The Multi-commodity Flow Problem: Double Dantzig-Wolfe decomposition

Zhang Fan
Huawei Technologies Ltd.,
Research Center, Hong-Kong
zhang.fan2@huawei.com

Wang Jiazheng
Huawei Technologies Ltd.,
Research Center, Hong-Kong
wang.Jiazheng@huawei.com

Mathieu Lacroix
Université Sorbonne Paris Nord, LIPN, Villetaneuse, France
lacroix@lipn.univ-paris13.fr

Roberto Wolfler Calvo
Université Sorbonne Paris Nord, LIPN, Villetaneuse, France
roberto.wolfler@lipn.univ-paris13.fr

Youcef Magnouche
Huawei Technologies Ltd.,
Paris Research Center, France
youcef.magnouche@huawei.com

Sebastien Martin
Huawei Technologies Ltd.,
Paris Research Center, France
sebastien.martin@huawei.com

***Abstract*—Traffic Engineering (TE) represents one of the most essential tools in modern telecommunication networks. The rapid growth of exchanged traffic has required tackling a known NP-hard problem called the Multi-Commodity Flow problem (MCF). Many studies in the literature have already considered different variants of this problem. In this paper, we propose a new way to use a double Dantzig-Wolfe decomposition formulation to improve the quality of the linear relaxation. We apply our method on the classical multi-commodity flow problem where the throughput acceptance is first maximized and then the routing cost is minimized. We provide a computational experiment and conduct an in-depth analysis of the algorithm based on realistic instances.**

***Index Terms*—Multi-Commodity Flow Problem, Column Generation, Integer Programming, Network, Optimization, Dantzig-Wolfe Decomposition, Linear Programming.**

I. Introduction

The Multi-commodity flow problem is one of the most famous problems related to telecommunications. It consists in routing a set of commodities from their source to their target in a capacitated network, each commodity being routed along a single path. This problem has been intensively studied by the Operation Research community and many exact and heuristic methods have been devised [15]. Many of them are based on a *Mixed Integer Linear Programming (MILP)* formulation containing a huge number of variables that is known as the *arc-path formulation* [1].

When the size of the instance is not "too big", the arc-path formulation can be solved in an exact way using a branch-and-price algorithm [2], [14]. However, this method rapidly shows its limitations when the size of the instance increases and the formulation can only be heuristically solved. A way to do it is to solve the linear relaxation of the arc-path formulation to generate paths and to construct a solution by routing commodities using only these generated paths. A drawback of this method is that the linear relaxation cannot take into account the requirement of routing every commodity along a single path, resulting sometimes in a heuristic providing poor-quality solutions.

To address this limitation, one may contemplate enhancing formulations [3], [12], [13]; however, this comes with the trade-off of a heightened computational time. Consequently, a balance must be struck between accuracy and efficiency. In this paper, we conduct an experimental comparison of various strategies to navigate this trade-off. Dantzig-Wolfe decomposition is used to tackle the Multi-commodity flow problem where a column generation algorithm combined with a rounding procedure allows finding a good heuristic solution [7], [9], [10], [16].

In Section II, we give a formal description of the multi-commodity flow problem. In Section III, we present a double Dantzig-wolfe decomposition. In Section IV, we describe several rounding algorithms to generate integer solutions from the linear relaxation. In Section V we present computational results to compare each method and we end with a conclusion.

II. Multi-commodity flow problem and original double decomposition model

In this section, we formally present the multicommodity flow problem. First, let us introduce some notation.

A. Notation

Given a graph $G = (V, A)$ and a vertex subset $W \subseteq V$, the *leaving cut* (resp. *entering cut*) associated with W is the set of arcs having their tail in W and their head in $V \setminus W$ (resp. their tail in $V \setminus W$ and their head in W) and it is denoted by $\delta^{\text{out}}(W)$ (resp. $\delta^{\text{in}}(W)$). When $W = \{v\}$, we write $\delta^{\text{out}}(v)$ and $\delta^{\text{in}}(v)$ instead of $\delta^{\text{out}}(\{v\})$ and $\delta^{\text{in}}(\{v\})$, respectively.

A *walk* in a directed graph (V, A) is a sequence $p = (v_0, a_1, v_1, \ldots, a_k, v_k)$ where $k \geq 0$, $v_0, v_1, \ldots, v_k \in V$, $a_1, \ldots, a_k \in A$ and $a_i = (v_{i-1}, v_i)$ for $i = 1, \ldots, k$. The walk is a *path* if $v_0, \ldots, v_k$ are distinct. In the following, a path will also refer to its set of arcs. Vertex v_0 is the *starting vertex* of p and v_k its *ending vertex*. For $s, t \in V$, a path is an *st-path* if its starting vertex is s and its ending vertex is t.

B. Problem Definition

Let $G = (V, A)$ be a directed graph. With each arc $a \in A$ we associate a capacity $c_a \in \mathbb{Z}_+^*$ (in Mbps per second) and a routing cost $r_a \in \mathbb{Z}_+^*$. Let K be a set of commodities. For $k \in K$, let $s_k \in V$ and $t_k \in V$ denote the source and target nodes of the commodity and b_k its bandwidth.

A solution to the *Multicommodity Flows Problem (MFP)* consists of a $|K|$-tuple of $s_k t_k$-paths, one for every commodity $k \in K$, such that the total bandwidth of the commodities being routed on each arc does not exceed the arc capacity. The solution cost is the sum of the cost of each path. This latter cost is equal to the bandwidth of the commodity routed along the path times the sum of the routing costs of the arcs in that path. The MFP consists in maximizing the throughput acceptance on the network as the first objective that minimizes the cost in the second objective. These two objectives are considered in a lexicographic order.

C. Arc-Path Formulation

We associate with each commodity $k \in K$ and path $p \in P^k$ a binary variable x_p^k indicating whether commodity k is routed along path p ($x_p^k = 1$) or not ($x_p^k = 0$). Let P_a^k be the set of paths in P^k which contain arc $a \in A$ for each commodity $k \in K$. A binary variable y_k is associated with each commodity $k \in K$ and indicates whether commodity k is routed ($y_k = 0$) or not ($y_k = 1$). Let $M \in \mathbb{R}^+$ be a constant being at least equal to the longest path in G with respect to arc cost vector r. The MFP can be formulated with the following *arc-path formulation*:

$$\min \quad \sum_{k \in K} b_k \left(M y_k + \sum_{p \in P^k} \sum_{a \in p} r_a x_p^k \right) \tag{1}$$

$$\sum_{k \in K} b_k \sum_{p \in P_a^k} x_p^k \leq c_a \qquad \forall a \in A, \tag{2}$$

$$\sum_{p \in P^k} x_p^k + y_k \geq 1 \qquad \forall k \in K, \tag{3}$$

$$x_p^k, y_k \in \{0, 1\} \qquad \forall k \in K, p \in P^k. \tag{4}$$

Inequalities (2) are the *arc capacity constraints* ensuring that the total commodity bandwidth on each arc does not exceed its capacity. Inequalities (3) assert that for each commodity $k \in K$, a penalty Mb_k is paid if it is not routed by forcing y_k to be equal to one in this case.

III. Double Dantzig-Wolfe decomposition

To strengthen the linear relaxation, we investigate a Dantzig-Wolfe decomposition. We first present the formulation introduced in [13] and present how we extend this formulation to speed-up its linear relaxation.

A. Original Double Dantzig-Wolfe decomposition

Park et al. [13] formulated the max acceptance version of MFP with a binary linear problem that is obtained by applying a Lagrangian decomposition on the classical compact formulation. For each arc $a = (i, j) \in A$, let $K^a = \{k \in K : s_k \neq j, t_k \neq i\}$. A subset of commodities $L \subseteq K^a$ such that the total bandwidth $\sum_{k \in L} b_k$ is not more than the arc capacity c_a is called a *pattern* associated with arc a. For all $a \in A$, let B_a denote the set of patterns of arc a and B_a^k denote the set of patterns of B_a containing commodity $k \in K$.

Since a commodity must be routed along a single path, the set of commodities routed along an arc $a \in A$ corresponds to a pattern of B_a. We consider the same variables x and y as in the arc-path Formulation (1)-(4) and introduce the following new variables. For each arc $a \in A$ and each pattern $b \in B_a$, let z_b equal one if the commodities routed through arc a are included in b, and zero otherwise. The max acceptance version of MFP can be formulated as:

$$\min \sum_{k \in K} b_k \left(M y_k + \sum_{p \in P^k} \sum_{a \in p} r_a x_p^k \right) \tag{5}$$

$$y_k + \sum_{p \in P^k} x_p^k \geq 1 \qquad \forall k \in K, \tag{6}$$

$$- \sum_{b \in B_a} z_b \geq -1 \qquad \forall a \in A, \tag{7}$$

$$\sum_{b \in B_a^k} z_b - \sum_{p \in P_a^k} x_p^k \geq 0 \qquad \forall a \in A, \forall k \in K, \tag{8}$$

$$x_p^k \in \{0, 1\} \qquad \forall k \in K, p \in P^k, \tag{9}$$

$$y_k \in \{0, 1\} \qquad \forall k \in K, \tag{10}$$

$$z_b \in \{0, 1\} \qquad \forall a \in A, b \in B_a. \tag{11}$$

The objective function (5) and inequalities (6) are the same as (1) and (3) respectively. Inequalities (7) ensure that at most one pattern is selected per arc. Capacity requirement is ensured with *linking constraints* (8) that force an arc pattern to contain a commodity to route this commodity along this arc. (9)-(11) are the binary constraints.

Note that the number of variables is an exponential number for the two families of variables, the path variables x_p^k and the pattern variables z_b. Then to solve the linear relaxation of Formulation (5)-(11) we need to run a column generation algorithm. The pricing problem is of two types: a shortest path problem for each commodity $k \in K$ to find a minimum negative reduced cost path of P^k, and a maximum knapsack problem for each arc $a \in A$ to find the minimum reduced cost pattern of B_a. These pricing problems will be presented in details in the next section.

B. Partial Double Dantzig-Wolfe decomposition

We present a formulation where the capacity requirement is ensured by capacity constraints (2) for some arcs and by patterns and linking constraints for the other arcs. Let $\tilde{A} \subseteq A$ be the set of arcs for which the classical capacity constraint (2) is replaced with pattern constraints (7) and linking constraints (8).

$$\min \sum_{k\in K} b_k \left(My_k + \sum_{p\in P^k} \sum_{a\in p} r_a x_p^k \right) \tag{12}$$

$$y_k + \sum_{p\in P^k} x_p^k \geq 1 \qquad \forall k \in K, \tag{13}$$

$$-\sum_{k\in K} b_k \sum_{p\in P_a^k} x_p^k \geq -c_a \qquad \forall a \in A \setminus \tilde{A}, \tag{14}$$

$$-\sum_{b\in B_a} z_b \geq -1 \qquad \forall a \in \tilde{A}, \tag{15}$$

$$\sum_{b\in B_a^k} z_b - \sum_{p\in P_a^k} x_p^k \geq 0 \qquad \forall a \in \tilde{A}, \forall k \in K, \tag{16}$$

$$x_p^k \in \{0,1\} \qquad \forall k \in K, p \in P^k, \tag{17}$$

$$y_k \in \{0,1\} \qquad \forall k \in K, \tag{18}$$

$$z_b \in \{0,1\} \qquad \forall a \in \tilde{A}, b \in B_a. \tag{19}$$

Formulation (12)-(19) is a mixed of the arc-path formulations and (5)-(11). Indeed, if $\tilde{A} = \emptyset$, then Formulation (12)-(19) is equivalent to the arc-path formulation, and if $\tilde{A} = A$, it is equivalent to (5)-(11). Thus, for $\emptyset \subsetneq \tilde{A} \subsetneq A$, the formulation is a compromise between the quality of the bound of the linear relaxation and the time needed to solve it.

1) Pricing problem: The linear relaxation of (12)-(19) contains an exponential number of variables so it requires a column generation algorithm to be solved. The linear relaxation consists in replacing (17)-(19) by

$$x_p^k \geq 0 \qquad \forall k \in K, p \in P^k, \tag{20}$$

$$y_k \geq 0 \qquad \forall k \in K, \tag{21}$$

$$z_b \geq 0 \qquad \forall a \in \tilde{A}, b \in B_a. \tag{22}$$

Denote by λ_k, μ_a, $\nu_{a,k}$ the nonnegative dual variables corresponding to constraint (13) associated with $k \in K$, to constraint (14) or (15) associated with arc $a \in A$[1], and to constraints (16) associated with arc $a \in \tilde{A}$ and commodity $k \in K$, respectively.

There are two types of pricing problems:

- The first pricing problem is the one associated with each commodity $k \in K$. It consists in determining whether there exists a path $p \in P^k$ having a negative reduced cost. This reduces to compute an $s_k t_k$-shortest path in G with respect to arc cost vector w defined as:

$$w_a = \begin{cases} r_a + \sum_{k\in K} \nu_{a,k} & \text{if } a \in \tilde{A}, \\ r_a + \mu_a & \text{otherwise,} \end{cases} \qquad \text{for each arc } a \in A.$$

  If the shortest path has a cost less than λ_k, it is a negative reduced cost column. Otherwise, there is no such column in P^k. Since $w \geq \mathbf{0}$, this pricing problem can be solved in polynomial time using Dijkstra algorithm [5].
- The second pricing problem is the one associated with each arc $a \in \tilde{A}$. It consists in determining whether there exists a pattern having a negative reduced cost. It reduces to computing a binary knapsack problem [8] obtained by considering an item for each commodity $k \in K$ with profit $\nu_{a,k}$ and weight b_k, the knapsack having a capacity of c_a. If the profit of the computed knapsack is greater than μ_a, the variable z_b with $b \in B_a$ has a negative reduced cost and is added to the restricted master problem. Otherwise, there does not exist a negative reduced cost column in B_a.

2) Pricing and cutting plane algorithm to solve the linear relaxation: We implement a pricing and cutting plane algorithm to solve the linear relaxation of Formulation (12)-(19) in a similar way as done in [13]. The pricing step is used to generate paths and patterns and the cutting plane comes from the fact that we consider linking constraints (16) in a lazy way.

We suppose that the set of arcs $\tilde{A}$ is fixed. In the experiments, we try different strategies to fix $\tilde{A}$ and compare the results we obtain for these different strategies in terms of gap and computation time.

The initial restricted master problem has variable y_k and the variable x_p^k associated with the shortest path p of P^k, for each commodity $k \in K$. Only the inequalities (16) associated with arcs used by shortest paths are initially considered for all commodities. No variable z is included.

At each iteration, after solving the restricted master problem, the first pricing problem is solved. If no negative reduced cost commodity path is found, the second pricing problem is solved. If patterns with negative reduced costs are found, they are added to the current restricted master problem. Otherwise, separation of violated inequalities (16) is performed by enumeration. If a violated inequality is found, then all inequalities associated with the same arc and all the commodities are added to the current restricted master problem. Otherwise, the solution of the restricted master problem is optimal for the linear relaxation of (12)-(19).

3) Strategies for the choice of $\tilde{A}$: We consider four different values for the choice of $\tilde{A}$. The first two are the extremal cases, that is when $\tilde{A} = \emptyset$ and $\tilde{A} = A$. They will correspond to strategies *EMPTY* and *FULL* respectively. As a trade-off between bound quality and computational time, we also propose to compute $\tilde{A}$ as a preprocessing step. To this aim, we solve the linear relaxation of Formulation (1)-(4) that corresponds to the case $\tilde{A} = \emptyset$. Denoting by x^*, y^* the obtained optimal linear solution, we denote by w the remaining arc capacity vector associated with this fractional solution, that is, $w_a = c_a - \sum_{k\in K} b_k \sum_{p\in P_a^k} x_p^*$ for each arc $a \in A$. The *saturated strategy (SAT)*, consists in considering $\tilde{A}$ as the set of arcs having no remaining capacity, that is, $\tilde{A} = \{a \in A : w_a = 0\}$. The *CUT* strategy computes $\tilde{A}$ as follows. Let K_{frac} be the set of commodities having a path with a fractional value in x^*, y^*, that is, $K_{frac} = \{k \in K : \exists p \in P^k \text{ with } 0 < x_p^* < 1\}$. Order K_{frac} following the decreasing commodity bandwidths. For each $k \in K_{frac}$, compute a minimum weighted $s_k t_k$-cut with weights w in G^k, which is the graph obtained by removing the arcs of the previous computed cuts (following the commodity

[1]Since there is exactly one constraint amoung (14) or (15) for each arc $a \in A$, there is a unique dual variable $\mu \in \mathbb{R}^{\tilde{A}}$ associated with these two families of constraints.

order). Denoting by $\tilde{A}^k$ the arcs of the computed $s_k t_k$-cut, we define $\tilde{A}$ as the union of A^k, that is $\tilde{A} = \cup_{k \in K_{frac}} \tilde{A}^k$.

IV. Rounding procedure

In this section, we present a comparison between different rounding procedures. Each rounding procedure tries to find the best solution using for each commodity $k \in K$ only the paths $\mathcal{C}_k \subseteq P^k$ that have been generated during column generation.

The first considered rounding procedure is called randomized rounding where paths are selected with a probability depending on the optimal solution of the linear relaxation. The second rounding procedure is based on the feasibility PUMP rounding [6] that consists in finding a solution close as possible to the best infeasible rounded solution.

Randomized rounding

The aim of the randomized rounding procedure is to randomly select paths where the probability is proportional to the linear relaxation. Thus, at the end of the column generation loops if a variable associated with a path has a high value then the path is a good candidate for the heuristic solution.

Each iteration of the randomized rounding procedure is described in the Algorithm 1. It can be summarized as shuffling the commodities, from this ordering, randomly selecting a path for each commodity $k \in K$ where the probability is proportional to the linear relaxation value, that is, each path $p \in \mathcal{C}_k$ has a probability $\frac{x_p^k}{1-y_k}$ to be selected. Algorithm 1 is run several times until the time limit is reached. To this end, we return the best solution.

Algorithm 1: Randomized rounding algorithm

Data:
$\{\mathcal{C}_k\}$ collection of columns (or paths) for each commodity $k \in K$ with their associated linear relaxation.

Result: Integer solution P^*

$K' \leftarrow \sigma(K)$; `/* shuffle the ordering of the commodities */`

$c'_a \leftarrow c_a \quad \forall a \in A$;

for $k \in K'$ **do**
  while $\mathcal{C}_k \neq \emptyset$ **do**
    $p \leftarrow \gamma(\mathcal{C}_k)$; `/* randomly select a path according to the linear relaxation */`
    if $\min_{a \in p} c'(a) \geq b_k$ **then**
      $P^* \leftarrow P^* \cup \{p\}$;
      for $a \in p$ **do**
        $c'_a \leftarrow c'_a - b_k$
      end
      break;
    else
      remove p from $\mathcal{C}_k$;
    end
  end
end

Improved feasibility PUMP rounding

For the second approach, we derive a set of algorithms called PUMP from the feasibility pump rounding algorithm presented in [6]. We consider the same structure of the algorithm but we introduce a Randomized Rounding phase at each iteration with a time limit. Indeed preliminary results show that the original version of the feasibility PUMP method is outperformed by the randomized rounding method.

Algorithm 2 describes the Improved feasibility PUMP rounding algorithm steps, where $[x]$ is the vector obtained from x by rounding each component to its closest integer. Let CGA be the model associated with (12)-(16) and (20)-(22). We denote by $\Delta(CGA, \tilde{x})$ the CGA model where the objective function is replaced by

$$\min \sum_{k \in K} \Big((1 - \tilde{x}(y_k)) y_k + \tilde{x}(y_k)(1 - y_k) \Big) + \sum_{k \in K} \sum_{p \in P^k} \Big((1 - \tilde{x}(x_p^k)) x_p^k + \tilde{x}(x_p^k)(1 - x_p^k) \Big) \tag{23}$$

where $\tilde{x}(x_p^k)$ (resp. $\tilde{x}(y_k)$) is the value of the variable x_p^k (resp. y_k) in the vector $\tilde{x}$. The goal of this objective function is to find a solution close to the ideal infeasible rounding and respecting the constraints.

Algorithm 2: Improved feasibility PUMP rounding algorithm

Data:
$\{\mathcal{C}_k\}$ collection of columns (or paths) for each commodity k with their associated linear relaxation

Result: Integer solution P^*

$t \leftarrow 0$ and $x^* \leftarrow \arg\min\{CGA\}$;

if x^* *is integer* **then**
  return the associated paths
end

$\tilde{x} \leftarrow [x^*]$; `/* rounding of x* */`

while *time* $<$ *TimeLimit* **do**
  $x^* \leftarrow \arg\min\{\Delta(CGA, \tilde{x})\}$;
  if x^* *is integer* **then**
    return the associated paths;
  else
    run randomized rounding on x^* with a time limit of 2 seconds;
    if $\tilde{x} \neq [x^*]$ **then**
      $\tilde{x} \leftarrow [x^*]$;
    else
      change the $rand(T/2, 3T/2)$ entries of $\tilde{x}$ with highest $|x^* - \tilde{x}|$ values;
    end
  end
end

V. Experimental results

Instance description

We consider three sets of instances. The first set called *PRC* corresponds to IP-RAN networks [7]. There are three sizes of

	# nodes	# links	# commodities
Small PRC	71	1025.2	60
Middle PRC	1271	4996.8	300
Large PRC	5021	16160.4	600
N600	600	2400.0	1000
N800	800	3200.0	2000

TABLE I
DESCRIPTION OF INSTANCES PRC AND NX.

Topology	# nodes	# links	# commodities
cost266	37	57	1332
di-yuan	11	42	22
france	25	45	300
geant	22	36	462
nobel-eu	28	41	378
nobel-germany	17	26	121
nobel-us	14	21	91
pdh	11	34	24
polska	12	18	66
sun	27	102	67
ta1	24	55	396
ta2	65	108	1869

TABLE II
SNDLIB INSTANCES DESCRIPTION.

instances: small, medium, and large, and 5 instances for each size.

The second set of instances called *NX* corresponds to telecommunication networks more connected than the classical IP-RAN networks. We consider two instances of different sizes: N600 and N800. For each instance, we consider the original instance and three additional ones in which the link capacities are divided by 2, 3, and 5 respectively.

The third set of instances is SNDLib composed of 12 instances from [11]. Since routing costs are all unitary costs in these instances, we change them in the same way as described in [4].

The average number of nodes, arcs, and commodities for each instance are reported in Table I for *PRC* and *NX*, and in Table II for *SNDLib*.

Settings

We compute for each instance the linear relaxation of Formulation (12)-(19) with the strategies *EMPTY*, *FULL*, *SAT* and *CUT* for the choice of $\tilde{A}$. Each time, we fix a time limit of 240 seconds for computing for the column generation.

We can remark that, due to the time limit, the FULL strategy is not able to improve the Dual Bound for NX instances. Furthermore, CUT strategy allows us to improve the Dual Bound of 45% of instances which is the best improvement.

In Figure 1, we see the computational time for each strategy. Clearly, strategy FULL reaches the time limit of 240 seconds

Strategy	PRC	NX	SNDlib
FULL	4/15	0/8	7/12
SAT	5/15	2/8	8/12
CUT	5/15	2/8	9/12

TABLE III
NUMBER OF INSTANCES WHERE THE DUAL BOUND (LINEAR RELAXATION) IS IMPROVED W.R.T. THE BASELINE (EMPTY STRATEGY).

Strategy	PRC	NX	SNDlib
FULL-RR	0/15	0/8	1/12
SAT-RR	6/15	1/8	1/12
CUT-RR	0/15	0/8	0/12
EMPTY-PUMP	1/15	3/8	2/12
FULL-PUMP	0/15	0/8	1/12
SAT-PUMP	7/15	3/8	1/12
CUT-PUMP	1/15	3/8	1/12

TABLE IV
NUMBER OF INSTANCES WHERE THE PRIMAL BOUND IS IMPROVED W.R.T. THE BASELINE (EMPTY STRATEGY).

in a lot of instances. Methods SAT and CUT are a good tradeoff to provide a better dual bound. Furthermore, CUT approach provides a better computational time and a better improvement of the linear relaxation.

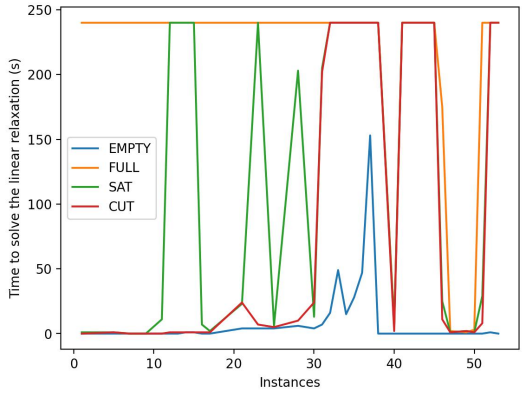

Fig. 1. Computational time for the linear relaxation

Now, we analyze the strength of the combination of partial double column generation with the two proposed rounding methods. For each rounding method, we consider a time limit of 100 seconds, and for the improved feasibility PUMP rounding, we set to 2 seconds the time limit of the internal randomized rounding. We compare the following combination of algorithms.

- EMPTY-RR: the baseline
- FULL-RR: Randomized Rounding after FULL column generation procedure.
- SAT-RR: Randomized rounding after SAT column generation procedure.
- CUT-RR: Randomized rounding after CUT column generation procedure.
- EMPTY-PUMP: our PUMP algorithm after EMPTY column generation procedure.
- FULL-PUMP: our PUMP algorithm after FULL column generation procedure.
- SAT-PUMP: our PUMP algorithm after SAT column generation procedure.
- CUT-PUMP: our PUMP algorithm after CUT column generation procedure.

Table IV provides the number of instances where the quality of the heuristic solution is better than the baseline EMPTY-

Strategy	PRC	NX	SNDlib
EMPTY-RR	0/15	0/8	0/12
FULL-RR	3/15	0/8	5/12
SAT-RR	9/15	0/8	2/12
CUT-RR	5/15	0/8	3/12
EMPTY-PUMP	0/15	0/8	0/12
FULL-PUMP	3/15	0/8	5/12
SAT-PUMP	10/15	0/8	1/12
CUT-PUMP	6/15	0/8	3/12

TABLE V
NUMBER OF INSTANCES WHERE THE GAP IS 0%.

Strategy	PRC	NX	SNDlib
FULL-RR	4/15	0/8	6/12
SAT-RR	11/15	1/8	7/12
CUT-RR	5/15	2/8	9/12
EMPTY-PUMP	1/15	3/8	2/12
FULL-PUMP	4/15	0/8	6/12
SAT-PUMP	12/15	3/8	5/12
CUT-PUMP	6/15	3/8	9/12

TABLE VI
NUMBER OF INSTANCES WHERE THE GAP IS IMPROVED W.R.T. THE BASELINE (EMPTY STRATEGY).

RR. The PUMP version of the algorithm allows for improving the quality of the heuristic for the PRC and NX instances. The SAT strategy of the partial double decomposition is the best one to improve the quality of the heuristic solution.

The gap presented in Table V and Table VI is computed as follows, $|\frac{\text{Heuristic solution - Linear relaxation}}{\text{Heuristic solution}}|$.

Table V shows the number of instances where the gap is closed, i.e., equal to 0. The algorithm closing the gap for a maximum of the tested instances is SAT-PUMP which closes the gap on more than 30%. We remark that the conclusion on the algorithm performance are the same as for Table IV.

Table VI presents the number of instances where the gap is improved in comparison with the Baseline. CUT strategy is better for improving the dual bound and SAT strategy is better for improving the heuristic solution. This table shows that to reduce the gap the SAT strategy with PUMP rounding method is the better one.

VI. CONCLUSION

In this paper, we have investigated the Multi-Commodity flow problem. We first have presented an arc-path formulation for the problem where the decision variable consists in selecting a path for each commodity. Then, we have applied the double Dantzig-wolfe decomposition, as in an existing paper from the literature, to strengthen the linear relaxation by introducing new arc pattern variables that are also exponential in number. In this paper, we introduced a slightly different formulation by considering patterns only on a subset of links. Different types of subsets are analyzed. Moreover, we compare different rounding methods that allowed us to obtain integer solutions from the linear relaxation. By combining the approaches, we obtained good performances where we closed the gap for 31% of instances and improved the gap for 57% of instances.

REFERENCES

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Network flows: theory, algorithms and applications. *Network*, 1:864, 1993. ISBN: 013617549X.

[2] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Integer multicommodity flow problems. In William H. Cunningham, S. Thomas McCormick, and Maurice Queyranne, editors, *Integer Programming and Combinatorial Optimization*, pages 58–71, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[3] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. *Operations Research*, 48(2):318–326, 2000.

[4] Amal Benhamiche, Morgan Chopin, and Sébastien Martin. Unsplittable shortest path routing: Extended model and matheuristic. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 926–931. IEEE, 2023.

[5] Edsger W Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022.

[6] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104:91–104, 2005.

[7] Nicolas Huin, Jérémie Leguay, Sébastien Martin, and Paolo Medagliani. Routing and slot allocation in 5g hard slicing. *Computer Communications*, 2023.

[8] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[9] Miguel Pineda Martín and Sébastien Martin. Unsplittable multicommodity flow problem via quantum computing. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 385–390. IEEE, 2023.

[10] M Yassine Naghmouchi, Shoushou Ren, Paolo Medagliani, Sébastien Martin, and Jérémie Leguay. Optimal admission control in damper-based networks: Branch-and-price algorithm. In *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 488–493. IEEE, 2023.

[11] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. Sndlib 1.0—survivable network design library. *Networks*, 55(3):276–286, 2010.

[12] Kyungchul Park, Seokhoon Kang, and Sungsoo Park. An Integer Programming Approach to the Bandwidth Packing Problem. *Management Science*, 42(9):1277–1291, 1996.

[13] Sungsoo Park, Deokseong Kim, and Kyungsik Lee. An integer programming approach to the path selection problems. In *Proceedings of the International Network Optimization Conference INOC, Evry-Paris, France*, pages 448–453, 2003.

[14] Mark Parker and Jennifer Ryan. A column generation algorithm for bandwidth packing. *Telecommunication Systems*, 2(1):185–195, December 1993.

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

[16] Jiachen Zhang, Youcef Magnouche, Pierre Bauguion, Sebastien Martin, and J. Christopher Beck. Computing bipath multicommodity flows with constraint programming–based branch-and-price-and-cut. *INFORMS Journal on Computing*, 2024.