http://pubsonline.informs.org/journal/ijoc

INFORMS JOURNAL ON COMPUTING
Vol. 00, No. 0, Xxxxx 0000, pp. 000–000
ISSN 0899-1499, EISSN 1526-5528

Submitted to *INFORMS Journal on Computing*

Computing Bi-Path Multi-Commodity Flows with Constraint Programming-based Branch-and-Price-and-Cut

Jiachen Zhang
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada, jasonzjc@mie.utoronto.ca

Youcef Magnouche, Pierre Bauguion, Sebastien Martin
Huawei Technologies, Paris Research Center, Boulogne-Billancourt, France, {firstname.lastname}@huawei.com

J. Christopher Beck
Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario M5S 3G8, Canada, jcb@mie.utoronto.ca

Abstract. We propose a constraint programming (CP) based branch-and-price-and-cut framework to exactly solve bi-path MCF: a multi-commodity flow (MCF) problem with two paths for each demand. The goal is to route demands in a capacitated network under the minimum cost. The two paths must have disjoint arcs and the delays accumulated along the two paths must be within a small deviation of each other. CP is used at multiple points in this framework: for solving pricing problems, for cut generation, and for primal and branching node heuristics. These modules use a CP solver designed for network routing problems and can be adapted to other combinatorial optimization problems. We also develop a novel, complete two-level branching scheme. On a set of diverse bi-path MCF instances, experimental results show that our algorithm significantly outperforms monolithic CP and mixed integer linear programming models and demonstrate the efficiency and flexibility brought by the tailored integration of linear programming and CP methodologies.

Key words: constraint programming, branch-and-price-and-cut, multi-commodity flow, network routing

1. Introduction

Telecommunication network size has significantly increased in recent years (Tang et al. 2015), with new applications leading to the next generation of traffic routing protocols and new requirements such as small end-to-end delay and resilience to link faults (Angilella et al. 2022). Larger networks and new requirements raise challenges for routing optimization and there have been several attempts to develop efficient optimization algorithms that handle all technical constraints through approaches including linear programming (LP)

(Bhatia et al. 2015), non-linear programming (D'Ambrosio et al. 2015), meta-heuristics (Risso et al. 2018), and machine learning (Tang et al. 2021).

In mathematical programming, several constraints appeared to be hard to model, requiring an exponential number of constraints, big-M parameters, and intermediate variables (Fortz et al. 2017). Constraint programming (CP), by contrast, allows greater flexibility and extensibility in the types of constraints (Laborie et al. 2018), resulting in the development of many global constraints for complex sub-problems in industrial contexts. However, CP performance depends substantially on local reasoning arising from inference algorithms embedded in each constraint (Rossi et al. 2006). Previous studies have shown that it is possible to benefit from a more global perspective through the combination of LP and CP: addressing optimality via LP relaxations and ensuring feasibility via CP inference (Focacci et al. 2002, Beck and Refalo 2003, Milano and Wallace 2010, Laborie and Rogerie 2016).

CP-based column generation (Junker et al. 1999, Rousseau et al. 2004) has been proposed as a different approach to hybridization of CP and LP. We extend this line of work by proposing CP-based branch-and-price-and-cut (BPC) (Desaulniers 2010) for a variant of the multi-commodity flow (MCF) problems (Barnhart et al. 2000). CP is embedded in all the functional components of the BPC framework: column generation, cut generation, primal heuristics, and branching node heuristics.

In bi-path MCF, multiple commodities need to be routed in a network. Each commodity requires unsplittable primary and secondary paths that are arc-disjoint (i.e., the two paths cannot share any arcs) and delay-similar (i.e., the difference in delay accumulated over each arc in the two paths must be less than a threshold value). Each commodity has a bandwidth to reflect its demand, while each arc in the network has a total bandwidth capacity limit. The goal is to minimize the total costs of bi-path arc usage by the commodities. This MCF variant arises mainly in the deterministic Internet Protocol (IP) network, with the main application being 1+1 protection (Finn 2018), where the two paths are used as working and protection channels, respectively. The bounded delay difference guarantees that the user experience during the failure of primary paths does not suffer too much. Another application is load balancing (Angilella et al. 2022), where the delay difference is related to the buffer sizes at the end of the routes and the bounded delay difference assures relatively small buffer sizes to reduce the fixed hardware costs. Bi-path MCF is also relevant to the primary/secondary cache placements for in-memory data grids (Sebbah et al. 2016).

Main contributions. Our first contribution is a CP-based branch-and-price-and-cut approach for the bi-path MCF problem. Differing from existing CP-based BPC algorithms where CP is used only for pricing problems (Junker et al. 1999, Rousseau et al. 2004, Hashemi Doulabi et al. 2016), the proposed approach also uses CP for primal heuristics, branching node heuristics, and cut generation. Our second and third contributions are the complexity proof and a two-level branching scheme for the bi-path MCF problem. Our fourth contribution is a thorough numerical evaluation on the bi-path MCF problem, showing the advantage of the CP-based approach over compact mixed-integer linear programming (MILP) and CP models.

Outline of the paper. Related research is reviewed in Section 2. In Section 3, the problem is defined with proof of its complexity, followed by two MILP formulations. In Section 4, a specialized CP solver for network routing problems is introduced, followed by an overview of the CP-based branch-and-price-and-cut framework in Section 5. In Sections 6 and 7, the CP-based column generation and cut generation are detailed. The CP-based primal heuristic is presented in Section 8. The CP-based branching node heuristic and a two-level branching scheme are described in Section 9. Finally, a thorough experimental evaluation is conducted in Section 10, followed by discussions and conclusions in Sections 11 and 12.

2. Literature Review

In this section, we summarize the bi-path MCF problem and CP-based approaches for network routing problems, followed by the presentation of the branch-and-price-and-cut approach.

2.1. The Bi-Path Multi-Commodity Flow Problem

With the recent trend of traffic engineering methodology where routes between nodes are explicitly specified (Foteinos et al. 2014), network routing problems are frequently abstracted as multi-commodity flow (MCF) problems (Assad 1978). In particular, MCF appears when multiple commodities (cargo, packets, or demands) need to be routed between specific node pairs without violating the capacity constraints associated with the arcs in the network. While the splittable MCF can be solved in polynomial time as a linear program, the unsplittable MCF is NP-hard (Karp 1975) and is often solved as a mixed-integer linear program with advanced techniques. MCF is a core problem in various application contexts, including communication (Minoux 2001), transportation (Caimi et al. 2011), and energy management (Shi et al. 2016).

A few works study the multi-commodity flow problem with primary (default) and secondary (backup) paths. Xia and Simonis (2005) studied an MCF variant with arc-disjoint primary and secondary paths, both under capacity constraints. However, they complicated the capacity constraints with nonlinearity and did not consider the delay (the time spent on traversing a link). Our mathematical model, by contrast, guarantees that the two paths meet a delay difference requirement and does not need nonlinear constraints or objectives.

A recent work studies bi-path MCF in the context of a large-scale deterministic IP network (Angilella et al. 2022). For network resilience, the work explicitly models arc-disjointness but not bounded difference of delay. The method for solving the abstracted optimization problem is price-and-branch heuristic, i.e., column generation at the root node, followed by an integer program with all the generated columns to compute a heuristic solution. In contrast, our approach is exact, with CP-based techniques to strengthen the relaxation and accelerate the solution process.

2.2. CP-based Approaches for Network Routing

The routing problem in networks has been studied as a constraint satisfaction problem (Frei and Faltings 1999) and a constraint optimization problem, with both node-based CP models (Ros et al. 2001) and arc-based CP models (Sakkout and Wallace 2000). A complete CP model using set variables for the paths,

continuous variables for the flows, and integer variables for the edge weights is proposed for a weight-setting MCF variant with an interior gateway protocol optimization metric (Ajili et al. 2005).

LP relaxations have been used to augment CP in solving routing problems for the open shortest path first routing protocol (Petterson et al. 2007). Lagrangian relaxation, with relaxed capacity or path constraints (Ouaja and Richards 2005), and CP were integrated to route traffic demands in data networks. Similarly, Lagrangian decomposition and CP were combined to solve the multi-cast network design problem, where Lagrangian decomposition provides a bound and the integration allows constraint propagation of all sub-structures at every dual iteration (Cronholm and Ajili 2004).

To solve traffic engineering problems in segment routing, a hybrid CP framework was proposed with novel route variables that can be considered as a relaxation of usual representations (Hartert et al. 2015). The framework allows the users to customize an efficient core system for specific routing problems. Recently, there are also approaches for routing problems using hybrid techniques including CP (Kinable et al. 2020). In particular, CP is used within a logic-based Benders decomposition framework (Hooker and Ottosson 2003) and the integration exhibits the best performance for the time-triggered periodic communication problem in time-sensitive networks (Vlk et al. 2021).

2.3. Branch-and-Price-and-Cut Approaches

The branch-and-price-and-cut (BPC) algorithm was introduced in a seminal paper (Barnhart et al. 2000) to exactly solve the unsplittable multi-commodity flow problem. In BPC, column generation provides a linear relaxation, strengthened by cutting planes, while a branch-and-bound tree searches the integer space. The BPC framework, and the complete branching scheme based on divergence nodes, are highly regarded exact techniques for MCF problems (Salimifard and Bigharaz 2022). BPC is also widely used for exactly solving vehicle routing problems (Archetti et al. 2011, Costa et al. 2019), one-dimensional bin packing problems (Wei et al. 2020), and cutting stock problems (Alves and de Carvalho 2008).

Only a limited number of papers have proposed CP-based branch-and-price-and-cut (CPBPC) algorithms. Most of these works use CP to solve pricing problems, i.e., generate columns. For example, this type of column generation algorithm has been applied to an airline crew assignment problem (Junker et al. 1999), vehicle routing problems with time windows (Rousseau et al. 2004, Lam et al. 2022), and an operating room planning and scheduling problem (Hashemi Doulabi et al. 2016). Although under a different name, a CP-based branch-and-price-and-check algorithm for a vehicle routing problem with location congestion follows the same paradigm as CPBPC by using CP to check the feasibility of the location resource constraints and to add combinatorial no-good cuts to the master problem in case of constraint violation (Lam and Hentenryck 2016).

3. Problem Definition and Formulations

In this section, we investigate a complex MCF variant where a primary and a secondary path must be allocated to each commodity. The two paths need to be arc-disjoint, i.e., no arc is used by both paths. Due to the time for a demand to traverse an arc, the two paths each have a total delay. We require the difference between the two path-delays to be no greater than a given threshold. Each commodity has a bandwidth to reflect its resource consumption in the network, while each arc in the network has a capacity for its bandwidth usage. The objective is to minimize the total costs of the primary paths and the secondary paths. We call this MCF variant bi-path MCF (BP-MCF).

The BP-MCF is defined on a graph $G = (V, A)$, where $V = \{v_1, ..., v_{|V|}\}$ represents the set of vertices and $A = \{a_1, ..., a_{|A|}\}$ represents the set of arcs. For each vertex v, $\delta^{\text{out}}(v)$ and $\delta^{\text{in}}(v)$ represent the set of outgoing and incoming arcs of v, respectively. For each arc a, we use a^- and a^+ to denote the origin and destination nodes of the arc, respectively. There is a set of commodities (demands) $K = \{k_1, ..., k_{|K|}\}$ that need to be routed as elementary paths in the graph. A path is called elementary if no vertex is repeated in the path. Each commodity k has an origin s_k, a destination t_k, and a bandwidth b_k. Each arc a has a bandwidth capacity c_a and a delay d_a for any commodity whose path contains a. The delay of a path is the sum of the delays of its arcs. The difference in the delays of the two paths for commodity k is required to be less than or equal to Δ_k. In addition, α_a and β_a are the arc cost coefficients for primary and secondary paths, respectively. Namely, if the primary (secondary) path of a commodity k contains arc a, the induced cost is $\alpha_a b_k$ ($\beta_a b_k$).

As capacities c_a and bandwidths b_k are often treated as integers in telecommunication applications, they are assumed to be non-negative integers in BP-MCF. By contrast, delays d_a, delay difference limits Δ_k, and cost coefficients α_a and β_a are assumed to be non-negative continuous. An example of BP-MCF is shown in Figure 1. The notation is summarized in Table 1.

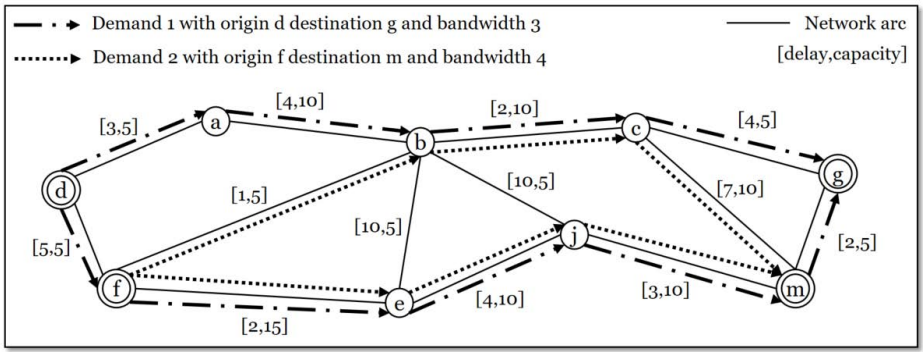

Figure 1 **An example of BP-MCF with two demands. The upper bound of delay difference is 2. The solution shown in the example is feasible as (1) the two paths for each demand are arc-disjoint; (2) the delay difference of the two paths for each demand is 1; and (3) the arc capacities are not exceeded.**

Table 1 Summary of symbols and definitions.

Symbol	Definition	Symbol	Definition	Symbol	Definition
$G=(V,A)$	graph (network)	a^-	origin of a	s_k	origin of k
$V=\{v\}$	set of vertices	a^+	destination of a	t_k	destination of k
$A=\{a\}$	set of arcs	$c_a \in \mathbb{Z}_{\geq 0}$	capacity of a	$b_k \in \mathbb{Z}_{\geq 0}$	bandwidth of k
$K=\{k\}$	set of demands	$d_a \in \mathbb{R}_{\geq 0}$	delay of a	μ_k	dual of (3b)
$\delta^{in}(v)$	incoming arcs of v	$\alpha_a \in \mathbb{R}_{\geq 0}$	primary cost of a	π_a	dual of (3c)
$\delta^{out}(v)$	outgoing arcs of v	$\beta_a \in \mathbb{R}_{\geq 0}$	secondary cost of a	η_a	dual of (21)
$\mathscr{C}=\{C\}$	set of covers	rv	route variable		
$\Delta_k \in \mathbb{R}_{\geq 0}$	bound of delay difference of k				
x_a^k/$\overline{x}_a^k$	=1 if arc a is used for the primary/secondary path of k, =0 otherwise				
$\hat{x}_a^k = x_a^k + \overline{x}_a^k$	=1 if arc a is used for demand k, =0 otherwise				
$z_{p,\overline{p}}^k$	=1 if primary and secondary paths p and $\overline{p}$ are used for k, =0 otherwise				
$w_{p,\overline{p}}^k$	total cost of primary and secondary paths p and $\overline{p}$ of demand k				
$P_{s,t}$	set of elementary paths from s to t				
K_a	set of demands using arc a				
A^s/A^c/A^o	set of set/closed/open arcs				
D_i^k	accumulated delay of column i for demand k				
θ_k/l_k/u_k	delay threshold/lower-bound/upper-bound for demand k				

PROPOSITION 1. *BP-MCF is NP-hard.*

Proof. In fact, the BP-MCF is NP-hard with only one commodity. We prove this NP-hardness via a reduction from the resource-constrained shortest path problem (RCSPP), which is NP-hard (Handler and Zang 1980). Specifically, for the single commodity with a delay-difference threshold Δ, we add an additional arc from the origin to the destination of the commodity, with a capacity greater than the bandwidth of the commodity, zero cost, and a delay of D. Then, with either the primary or the secondary path assigned to the additional arc, the rest of the problem is to determine the other path with its delay in the range of $[D-\Delta, D+\Delta]$ while minimizing the cost. This problem is exactly the RCSPP. Thus, BP-MCF is NP-hard. □

3.1. Mathematical Programming Formulations

We propose two mathematical programming formulations for BP-MCF, where the first is compact and the second is typically solved with column generation.

3.1.1. Compact formulation. The compact mixed-integer linear programming (MILP) model with arc variables for commodities is inspired by an existing model in the literature (Xia and Simonis 2005). There are only binary decision variables $\mathbf{X}=\{x_a^k\}$ and $\overline{\mathbf{X}}=\{\overline{x}_a^k\}$, where $x_a^k=1$ if the primary path of commodity k uses arc a, and 0 otherwise; and $\overline{x}_a^k=1$ if the secondary path of commodity k uses arc a, and 0 otherwise. The MILP model is as follows:

$$\texttt{min} \sum_{k\in K} b_k \Big(\sum_{a\in A} \alpha_a x_a^k + \sum_{a\in A} \beta_a \overline{x}_a^k\Big) \tag{1a}$$

$$\sum_{a\in\delta^{\text{out}}(v)} x_a^k - \sum_{a\in\delta^{\text{in}}(v)} x_a^k = \begin{cases} 1 & \text{if } v = s_k, \\ -1 & \text{if } v = t_k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall v \in V, \forall k \in K, \tag{1b}$$

$$\sum_{a\in\delta^{\text{out}}(v)} \overline{x}_a^k - \sum_{a\in\delta^{\text{in}}(v)} \overline{x}_a^k = \begin{cases} 1 & \text{if } v = s_k, \\ -1 & \text{if } v = t_k, \\ 0 & \text{otherwise,} \end{cases} \qquad \forall v \in V, \forall k \in K, \tag{1c}$$

$$\sum_{k\in K} b_k(x_a^k + \overline{x}_a^k) \le c_a \qquad \forall a \in A, \tag{1d}$$

$$-\Delta_k \le \sum_{a\in A} d_a x_a^k - \sum_{a\in A} d_a \overline{x}_a^k \le \Delta_k \qquad \forall k \in K, \tag{1e}$$

$$x_a^k + \overline{x}_a^k \le 1 \qquad \forall a \in A, \forall k \in K, \tag{1f}$$

$$x_a^k, \overline{x}_a^k \in \{0,1\} \qquad \forall a \in A, \forall k \in K, \tag{1g}$$

$$\texttt{Cycle Elimination Constraints.} \tag{1h}$$

Term (1a) represents the total weighted cost of arc usage by the primary and secondary paths of each commodity. Constraints (1b) and (1c) state the path flow constraints. Constraints (1d) address the arc capacity for primary and secondary paths. Constraint (1e) ensures that the delay difference between the two paths of the same commodity is under the threshold. Constraint (1f) guarantees that the two paths are arc-disjoint.

We also need to lazily add constraints to eliminate cycles that may be found. For example, for a commodity k, given that the delay difference of the primary and secondary paths is limited, if a low-cost but high-delay primary path is selected, then the secondary path may include cycles in order to close the delay gap. Such solutions to the MILP model need to be eliminated to ensure elementary paths. Thus, if there exist cycles in the optimal solution of model (1a)-(1g), constraints (1h) are added lazily during the solving process, i.e., via callback of commercial MILP solvers. We use depth-first search to detect cycles in the current best solution. If a cycle in the primary path for commodity k is detected as S_k, a set of all arcs in the cycle, the cycle elimination constraint is the following inequality:

$$\sum_{a\in S_k} x_a^k \le |S_k| - 1. \tag{2}$$

An analogous constraint is added if the cycle is in the secondary path. All such constraints (if any) are lazily added to the compact model, which is solved again until no cycle is detected in the current best solution.

3.1.2. Bi-path formulation and column generation. We present a MILP formulation with bi-path decision variables $\mathbf{Z} = \{z_{p,\overline{p}}^k\}$, where $z_{p,\overline{p}}^k = 1$ if primary path p and secondary path $\overline{p}$ are used for commodity k, and 0 otherwise. For convenience, we use PP_k to represent the set of potential bi-paths for commodity k. Any bi-path in the set satisfies the corresponding path flow, arc-disjoint, and delay-difference

constraints. Thus, we can directly obtain the cost coefficient $w^k_{p,\overline{p}}$ associated with a bi-path $(p,\overline{p})$ of commodity k. The MILP formulation is as follows:

$$\texttt{min} \sum_{k\in K}\sum_{(p,\overline{p})\in PP_k} w^k_{p,\overline{p}} z^k_{p,\overline{p}} \tag{3a}$$

$$\sum_{(p,\overline{p})\in PP_k} z^k_{p,\overline{p}} = 1 \qquad \forall k\in K, \tag{3b}$$

$$\sum_{k\in K} b_k \sum_{(p,\overline{p})\in PP_k | a\in p \text{ or } a\in\overline{p}} z^k_{p,\overline{p}} \le c_a \qquad \forall a\in A, \tag{3c}$$

$$z^k_{p,\overline{p}} \in \{0,1\} \qquad \forall k\in K, \forall (p,\overline{p})\in PP_k. \tag{3d}$$

The bi-path MILP model requires an exponential number of decision variables, which is intractable when the graph size is large. Thus, a column generation approach inspired by Barnhart et al. (1998) and Barnhart et al. (2000) is used. In column generation, model (3a)-(3d) with a subset of bi-paths for each commodity is referred to as the (restricted) master problem. The bi-path variables of a commodity are generated by solving pricing problems, where arc-disjoint and delay-similar bi-paths are found. We define the dual value of constraint (3b) as $\{\mu_k\}$ and the dual value of constraint (3c) as $\{\pi_a\}$. By using binary decision variables $\{x_a\}$ ($\{\overline{x}_a\}$) to express whether the primary (secondary) path uses arc a, the compact formulation of the pricing problem for commodity k is presented as follows:

$$\texttt{min} \sum_{a\in A}(\alpha_a+\pi_a)b_k x_a + \sum_{a\in A}(\beta_a+\pi_a)b_k\overline{x}_a - \mu_k \tag{4a}$$

$$\sum_{a\in\delta^{\text{out}}(v)} x_a - \sum_{a\in\delta^{\text{in}}(v)} x_a = \begin{cases} 1 & \text{if } v=s_k,\\ -1 & \text{if } v=t_k,\\ 0 & \text{otherwise,}\end{cases} \qquad \forall v\in V, \tag{4b}$$

$$\sum_{a\in\delta^{\text{out}}(v)} \overline{x}_a - \sum_{a\in\delta^{\text{in}}(v)} \overline{x}_a = \begin{cases} 1 & \text{if } v=s_k,\\ -1 & \text{if } v=t_k,\\ 0 & \text{otherwise,}\end{cases} \qquad \forall v\in V, \tag{4c}$$

$$-\Delta_k \le \sum_{a\in A} d_a x_a - \sum_{a\in A} d_a\overline{x}_a \le \Delta_k, \tag{4d}$$

$$x_a + \overline{x}_a \le 1 \qquad \forall a\in A, \tag{4e}$$

$$x_a, \overline{x}_a \in \{0,1\} \qquad \forall a\in A, \tag{4f}$$

$$\texttt{Cycle Elimination Constraints.} \tag{4g}$$

The pricing problem does not have the arc capacity constraints, as they are enforced in the master problem. The explanations of the model (4a)-(4g) are omitted as they are similar to model (1a)-(1h).

PROPOSITION 2. *The pricing problem of BP-MCF is NP-hard.*

Proof. We have proved in Proposition 1 that the BP-MCF with only one commodity is NP-hard, which implies that the pricing problem is also NP-hard by the same reduction from the resource-constrained shortest path problem. □

We can develop a variety of approaches to generate bi-paths: dynamic programming, MILP, constraint programming (CP), and problem-specific algorithms. In this work, we use CP as it can deal with various constraints and is efficient, especially in failure detection.

4. A CP Solver for Multi-Commodity Flow Problems

In this section, we introduce a specialized CP solver designed for MCF problems.

4.1. Route Variable

As a type of graph variable (Dooms et al. 2005, Laborie 2009), a route variable for a commodity is defined as

$$rv := (s, t, b, G, A), \tag{5}$$

where s and t are the origin and destination of the commodity, and b represents the bandwidth, which we also refer to as the flow. G is the graph where the commodity flow problem is defined and A is the set of all arcs in G. An arc $a \in A$ has three possible statuses: set, closed, and open. If a is set, it is selected as a part of the elementary path of the commodity. If a is closed, it is removed from the commodity's graph. If a is neither set nor closed, then it is open: it may or may not be added to the elementary path. We use $A = A^s \cup A^c \cup A^o$ to distinguish the three sets of mutual exclusive arcs for a commodity.

The domain of a route variable contains all the elementary paths from s to t in the graph defined implicitly via arc status. The route variable manipulates the status of arcs in the graph to determine its value, i.e., the elementary path it is assigned to. Formally we write the domain of a route variable as $D(rv) = P_{s,t}$, where $P_{s,t}$ is the set of all elementary paths from s to t. The route variable maintains a partial path originating from the origin, which is used to prune arcs via inference. An example of the partial path and the statuses of arcs is shown in Fig. 2. We base our implementation on ideas in the literature on route variables (Le Pape et al. 2002, Hartert et al. 2015).

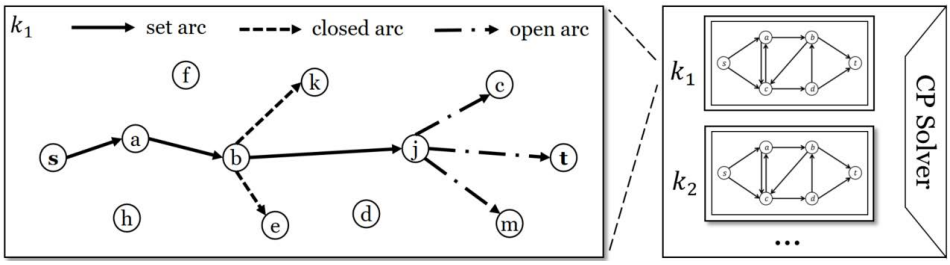

Figure 2 **Three types of arcs in the specialized CP solver: $\{sa, ab, bj\}$ are the set arcs that form the partial path for commodity k_1; $\{be, bk\}$ are the closed arcs as there already exists one outgoing arc from node b that is set; $\{jc, jm, jt\}$ are the open arcs as no decision has been made for the next set arc of the path.**

4.2. Constraints

Constraints in CP are implemented by filtering algorithms that preserve consistency among variable domains (Régin 1996). Thus, CP supports high-level, independent, and composable constraints. Here we present all the relevant CP constraints in this paper.

4.2.1. Arc-Disjoint Constraint. In the case where two paths need to be arc-disjoint, the constraint is defined as follows:

$$\texttt{ArcDisjoint}(rv_1, rv_2), \tag{6}$$

where $rv_1 = (s_1, t_1, b_1, G, A_1)$ and $rv_2 = (s_2, t_2, b_2, G, A_2)$ are two route variables. This constraint ensures that if an arc is set for a variable, the same arc is closed for the other variable, i.e.,

$$\forall a \in A_1^s \rightarrow a \in A_2^c \text{ and } \forall a \in A_2^s \rightarrow a \in A_1^c. \tag{7}$$

The implementation of the `ArcDisjoint` constraint is as simple as (7), which is triggered by the setting operation of an arc in a route variable.

4.2.2. Delay-Range Constraint. When arc delays are considered in MCF, the accumulated delay along a path can be constrained by the following constraint:

$$\texttt{DelayRange}(rv, l, u), \tag{8}$$

where $rv = (s, t, b, G, A)$ is a route variable. The l (u) is the lower bound (upper bound) of the accumulated delay along the path.

During the construction of the path, when considering an arc candidate a to set, we use shortest path algorithms (e.g., Dijkstra (1959)) to compute the smallest path delay $D^{'}$ with the arc a set according to the following formula:

$$D^{'}(a) = \sum_{i \in A^s} d_i + d_a + h(a), \tag{9}$$

where h_a is the smallest delay of the partial path originating from the end node of arc a. This formula is illustrated in Fig. 3. If $D^{'}(a) > u$, the upper bound of delay is exceeded after setting arc a, which implies that arc a can be closed.

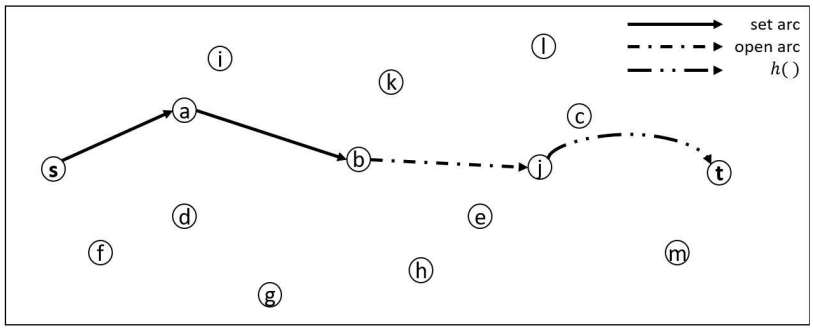

Figure 3 **Inference of the delay in the delay-range constraint: the arc bj is the candidate under consideration, $h(jt)$ is the shortest delay of the partial path from j to the destination t.**

4.2.3. Delay-Difference Constraint. Constraints that place an upper limit on the delay difference between two paths are defined as follows:

$$\texttt{DelayDiffUB}(rv_1, rv_2, \Delta), \tag{10}$$

where Δ represents the threshold of the delay difference between the two paths, i.e.,

$$|\sum_{a \in A_1^s} d_a - \sum_{a \in A_2^s} d_a| \leq \Delta. \tag{11}$$

In general, the inference of the delay difference is weak when neither the first nor the second path is fixed. However, with an appropriate search strategy, if one of the paths can be fixed first, then its delay D can be obtained. With the fixed path delay and the upper limit of the delay difference, a range of the other path delay $D^{'}$ is calculated as follows:

$$D - \Delta \leq D^{'} \leq D + \Delta. \tag{12}$$

The inference regarding this delay range is the same as the constraint `DelayRange`.

4.2.4. Arc Capacity Constraint. The arc capacity constraint of MCF is as follows:

$$\texttt{ArcCapacity}(\{rv_1, rv_2, ...rv_n\}, \{c_{a_1}, c_{a_2}, ..., c_{a_m}\}). \tag{13}$$

The `ArcCapacity` constraint involves a set of route variables and a set of arc capacities. This global constraint is equivalent to a group of capacity constraints for each arc, i.e.,

$$\sum_{a \in A_i^s, \forall i=1,...,n} b_i \leq c_a, \ \forall a = a_1, ..., a_m. \tag{14}$$

The constraint closes any arc of a variable of a commodity that exceeds the capacity of the arc, based on the accumulated bandwidth on an arc.

4.3. Implementation Details

Our CP solver does not represent constrained floating point variables: in fact, all constrained variables used in the models in this paper are Boolean (i.e., 0/1) variables. However, reasoning about floating point values is needed in some global constraints and with respect to costs as both arc delays and arc costs are assumed to be continuous parameters. In such cases, our solver uses C++ double variables to represent bounds on continuous values. Our implementation makes use of frequent epsilon comparisons to handle numerical errors that could occur during the propagation.

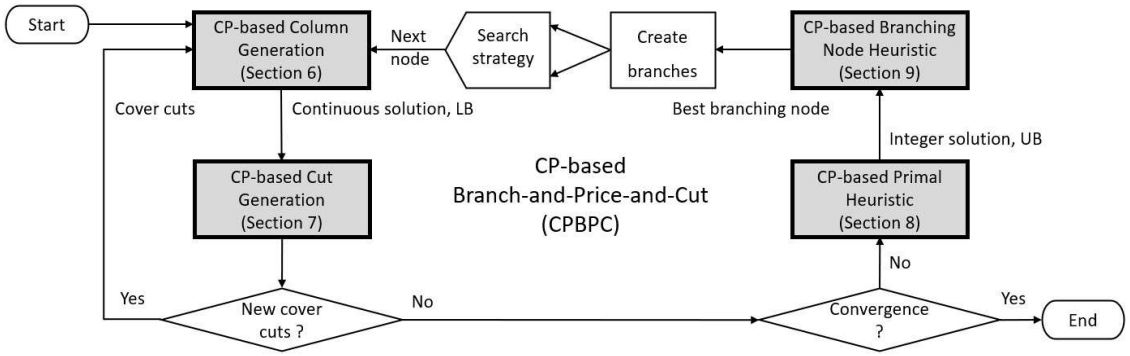

Figure 4 Flowchart of the CP-based branch-and-price-and-cut: blocks with bold frames are the CP modules.

5. CP-based Branch-and-Price-and-Cut

Branch-and-Price-and-Cut (BPC) is an exact framework for solving combinatorial optimization problems, where the relaxation is solved by column generation and strengthened by cutting planes, with the help of branching heuristics to search in the global solution space and close the gap between the primal and dual bounds. We incorporate the specialized CP solver into four algorithmic modules in a CP-based BPC (CPBPC) framework, as shown in Fig. 4.

The four modules are CP-based column generation (CPCG), CP-based cut generation (CPCT), CP-based primal heuristic (CPPH), and CP-based branching node heuristic (CPBNH). Our implementation iteratively solves CPCG and CPCT until no new cuts are generated, then it turns to CPPH with the latest CPCG

Algorithm 1: CPBPC

Input: ins - a MCF problem instance.

Output: lb/ub - global lower/upper bound; $\hat{\mathbf{X}}^*$ - the best integer solution

$lb \leftarrow -\infty$; $ub \leftarrow \infty$; $C \leftarrow \emptyset$; $\hat{\mathbf{X}}^* \leftarrow \emptyset$; $T \leftarrow \{n_0\}$;
while $T \neq \emptyset \wedge ub - lb > \delta$ **do**
  $n \leftarrow \texttt{PopTree}(T)$;
  repeat
    $(lb', \hat{\mathbf{X}}') \leftarrow \texttt{CPCG}(ins, n, C)$;
    if $\hat{\mathbf{X}}' = \emptyset$ or $lb' \geq ub$ **then** $n \leftarrow \texttt{PopTree}(T)$;
    else $C \leftarrow \texttt{CPCT}(ins, n, \hat{\mathbf{X}}')$;
  until C has no new cuts;
  $(ub', \hat{\mathbf{X}}) \leftarrow \texttt{CPPH}(ins, n, \hat{\mathbf{X}}')$; $lb \leftarrow \texttt{UpdateLB}(lb')$; $(ub, \hat{\mathbf{X}}^*) \leftarrow \texttt{UpdateUB}(ub', \hat{\mathbf{X}})$;
  if $ub - lb > \delta$ **then**
    $n_1, n_2 \leftarrow \texttt{CPBNH}(ins, n, \hat{\mathbf{X}}')$;
    $T \leftarrow \texttt{CreateBranch}(T, \{n_1, n_2\})$;
return $lb, ub, \hat{\mathbf{X}}^*$;

solution. Combined with a proposed two-level branching scheme evaluated by CPBNH, the entire procedure is summarized as Algorithm 1 in a minimization problem setting.

In the CPBPC algorithm, the initialization is conducted in line 1. T is the search tree and n_0 is the root node of the tree. Starting from line 2, a node of the search tree T is evaluated. From line 4 to line 8, the algorithm iteratively solves CPCG and CPCT until no new cuts are generated. In line 6, the current node is discarded if it induces an infeasible relaxation or the local lower bound is worse than the best upper bound. The best fractional solution along with the generated columns are then passed to CPPH to find an upper bound and an integer feasible solution in line 9. If the best upper bound is greater than the best lower bound by a threshold δ, the convergence criteria are not satisfied and thus the CPBNH is performed to generate two branches in line 11. The two branches are then explicitly added to the search tree T in line 12. Otherwise, if the algorithm converges, the best lower and upper bounds and the best integer feasible solutions are returned.

We detail the CP-based column generation in Section 6, the CP-based cut generation in Section 7, the CP-based primal heuristics in Section 8, and the CP-based branching node heuristics in Section 9.

6. CP-based Column Generation

In our column generation approach, CPCG, the CP solver introduced above is used for solving the pricing problems. The linear relaxation of the model (3a)-(3d) with a subset of all bi-paths is referred to as the (restricted) master problem. Specifically, constraint (3d) is replaced by the following constraint:

$$z^k_{p,\overline{p}} \geq 0 \qquad \forall k \in K, \forall (p,\overline{p}) \in PP_k. \tag{3d'}$$

In the rest of this paper, (3a)-(3d') represents the master problem in column generation.

6.1. CP Model for Bi-Path Generation

The pricing problem that needs to be solved for each commodity k is the following constraint satisfaction problem:

$$\texttt{ArcDisjoint}(rv_k, \overline{rv}_k), \tag{15a}$$

$$\texttt{DelayDiffUB}(rv_k, \overline{rv}_k, \Delta_k), \tag{15b}$$

$$\sum_{a \in A^s_k} (\alpha_a + \pi_a) b_k + \sum_{a \in \overline{A}^s_k} (\beta_a + \pi_a) b_k < \mu_k, \tag{15c}$$

$$rv_k = (s_k, t_k, b_k, G, A_k), \tag{15d}$$

$$\overline{rv}_k = (s_k, t_k, b_k, G, \overline{A}_k). \tag{15e}$$

The rv_k and $\overline{rv}_k$ are the route variables of the primary and secondary paths of commodity k. The A_k and $\overline{A}_K$ are the arc sets for the two variables. Constraints (15a) and (15b) are the arc-disjoint and delay-difference constraints of the two paths. Constraint (15c) reformulates the reduced cost and forces it to

be negative, i.e., solutions with non-negative reduced cost are infeasible. Different from standard pricing problem formulations, we model ours as a constraint satisfaction problem enforcing the negative reduced costs via a constraint.

6.2. Search Strategy for Bi-Path Generation

The proposed CP solver is compatible with any valid search strategy, with depth-first search as default. We use a strategy that first determines an elementary path for the primary and then for the secondary route variable, guided by the shortest path from the origin to the destination of the variables. With the primary path fixed, the arcs used by the primary path are closed for the secondary route variable and, as the accumulated delay of the primary path is fixed, the range of the delay of the secondary path is determined. Efficient algorithms tailored for resource-constrained shortest path problems (e.g., "pulse" (Lozano and Medaglia 2013)) are hence used. This search strategy is particularly advantageous as the `DelayDiffUB` constraint cannot make strong inferences about the delay difference of the two incomplete paths.

Similarly, with the total cost of the primary path fixed, the cost-based filtering for the secondary route variable is stronger. If the CP solver detects a dead-end, it backtracks to another (partial) primary path.

7. CP-based Column Generation

To strengthen the linear relaxation obtained from the CP-based column generation, we propose a CP-based cut generation method. A cutting planes (cut generation) method iteratively refines a feasible set by adding cuts, i.e., linear inequalities, to LP and MILP problems (Dantzig et al. 1954). For MCF variants, cover cuts are stronger than the capacity constraints and can be generated to strengthen the linear relaxation (Balas 1975).

7.1. Generating Minimal Cover Cuts

A cover is any subset of K that exceeds the arc capacity for some arc a. Given a cover C, at most $|C|-1$ commodities of C can be routed via arc a. A cover cut (Balas 1975) is then introduced to strengthen the capacity constraint as follows:

$$\sum_{k\in C} x_a^k \leq |C|-1, \tag{16}$$

where $x_a^k = 1$ if commodity k uses arc a as a part of its path.

In BP-MCF, an arc cannot be used by both paths of the same commodity. Thus, the following inequality using decision variables in the compact formulation is satisfied by all integer solutions:

$$\sum_{k\in C}(x_a^k + \overline{x}_a^k) \leq |C|-1. \tag{17}$$

This inequality is a cover cut for BP-MCF. Moreover, a cover C is minimal if

$$\sum_{k\in C} b_k > c_a \quad \texttt{and} \quad \sum_{k\in C-\{i\}} b_k \leq c_a, \forall i \in C. \tag{18}$$

The minimal cover inequalities dominate all others (Balas 1975).

According to the arc-disjointness, for any commodity k, $x_a^k + \overline{x}_a^k \leq 1$ always holds. When the variables are discrete, we can use variable $\hat{x}_a^k = x_a^k + \overline{x}_a^k$ to replace x_a^k and $\overline{x}_a^k$. We then use $\hat{\mathbf{X}}$ to represent the set of $\hat{x}_a^k$ with domain of $\{0, 1\}$. Note that this replacement is for the convenience of presentation. In practice, the two sets of variables are manipulated separately. Let $\hat{\mathbf{X}}'$ ($\mathbf{X}'$ and $\overline{\mathbf{X}}'$) be the fractional solution obtained from CPCG. The problem of finding a cover cut violated by $\hat{\mathbf{X}}'$, or deciding that none exists, is the separation problem for the cover cuts: deciding if there exists a subset $C \subseteq K$ such that

(i). $\sum_{k \in C} b_k > c_a$,

(ii). $\sum_{k \in C} \hat{x}_a'^k > |C| - 1$.

Condition (i) ensures that C is a cover, while condition (ii) indicates that $\hat{\mathbf{X}}'$ violates the corresponding cover cut.

We propose a CP-based cut generation (CPCT) method to heuristically generate cover cuts violated by the current fractional solution. The process is conducted with an independent CP model considering capacity constraints and all commodities on small graphs (one for each route variable) constructed from active columns (obtained from column generation) to efficiently generate cuts. The detected cuts are guaranteed to be minimal. The CPCT method detects a cover cut by seeking a set of commodities and an arc that satisfies inequalities (i) and (ii).

7.2. Algorithm Details

CPCT is performed after each run of CP-based column generation (CPCG) to add cuts to the master problem (3a)-(3d') and strengthen the linear relaxation. After an iteration of CPCG, the generated columns form partial graphs for each commodity. We create a single multi-commodity CP model with these partial graphs and arc capacity constraints. Our intuition is that the newly generated columns are likely to be incorporated into the next solution to the master problem. By generating cuts specifying incompatibilities between these columns, we are tightening the subsequent master relaxation. During the solving process of the CP model, whenever a capacity constraint is violated on an arc, condition (i) is satisfied. Then based on the fractional variable solutions obtained from CPCG we evaluate condition (ii), which generates a cover cut if satisfied. The CP model for cut generation is as follows:

$$\texttt{ArcDisjoint}(rv_k, \overline{rv}_k) \qquad \forall k \in K, \tag{19a}$$

$$\texttt{DelayDiffUB}(rv_k, \overline{rv}_k, \Delta_k) \qquad \forall k \in K, \tag{19b}$$

$$\texttt{ArcCapacity}(\{rv_k, \overline{rv}_k, \forall k \in K\}, \{c_a, \forall a \in A\}) \tag{19c}$$

$$rv_k = (s_k, t_k, b_k, G, A_k') \qquad \forall k \in K, \tag{19d}$$

$$\overline{rv}_k = (s_k, t_k, b_k, G, \overline{A}_k') \qquad \forall k \in K. \tag{19e}$$

Algorithm 2: CPCT

Input: K - the set of commodities; A - the set of arcs; $\hat{\mathbf{X}}^{'}$ - the fractional solution; $\mathcal{M}$ - the CP model; $\{rv_k, \overline{rv}_k, \forall k \in K\}$ - the set of route variables.

Output: $\mathscr{C}$ - the set of covers found by CPCT.

1 $K^{'} \leftarrow \texttt{SortNonIncreasing}(K)$; $\mathscr{C} \leftarrow \emptyset$;
2 In the CP model (19a)-(19e) **while** stop condition not met **do**
3 **for** $k \in K^{'}$ **do**
4 **repeat**
5 $\texttt{SetPrimaryThenSecondaryPath}(\mathcal{M}, rv_k, \overline{rv}_k)$;
6 $(a, K_a) \leftarrow \texttt{FindViolatedArc}(\mathcal{M}, A)$;
7 **if** $K_a \neq \emptyset$ and $\texttt{EvalCondition2}(\hat{\mathbf{X}}^{'}, K_a) =$ True **then** $\mathscr{C} \leftarrow \mathscr{C} + \{K_a\}$;
8 **until** rv_k and $\overline{rv}_k$ are set or stop condition are met;
9 **return** $\mathscr{C}$;

For commodity k, rv_k and $\overline{rv}_k$ are the route variables of the primary and secondary paths. The sets $A_k^{'}$ and $\overline{A}_k^{'}$ contain the arcs of the partial graphs formed by the generated columns for commodity k. Constraints (19a) and (19b) are the arc-disjoint and delay-difference constraints. Constraint (19c) is the capacity constraint over all the route variables and arcs in the network. Note that there is no objective function in the CP model. The goal of using this model in CPCT is to make inferences based on the constraint violation.

The search strategy of the CP model is designed to first set the primary and then the secondary paths for a commodity. All the commodities are sorted in non-increasing order of bandwidth and their bi-paths are set one commodity at a time in this order, with backtracking in case of constraint violation. The procedure is presented in Algorithm 2.

In line 1 of the algorithm, the set of commodities is sorted in non-increasing order of bandwidth. Then we start solving the CP model: for each commodity, the primary path and secondary path are determined, with a priority on setting the primary path, as shown in line 5. If the capacity of an arc a is violated, the set of all the commodities using arc a (i.e., K_a) is returned in line 6. The backtracking is implicitly allowed in the while loop as shown in line 4. If condition (ii) is also satisfied according to line 7, K_a will be added to the set of covers $\mathscr{C}$. The cover cut corresponding to K_a is as follows:

$$\sum_{k \in K_a} \hat{x}_a^k \leq |K_a| - 1. \tag{20}$$

In the context of CPCG, all such cover cuts in the returned $\mathscr{C}$ are added to the master problem (3a)-(3d') in the following form:

$$\sum_{k \in K_a} \sum_{(p,\overline{p}) \in PP_k | a \in p \text{ or } a \in \overline{p}} z_{p,\overline{p}}^k \leq |K_a| - 1. \tag{21}$$

Let the corresponding dual variable in the master problem be η_a. Then the reduced cost constraint (15c) of the CP-based pricing problem for commodity $k \in C$ with the new dual values becomes

$$\sum_{a \in A_k^s} [(\alpha_a + \pi_a)b_k + \eta_a] + \sum_{a \in \overline{A}_k^s} [(\beta_a + \pi_a)b_k + \eta_a] < \mu_k. \tag{12c'}$$

PROPOSITION 3. *Any cover cut found by CPCT is minimal.*

Proof. Since the commodities are sorted in decreasing order of their bandwidths (b_k) in CPCT, when a capacity violation of arc a is detected, the trigger is the commodity with the smallest bandwidth in the current cover. Since the previous set of commodities using the arc does not form a valid cover, they only occupy a total bandwidth of $B \leq c_a$. As the bandwidth of all the other commodities in the cover is no smaller than the newly added commodity, removing any of them from the cover would induce a smaller total bandwidth occupancy B' and $B' \leq B \leq c_a$. Thus, the cover cut found by CPCT is minimal. □

7.3. Extended Cover Cuts

The generated cover cuts could be extended (Wolsey and Nemhauser 1999). If $C \subseteq K$ is a cover, the extended cover $E(C)$ is defined as $E(C) = C \cup \{i \in K | b_i \geq b_k, \forall k \in C\}$. Accordingly, an extended cover cut associated with the capacity constraint of arc a is

$$\sum_{k \in E(C)} \hat{x}_a^k \leq |C| - 1, \tag{22}$$

and the extended cover cut $E(C)$ dominates the original cover cut C. In our framework, all the generated cover cuts are extended (if possible) to strengthen the linear relaxation.

8. CP-based Primal Heuristic

Column generation provides a solution to the linear relaxation of BP-MCF. Within a branch-and-price-and-cut approach, it is often useful to employ primal heuristics to find integral feasible solutions. There are generally two options:

(a). Round the continuous solution.

(b). Solve a restricted integer optimization problem using only the generated columns.

Here we propose a CP-based approach for (b). We use CP as a primal heuristic to solve a relaxed version of the restricted master problem where the possible network for commodity k is constructed from all the arcs in the generated columns for commodity k.

8.1. CP Model for the Primal Heuristic

The CP model for the primal heuristic is presented as follows:

$$\texttt{min} \sum_{k \in K} (\sum_{a \in A_k^s} \alpha_a b_k + \sum_{a \in \overline{A}_k^s} \beta_a b_k) \tag{24}$$
$$(19a) - (19e).$$

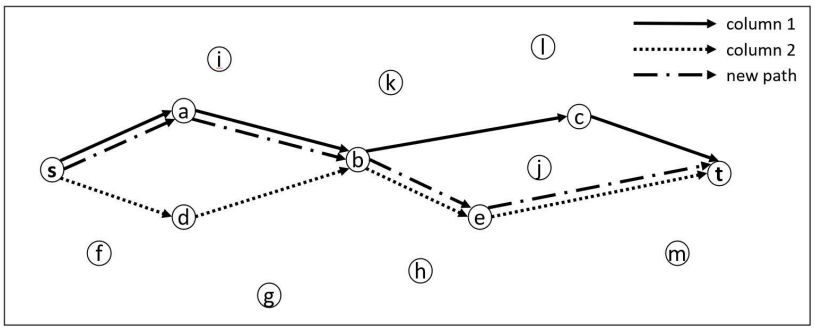

Figure 5 More paths than columns in the graph of a route variable.

The objective (24) minimizes the total arc cost in the two paths of each commodity. The constraints are exactly the same as the CP model used for CP-based cut generation. Note that a feasible integral solution does not necessarily exist when the relaxed problem is feasible, thus the CP model may prove infeasibility.

There are several advantages of using CP as a primal heuristic. First, the graph for each commodity is much smaller than the original graph as it is constructed from the arcs of the generated columns for that commodity, as shown in (19d) and (19e). With smaller graphs, the `ArcDisjoint`, `DelayDiffUB`, and `ArcCapacity` constraints are stronger.

Second, as shown in Fig. 5, the graph of each variable may contain more paths than just the active columns because the formulation is arc-based: arcs from different columns may be combined to form a novel path. The arc-based CP model searches through all paths that can be constructed by arcs that participate in the routes of the current node column generation. A similar idea is used by Alvelos and de Carvalho (2007).

Finally, cycles are eliminated with the internal inference of the CP solver.

8.2. Search Strategy for the Primal Heuristic

We use a search strategy similar to the bi-path generation of BP-MCF. Each commodity is processed in sequence and the primary path is determined first. With a primary path fixed, the CP solver can quickly find a valid secondary path or prove that none exists, backtracking to another primary path for the commodity. The `ArcCapacity` constraint filters arcs as more and more primary and secondary paths are determined.

The order in which commodities are processed may matter to the performance of the CP solver. In our implementation, the commodities are processed in random order. We leave the investigation of other variable ordering heuristics to future work. Also note that since the CP solver looks for feasible solutions with small objective values, the primary path is determined with a priority for small costs.

9. CP-based Branching Node Heuristic

The CP-based column generation, CP-based cut generation, and CP-based primal heuristic modules are performed at every node of a search tree that is constructed with a two-level CP-based branching strategy: the first-level branches on accumulated path delays of primary route variables and the second-level branches

on removing arcs of primary and secondary route variables. In this section, we first introduce the two-level branching scheme and then the CP-based branch node heuristic for node selection.

9.1. First-Level Branching Scheme

The first level strategy branches on primary delays. Consider a fractional solution where, for a commodity, several bi-path variables take fractional values. We use their weighted average delay as the branching threshold. Formally, for commodity k in a CPCG solution, n_k columns take fractional values $\{w_1^k, w_2^k, ..., w_{n_k}^k\}$ where $\sum_{i=1}^{n_k} w_i^k = 1$ and $0 < w_i^k < 1, \forall i = 1, ..., n_k$. Let $\{D_1^k, D_2^k, .., D_{n_k}^k\}$ be the accumulated delays of the n_k columns. The branching threshold is obtained as

$$\theta_k := \sum_{i=1}^{n_k} w_i^k D_i^k. \tag{24}$$

Then the two branches are $\sum_{a \in A_k^s} d_a \leq \theta_k$ and $\sum_{a \in A_k^s} d_a > \theta_k$. In our CP framework, we enforce the two branches by adding constraints on the delay range of the primary route variable as follows:

$$\texttt{DelayRange}(rv_k, l_k, \theta_k) \ \texttt{\textbackslash/} \ \texttt{DelayRange}(rv_k, \theta_k + \epsilon, u_k). \tag{25}$$

In the right branch, ϵ is a sufficiently small numeric value. If this commodity is branched on again in a deeper node, the primary delay ranges are updated in the `DelayRange` constraint rather than adding a new constraint. Thus, for each node in the search tree, there is at most one `DelayRange` constraint on the primary route variable of a commodity. For branching node selection, we select the commodity with the biggest delay variance among its non-zero weighted paths.

To the best of our knowledge, this is the first time that a delay-based branching scheme is proposed for solving MCF problems though there are some conceptual similarities with branching on time windows in vehicle routing problems (Gélinas et al. 1995). However, this delay-based branching scheme is not complete. There are two cases where it cannot distinguish two bi-paths: (1) if more than one bi-path of a commodity has the same primary delay, and (2) if there exists more than one secondary path with the same cost and the same primary path. Thus, we need a second-level branching scheme to ensure completeness.

9.2. Second-Level Branching Scheme

If the first-level branching scheme cannot distinguish several bi-paths, the second-level branching scheme branches by removing arcs via the divergence node-based branching approach (Barnhart et al. 2000). In this approach, if a commodity takes two fractional values for its (primary or secondary) paths in the linear relaxation, we find the divergence node: the first node where the two paths diverge. Then all the outgoing arcs from the divergence node are divided into two sets: A_1 and A_2, where the two fractional arcs are not in the same one. The two resulting branches are then removing A_1 and removing A_2, respectively.

A problem with the second-level branching scheme is that it may not efficiently partition the solution space such that the optimal solution is in exactly one branch. If the divergence node of a commodity is not

visited in the optimal solution, then that solution is in both branches and the current branching node is not well selected. For BP-MCF, this second-level branching scheme is a suitable complement for the first-level branching scheme, as it will always distinguish the indistinguishable cases in level 1: as long as the primary or secondary paths of two bi-paths are different, a divergence node will exist.

9.3. CP-based Branch Evaluation

In order to improve the efficiency of the second-level branching scheme, we propose a constraint propagation method for selecting branching nodes. For the two branches of a branching node, CP propagates (i.e., makes valid inferences according to the arc status of route variables) on each of the two branches to remove more arcs. A metric score, calculated after the two branches are propagated, is used to make the branching decision.

Specifically, in a fractional solution with multiple divergence nodes in different commodity graphs, we use the CP model (19a)-(19e) for constraint propagation. For the divergence node i, define N_1^i as the number of open arcs in the network after propagating on the first branch and N_2^i as the number for the second branch. The score N^i for node i is

$$N^i = N_1^i + N_2^i + |N_1^i - N_2^i|. \tag{26}$$

We select the branching node with the smallest score. The intuition behind the selection criteria is to obtain smaller commodity networks to detect dead-end or feasible solutions faster. The absolute value term ensures that the two branches are not too unbalanced.

10. Experimental Evaluation

In this section, we present the numerical experiments on diverse instances of bi-path MCF generated in the context of telecommunication applications.

10.1. Instance Generation

The majority of problem instances for experiments are generated from two well-known libraries: SNDlib, a library for telecommunication network design (`http://sndlib.zib.de`); and The Internet Topology Zoo, a collection of real networks (`http://www.topology-zoo.org`). The rest of the instances are generated randomly. The three sets of instances follow a similar generation scheme. In this section, we briefly describe the instances and refer readers to the appendix (Zhang et al. 2023a) for the detailed generation processes.

We select 10 topologies from SNDlib, 10 topologies from The Internet Topology Zoo, and generate 9 topologies randomly. The number of nodes in randomly generated topologies varies in $\{20, 25, 30\}$ and the density of the graph varies in $\{0.3, 0.4, 0.5\}$. The edge costs for primary paths are multi-valued, while the edge costs for secondary paths have two configurations: all-1 or all-0. With the total number of feasible demands being K_f for each instance, three configurations $\{K_f/1, K_f/2, K_f/3\}$ are used in different

Table 2 Components of the CPBPC variants

Algorithm	column generation	cut generation	primal heuristic	branching node heuristic
CPBPC-div	✓	✓	✓	✓
CPBPC-div-random	✓	✓	✓	-
CPBPC-del&div	✓	✓	✓	✓
CPBPC-del&div-noCut	✓	-	✓	✓
CPBPC-del&div-noPH	✓	✓	-	✓
CPBPC-del&div-noCutPH	✓	-	-	✓

instances. Yen's algorithm (Yen 1971) is used to find the top 40 delay-shortest paths. Then the delay differences between the 10^{th} and the $\{20^{th}, 30^{th}, 40^{th}\}$ paths are used as the delay threshold.

In summary, we generate $10 \times 2 \times 3 \times 3 = 180$ instances from SNDlib, $10 \times 2 \times 3 \times 3 = 180$ instances from the Internet Topology Zoo, and $3 \times 3 \times 2 \times 3 \times 3 = 162$ instances randomly. All the 522 instances and results can be found online (Zhang et al. 2023b).

10.2. Experiment Setting

We summarize the components of the evaluated CPBPC variants in Table 2. In the table, 'del' refers to the delay-based branching scheme and 'div' refers to the divergence node-based branching scheme. The algorithms are implemented in C++ on a machine with Intel(R) Xeon(R) CPU E5-4627 v2 of 3.30GHz with 504GB RAM, running under Linux 64-bit OS. CPLEX 12.6 IBM (2023) is used as the solver for LP and MILP. For every single run, one thread is used with a 1000s time limit and a 5Gb memory limit. We solve all CP models with the proprietary CP solver described above developed at Paris Research Center, Huawei France.

For CPBPC, we maintain a branch-and-price binary search tree and search for solutions using breadth-first search (BFS). BFS updates the global lower bound efficiently but may not be able to update the global upper bound at an early stage.

For column generation, the pricing problem of each demand has a time limit of 100 ms, while the master problem is run for up to 6000 ms. The CP-based cut generation is run for 100 ms. Similarly, the primal heuristic is constrained by a time limit of 100 ms but the branching node heuristic has no time limit as it is doing propagation only.

For comparison, a monolithic compact CP model is also solved with the specialized CP solver. The model is represented by (24), (19a) - (19e) with the entire network for all demands instead of partial networks constructed from active columns.

10.3. Numerical Results

The numerical results are summarized in Table 3. The 'gap' column is the mean optimality gap over all the instances. All algorithms calculate both a lower bound (LB) and an upper bound (UB) so the gap

Table 3 Results over 522 instances of the CPBPC algorithms and the compact MILP model

Algorithm	gap	time(s)	#nodes	#feas	#opt	#cols	#cuts	#PHs
CPBPC-div	7.27%	418	**20**	363	312	167	605	289
CPBPC-div-random	6.97%	406	39	367	313	173	652	178
CPBPC-del&div	**6.77%**	**386**	24	**380**	**332**	176	540	186
CPBPC-del&div-noCut	6.91%	404	59	379	326	169	-	365
CPBPC-del&div-noPH	8.46%	439	45	339	310	164	664	-
CPBPC-del&div-noCutPH	10.73%	485	108	302	285	150	-	-
Compact MILP	30.36%	854	27470	139	101	-	458	-
Compact CP	38.99%	929	210060	55	47	-	-	-

is determined by the corresponding algorithm. If no feasible solution is found within the time limit, a default upper bound found by a greedy heuristic used during instance generation is applied to calculate the optimality gap. The 'time(s)' column is the mean runtime over all the instances. The '#nodes' column is the mean number of evaluated nodes over the instances that are solved to optimality by the associated algorithm. The '#feas' column is the number of instances with at least one feasible solution found while the '#opt' column is the number of instances with the optimal solution found and proved. The '#cols' column is the mean number of generated columns over the instances solved to optimality. The columns labelled '#cuts' and '#PHs' are the mean number of generated cuts and calls to the primal heuristic, respectively. For CPBPC variants, the generated cuts are cover cuts, while for the compact MILP model, the generated cuts are cycle elimination constraints.

From the table, we can see that all the CPBPC variants are substantially better than the compact MILP and CP models in both solution quality and run-time. CPBPC-div and CPBPC-div-random find almost the same number of optimal solutions, while the mean number of evaluated nodes of CPBPC-div is just half of the number of CPBPC-div-random, demonstrating the effectiveness of CPBNH in finding good branching nodes. However, CPBNH can be time-consuming when there are multiple divergence nodes for many demands, leading to a better mean optimality gap and runtime of CPBPC-div-random. For the algorithm variants with the two-level branching scheme, CPBPC-del&div is the best. We can also see that removing CPCT and/or CPPH leads to worse results and removing the latter results in more significant degradation.

The detailed results of the eight algorithms for different groups of instances are summarized in Table 4. For the instances from SNDlib, about half of the instances have feasible solutions found by CPBPC algorithms and more than one-third are solved to optimality. Algorithms with two-level branching schemes do not have obvious advantages. CPBPC-del&div and CPBPC-del&div-noCut perform the best and demonstrate the importance of the primal heuristic in the CPBPC framework.

For Internet Topology Zoo instances, almost all instances have feasible solutions found by CPBPC algorithms, and more than three-quarters of the instances are solved to optimality. For CPBPC algorithms, these

Table 4 Results of the CPBPC algorithm and the compact MILP model for different instance groups

Instances	Algorithm	gap	time(s)	#nodes	#feas	#opt
SNDlib	CPBPC-div	8.36%	672	**57**	82	61
	CPBPC-div-random	8.00%	662	133	83	62
	CPBPC-del&div	8.19%	**633**	64	92	**69**
	CPBPC-del&div-noCut	**7.79**%	650	137	**93**	67
	CPBPC-del&div-noPH	8.71%	668	114	75	64
	CPBPC-del&div-noCutPH	9.44%	705	222	66	58
180	Compact MILP	19.60%	909	29464	29	20
instances	Compact CP	27.92%	933	226274	16	14
Internet	CPBPC-div	1.86%	198	**8**	168	147
Topology	CPBPC-div-random	1.78%	192	12	169	146
Zoo	CPBPC-del&div	**1.18**%	165	12	173	154
	CPBPC-del&div-noCut	1.20%	**163**	32	**174**	**156**
	CPBPC-del&div-noPH	2.99%	211	23	162	149
	CPBPC-del&div-noCutPH	5.13%	240	70	150	142
180	Compact MILP	28.32%	714	26777	79	57
instances	Compact CP	41.81%	875	191332	29	27
Random	CPBPC-div	12.05%	381	**13**	113	104
	CPBPC-div-random	11.60%	360	17	**115**	105
	CPBPC-del&div	**11.40**%	**356**	16	**115**	**109**
	CPBPC-del&div-noCut	12.29%	399	49	112	103
	CPBPC-del&div-noPH	14.27%	438	32	102	97
	CPBPC-del&div-noCutPH	18.39%	510	94	86	85
162	Compact MILP	45.57%	975	63372	13	6
instances	Compact CP	48.14%	985	256504	10	6

instances are easier than the instances from SNDlib. Similarly, the compact MILP/CP approaches find feasible and optimal solutions for more instances from this group than SNDlib, though they perform worse in terms of the mean optimality gap. Algorithms with the two-level branching scheme have a significant advantage over the algorithms with the divergence-based branching scheme only.

For randomly generated instances, all CPBPC algorithms behave similarly by providing feasible or optimal solutions for more than half of the instances. These instances are harder than other instances according to the mean optimality gap of CPBPC and compact MILP/CP approaches. The reason fewer feasible and optimal solutions are found is that problems get harder as the number of demands, the size, and the density of the network increase. For these instances, CPBPC-del&div achieves the best results.

Recall that we use three different configurations of the threshold for delay difference. We hence reorganize the results of CPBPC-div, CPBPC-del&div, and the compact MILP/CP approaches according to the three configurations in Table 5.

In the table, the relative UBs (LBs) are calculated as the mean relative value of the UB (LB) divided by the LB value of the corresponding instance and algorithm when the delay-difference threshold is 10^{th} to 40^{th} as shown in (27). The wider range of the delay difference results in a less constrained problem and

Table 5 Results with different upper bounds of delay difference

Algorithm	delay-diff UB	gap	time(s)	relative UB	relative LB	#feas	#opt
CPBPC-div	10^{th} - 20^{th}	8.00%	463	1.20	1.11	114	96
	10^{th} - 30^{th}	7.21%	428	1.10	1.03	123	102
	10^{th} - 40^{th}	6.59%	363	1.07	1.00	126	114
CPBPC-del&div	10^{th} - 20^{th}	7.53%	434	1.19	1.11	121	101
	10^{th} - 30^{th}	7.08%	391	1.10	1.03	128	111
	10^{th} - 40^{th}	5.70%	332	1.06	1.00	131	120
compact MILP	10^{th} - 20^{th}	37.93%	944	1.44	1.05	18	12
	10^{th} - 30^{th}	30.51%	872	1.33	1.02	39	24
	10^{th} - 40^{th}	23.55%	770	1.24	1.00	64	47
compact CP	10^{th} - 20^{th}	47.98%	974	1.49	1.04	5	5
	10^{th} - 30^{th}	35.91%	926	1.38	1.02	22	15
	10^{th} - 40^{th}	33.07%	887	1.33	1.00	28	black27

Table 6 Results with different secondary costs

Algorithm	secondary cost	gap	time(s)	#feas	#opt
CPBPC-div	all-1	8.19%	446	180	149
	all-0	6.34%	391	183	163
CPBPC-del&div	all-1	7.44%	424	187	157
	all-0	6.10%	347	193	175
Compact MILP	all-1	25.43%	855	69	47
	all-0	34.33%	825	58	42
Compact CP	all-1	41.14%	928	27	21
	all-0	34.71%	890	28	26

potentially smaller objective value in minimization problems. As a result, with the proposed denominator, the relative UB (LB) is always ≥ 1.00.

$$relative\, UB_{10^{th}-r^{th}} = \frac{UB_{10^{th}-r^{th}}}{LB_{10^{th}-40^{th}}} \tag{27}$$

The results show that with the delay-difference threshold increasing, the problem becomes easier: the gap and runtime get smaller while the number of instances with feasible or optimal solutions found increases. In addition, the obtained objective decreases, which means better solutions are found with looser delay-difference thresholds.

Finally, we present the results of CPBPC-div, CPBPC-del&div, and the compact MILP approach with two types of different secondary costs, i.e., all-1 or all-0 costs in Table 6.

We can see that the two CPBPC algorithms and the compact CP approach find feasible and optimal solutions for more instances with all-0 secondary costs, while the compact MILP approach finds feasible and optimal solutions for fewer instances with all-0 secondary costs. All-0 costs mean that the secondary

paths only need to satisfy the arc-disjoint and delay-difference constraints: an easier task for each pricing problem and for the compact CP model. However, for the compact MILP approach, all-0 costs make it harder to distinguish secondary paths with costs, which might more frequently create cycles to satisfy the delay-difference threshold constraints.

11. Discussions

We selected branch-and-price-and-cut as the decomposition framework because of its suitability for the MCF problem variants and its scalability for large industrial instances. In particular, column generation leads to high-quality dual bounds and simple sub-problems. The branch-and-price-and-cut framework also takes advantage of the versatility of CP, which stems from a core model that is applicable at different stages in industrial settings (Kadıoğlu 2019). For comparison, logic-based Benders decomposition (LBBD) (Hooker and Ottosson 2003) might concentrate too much computation burden on a master problem and induce a large number of cuts. Large neighborhood search (LNS) (Ropke and Pisinger 2006) can be effective by fixing a subset of route variables and searching for better solutions over the remaining variables. However, LNS sacrifices the exactness of the approach and our intuition is that the search space is too large for it to be promising.

In the branch-and-price-and-cut framework, our results are obtained with a specialized CP solver, but similar results could be expected with other CP solvers that can handle route variables. There is a certain amount of effort needed to formulate/implement the `ArcDisjoint`, the `DelayDiffUB`, and the cycle elimination constraints. However, the propagation algorithms of these constraints are not complicated.

As shown in the results, the four CP-based algorithmic modules all contribute to the superior performance of the CPBPC approach. To achieve similar performance, the minimum required components are route variables, several global constraints on route variables, and the CP-based column generation module. However, a primal heuristic, not necessarily CP-based, also provides substantial benefits. Intuitively, column generation provides high-quality dual bounds and the primal heuristic generates good primal bounds, speeding up the convergence of the algorithms.

12. Conclusions

In this paper, we integrated Constraint Programming (CP) and Linear Programming (LP) in a CP-based branch-and-price-and-cut (CPBPC) approach for Multi-Commodity Flow (MCF) problems. In addition to the column generation itself, we propose the novel use of CP for the cut generation, primal heuristics, and branching node heuristics in the CPBPC framework. With minor modifications, the approach can be applied to different routing problems and, with more effort, to other combinatorial optimization problems.

The proposed solution approach is applied to a bi-path multi-commodity flow (BP-MCF) optimization problem, where a primary path delivers each demand and an arc-disjoint, delay-similar secondary path is

reserved in case of the failure of the primary path. The goal is to route demands in a capacitated network under the minimum cost. We proved the NP-hardness of BP-MCF and its pricing problem, provided two mixed-integer linear programming formulations and a constraint programming formulation, and developed a delay-based branching scheme.

We evaluated the performance of the proposed CPBPC approach, a compact MILP model, and a compact CP model on a diverse set of BP-MCF instances. Experimental results show that the CPBPC approach is significantly superior to the compact MILP and CP models in terms of optimality gap, runtime, size of the search tree, and the number of instances solved. Thus, this work represents an important contribution to exact methods for MCF and demonstrates the promising prospect of CP-LP integration.

Our proposed methodology can be viewed as a general CP-based framework, where every component except for the master problem solver is a CP module. Each module in the framework can be useful in general. The CP-based cut generation can be applied to other MCF variants with capacity constraints. The idea of incorporating more paths than columns in the CP-based primal heuristic can be adapted to other problems based on path generation. The CP-based branching node heuristic is effective when branching can lead to domain pruning of decision variables. Lastly, the two-level branching scheme can be adapted to other problems for which no efficient and complete branching scheme exists.

For future work, the proposed approach can be adapted to other MCF variants, and a more efficient filtering algorithm of the capacity constraint can further improve the performance of the primal heuristics, cut generation, and cut lifting components based on CP. Other ways of CP-LP integration are worth exploring. In particular, using the fractional solutions of the embedded LP relaxation in the CP solver to guide the CP search for MCF is an interesting future direction.

References

Ajili F, Rodošek R, Eremin A (2005) A branch-price-and-propagate approach for optimizing IGP weight setting subject to unique shortest paths. *Proceedings of the 2005 ACM Symposium on Applied Computing*, 366–370.

Alvelos FP, de Carvalho JV (2007) An extended model and a column generation algorithm for the planar multicommodity flow problem. *Networks: An International Journal* 50(1):3–16.

Alves C, de Carvalho JV (2008) A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem. *Computers & Operations Research* 35(4):1315–1328.

Angilella V, Krasniqi F, Medagliani P, Martin S, Leguay J, Shoushou R, Xuan L (2022) High capacity and resilient large-scale deterministic IP networks. *Journal of Network and Systems Management* 30(4):1–28.

Archetti C, Bouchard M, Desaulniers G (2011) Enhanced branch and price and cut for vehicle routing with split deliveries and time windows. *Transportation Science* 45(3):285–298.

Assad AA (1978) Multicommodity network flows - a survey. *Networks* 8(1):37–91.

Balas E (1975) Facets of the knapsack polytope. *Mathematical Programming* 8(1):146–164.

Barnhart C, Hane CA, Vance PH (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research* 48(2):318–326.

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MW, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329.

Beck JC, Refalo P (2003) A hybrid approach to scheduling with earliness and tardiness costs. *Annals of Operations Research* 118:49–71.

Bhatia R, Hao F, Kodialam M, Lakshman T (2015) Optimized network traffic engineering using segment routing. *2015 IEEE Conference on Computer Communications (INFOCOM)*, 657–665 (IEEE).

Caimi G, Chudak F, Fuchsberger M, Laumanns M, Zenklusen R (2011) A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. *Transportation Science* 45(2):212–227.

Costa L, Contardo C, Desaulniers G (2019) Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science* 53(4):946–985.

Cronholm W, Ajili F (2004) Strong cost-based filtering for lagrange decomposition applied to network design. *International Conference on Principles and Practice of Constraint Programming*, 726–730 (Springer).

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America* 2(4):393–410.

Desaulniers G (2010) Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research* 58(1):179–192.

Dijkstra EW, et al. (1959) A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1):269–271.

Dooms G, Deville Y, Dupont P (2005) Cp (graph): Introducing a graph computation domain in constraint programming. *Principles and Practice of Constraint Programming-CP 2005: 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005. Proceedings 11*, 211–225 (Springer).

D'Ambrosio C, Lodi A, Wiese S, Bragalli C (2015) Mathematical programming techniques in water network optimization. *European Journal of Operational Research* 243(3):774–788.

Finn N (2018) Introduction to time-sensitive networking. *IEEE Communications Standards Magazine* 2(2):22–28.

Focacci F, Lodi A, Milano M (2002) Embedding relaxations in global constraints for solving tsp and tsptw. *Annals of Mathematics and Artificial Intelligence* 34(4):291–311.

Fortz B, Gouveia L, Joyce-Moniz M (2017) Models for the piecewise linear unsplittable multicommodity flow problems. *European Journal of Operational Research* 261(1):30–42.

Foteinos V, Tsagkaris K, Peloso P, Ciavaglia L, Demestichas P (2014) Operator-friendly traffic engineering in IP/MPLS core networks. *IEEE Transactions on Network and Service Management* 11(3):333–349.

Frei C, Faltings B (1999) Resource allocation in networks using abstraction and constraint satisfaction techniques. *International Conference on Principles and Practice of Constraint Programming*, 204–218 (Springer).

Gélinas S, Desrochers M, Desrosiers J, Solomon MM (1995) A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research* 61:91–109.

Handler GY, Zang I (1980) A dual algorithm for the constrained shortest path problem. *Networks* 10(4):293–309.

Hartert R, Schaus P, Vissicchio S, Bonaventure O (2015) Solving segment routing problems with hybrid constraint programming techniques. *International Conference on Principles and Practice of Constraint Programming*, 592–608 (Springer).

Hashemi Doulabi SH, Rousseau LM, Pesant G (2016) A constraint-programming-based branch-and-price-and-cut approach for operating room planning and scheduling. *INFORMS Journal on Computing* 28(3):432–448.

Hooker JN, Ottosson G (2003) Logic-based benders decomposition. *Mathematical Programming* 96(1):33–60.

IBM (2023) IBM ILOG CPLEX Optimizer `https://www.ibm.com/de-de/analytics/cplex-optimizer`.

Junker U, Karisch SE, Kohl N, Vaaben B, Fahle T, Sellmann M (1999) A framework for constraint programming based column generation. *International Conference on Principles and Practice of Constraint Programming*, 261–274 (Springer).

Kadıoğlu S (2019) Core group placement: allocation and provisioning of heterogeneous resources. *EURO Journal on Computational Optimization* 7(3):243–264.

Karp RM (1975) On the computational complexity of combinatorial problems. *Networks* 5(1):45–68.

Kinable J, van Hoeve WJ, Smith SF (2020) Snow plow route optimization: A constraint programming approach. *IISE Transactions* 53(6):685–703.

Laborie P (2009) Ibm ilog cp optimizer for detailed scheduling illustrated on three problems. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 6th International Conference, CPAIOR 2009 Pittsburgh, PA, USA, May 27-31, 2009 Proceedings 6*, 148–162 (Springer).

Laborie P, Rogerie J (2016) Temporal linear relaxation in IBM ILOG CP Optimizer. *Journal of Scheduling* 19(4):391–400.

Laborie P, Rogerie J, Shaw P, Vilím P (2018) IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. *Constraints* 23:210–250.

Lam E, Desaulniers G, Stuckey PJ (2022) Branch-and-cut-and-price for the electric vehicle routing problem with time windows, piecewise-linear recharging and capacitated recharging stations. *Computers & Operations Research* 145:105870.

Lam E, Hentenryck PV (2016) A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints* 21:394–412.

Le Pape C, Perron L, Régin JC, Shaw P (2002) Robust and parallel solving of a network design problem. *Principles and Practice of Constraint Programming-CP 2002: 8th International Conference, CP 2002 Ithaca, NY, USA, September 9–13, 2002 Proceedings 8*, 633–648 (Springer).

Lozano L, Medaglia AL (2013) On an exact method for the constrained shortest path problem. *Computers & operations research* 40(1):378–384.

Milano M, Wallace M (2010) Integrating operations research in constraint programming. *Annals of Operations Research* 175(1):37–76.

Minoux M (2001) Discrete cost multicommodity network optimization problems and exact solution methods. *Annals of Operations Research* 106(1-4):19–46.

Ouaja W, Richards B (2005) Hybrid lagrangian relaxation for bandwidth-constrained routing: knapsack decomposition. *Proceedings of the 2005 ACM Symposium on Applied Computing*, 383–387.

Petterson M, Szymankek R, Kuchcinski K (2007) A CP-LP hybrid method for unique shortest path routing optimization. *International Network Optimization Conference (INOC)* (Citeseer).

Régin JC (1996) Generalized arc consistency for global cardinality constraint. *Proceedings of the 13th National Conference on Artificial Intelligence - Volume 1*, 209–215.

Risso C, Nesmachnow S, Robledo F (2018) Metaheuristic approaches for IP/MPLS network design. *International Transactions in Operational Research* 25(2):599–625.

Ropke S, Pisinger D (2006) An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science* 40(4):455–472.

Ros L, Creemers T, Tourouta E, Riera J (2001) A global constraint model for integrated routing and scheduling on a transmission network. *International Conference on Information Networks, Systems and Technologies*.

Rossi F, Van Beek P, Walsh T (2006) *Handbook of Constraint Programming* (Elsevier).

Rousseau LM, Gendreau M, Pesant G, Focacci F (2004) Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research* 130:199–216.

Sakkout HE, Wallace M (2000) Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints* 5(4):359–388.

Salimifard K, Bigharaz S (2022) The multicommodity network flow problem: state of the art classification, applications, and solution methods. *Operational Research* 22(1):1–47.

Sebbah S, Bagley C, Colena M, Kadioglu S (2016) Availability optimization in cloud-based in-memory data grids. *Principles and Practice of Constraint Programming: 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings 22*, 666–679 (Springer).

Shi H, Blaauwbroek N, Nguyen PH, Kamphuis RI (2016) Energy management in multi-commodity smart energy systems with a greedy approach. *Applied Energy* 167:385–396.

Tang F, Mao B, Kawamoto Y, Kato N (2021) Survey on machine learning for intelligent end-to-end communication toward 6G: From network access, routing to traffic control and streaming adaption. *IEEE Communications Surveys & Tutorials* 23(3):1578–1598.

Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: Large-scale information network embedding. *Proceedings of the 24th International Conference on World Wide Web*, 1067–1077.

Vlk M, Hanzálek Z, Tang S (2021) Constraint programming approaches to joint routing and scheduling in time-sensitive networks. *Computers & Industrial Engineering* 157:107317.

Wei L, Luo Z, Baldacci R, Lim A (2020) A new branch-and-price-and-cut algorithm for one-dimensional bin-packing problems. *INFORMS Journal on Computing* 32(2):428–443.

Wolsey LA, Nemhauser GL (1999) *Integer and combinatorial optimization*, volume 55 (John Wiley & Sons).

Xia Q, Simonis H (2005) Primary/secondary path generation problem: Reformulation, solutions and comparisons. *International Conference on Networking*, 611–619 (Springer).

Yen JY (1971) Finding the k shortest loopless paths in a network. *Management Science* 17(11):712–716.

Zhang J, Magnouche Y, Bauguion P, Martin S, Beck JC (2023a) Appendix to Computing Bi-Path Multi-Commodity Flows with Constraint Programming-based Branch-and-Price-and-Cut `https://tidel.mie.utoronto.ca/pubs/Computing_Bi-Path_MCP_with_CP-based_B&P&C(Appendix).pdf`.

Zhang J, Youcef M, Bauguion P, Martin S, Beck JC (2023b) Computing Bi-Path Multi-Commodity Flows with Constraint Programming-based Branch-and-Price-and-Cut URL `http://dx.doi.org/10.1287/ijoc.2023.0128.cd`, available for download at https://github.com/INFORMSJoC/2023.0128.