, Aristide Mingozzi, and Roberto Roberti. New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, 59(5):1269–1283, 2011.

David Bergman, Andre A Cire, Willem-Jan van Hoeve, and John Hooker. *Decision diagrams for optimization*, volume 1. Springer, 2016.

Natashia Boland, John Dethridge, and Irina Dumitrescu. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, 34(1):58–68, 2006.

Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017.

Margarita P Castro, Andre A Cire, and J Christopher Beck. Decision diagrams for discrete optimization: A survey of recent advances. *INFORMS Journal on Computing*, 34(4):2271–2295, 2022.

Vasek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4): 305–337, 1973.

Vinícius L de Lima, Cláudio Alves, François Clautiaux, Manuel Iori, and José M Valério de Carvalho. Arc flow formulations based on dynamic programming: Theoretical foundations and applications. *European Journal of Operational Research*, 296(1):3–21, 2022.

Vinícius Loti de Lima, Manuel Iori, and Flávio Keidi Miyazawa. Exact solution of network flow models with strong relaxations. *Mathematical Programming*, 197(2):813–846, 2023.

Guy Desaulniers, Jacques Desrosiers, and Simon Spoorendonk. Cutting planes for branch-and-price algorithms. *Networks*, 58(4):301–310, 2011.

Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

Jacques Desrosiers, Marco Lübbecke, Guy Desaulniers, and Jean Bertrand Gauthier. *Branch-and-price*. GERAD, HEC Montréal, 2024.

Lester R Ford and Delbert Ray Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.

Mads Jepsen, Bjørn Petersen, Simon Spoorendonk, and David Pisinger. Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2):497–511, 2008.

Anthony Karahalios. *Column Elimination: Iterative Refinement for Solving Arc Flow Formulations*. PhD thesis, Tepper School of Business, Carnegie Mellon University, May 2025.

Gilbert Laporte. Generalized subtour elimination constraints and connectivity constraints. *Journal of the Operational Research Society*, 37(5):509–514, 1986.

R Kipp Martin, Ronald L Rardin, and Brian A Campbell. Polyhedral characterization of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990.

Dimitrios Michail, Joris Kinable, Barak Naveh, and John V Sichi. JGraphT—A Java library for graph data structures and algorithms. *ACM Transactions on Mathematical Software*, 46(2):1–29, 2020.

George L Nemhauser and Laurence A Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, 1999.

Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

Bjørn Petersen, David Pisinger, and Simon Spoorendonk. Chvátal-Gomory rank-1 cuts used in a Dantzig-Wolfe decomposition of the vehicle routing problem with time windows. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 397–419. Springer, 2008.

Ted K Ralphs and Matthew V Galati. Decomposition and dynamic cut generation in integer linear programming. *Mathematical Programming*, 106:261–285, 2006.

Giovanni Righini and Matteo Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Networks: An International Journal*, 51(3):155–170, 2008.

Ruslan Sadykov and François Vanderbeck. Column generation for extended formulations. *EURO Journal on Computational Optimization*, 1(1-2):81–115, 2013.

Jeremy F Shapiro. Dynamic programming algorithms for the integer programming problem—I: The integer programming problem viewed as a knapsack type problem. *Operations Research*, 16(1):103–121, 1968.

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

W.-J. van Hoeve. An Introduction to Decision Diagrams for Optimization. In *INFORMS TutORials in Operations Research*, pages 117–145. INFORMS, 2024.

Willem-Jan van Hoeve. Graph coloring with decision diagrams. *Mathematical Programming*, 192(1):631–674, 2022.

François Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

Daniel Villeneuve, Jacques Desrosiers, Marco E Lübbecke, and François Soumis. On compact formulations for integer programs solved by column generation. *Annals of Operations Research*, 139:375–388, 2005.

Laurence A Wolsey. Generalized dynamic programming methods in integer programming. *Mathematical Programming*, 4(1):222–232, 1973.

Daniel Yamín, Guy Desaulniers, and Jorge E Mendoza. The electric vehicle routing and overnight charging scheduling problem on a multigraph. *INFORMS Journal on Computing*, 37(4):808–830, 2025.

A Block-Diagonal Structure

In many applications, (IP) has the following structure:

$$\max\left\{\mathbf{cx} \,:\, \mathbf{Ax} \le \mathbf{b},\, \mathbf{D}^k\mathbf{x}^k \le \mathbf{d}^k \;\; k = 1,\dots K,\, \mathbf{x} \in \mathbb{Z}^n_+\right\},$$

and the set X decomposes into K independent subproblems $X^k := \{\mathbf{x} \in \mathbb{Z}^{n_k}_+ \,:\, \mathbf{D}^k\mathbf{x} \le \mathbf{d}^k\}$. By reformulating each X^k as in (1), the DW reformulation follows:

$$\max\left\{\sum_{k=1}^{K}\sum_{q\in Q^k}(\mathbf{c}^k\mathbf{x}^{q,k})\lambda^k_q \;:\; \sum_{k=1}^{K}\sum_{q\in Q^k}(\mathbf{A}^k\mathbf{x}^{q,k})\lambda^k_q \le \mathbf{b},\; \sum_{q\in Q^k}\lambda^k_q = 1 \;\; \forall k,\; \boldsymbol{\lambda}^k \in \{0,1\}^{|Q^k|} \;\; \forall k\right\},$$

where $\mathbf{c}^k$ and $\mathbf{A}^k$ are the corresponding blocks in $\mathbf{c}$ and $\mathbf{A}$, and $\{\mathbf{x}^{q,k}\}_{q\in Q^k}$ are the integer points in X^k. When blocks are identical, the variables λ^k_q across different k correspond to the same integer point $q \in Q$ and can be *aggregated* into a single variable $\lambda_q := \sum_{k=1}^K \lambda^k_q$, yielding an aggregated convexity constraint: $\sum_{q\in Q}\lambda_q = K$. For AF, a DAG $\mathcal{D}^k = (\mathcal{N}^k, \mathcal{E}^k)$ satisfying Definition 1(i)-(ii) with respect to X^k must be defined for each block $k = 1,\dots,K$. The AF reformulation follows:

$$\max\left\{\sum_{k=1}^{K}(\mathbf{c}^k\mathbf{T}^k)\mathbf{y}^k \;:\; \sum_{k=1}^{K}(\mathbf{A}^k\mathbf{T}^k)\mathbf{y}^k \le \mathbf{b},\; \mathbf{F}^k\mathbf{y}^k = \mathbf{f}^k \;\; \forall k,\; \mathbf{y}^k \in \{0,1\}^{|\mathcal{E}^k|} \;\; \forall k\right\},$$

where $\mathbf{F}^k\mathbf{y}^k = \mathbf{f}^k$ denotes the flow constraints (AF-c)–(AF-d) on $\mathcal{D}^k$. In the case of identical blocks, we have K identical DAGs that can be aggregated into a single one $\mathcal{D} = (\mathcal{N}, \mathcal{E})$. Accordingly, for each arc $e \in \mathcal{E}$, the variables y^k_e aggregate into a single variable $y_e := \sum_{k=1}^K y^k_e$, leading to aggregated flow constraints: $\sum_{e\in\delta^+(r)} y_e = K$ and $\sum_{e\in\delta^+(u)} y_e - \sum_{e\in\delta^-(u)} y_e = 0$ for all $u \in \mathcal{N}\setminus\{r,t\}$.

B Additional Computations for the Illustrative Example

B.1 Projection Matrix, Cost, and Constraint Coefficients for the VRPTW Arc-Flow Reformulation

Table 5 shows the projection matrix **T** and the computations (**cT**) and (**AT**) for the illustrative example in Figure 2.

		y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}	y_{17}	y_{18}	y_{19}	y_{20}	y_{21}
	$x_{\underline{0},\overline{0}}$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	$x_{\underline{0},1}$	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$x_{\underline{0},2}$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$x_{\underline{0},3}$	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$x_{1,2}$	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T :=	$x_{2,1}$	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	$x_{2,3}$	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	$x_{3,2}$	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	$x_{1,\overline{0}}$	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0
	$x_{2,\overline{0}}$	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0
	$x_{3,\overline{0}}$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1
(**cT**) =	$(c_e)_{e\in\mathcal{E}}$	5	10	5	10	10	10	10	10	10	10	10	0	5	10	5	5	10	5	5	10	5
		0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0
(**AT**) =	$(z_i)_{i\in N}$ =	0	0	0	0	1	1	0	0	1	1	0	0	0	1	0	0	1	0	0	1	0
		0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	1

Table 5: Projection matrix **T** and computations (**cT**) and (**AT**) for the illustrative example.

B.2 CG-Rank 1 Cut in the DW and AF Spaces

Figure 5 shows a CG rank-1 cut with multipliers $\mathbf{u} = (1/2, 1/2, 0)$ in the DW and AF spaces in our illustrative example (Figure 2). These cuts are: $\sum_{k=1}^{K}(\lambda_4^k + \lambda_5^k + \lambda_8^k + \lambda_9^k + \lambda_{10}^k + \lambda_{11}^k + \lambda_{12}^k) \leq 1$ and $\sum_{k=1}^{K}(y_4^k + y_5^k + y_{10}^k + y_{13}^k) \leq 1$.

B.3 Decremental State-Space Relaxation and Column Elimination

The *ng*-relaxation is a state-space relaxation. Figure 6 illustrates its strengthening in our illustrative example (Figure 2). Decremental state-space relaxation forbids cycles of the form $1 \to 2 \to 1$ globally by adding customer 1 to the *ng*-set of customer 2, yielding the DAG in Figure 6(a) with three additional arcs and two additional nodes. As Figure 6(c) illustrates, column elimination instead locally removes the conflicting path $\underline{0} \to 1 \to 2 \to 1 \to \overline{0}$ by adding only two arcs and one node.

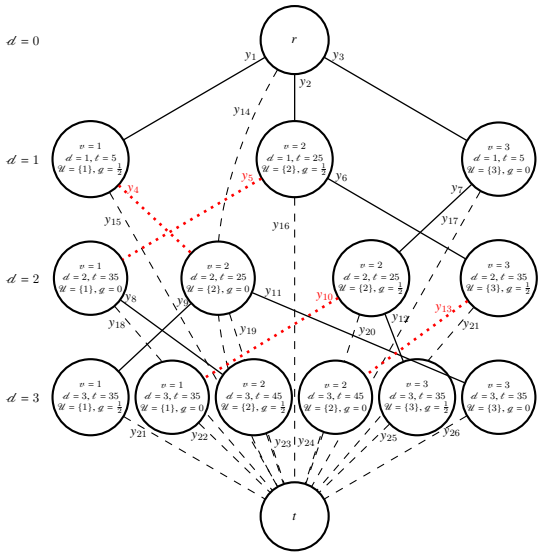
C Detailed Computational Results

C.1 Dantzig-Wolfe and Arc-Flow Performance (Without Subset-Row Cuts)

Table 6 summarizes the performance of DW and AF for both $\Delta = 6$ and $\Delta = 12$, by instance group. For DW, Columns 2–6 present the lower bound, the time spent on the restricted master problem (RMP), the time spent on the pricing problem (PP), the total CPU time, and the number

q	Route	$\gamma_q = \lfloor \frac{1}{2}(z_1^q + z_2^q) \rfloor$
1	$0 \to 1 \to 0$	0
2	$0 \to 2 \to 0$	0
3	$0 \to 3 \to 0$	0
4	$0 \to 1 \to 2 \to 0$	**1**
5	$0 \to 2 \to 1 \to 0$	**1**
6	$0 \to 2 \to 3 \to 0$	0
7	$0 \to 3 \to 2 \to 0$	0
8	$0 \to 1 \to 2 \to 1 \to 0$	**1**
9	$0 \to 1 \to 2 \to 3 \to 0$	**1**
10	$0 \to 2 \to 1 \to 2 \to 0$	**1**
11	$0 \to 2 \to 3 \to 2 \to 0$	**1**
12	$0 \to 3 \to 2 \to 1 \to 0$	**1**
13	$0 \to 3 \to 2 \to 3 \to 0$	0

(a) Cut in the DW space.



(b) Cut in the AF space.

Figure 5: CG rank-1 cut with $\mathbf{u} = (\frac{1}{2}, \frac{1}{2}, 0)$. Variables in the cut are shown in bold red and dotted lines.

of instances solved within the time limit. Columns 7–12 present the corresponding AF metrics, with Column 11 additionally reporting the ratio of DAG paths to priced routes as a measure of path recombination. Finally, Columns 13–15 report DW-to-AF ratios for the number of variables, pricing iterations, and total CPU time.

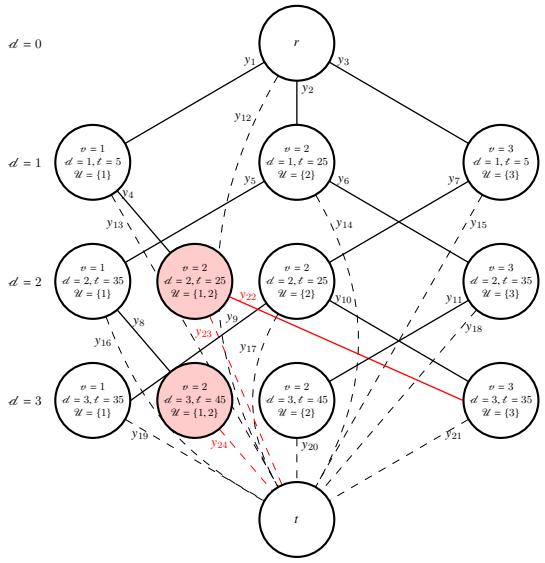
C.2 Dantzig-Wolfe and Arc-Flow Performance With Subset-Row Cuts

Table 7 reports the same information as Table 6 when subset-row cuts (SRCs) are introduced. Columns 3 and 9 additionally report the number of cuts added to DW and AF, respectively. We report the arithmetic mean in these columns since some values are zero.

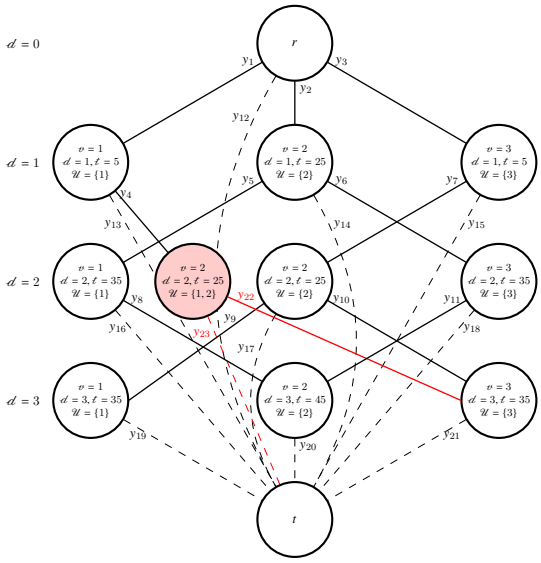
C.3 Dantzig-Wolfe and Arc-Flow Performance Under Explicit and Implicit DAG Representations

We compare DW and AF under explicit and implicit DAG representations. In the explicit approach, the unfolded *ng*-relaxation state space is stored in memory, and the exact pricing problem is solved using a standard shortest path algorithm. We implement a *beam search*-based heuristic pricing procedure that explores only the three best extensions from each node. In contrast, the implicit approach relies on label extensions and dominance checks (see §5.1). For this experiment, we consider $\Delta = 0$ and only type 1 instances (i.e., with tight time windows).

Table 8 reports the CPU times (in seconds) of DW and AF under the two approaches. For DW, Columns 2–4 report the CPU times with the explicit DAG, the implicit DAG, and their ratio for the 25-customer instances. Columns 5–7 provide the corresponding results for AF. Finally, Columns 9–14 present the same information as Columns 2–7 for the 50-customer instances. For DW, the implicit approach is about three 3.7 and 3.0 times faster than the explicit approach on



(a) Decremental state-space relaxation.



(b) Column elimination.

Figure 6: Strengthening subproblem relaxations.

	Dantzig-Wolfe					Arc-Flow						DW-to-AF Ratios		
Instances	LB	RMP (s)	PP (s)	Time (s)	Solved	LB	RMP (s)	PP (s)	Time (s)	Recomb.	Solved	Variables	Iterations	Time
						$\Delta = 6$								
R1-25	457.24	0.03	0.28	0.34	12/12	457.24	0.08	0.31	0.46	1.02	12/12	0.27	0.88	0.74
C1-25	190.58	0.06	0.79	0.91	9/9	190.58	0.15	0.52	0.80	1.41	9/9	0.30	1.46	1.13
RC1-25	340.78	0.06	0.58	0.70	8/8	340.78	0.12	0.56	0.80	1.14	8/8	0.27	1.13	0.88
R2-25	376.73	0.07	1.26	1.38	11/11	376.73	0.28	1.20	1.64	1.12	11/11	0.21	1.05	0.85
C2-25	214.30	0.15	2.20	2.48	8/8	214.30	0.57	1.59	2.34	3.39	8/8	0.25	1.62	1.06
RC2-25	309.91	0.12	3.96	4.29	8/8	309.91	0.62	3.15	4.15	1.79	8/8	0.25	1.29	1.03
Geo. Mean	**300.95**	**0.07**	**1.06**	**1.21**	**56/56**	**300.95**	**0.23**	**0.90**	**1.29**	**1.49**	**56/56**	**0.26**	**1.21**	**0.94**
R1-50	747.96	0.14	3.30	3.61	12/12	747.96	0.64	3.62	4.49	1.04	12/12	0.22	0.94	0.80
C1-50	361.69	0.18	5.82	6.14	9/9	361.69	0.73	4.26	5.24	1.53	9/9	0.25	1.33	1.17
RC1-50	658.20	0.21	4.42	4.78	8/8	658.20	0.78	4.48	5.61	1.20	8/8	0.21	1.03	0.85
R2-50	599.64	0.61	69.12	70.51	11/11	599.64	7.41	61.57	71.43	1.30	11/11	0.16	1.02	0.99
C2-50	357.48	7.75	162.85	175.31	8/8	357.48	17.02	33.12	52.73	19.64	8/8	0.39	4.67	3.32
RC2-50	571.84	0.83	97.60	99.51	7/8	571.84	6.82	64.01	73.55	2.84	8/8	0.23	1.49	1.35
Geo. Mean	**528.65**	**0.52**	**21.30**	**22.52**	**55/56**	**528.65**	**2.61**	**14.43**	**18.22**	**2.27**	**56/56**	**0.23**	**1.45**	**1.23**
						$\Delta = 12$								
R1-25	457.24	0.03	0.27	0.34	12/12	457.24	0.07	0.32	0.46	1.01	12/12	0.26	0.85	0.75
C1-25	190.58	0.06	0.81	0.93	9/9	190.58	0.13	0.65	0.91	1.10	9/9	0.26	1.29	1.02
RC1-25	340.86	0.04	0.52	0.59	8/8	340.86	0.08	0.51	0.72	1.10	8/8	0.25	1.05	0.83
R2-25	377.79	0.06	1.52	1.65	11/11	377.80	0.25	1.42	1.83	1.08	11/11	0.21	0.99	0.90
C2-25	214.30	0.16	2.22	2.47	8/8	214.30	0.82	1.93	3.03	1.89	8/8	0.20	1.32	0.82
RC2-25	318.11	0.06	2.40	2.59	8/8	318.11	0.39	2.79	3.44	1.34	8/8	0.22	0.98	0.75
Geo. Mean	**302.42**	**0.06**	**0.99**	**1.12**	**56/56**	**302.42**	**0.20**	**0.97**	**1.34**	**1.22**	**56/56**	**0.23**	**1.07**	**0.84**
R1-50	748.50	0.17	3.46	3.78	12/12	748.50	0.70	3.69	4.74	1.02	12/12	0.22	0.93	0.80
C1-50	361.69	0.22	6.27	6.74	9/9	361.69	1.03	5.47	6.90	1.10	9/9	0.20	1.09	0.98
RC1-50	659.34	0.19	3.69	4.02	8/8	659.34	0.61	4.37	5.23	1.10	8/8	0.20	0.98	0.77
R2-50	617.60	0.45	48.06	49.00	10/11	617.61	6.24	51.40	60.26	1.11	11/11	0.16	0.98	0.81
C2-50	357.48	7.17	146.79	158.18	8/8	357.48	32.37	42.19	81.62	5.78	8/8	0.27	3.47	1.94
RC2-50	607.18	0.32	22.49	23.23	6/8	607.18	2.76	27.72	31.45	1.66	6/8	0.19	0.96	0.74
Geo. Mean	**536.81**	**0.44**	**15.27**	**16.25**	**53/56**	**536.81**	**2.50**	**13.21**	**17.26**	**1.54**	**54/56**	**0.20**	**1.22**	**0.94**

Table 6: Summary of DW and AF performance for both $\Delta = 6$ and $\Delta = 12$.

	Dantzig-Wolfe						Arc-Flow							DW-to-AF Ratios		
Instances	LB	Cuts	RMP (s)	PP (s)	Time (s)	Solved	LB	Cuts	RMP (s)	PP (s)	Time (s)	Recomb.	Solved	Variables	Iterations	Time
							$\Delta = 6$									
R1-25	459.22	12.3	0.04	0.38	0.47	12/12	459.22	12.3	0.15	0.43	0.69	1.02	12/12	0.26	0.86	0.68
C1-25	190.58	0.0	0.05	0.81	0.91	9/9	190.58	0.0	0.14	0.60	0.82	1.41	9/9	0.30	1.46	1.11
RC1-25	346.26	9.1	0.06	0.78	0.92	8/8	346.26	9.3	0.21	0.65	1.03	1.14	8/8	0.27	1.18	0.89
R2-25	380.62	24.8	0.10	3.90	4.21	11/11	380.62	28.1	1.21	6.28	8.05	1.11	11/11	0.20	0.97	0.52
C2-25	214.45	1.3	0.15	2.59	2.86	8/8	214.45	1.3	0.62	1.88	2.75	3.00	8/8	0.25	1.64	1.04
RC2-25	330.37	13.0	0.14	3.20	3.47	6/8	330.37	12.5	0.58	2.77	3.62	1.54	6/8	0.30	1.43	0.96
Geo. Mean	**305.76**	**10.1**	**0.08**	**1.41**	**1.59**	**54/56**	**305.76**	**10.6**	**0.35**	**1.33**	**1.90**	**1.43**	**54/56**	**0.26**	**1.22**	**0.84**
R1-50	754.36	48.5	0.37	7.30	8.13	12/12	754.34	48.9	2.81	8.67	12.19	1.02	12/12	0.22	0.87	0.67
C1-50	361.69	0.0	0.23	5.73	6.18	9/9	361.69	0.0	0.66	4.32	5.25	1.53	9/9	0.25	1.33	1.18
RC1-50	712.11	84.5	0.66	28.72	30.43	8/8	712.41	78.5	6.37	31.28	41.06	1.16	8/8	0.20	0.91	0.74
R2-50	645.32	62.5	0.94	121.85	124.01	8/11	645.32	65.0	33.90	223.42	284.78	1.08	8/11	0.14	0.88	0.44
C2-50	357.48	0.0	7.80	163.05	175.59	8/8	357.48	0.0	16.95	33.13	52.67	19.64	8/8	0.39	4.67	3.33
RC2-50	617.39	34.4	0.90	80.04	81.70	5/8	617.39	35.8	15.15	52.17	68.56	2.45	5/8	0.24	1.34	1.19
Geo. Mean	**549.97**	**38.3**	**0.85**	**35.23**	**37.36**	**50/56**	**550.01**	**38.0**	**6.84**	**27.71**	**37.32**	**2.13**	**50/56**	**0.23**	**1.34**	**1.00**
							$\Delta = 12$									
R1-25	459.22	12.3	0.05	0.41	0.49	12/12	459.22	13.4	0.12	0.41	0.63	1.01	12/12	0.25	0.87	0.78
C1-25	190.58	0.0	0.05	0.83	0.93	9/9	190.58	0.0	0.15	0.64	0.90	1.10	9/9	0.26	1.29	1.02
RC1-25	346.26	7.9	0.05	0.61	0.71	8/8	346.26	6.9	0.13	0.59	0.83	1.10	8/8	0.25	1.05	0.85
R2-25	380.62	24.4	0.14	4.69	4.94	11/11	380.62	25.1	1.00	6.52	8.25	1.08	11/11	0.18	0.93	0.60
C2-25	214.45	1.3	0.17	2.68	3.06	8/8	214.45	1.3	0.79	2.37	3.45	1.76	8/8	0.20	1.35	0.89
RC2-25	318.11	0.0	0.06	2.47	2.67	8/8	318.11	0.0	0.36	2.75	3.37	1.34	8/8	0.22	0.98	0.79
Geo. Mean	**303.84**	**7.3**	**0.08**	**1.36**	**1.53**	**56/56**	**303.84**	**7.8**	**0.30**	**1.37**	**1.89**	**1.21**	**56/56**	**0.22**	**1.06**	**0.81**
R1-50	754.30	48.3	0.35	7.84	8.59	12/12	754.33	46.9	2.62	9.20	12.59	1.01	12/12	0.21	0.89	0.68
C1-50	361.69	0.0	0.20	6.27	6.69	9/9	361.69	0.0	1.04	5.40	6.86	1.10	9/9	0.20	1.09	0.98
RC1-50	712.45	81.9	0.67	18.59	19.90	8/8	712.55	76.8	4.56	18.49	25.59	1.10	8/8	0.21	0.96	0.78
R2-50	645.75	55.0	0.83	106.96	109.10	8/11	645.77	55.0	21.50	152.97	191.53	1.07	8/11	0.15	0.92	0.57
C2-50	357.48	0.0	7.00	144.65	155.66	8/8	357.48	0.0	32.45	42.51	81.79	5.78	8/8	0.27	3.47	1.90
RC2-50	607.18	0.0	0.32	22.87	23.62	6/8	607.18	0.0	2.91	27.69	31.74	1.66	6/8	0.19	0.96	0.74
Geo. Mean	**548.54**	**30.9**	**0.67**	**26.20**	**27.77**	**51/56**	**548.56**	**29.8**	**5.42**	**23.43**	**32.12**	**1.52**	**51/56**	**0.20**	**1.19**	**0.86**

Table 7: Summary of DW and AF performance with SRCs for both $\Delta = 6$ and $\Delta = 12$.

the 25- and 50-customer instances, respectively. For AF, these factors are 3.6 and 3.8, indicating that the implicit approach is generally much faster for both reformulations, although the explicit representation remains competitive in some cases. These results provide context for the discussion in §2.4 and §6.5, but they are not generalizable, as performance highly depends on implementation. For example, Karahalios (2025) found the explicit DAG approach to be competitive with state-of-the-art implicit methods when the *ng*-route relaxation was combined with time bucketing, relaxed capacity constraints, and a specialized minimum-update successive shortest path algorithm, among other enhancements.

C.4 Dantzig-Wolfe and Arc-Flow Performance with Iterative Subproblem Strengthening

Table 9 reports the same information as Table 4, but aggregated by instance group and with the number of solved instances.

	Dantzig-Wolfe			Arc-Flow				Dantzig-Wolfe			Arc-Flow		
Instance	Explicit (s)	Implicit (s)	Ratio	Explicit (s)	Implicit (s)	Ratio	Instance	Explicit (s)	Implicit (s)	Ratio	Explicit (s)	Implicit (s)	Ratio
R101-25	0.07	0.07	1.00	0.11	0.14	0.79	R101-50	0.17	0.31	0.55	0.29	0.44	0.66
R102-25	0.88	0.17	5.18	0.99	0.28	3.54	R102-50	26.55	1.23	21.59	30.43	1.48	20.56
R103-25	3.38	0.42	8.05	3.29	0.68	4.84	R103-50	152.80	5.96	25.64	118.98	7.25	16.41
R104-25	12.85	0.65	19.77	15.11	0.91	16.60	R104-50	474.93	43.26	10.98	595.52	43.94	13.55
R105-25	0.11	0.14	0.79	0.18	0.22	0.82	R105-50	0.64	0.59	1.09	0.89	0.94	0.95
R106-25	1.79	0.29	6.17	2.88	0.45	6.40	R106-50	43.05	3.06	14.07	50.79	3.86	13.16
R107-25	7.04	0.54	13.04	6.30	0.80	7.88	R107-50	197.75	10.51	18.82	185.51	12.19	15.22
R108-25	20.13	0.86	23.41	19.68	1.21	16.26	R108-50	474.09	56.35	8.41	590.31	57.40	10.28
R109-25	0.42	0.29	1.45	0.57	0.51	1.12	R109-50	8.00	2.65	3.02	7.46	3.68	2.03
R110-25	1.76	0.53	3.32	1.64	0.79	2.08	R110-50	33.56	7.72	4.35	37.79	8.69	4.35
R111-25	4.61	0.57	8.09	4.25	0.72	5.90	R111-50	105.99	12.43	8.53	102.63	13.42	7.65
R112-25	3.93	1.00	3.93	4.98	1.45	3.43	R112-50	98.36	30.57	3.22	99.97	31.73	3.15
C101-25	0.34	0.29	1.17	0.25	0.33	0.76	C101-50	2.45	1.39	1.76	2.11	1.05	2.01
C102-25	21.08	1.21	17.42	20.15	0.94	21.44	C102-50	100.15	11.71	8.55	106.14	4.57	23.23
C103-25	78.23	4.10	19.08	79.60	1.59	50.06	C103-50	564.86	63.87	8.84	408.89	19.97	20.48
C104-25	129.05	5.62	22.96	101.45	2.57	39.48	C104-50	962.01	169.35	5.68	1,205.83	91.76	13.14
C105-25	0.84	0.34	2.47	0.74	0.35	2.11	C105-50	11.48	1.69	6.79	7.48	1.34	5.58
C106-25	0.38	0.28	1.36	0.34	0.33	1.03	C106-50	6.04	1.45	4.17	6.20	1.07	5.79
C107-25	3.17	0.50	6.34	2.29	0.40	5.73	C107-50	18.50	2.11	8.77	22.67	1.50	15.11
C108-25	6.34	1.06	5.98	9.27	0.97	9.56	C108-50	45.03	13.09	3.44	78.93	5.92	13.33
C109-25	16.51	1.48	11.16	20.21	1.40	14.44	C109-50	127.70	25.23	5.06	108.40	13.95	7.77
RC101-25	0.14	0.26	0.54	0.20	0.33	0.61	RC101-50	0.39	0.84	0.46	0.56	1.00	0.56
RC102-25	1.08	0.70	1.54	1.21	0.72	1.68	RC102-50	3.62	3.76	0.96	4.15	3.19	1.30
RC103-25	2.77	1.43	1.94	2.68	1.25	2.14	RC103-50	10.16	14.62	0.70	12.95	10.59	1.22
RC104-25	6.57	2.09	3.14	5.66	1.52	3.72	RC104-50	34.61	71.94	0.48	34.34	55.91	0.61
RC105-25	0.59	0.51	1.16	0.60	0.50	1.20	RC105-50	3.07	2.56	1.20	3.50	2.62	1.34
RC106-25	0.63	0.67	0.94	0.95	0.94	1.01	RC106-50	2.48	4.22	0.59	3.43	4.49	0.76
RC107-25	2.56	1.64	1.56	2.81	1.70	1.65	RC107-50	8.18	17.77	0.46	9.17	14.54	0.63
RC108-25	4.16	3.37	1.23	4.90	3.24	1.51	RC108-50	16.56	46.25	0.36	15.67	47.54	0.33
Geo. Mean	**2.38**	**0.65**	**3.68**	**2.58**	**0.73**	**3.56**	**Geo. Mean**	**22.26**	**7.20**	**3.09**	**24.33**	**6.33**	**3.85**

Table 8: CPU time (in seconds) of DW and AF with an explicit and implicit DAG representation ($\Delta = 0$).

	DW-DSSR				AF-DSSR					DW-CE				AF-CE				
Instances	Strength. Iter.	Elim. Routes	Time (s)	Solved	Strength. Iter.	Elim. Arcs	Time (s)	Change in DAG	Solved	Strength. Iter.	Elim. Routes	Time (s)	Solved	Strength. Iter.	Elim. Arcs	Time (s)	Change in DAG	Solved
R1-25	2.5	250.1	7.35	12/12	2.7	903.3	8.49	893,894.3	12/12	7.4	85.4	3.99	12/12	7.4	21.1	4.88	1,330.9	12/12
C1-25	1.0	398.5	11.78	9/9	1.0	1,576.3	10.58	4,864,992.4	9/9	4.8	81.3	8.34	9/9	4.7	10.9	7.42	377.0	9/9
RC1-25	2.8	533.9	8.23	8/8	2.8	1,669.3	9.03	1,002,700.6	8/8	11.2	340.8	4.81	8/8	10.8	52.0	4.97	1,786.3	8/8
Geo. Mean	**2.1**	**366.3**	**8.78**	**29/29**	**2.2**	**1,281.7**	**9.25**	**1,221,596.3**	**29/29**	**7.8**	**139.7**	**5.28**	**29/29**	**7.7**	**25.2**	**5.59**	**970.8**	**29/29**
R1-50	4.0	346.8	28.06	6/12	2.7	790.0	27.77	3,959,035.3	6/12	8.3	130.2	13.54	8/12	8.1	33.3	17.37	3,846.8	8/12
C1-50	1.0	846.2	62.72	8/9	1.0	2,743.5	70.31	11,766,968.5	8/9	5.4	112.5	49.51	9/9	4.0	12.4	44.01	593.3	9/9
RC1-50	5.7	1,796.9	108.27	8/8	5.7	6,977.5	129.31	7,119,176.3	8/8	172.7	1,427.6	83.02	8/8	170.6	620.0	83.05	14,928.8	8/8
Geo. Mean	**3.4**	**986.6**	**61.42**	**22/29**	**3.1**	**3,205.1**	**68.11**	**4,944,384.5**	**22/29**	**34.1**	**415.7**	**41.96**	**25/29**	**31.1**	**112.2**	**43.03**	**4,878.8**	**25/29**

Table 9: Summary of DW and AF performance with DSSR and CE.