

# Comparing branch-and-price algorithms for the Multi-Commodity k-splittable Maximum Flow Problem

**Gamst, Mette; Petersen, Bjørn**

*Published in:*
European Journal of Operational Research

*Link to article, DOI:*
10.1016/j.ejor.2011.10.001

*Publication date:*
2012

# Comparing branch-and-price algorithms for The Multi-Commodity $k$-splittable Maximum Flow Problem

M. Gamst* and B. Petersen◇
DTU Management Engineering, Produktionstorvet 426, DK-2800 Kgs. Lyngby, Denmark
* gamst@man.dtu.dk, ◇ bjop@man.dtu.dk

**Abstract**

The Multi-Commodity $k$-splittable Maximum Flow Problem consists in routing as much flow as possible through a capacitated network such that each commodity uses at most $k$ paths and the capacities are satisfied. The problem appears in telecommunications, specifically when considering *Multi-Protocol Label Switching*. The problem has previously been solved to optimality through branch-and-price. In this paper we propose two exact solution methods both based on an alternative decomposition. The two methods differ in their branching strategy. The first method, which branches on forbidden edge sequences, shows some performance difficulty due to large search trees. The second method, which branches on forbidden and forced edge sequences, demonstrates much better performance. The latter also outperforms a leading exact solution method from the literature. Furthermore, a heuristic algorithm is presented. The heuristic is fast and yields good solution values.

**Keywords**: *Branch and bound; Combinatorial optimization; Multi-commodity flow; k-Splittable; Dantzig-Wolfe decomposition; Heuristic*

## 1 Introduction

The Multi-Commodity $k$-splittable Maximum Flow Problem (MC$k$MFP) consists in maximizing the amount of routed flow through a capacitated network such that each commodity uses at most $k$ paths and the capacities are satisfied. The MC$k$MFP appears in telecommunications, specifically when considering *Multi-Protocol Label Switching* (MPLS). In MPLS, several data packets are gathered under a single label in order to limit the size of the routing tables and to increase the quality of data transmission. Also, encapsulating packets of different network protocols and only considering the labels eliminates the need for the network to support several data link layer technologies. The cost of sending data increases with the number of *Label Switch Paths* (LSP). By limiting the number of used labels (i.e. paths) the total cost can be reduced. However, we must still ensure that all or as much data as possible is transmitted. This corresponds to the MC$k$MFP; given an upper bound on the number of paths to use, we try to maximize the total throughput in the network. See Evans and Filsfils [7] for more details on the MPLS.

When $k = 1$ the MC$k$MFP collapses to the Multi-Commodity unsplittable Maximum Flow Problem for which specialized algorithms exists, see e.g. Alvelos and de Carvalho [1], Barnhart et al. [3] and Kleinberg [9]. We thus assume that $k > 1$.

The Multi-Commodity $k$-splittable Flow Problem (MC$k$FP) was presented by Baier et al. [2], who solved the Maximum Budget-Constrained Single- and Multi-Commodity $k$-splittable Flow Problems using approximation algorithms. The authors proved that the Maximum Single-Commodity $k$-splittable Flow Problem is $\mathcal{NP}$-hard in the strong sense for directed graphs. Finally, they noted that for $k \geq |E|$, a $k$-splittable $(s,t)$ flow problem degenerates to an ordinary $(s,t)$ flow problem.

Koch et al. [11] proved that the MC$k$MFP is $\mathcal{NP}$-hard in the strong sense for directed as well as undirected graphs, and showed that when $\mathcal{P} \neq \mathcal{NP}$, the best possible approximation factor is $\frac{5}{6}$. Koch et al. [10] considered the MC$k$MFP as a two-stage problem, where the first stage consists of the decision on the $k$ paths (routing) and the second of the amount of flow on the paths (packing). If $k$ is a constant then it suffices to consider a polynomial number of packing alternatives, which can be constructed in polynomial time. If $k$ is part of the input, they proposed an approximation algorithm having approximation factor $(1-\epsilon)$, $\epsilon > 0$.

Martens and Skutella [14] considered a variant of the MC$k$FP with extra constraints on the amount of flow on paths and where the objective was to minimize the network congestion. They showed that when reducing the problem to an unsplittable flow problem, only a constant factor is lost in the performance ratio.

Kolliopoulos [12] considered the single-source Minimum Cost 2-splittable flow problem with budget constraints and with the assumption that the minimum edge capacity is larger than the maximum commodity demand. The author presented an approximation algorithm with factor $(2,1)$. This result was generalized to the $k$-splittable problem by Salazar and Skutella [15] with a resulting approximation factor of $(1+\frac{1}{k}+\frac{1}{2k-1},1)$.

Caramia and Sgalambro [5] proposed a heuristic for the Maximum Concurrent $k$-splittable Flow Problem, where commodities are first routed using an augmenting path algorithm and then a local search routine re-routes part of the paths. The solution quality of the heuristic was shown to increase with the size of $k$.

Truffot and Duhamel [16] used branch-and-price to solve the Single-Commodity $k$-splittable Maximum Flow Problem (SC$k$MFP). A 3-index edge-path model and a corresponding branch-and-price algorithm were presented. The pricing problem for the column generation is a shortest path problem solvable in polynomial time. Furthermore, Truffot et al. [17] applied their 3-index branch-and-price algorithm to the MC$k$MFP.

Truffot et al. [18] also considered a non-linear variant of the MC$k$FP with end-to-end delay bounds on each path and quality of service (QoS) requirements. The problem was solved using branch-and-price. The results showed that QoS requirements and CPU time are correlated and that minimizing the delay on edges improves the solution value and decreases the computation time.

Gamst et al. [8] used branch-and-price to solve the Minimum Cost Multi-Commodity $k$-splittable Flow Problem (MCMC$k$FP). They applied the algorithm of Truffot et al. [17] to the MCMC$k$FP and proposed a new branch-and-price algorithm based on a 2-index model. The latter showed very good performance and outperformed the existing branch-and-price algorithm.

A different exact solution approach from the literature consists of solving an edge-based formulation of the Maximum Concurrent $k$-splittable Flow Problem in a branch-and-bound scheme, see Caramia and Sgalambro [4]. Branching fixes usage of edges and bounding consists of solving the LP-relaxed edge-based formulation with branching constraints on used edges. The approach outperformed standard MIP-solvers, but suffered from large branch-and-bound trees.

The MC$k$MFP can be represented by a directed graph $G = (V, E, L)$, where $V$ is the set of vertices and $E$ the set of edges. A positive capacity $u_e$ is associated with every edge $e \in E$. The set of commodities is denoted $L$ and each commodity $l \in L$ has a source $s_l \in E$ and a destination $t_l \in E$. The maximal number of routes each commodity may use is denoted $k$.

In this paper three exact solution methods for the MC$k$MFP are presented and compared. The 3-index branch-and-price algorithm (3BP) by Truffot et al. [17] is extended with a heuristic proposed by Gamst et al. [8] to reach feasible solutions faster. The extended 3BP is compared to two algorithms based on a 2-index branch-and-price formulation applied to the MC$k$MFP by Gamst et al. [8]. The two algorithms for the MC$k$MFP differ in their branching schemes. The first algorithm (2BP) uses a branching strategy from the literature where certain subpaths are forbidden, and the second algorithm (2BP+B) uses a new branching strategy where the use of certain paths is either forced or forbidden and where branch cuts are added to the master problem.

The main contribution of this paper is to apply the 2BP algorithm to the MC$k$MFP and especially to introduce the new branching scheme and the branch cuts of the 2BP+B algorithm. Furthermore, a heuristic use of the 2BP and 2BP+B algorithms is presented.

The paper is organized as follows. In Section 2 we summarize and extend methods from the literature. The 3BP algorithm is extended with a heuristic in Section 2.1 and the 2BP algorithm is presented in Section 2.2. Then in Section 3 the new algorithm 2BP+B is proposed. This is followed by Section 4, which transforms the exact methods into heuristics. All algorithms are compared in Section 5. Section 6 gives final conclusions. Note that throughout this paper, we refer to a "restricted master problem" simply as a "master problem".

## 2 Branch-and-price algorithms from the literature

In this section we briefly introduce two exact algorithms for the MC$k$MFP from the literature. First the 3-index branch-and-price algorithm (3BP) by Truffot et al. [17] is presented. The 3BP motivates the need for the 2-index branch-and-price algorithm (2BP), which was originally applied to the MCMC$k$FP.

### 2.1 The 3-index branch-and-price algorithm (3BP)

Truffot et al. [17] solved the MC$k$MFP by applying Dantzig-Wolfe decomposition [6], such that the pricing problem generates paths for each commodity and the master problem merges the paths into an overall solution. We denote their branch-and-price algorithm 3BP.

Let $L$ be the set of commodities and $h \in \{1, \ldots, k\}$ be a path index indicating the first, second, ..., $k$th path used by a commodity. A generated path $p$ for commodity $l$ travels from $s_l$ to $t_l$, has capacity $u_p = \min_{e \in p} u_e$, and is kept in the set $P^l$. In the master problem, the variable $x_p^{hl} \geq 0$ denotes the amount of flow on path $p$ for the $h$th path of commodity $l$ and the binary variable $y_p^{hl}$ denotes whether or not path $p$ is used as the $h$th path for commodity $l$. Furthermore, if $\delta_e^p = 1$ then path $p$ travels on edge $e$, otherwise $\delta_e^p = 0$. The 3BP master

problem is:

$$\max \quad \sum_{l\in L}\sum_{h=1}^{k}\sum_{p\in P^l} x_p^{hl}$$

$$\text{s.t.} \quad \sum_{l\in L}\sum_{h=1}^{k}\sum_{p\in P^l} \delta_e^p x_p^{hl} \le u_e \quad \forall e \in E \tag{1}$$

$$x_p^{hl} - u_p y_p^{hl} \le 0 \qquad \forall l \in L, h \in \{1,\dots,k\}, \forall p \in P^l \tag{2}$$

$$\sum_{p\in P^l} y_p^{hl} \le 1 \qquad \forall l \in L, h \in \{1,\dots,k\} \tag{3}$$

$$x_p^{hl} \ge 0 \qquad \forall l \in L, h \in \{1,\dots,k\}, \forall p \in P^l$$

$$y_p^{hl} \in \{0,1\} \qquad \forall l \in L, h \in \{1,\dots,k\}, \forall p \in P^l$$

The objective function maximizes the total amount of routed flow. Constraints (1) ensure that edge capacities are satisfied. Constraints (2) force the decision variable $y_p^{hl}$ to be set to one if there is flow on the corresponding path $x_p^{hl}$. Constraints (3) ensure that at most one path is used as the $h$th path of a commodity $l$. The path index $h \in \{1,\dots,k\}$ causes symmetry in the solution space, hence a symmetry-breaking constraint is added to the formulation:

$$\sum_{p\in P^l} x_p^{h+1,l} - \sum_{p\in P^l} x_p^{hl} \le 0 \quad \forall h \in \{1,\dots,k-1\}, \forall l \in L \tag{4}$$

The constraint forces the order of selected paths to be such that the corresponding amount of path flow is non-increasing. This eliminates many identical solutions, but does not prevent symmetric solutions where paths carry the same amount of flow. The 3-index model is LP-relaxed and then the model is simplified by substituting $x_p^{hl}/u_p$ for $y_p^{hl}$, which is feasible according to constraints (2) and (3) and to the fact that $u_p > 0$. Constraints (2) and the bounds on the $y_p^{hl}$ variables are removed from the formulation and constraints (3) are rewritten as:

$$\sum_{p\in P^l} \frac{x_p^{hl}}{u_p} \le 1 \quad \forall l \in L, h \in \{1,\dots,k\} \tag{5}$$

Let $\pi \ge \mathbf{0}$ be the dual variables of constraints (1), $\lambda \ge \mathbf{0}$ the dual variables of constraints (5), and $\omega \ge \mathbf{0}$ the dual variables of constraints (4). Furthermore, let $\bar{\omega}^{hl} = -\omega^{hl}, h = 1$, $\bar{\omega}^{hl} = -\omega^{hl} + \omega^{(h-1)l}, h = 2,\dots,k-1$, and $\bar{\omega}^{hl} = \omega^{(h-1)l}, h = k$. The reduced cost of a path $p \in P^l$ with path index $h \in \{1,\dots,k\}$ for a commodity $l \in L$ is:

$$c_p^{hl} = 1 - \sum_{e\in E} \delta_e^p \pi_e - \frac{\lambda^{hl}}{u_p} - \bar{\omega}^{hl} \ge 0$$

The pricing problems generate columns for each commodity $l$ and path index $h$ with positive reduced cost, i.e., where the following holds true:

$$\sum_{e\in E} \delta_e^p \pi_e < 1 - \frac{\lambda^{hl}}{u_p} - \bar{\omega}^{hl}$$

The dual variables $\bar{\omega}^{hl}$ and $\lambda^{hl}$ are thus constants, while the path capacity $u_p$ is not known until the path has been generated. The pricing problem fixes the value of $u_p$, and then tries to generate a column with positive reduced cost for commodity $l$ by solving a shortest path problem from $s_l$ to $t_l$ with edge weights $\pi_e, \forall e \in E : u_e \geq u_p$. The path capacity $u_p$ can be fixed to at most $|E|$ different values (one for each different $u_e : e \in E$), hence the pricing problem can be solved in polynomial time by considering at most $|E|$ shortest path problems. The algorithm for solving the pricing problem is illustrated in Algorithm 1, where $u_p$ is given by *cap*.

**Algorithm 1** Algorithm for solving the pricing problem.

1: $CAP \leftarrow$ a list consisting of edge capacities, $u_e$, sorted in increasing order and without duplicates
2: **for** (each $cap \in CAP$) **do**
3: $\quad E' \leftarrow E \backslash \{e : u_e < cap\}$
4: $\quad$ Solve shortest path problem from $s_l$ to $t_l$ on edges $e \in E'$ with edge weights $\pi_e$
5: $\quad$ Save path, if its reduced cost is positive
6: **end for**

Gamst et al. [8] applied the 3BP algorithm to the MCMC$k$FP and improved the 3BP algorithm by including a heuristic, which merges certain fractional columns such that a feasible solution was possibly reached. Specifically, one of the following two situations may occur:

1. For a commodity, several identical paths are used but with different values of $h$. In this case, the heuristic merges the paths into one single path.

2. More than one path is used for a single value of $h$ for a commodity. Here, the heuristic assigns a unique value of $h$ to each path, if possible.

The heuristic improved the 3BP algorithm by reducing the size of the branch-and-bound tree for solved instances and by improving the bounds of unsolved instances [8].

The heuristic is included just before branching, i.e., whenever a fractional lower bound is found in a branch-and-bound node. This gives us the final 3BP algorithm.

## 2.2 The 2-index branch-and-price algorithm (2BP)

The MC$k$MFP is Dantzig-Wolfe decomposed such that the variables of the resulting master problem do not contain the $h$-index. This decomposition gives a pricing problem, which generates a path for each commodity, and a master problem, which merges the paths into an overall feasible solution.

Recall the notation introduced for the 3BP algorithm. Furthermore, let $x_p^l \geq 0$ denote the amount of flow on path $p$ for commodity $l$ and let $y_p^l \in \{0, 1\}$ denote whether or not path $p$ is

used by commodity $l$. The master problem is:

$$\max \quad \sum_{l \in L} \sum_{p \in P^l} x_p^l \tag{6}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{p \in P^l} \delta_e^p x_p^l \leq u_e \quad \forall e \in E \tag{7}$$

$$x_p^l - u_p y_p^l \leq 0 \qquad \forall l \in L, \forall p \in P^l \tag{8}$$

$$\sum_{p \in P^l} y_p^l \leq k \qquad \forall l \in L \tag{9}$$

$$x_p^l \geq 0 \qquad \forall l \in L, \forall p \in P^l$$

$$y_p^l \in \{0,1\} \qquad \forall l \in L, \forall p \in P^l$$

The objective function maximizes the total amount of routed flow. Constraints (7) ensure edge capacities are never violated and constraints (8) force the decision variables to take on value 1, whenever the amount of flow on the corresponding path is positive. Constraints (9) limit the number of paths for commodity $l$ to at most $k$.

The binary variables $y_p^l$ are LP-relaxed and $y_p^l$ is replaced by $x_p^l/u_p$, which is feasible according to constraints (8) and (9). Constraints (8) and the bounds on $y_p^l$ are redundant and thus removed from the formulation. Furthermore, constraints (9) are reformulated to:

$$\sum_{p \in P^l} \frac{x_p^l}{u_p} \leq k \quad \forall l \in L \tag{10}$$

Let $\pi \geq \mathbf{0}$ and $\lambda \geq \mathbf{0}$ be the dual variables for constraints for (7) and (10). The reduced cost of a path $p \in P^l$ for a commodity $l \in L$ is:

$$c_p^l = 1 - \sum_{e \in E} \delta_e^p \pi_e - \frac{\lambda^l}{u_p}, \quad \forall l \in L,\ \forall p \in P^l \tag{11}$$

Only paths with positive reduced cost will be considered, i.e., we seek to generate paths for each commodity $l$ where

$$\sum_{e \in E} \delta_e^p \pi_e < 1 - \frac{\lambda^l}{u_p}, \quad \forall l \in L,\ \forall p \in P^l$$

The dual variable $\lambda^l$ is considered a constant, while the path capacity $u_p$ is not known until the path has been generated. The pricing problem is solved using Algorithm 1, which runs in polynomial time.

**Branching scheme – forbidding edge sequences**

The branching scheme forbids edge sequences. Let the divergence vertex for a commodity be defined as the first vertex where at most one incoming edge is visited and several outgoing edges are visited by the commodity. If the number of paths emanating from the divergence vertex for some commodity $l$ is greater than $k$ then branching can be applied.

We consider the set of all paths with positive flow $\bar{P}^l$ of commodity $l$. Let $q(p)$ be the subpath of $p \in \bar{P}^l$ starting at the divergence vertex. Also, let a prefix path of $q(p)$ be a subpath of $q(p)$ beginning in the source of $q(p)$. Denote $s(p)$ the smallest prefix path of $q(p)$ such that $s(p)$ is no prefix of any other subpath $q(p')$, $p' \in \bar{P}^l : p' \neq p$. Now, $s(p)$ is the unique edge sequence for $p$. Each unique edge sequence is forbidden in a branching child. The unique edge sequences are identified by enumerating and comparing the subpaths of all $p \in \bar{P}^l$.

The number of branching children is kept equal to $k+1$, so if more than $k+1$ paths emanate from the divergence vertex, then we let some branching children forbid more than one unique edge sequence. The reason for this is to reduce the width of the search tree. A branching child can forbid several unique edge sequences, because all combinations of $k$ paths from the divergence vertex are available in at least one other branching child.

An illustration of the branching strategy is seen in Figure 1. A graph with four vertices is given and a single commodity with source $s$ and destination $t$ is to be routed using at most two paths. In the current solution three paths are used: $p_1 = (e_1, e_4, e_5), p_2 = (e_1, e_3, e_5)$, and $p_3 = (e_2, e_3, e_5)$. Assume that the optimal solution consists of path $p_1$ and $p_3$. The divergence vertex is $s$ and $k+1$ subpaths emanating from $s$ are found: $(e_1, e_3)$, $(e_1, e_4)$ and $(e_2)$. The optimal solution is found in the branching child, which forbids the use of edge sequence $(e_1, e_3)$.

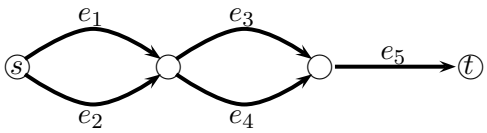


Figure 1: A graph used to illustrate the branching scheme. The graph consists of four vertices, the source vertex is denoted $s$, and the destination vertex $t$. Edges are $e_1, e_2, e_3, e_4$, and $e_5$.

The branching scheme changes the pricing problem. When solving the shortest path problem we need to ensure that we do not use the forbidden edge sequences. The shortest path problem with forbidden paths is a polynomial problem and can be solved by applying a shortest path algorithm on an extended graph, see Villeneuve and Desaulniers [19].

# 3 New branching scheme for the 2BP

In this section we present the new branch-and-price algorithm 2BP+B. The 2BP+B algorithm only differs from the 2BP algorithm in the branching scheme. The master problem (6),(7) and (10) is the same and the reduced cost is given by (11).

This branching scheme is based on the idea of forbidding or forcing the use of a certain path $p'$ for a fixed commodity $l \in L$. This corresponds to setting $y^l_{p'} = 0$ or $y^l_{p'} = 1$, respectively, in the non-relaxed master problem. In the remainder of this section a fixed commodity $l \in L$ is assumed.

The effect of the branching scheme on the non-relaxed master problem, specifically constraint (9) is considered. Both when $y^l_{p'} = 0$ and $y^l_{p'} = 1$, the variable can be left out of the

constraint. If $y^l_{p'} = 1$ the constraint is rewritten as:

$$\sum_{p \in P^l \setminus \{p'\}} y^l_p \leq k - 1$$

Now, the effect of the branching scheme on the relaxed master problem, specifically constraint (10) is considered. When path $p'$ is forbidden for commodity $l$ then $x^l_{p'} = 0$. When the use of path $p'$ is forced then we set $x^l_{p'} \geq \epsilon$, where $\epsilon > 0$ is a suitably small number. Constraint (10) is rewritten as:

$$\sum_{p \in P^l \setminus \{p'\}} \frac{x^l_p}{u_p} \leq k - 1 \tag{12}$$

This is stronger than the original constraint when $x^l_{p'} < u_{p'}$, hence the bound of the branching child is strengthened in this case.

The number of branching children depends on the current fractional solution. Assume that the current solution consists of $k + \alpha, \alpha > 0$ paths for commodity $l$. At least $\alpha + 1$ paths in the current fractional solution have $x^l_p < u_p$, otherwise constraint (10) would be violated. Let $\bar{P}^l$ denote these paths. An optimal solution may not include any of the paths in $\bar{P}^l$, hence we propose the following branching strategy. Generate $|\bar{P}^l| + 1$ branching children. Usage of each path in $\bar{P}^l$ is respectively forced in the first $|\bar{P}^l|$ children. In the last child, usage of any path in $\bar{P}^l$ is forbidden.

The first $|\bar{P}^l|$ children cause symmetry in the solution space. Consider an example with $|\bar{P}^l| + 1 = 4$ branching children $b_1$, $b_2$, $b_3$ and $b_4$, forcing usage of path $p_1$, $p_2$, and $p_3$, and forbidding all paths $p_1$, $p_2$, and $p_3$, respectively. Now, a solution containing paths $p_1$ and $p_3$ can be generated in both the subtree of branching child $b_1$ and $b_3$.

To prevent such symmetry in the branching children, the branching strategy is changed into letting the first $|\bar{P}^l|$ children force and forbid usage of certain paths. Recall the previous example. Child $b_1$ still forces the use of $p_1$. Child $b_2$ forces the use of $p_2$ and forbids the use of $p_1$. Similarly, child $b_3$ forces the use of $p_3$ and forbids the use of $p_1$ and $p_2$. In this way, the solution using $p_1$ and $p_3$ is only available in the subtree of $b_1$.

In practice we would rather add a cut than rewrite constraints (10) when the use of a path is forced. Constraints (10) are thus always kept in the master problem, even if stronger constraints are added due to the branching strategy. Recall inequality (12) when forcing the use of path $p'$. This inequality is denoted the branch cut. Branching children which force the usage of paths, have the corresponding branch cuts added to their master problem. The set of branch cuts for commodity $l$ is denoted $B^l$. Let $\omega_{bl} \geq 0$ be the dual of branch cut $b \in B^l$ for commodity $l$. The resulting reduced cost for path $p \in P^l$ for commodity $l \in L$ is:

$$c^l_p = 1 - \sum_{e \in E} \delta^e_p \pi_e - \frac{\lambda_l}{u_p} - \sum_{b \in B^l} \frac{\omega_{bl}}{u_p} = 1 - \sum_{e \in E} \delta^e_p \pi_e - \frac{\lambda_l + \sum_{b \in B^l} \omega_{bl}}{u_p} \tag{13}$$

When solving the pricing problem for commodity $l$ the dual costs $\omega_{bl}$ are constants. Hence the branch cut does not affect the structure of the pricing problem. The pricing problem must, however, be able to avoid using forbidden paths as for the 2BP algorithm.

# 4 Heuristic approach

The presented exact algorithms can be used as heuristics by only computing the root node and then returning the best feasible solution. The approach of only computing the root node does not guarantee a polynomial running time, since an exponential number of columns potentially needs to be added in the root. In practice, however, we expect low running times.

The heuristic usage of the 3BP algorithm is denoted 3HEUR. Because no branching occurs the heuristic usage of the 2BP and the 2BP+B algorithms is identical and is denoted 2HEUR. Truffot and Duhamel [16] argue that the 3-index and 2-index formulations are equivalent, also after LP-relaxation and elimination of the binary variables. Even though the formulations give the same bounds, we may not reach the same feasible solutions in the root node. Hence we investigate the performance of 3HEUR and 2HEUR empirically.

The 2HEUR may give infeasible solutions where more than $k$ paths are used for each commodity. In this case we try to move the flow between the paths in order to find a feasible solution using at most $k$ paths for each commodity. The approach takes as input $\{G, P, X\}$, where $G$ is the graph representing the problem instance, $P = \bigcup_{l \in L} P^l$ is the set of paths in the current fractional solution with $x_p > 0$, and $X$ is the set of non-zero variables in the current fractional solution, i.e., $x_p > 0, \forall x_p \in X$. The steps of the approach are seen in Algorithm 2. In line 1, an empty solution is initialized. Then, for each commodity $l$, the approach first

**Algorithm 2** Flow-moving approach with $\{G, P, X\}$ as input

```
1:  Initialize empty solution sol ← ∅
2:  for (each l ∈ L) do
3:     P̄^l ← list of p ∈ P^l ⊆ P sorted according to decreasing u_p : u_p ← min_{e∈p} u_e
4:     for (each p ∈ P̄^l with min_{e∈p} u_e > 0) do
5:        x̄^l_p ← min_{e∈p} u_e
6:        u_e ← u_e − x̄^l_p, ∀e ∈ p
7:        sol ← {p, x̄^l_p}
8:        if (Flow is assigned to k paths for commodity l) then
9:           break
10:       end if
11:    end for
12: end for
13: Return solution sol
```

sorts path for the commodity according to decreasing path capacity. Then for each path in the sorted list, the approach identifies the amount of flow that can be sent on the path (line 5), subtracts this flow from the edges of the path (line 6) and stores the path and its flow in the solution (line 7). If $k$ paths are stored for commodity $l$ the approach considers the next commodity (line 9-11). When all commodities are considered, the approach returns the generated solution (line 13). Note that the order of paths in $\bar{P}^l$ does not change in line 4-11, even though edge capacities are changed in line 6.

The flow-moving approach has time complexity $\mathcal{O}(|L|(|P| \log |P| + |P||E|))$, where $P = \cup_{l \in L} P^l$. This is polynomial in the input size of the approach. Including this flow-moving approach in 2HEUR gives the final heuristic denoted 2HEUR+FM.

| Name | $\|V\|$ | $\|E\|$ | $\|L\|$ |
|---|---|---|---|
| Random5-35 | 5 | 35 | 1 |
| Random10-45 | 10 | 45 | 1 |
| Random15-60 | 15 | 60 | 1 |
| Random20-140 | 20 | 140 | 1 |
| tg10-2 | 12 | 40 | 1 |
| tg20-2 | 22 | 80 | 1 |
| tg40-1 | 42 | 154 | 1 |
| tg40-5 | 42 | 154 | 1 |
| tg80-1 | 82 | 322 | 1 |
| tg100-2 | 102 | 400 | 1 |
| Random10-40 | 10 | 40 | 3 |
| Random11-42 | 11 | 42 | 11 |
| Random20-80 | 20 | 80 | 20 |
| Random22-56 | 22 | 56 | 22 |

Table 1: Sizes of test instances. First column denotes the instance name, then follows the number of vertices, the number of edges, and finally the number of commodities.

# 5 Computational results

A computational evaluation is performed on a dual 2.66GHz Intel® Xeon® X5355 machine with 16 GB of RAM. Note that CPU times in the following stem from using one core only.

We have tested the proposed algorithms; the 3BP extended with the heuristic to reach feasible solutions faster, the 2BP, the 2BP+B, and the three heuristics: 3HEUR, 2HEUR, and 2HEUR+FM. We implemented all algorithms using the COIN framework [13] with `ILOG CPLEX 10.2` as LP-solver. Computations concerning the selection of branching candidates and branching children are handled by COIN.

The solution methods are tested on benchmark instances from the literature [16]: the `Random` instances are randomly generated and the `tg` instances are generated by the Transit Grid generator[1] using topologies from transportation networks. See Table 1 for details.

Two different types of tests have been performed. First the three exact algorithms are computationally evaluated on the described instances and results are compared. Then we test the performance of the heuristics, 3HEUR, 2HEUR and 2HEUR+FM, to investigate their heuristic solution values and running times.

## 5.1 Exact approach

The three algorithms are computationally evaluated on the described instances. Results for the single-commodity `Random` instances are summarized in Table 2 and results for the single-commodity `tg` instances are summarized in Table 3. The multi-commodity instances are all of the `Random` type and results are summarized in Table 4.

In the tables the first column holds the name of the problem instance, the second column holds the value of $k$ and the third column holds the optimal value. Then follows the size and depth of the search tree, the number of generated variables, the gap in percent between the upper and lower bound, and the time in seconds spent on solving the instance for the 3BP, the 2BP, and the 2BP+B algorithms, respectively. If a test run is marked with "-" then it has run out of memory. If the gap is also marked with "-" then no lower bound was found. The

[1] `http://www.informatik.uni-trier.de/~naeher/Professur/research/generators/maxflow/tg/index.html`

| | | | 3BP | | | | | 2BP | | | | | 2BP+B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | **k** | **z** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** |
| Random5-35 | 1 | 66 | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** |
| | 2 | 128 | 1 | 0 | 14 | 0.00% | **0.00** | 1 | 0 | 7 | 0.00% | **0.00** | 1 | 0 | 7 | 0.00% | **0.00** |
| | 3 | 182 | 1 | 0 | 27 | 0.00% | 0.01 | 1 | 0 | 9 | 0.00% | **0.00** | 1 | 0 | 9 | 0.00% | **0.00** |
| | 4 | 223 | 13 | 6 | 48 | 0.00% | 0.01 | 1 | 0 | 12 | 0.00% | **0.00** | 1 | 0 | 12 | 0.00% | **0.00** |
| | 5 | 262 | 19 | 9 | 60 | 0.00% | 0.03 | 1 | 0 | 12 | 0.00% | **0.00** | 1 | 0 | 12 | 0.00% | **0.00** |
| | 6 | 297 | 21 | 10 | 78 | 0.00% | 0.03 | 1 | 0 | 14 | 0.00% | 0.01 | 1 | 0 | 14 | 0.00% | **0.00** |
| | 7 | 326 | 67 | 12 | 98 | 0.00% | 0.10 | 1 | 0 | 14 | 0.00% | **0.00** | 1 | 0 | 14 | 0.00% | **0.00** |
| | 8 | 326 | 1 | 0 | 104 | 0.00% | **0.00** | 1 | 0 | 11 | 0.00% | 0.01 | 1 | 0 | 11 | 0.00% | **0.00** |
| Random10-45 | 1 | 73 | 1 | 0 | 6 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** |
| | 2 | 142 | 5 | 2 | 15 | 0.00% | **0.01** | 4 | 1 | 9 | 0.00% | **0.01** | 8 | 2 | 9 | 0.00% | **0.01** |
| | 3 | 209 | 9 | 3 | 33 | 0.00% | **0.02** | 21 | 3 | 15 | 0.00% | 0.03 | 20 | 3 | 12 | 0.00% | **0.02** |
| | 4 | 260 | 45 | 17 | 68 | 0.00% | 0.08 | 411 | 12 | 24 | 0.00% | 0.56 | 34 | 4 | 20 | 0.00% | **0.03** |
| | 5 | 306 | 369 | 22 | 102 | 0.00% | 0.80 | 23599 | 18 | 34 | 0.00% | 44.96 | 40 | 4 | 20 | 0.00% | **0.07** |
| | 6 | 345 | 973 | 26 | 137 | 0.00% | 2.90 | >427099 | >26 | 39 | 2.36% | - | 135 | 6 | 26 | 0.00% | **0.22** |
| | 7 | 381 | 4281 | 36 | 219 | 0.00% | 16.55 | >354551 | >22 | 46 | -% | - | 313 | 8 | 34 | 0.00% | **0.64** |
| | 8 | 413 | 22985 | 43 | 265 | 0.00% | 102.51 | >431299 | >29 | 46 | 2.93% | - | 606 | 9 | 40 | 0.00% | **1.31** |
| | 9 | 429 | >110199 | >58 | 380 | 6.43% | - | >388228 | >26 | 60 | -% | - | 2507 | 11 | 46 | 0.00% | **5.97** |
| | 10 | 451 | >104999 | >57 | 448 | 5.74% | - | >456699 | >41 | 74 | 6.57% | - | 2355 | 12 | 46 | 0.00% | **5.91** |
| Random15-60 | 1 | 86 | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 6 | 0.00% | **0.00** | 1 | 0 | 6 | 0.00% | **0.00** |
| | 2 | 163 | 1 | 0 | 16 | 0.00% | **0.00** | 1 | 0 | 8 | 0.00% | **0.00** | 1 | 0 | 8 | 0.00% | **0.00** |
| | 3 | 221 | 9 | 3 | 34 | 0.00% | **0.02** | 41 | 6 | 15 | 0.00% | 0.06 | 12 | 2 | 12 | 0.00% | **0.02** |
| | 4 | 248 | 111 | 10 | 70 | 0.00% | 0.32 | >100454 | >26 | 50 | -% | - | 111 | 6 | 20 | 0.00% | **0.22** |
| | 5 | 268 | 557 | 18 | 101 | 0.00% | 551.83 | >176599 | >29 | 52 | 2.86% | - | 322 | 7 | 29 | 0.00% | **0.76** |
| | 6 | 287 | 419 | 21 | 135 | 0.00% | 1.59 | >277801 | >31 | 45 | 2.74% | - | 354 | 9 | 30 | 0.00% | **0.79** |
| | 7 | 295 | 19097 | 35 | 194 | 0.00% | 72.91 | >387565 | >23 | 49 | -% | - | 836 | 10 | 27 | 0.00% | **1.74** |
| | 8 | 301 | >88799 | >47 | 231 | 2.90% | - | >413343 | >33 | 55 | 2.90% | - | 4995 | 11 | 30 | 0.00% | **11.32** |
| | 9 | 306 | >153099 | >51 | 229 | 1.29% | - | >547079 | >28 | 48 | -% | - | 2263 | 11 | 19 | 0.00% | **4.42** |
| Random20-140 | 1 | 81 | 1 | 0 | 4 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | **0.00** |
| | 2 | 158 | 1 | 0 | 14 | 0.00% | **0.00** | 1 | 0 | 7 | 0.00% | **0.00** | 1 | 0 | 7 | 0.00% | **0.00** |
| | 3 | 228 | 1 | 0 | 30 | 0.00% | 0.02 | 1 | 0 | 11 | 0.00% | **0.00** | 1 | 0 | 11 | 0.00% | **0.00** |
| | 4 | 253 | 9935 | 31 | 103 | 0.00% | 75.25 | >41444 | >42 | 68 | -% | - | 90 | 18 | 67 | 0.00% | **1.04** |
| | 5 | 274 | >39999 | >41 | 146 | 1.86% | - | >68299 | >66 | 87 | 1.86% | - | 819 | 22 | 51 | 0.00% | **12.65** |
| | 6 | 294 | >30199 | >61 | 184 | 1.78% | - | >60299 | >86 | 107 | 1.78% | - | >14106 | >32 | 113 | 1.78% | - |
| | 7 | - | >28999 | >70 | 227 | 1.81% | - | >75894 | >46 | 91 | -% | - | >14299 | >32 | 109 | 1.69% | - |
| | 8 | 319 | >30599 | >80 | 267 | 1.91% | - | >94699 | >101 | 120 | 1.91% | - | 4028 | 22 | 29 | 0.00% | **52.95** |
| | 9 | 325 | >39599 | >93 | 315 | 0.84% | - | >108990 | >63 | 105 | -% | - | 130 | 9 | 25 | 0.00% | **0.32** |
| | 10 | 327 | 2907 | 109 | 326 | 0.00% | 19.15 | >272685 | >49 | 68 | 0.61% | - | 17 | 3 | 22 | 0.00% | **0.02** |
| | 11 | 327 | 1325 | 86 | 301 | 0.00% | 8.75 | 49 | 3 | 22 | 0.00% | **0.03** | 20 | 5 | 20 | 0.00% | **0.03** |
| Best | | | | | | | **11** | | | | | **14** | | | | | **36** |

Table 2: Results from solving the single-commodity `Random` instances exactly.

total number of times each algorithm has best performance is found at the bottom of each table. Also, for each instance the best performance is written in **bold**.

The 2BP algorithm performs much better than the 3BP algorithm for the MCMC$k$FP [8]; however, this is generally not the case for the MC$k$MFP. Although the number of times the algorithm has best performance is larger for the 2BP, the 3BP algorithm is capable of solving more instances. The change of objective function has a great impact on the problem; the algorithms always try to push as much flow through the network as possible, thus potentially exploiting the somewhat weakly formulated bound on the number of used paths. The formulation has less impact on the minimum cost problem because it may not always be beneficial to increase the number of used paths. The 2BP algorithm suffers from large search trees because of the existence of potentially many solutions using more than $k$ paths per commodity and because the branching scheme allows much symmetry in the branching children. The 2BP algorithm, however, performs somewhat better than the 3BP for the multi-commodity `Random` instances with respect to running times.

The 2BP+B algorithm generally performs much better than the 3BP algorithm. Exceptions are `tg40-5`, $k = 4$ and `Random20-80`, $k = 5$, for which the 2BP+B algorithm spends more time on solving. Furthermore, 2BP+B is unable to find an optimal solution for `Random20-80`, $k = 4$. For the far majority of test instances, however, the 2BP+B algorithm is capable of finding an optimal solution in short time, even when the 3BP algorithm shows great difficulty.

| | | | 3BP | | | | | 2BP | | | | | 2BP+B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | **k** | **z** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** |
| tg10-2 | 1 | 389 | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 4 | 0.00% | **0.00** | 1 | 0 | 4 | 0.00% | **0.00** |
| | 2 | 557 | 215 | 13 | 26 | 0.00% | 0.21 | 355 | 14 | 10 | 0.00% | 0.21 | 41 | 5 | 11 | 0.00% | **0.04** |
| | 3 | 716 | 553 | 19 | 58 | 0.00% | 0.70 | 39505 | 20 | 28 | 0.00% | 32.49 | 53 | 5 | 15 | 0.00% | **0.06** |
| | 4 | 815 | 83 | 17 | 52 | 0.00% | 0.10 | 6 | 1 | 8 | 0.00% | **0.00** | 5 | 1 | 8 | 0.00% | **0.00** |
| | 5 | 815 | 1 | 0 | 40 | 0.00% | **0.00** | 1 | 0 | 8 | 0.00% | **0.00** | 1 | 0 | 8 | 0.00% | **0.00** |
| tg20-2 | 1 | 385 | 1 | 0 | 4 | 0.00% | **0.00** | 1 | 0 | 4 | 0.00% | **0.00** | 1 | 0 | 4 | 0.00% | **0.00** |
| | 2 | 643 | 1 | 0 | 10 | 0.00% | **0.00** | 1 | 0 | 6 | 0.00% | **0.00** | 1 | 0 | 6 | 0.00% | **0.00** |
| | 3 | 832 | 5 | 2 | 33 | 0.00% | 0.04 | 1 | 0 | 10 | 0.00% | **0.00** | 1 | 0 | 10 | 0.00% | **0.00** |
| | 4 | 853 | 1 | 0 | 40 | 0.00% | 0.01 | 1 | 0 | 10 | 0.00% | **0.00** | 1 | 0 | 10 | 0.00% | **0.00** |
| tg40-1 | 1 | 517 | 1 | 0 | 5 | 0.00% | **0.00** | 1 | 0 | 5 | 0.00% | 0.01 | 1 | 0 | 5 | 0.00% | **0.00** |
| | 2 | 750 | 5 | 2 | 16 | 0.00% | 0.07 | 4 | 1 | 10 | 0.00% | **0.02** | 10 | 3 | 12 | 0.00% | 0.07 |
| | 3 | 908 | >9999 | >40 | 96 | 2.61% | - | >83282 | >61 | 94 | -% | - | 231 | 11 | 21 | 0.00% | **3.32** |
| | 4 | 994 | >7799 | >57 | 143 | 1.00% | - | >82770 | >45 | 64 | -% | - | 893 | 18 | 33 | 0.00% | **25.15** |
| | 5 | 1004 | 15 | 7 | 65 | 0.00% | 0.09 | 703 | 27 | 20 | 0.00% | 1.41 | 11 | 2 | 18 | 0.00% | **0.03** |
| | 6 | 1004 | 1 | 0 | 96 | 0.00% | 0.03 | 29 | 3 | 13 | 0.00% | **0.02** | 43 | 6 | 13 | 0.00% | 0.07 |
| tg40-5 | 1 | 487 | 1 | 0 | 8 | 0.00% | 0.01 | 1 | 0 | 6 | 0.00% | **0.00** | 1 | 0 | 6 | 0.00% | **0.00** |
| | 2 | 828 | >20599 | >46 | 80 | 4.11% | - | >64248 | >45 | 57 | 5.70% | - | 144 | 9 | 23 | 0.00% | **1.49** |
| | 3 | 1062 | >17299 | >59 | 139 | 0.28% | - | >77103 | >44 | 65 | -% | - | 276 | 8 | 22 | 0.00% | **4.20** |
| | 4 | 1078 | 181 | 47 | 68 | 0.00% | **0.61** | >148934 | >22 | 50 | -% | - | 1520 | 21 | 22 | 0.00% | 26.53 |
| | 5 | 1078 | 3 | 1 | 90 | 0.00% | **0.03** | 61 | 4 | 16 | 0.00% | 0.04 | 76 | 20 | 16 | 0.00% | 1.72 |
| tg80-1 | 1 | 549 | 1 | 0 | 7 | 0.00% | 0.04 | 1 | 0 | 6 | 0.00% | **0.02** | 1 | 0 | 6 | 0.00% | **0.02** |
| | 2 | 984 | 1591 | 29 | 80 | 0.00% | 65.22 | 2308 | 22 | 25 | 0.00% | 59.16 | 288 | 11 | 39 | 0.00% | **8.72** |
| | 3 | 1411 | >2199 | >36 | 162 | 3.85% | - | >51476 | >49 | 107 | -% | - | 1914 | 10 | 38 | 0.00% | **110.38** |
| tg100-2 | 1 | 530 | 1 | 0 | 7 | 0.00% | 0.07 | 1 | 0 | 6 | 0.00% | 0.03 | 1 | 0 | 6 | 0.00% | **0.02** |
| | 2 | 1007 | 3 | 1 | 16 | 0.00% | 0.20 | 1 | 0 | 8 | 0.00% | **0.04** | 1 | 0 | 8 | 0.00% | **0.04** |
| | 3 | 1407 | >1099 | >31 | 115 | 0.39% | - | >29087 | >60 | 113 | -% | - | 229 | 6 | 51 | 0.00% | **29.14** |
| | 4 | 1768 | >1499 | >72 | 234 | 1.51% | - | >56256 | >40 | 167 | -% | - | 2118 | 9 | 82 | 0.00% | **284.41** |
| Best | | | | | | | **7** | | | | | **12** | | | | | **23** |

Table 3: Results from solving the `tg` instances exactly.

| | | | 3BP | | | | | 2BP | | | | | 2BP+B | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | **k** | **z** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** | **size** | **depth** | **vars** | **gap** | **time** |
| Random10-40 | 1 | 110 | 5 | 2 | 18 | 0.00% | 0.02 | 5 | 2 | 15 | 0.00% | **0.00** | 4 | 1 | 14 | 0.00% | 0.01 |
| | 2 | 194 | 1 | 0 | 26 | 0.00% | **0.00** | 34 | 5 | 21 | 0.00% | 0.04 | 4 | 1 | 18 | 0.00% | 0.01 |
| | 3 | 258 | 183 | 18 | 80 | 0.00% | 0.39 | 213 | 12 | 24 | 0.00% | 0.18 | 50 | 6 | 23 | 0.00% | **0.06** |
| | 4 | 293 | 695 | 36 | 129 | 0.00% | 2.23 | 2956 | 16 | 41 | 0.00% | 4.25 | 112 | 7 | 32 | 0.00% | **0.20** |
| | 5 | 309 | 1989 | 30 | 176 | 0.00% | 7.91 | >253716 | >25 | 56 | 1.24% | **1.06** | 561 | 12 | 39 | 0.00% | **1.06** |
| | 6 | 318 | 15905 | 36 | 253 | 0.00% | 73.84 | >610006 | >24 | 59 | 1.35% | - | 1294 | 13 | 60 | 0.00% | **2.63** |
| | 7 | 321 | >153199 | >56 | 286 | 0.01% | - | >335959 | >26 | 54 | -% | - | 26182 | 18 | 47 | 0.00% | **57.10** |
| | 8 | 323 | 113 | 37 | 299 | 0.00% | **0.52** | 2008 | 14 | 37 | 0.00% | 1.23 | 2051 | 15 | 36 | 0.00% | 2.43 |
| | 9 | 323 | 333 | 49 | 318 | 0.00% | 1.38 | 11 | 1 | 32 | 0.00% | **0.01** | 18 | 5 | 32 | 0.00% | 0.02 |
| Random11-42 | 1 | 291 | 5 | 2 | 29 | 0.00% | 0.02 | 7 | 3 | 28 | 0.00% | **0.01** | 7 | 2 | 27 | 0.00% | 0.02 |
| | 2 | 343 | 1 | 0 | 50 | 0.00% | **0.01** | 7 | 2 | 27 | 0.00% | **0.01** | 6 | 1 | 27 | 0.00% | **0.01** |
| | 3 | 344 | 1 | 0 | 75 | 0.00% | **0.00** | 1 | 0 | 26 | 0.00% | **0.00** | 1 | 0 | 26 | 0.00% | **0.00** |
| | 4 | 344 | 1 | 0 | 100 | 0.00% | **0.00** | 1 | 0 | 26 | 0.00% | **0.00** | 1 | 0 | 26 | 0.00% | **0.00** |
| Random20-80 | 1 | 347 | 7 | 3 | 55 | 0.00% | 0.06 | 3 | 1 | 51 | 0.00% | **0.02** | 7 | 2 | 53 | 0.00% | 0.04 |
| | 2 | 553 | 3 | 1 | 100 | 0.00% | 0.03 | 4 | 1 | 50 | 0.00% | 0.02 | 4 | 1 | 51 | 0.00% | **0.01** |
| | 3 | 584 | 1063 | 24 | 188 | 0.00% | 6.14 | 57 | 7 | 59 | 0.00% | **0.16** | 1020 | 16 | 62 | 0.00% | 3.45 |
| | 4 | 601 | 5599 | 33 | 277 | 0.00% | 40.05 | 1041 | 10 | 60 | 0.00% | **2.02** | >81550 | >548 | 601 | 2.01% | - |
| | 5 | 617 | 13291 | 44 | 340 | 0.00% | 117.96 | 4363 | 14 | 66 | 0.00% | **7.35** | 49695 | 34 | 67 | 0.00% | 198.61 |
| | 6 | 621 | >48999 | >40 | 380 | 0.01% | - | 3998 | 11 | 63 | 0.00% | **6.42** | 32552 | 29 | 58 | 0.00% | 100.08 |
| | 7 | 626 | 413 | 37 | 412 | 0.00% | 3.48 | 17 | 2 | 57 | 0.00% | **0.02** | 116 | 14 | 57 | 0.00% | 0.22 |
| | 8 | 626 | 1 | 0 | 440 | 0.00% | 0.03 | 1 | 0 | 57 | 0.00% | **0.01** | 1 | 0 | 57 | 0.00% | **0.01** |
| Random22-56 | 1 | 365 | 9 | 3 | 52 | 0.00% | **0.02** | 7 | 3 | 42 | 0.00% | **0.02** | 7 | 2 | 44 | 0.00% | **0.02** |
| | 2 | 389 | 11 | 4 | 81 | 0.00% | 0.02 | 10 | 3 | 42 | 0.00% | 0.02 | 9 | 3 | 42 | 0.00% | **0.01** |
| | 3 | 407 | 1 | 0 | 108 | 0.00% | **0.01** | 1 | 0 | 41 | 0.00% | **0.01** | 1 | 0 | 41 | 0.00% | **0.01** |
| | 4 | 407 | 1 | 0 | 144 | 0.00% | 0.01 | 1 | 0 | 41 | 0.00% | **0.00** | 1 | 0 | 41 | 0.00% | **0.00** |
| Best | | | | | | | **7** | | | | | **17** | | | | | **14** |

Table 4: Results from solving the multi-commodity instances exactly.

The 2BP+B algorithm generally also generates smaller gaps for instances, which are not solved to optimality. Reasons are that the search tree sizes are generally smaller for the 2BP+B, the number of variables in the master problem is smaller, and much symmetry is eliminated because of the lacking $h$-indices.

The 2BP+B algorithm generally also performs much better than the 2BP algorithm. Exceptions are `Random20-80`, $k = 4, 5$, and 6 where the 2BP has overall best performance. The reason for the generally superior performance of the 2BP+B algorithm is that the branching scheme gives better bounds in the branching children: forcing the use of a path is much stronger than forbidding a path. Also forbidding the use of all paths with positive flow is stronger than forbidding a subset of the paths.

All three algorithms suffer from the same weakness in the formulation, specifically the bounding of the number of used paths per commodity: constraints (3) for the 3BP and (10) for the 2BP and the 2BP+B algorithms. Because the objective is to maximize the total amount of flow, the algorithms are very likely to exceed $k$ paths per commodity whenever the mentioned constraints are not tight. The constraints will rarely be tight, especially when several paths share the same edges and the corresponding $x_p^l/u_p < 1$. The 2BP+B reduces this problem to some extend with the branching cut (12).

Finally, it is noted that including the flow-moving approach from Algorithm 2 in the exact 2BP and 2BP+B approaches does not improve performance; see the tables at `http://www.diku.dk/~gamst/heuristic_results.pdf` for documentation.

### 5.2 Heuristic approach

The three heuristics 3HEUR, 2HEUR, and 2HEUR+FM are evaluated on the previously described instances. Test results are summarized in tables 5, 6, and 7.

The first column of each table holds the name of the problem instance, the second column holds the value of $k$, and the third column holds the optimal value. Then follows for each of the algorithms 3HEUR, 2HEUR, and 2HEUR+FM; the number of iterations in the column generation, the gap between the heuristic and the optimal value, and the time in seconds spent on solving the instance. An entry marked with "-" indicates that no feasible solution was found. The average number of iterations, gap, and time usage are given at the bottom of each table.

The results show that the 3HEUR algorithm often gives poor heuristic solutions with gaps of up to 94%. For three multi-commodity `Random` instances the 3BP algorithm is even unable to find a feasible solution in the root node. The 2HEUR algorithm generally finds much better solution values than the 3HEUR algorithm. The 2HEUR+FM, however, shows superior performance by solving the majority of the instances to optimality and with the largest gap of those not solved being 20%. All heuristics have very low running times and terminate in less than a second.

## 6 Conclusion

Two new exact solution methods for the MC$k$MFP problem have been introduced. They are both based on Dantzig-Wolfe decomposition, where the master problem is a 2-index formulation merging paths for commodities into an overall solution. The two methods differ in their branching schemes: the first method forbids subpaths (2BP), while the second forces or

| | | | 3HEUR | | | 2HEUR | | | 2HEUR+FM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | **k** | **z** | **iter.** | **gap** | **time** | **iter.** | **gap** | **time** | **iter.** | **gap** | **time** |
| Random5-35 | 1 | 66 | 5 | 0.00 | 0.00 | 3 | 0.00 | 0.00 | 3 | 0.00 | 0.01 |
| | 2 | 128 | 7 | 0.00 | 0.00 | 5 | 0.00 | 0.00 | 5 | 0.00 | 0.00 |
| | 3 | 182 | 8 | 0.00 | 0.00 | 7 | 0.00 | 0.00 | 7 | 0.00 | 0.00 |
| | 4 | 223 | 12 | 16.60 | 0.00 | 10 | 0.00 | 0.00 | 10 | 0.00 | 0.00 |
| | 5 | 262 | 12 | 54.96 | 0.00 | 10 | 0.00 | 0.00 | 10 | 0.00 | 0.00 |
| | 6 | 297 | 13 | 80.37 | 0.00 | 12 | 0.00 | 0.00 | 12 | 0.00 | 0.00 |
| | 7 | 326 | 14 | 54.29 | 0.00 | 12 | 0.00 | 0.00 | 12 | 0.00 | 0.00 |
| | 8 | 326 | 13 | 0.00 | 0.01 | 11 | 0.00 | 0.00 | 11 | 0.00 | 0.00 |
| Random10-45 | 1 | 73 | 6 | 0.00 | 0.00 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 |
| | 2 | 142 | 7 | 12.68 | 0.00 | 6 | 12.68 | 0.00 | 6 | 0.00 | 0.00 |
| | 3 | 209 | 10 | 22.00 | 0.01 | 10 | 15.31 | 0.00 | 10 | 0.00 | 0.00 |
| | 4 | 260 | 13 | 65.38 | 0.01 | 13 | 18.08 | 0.00 | 13 | 0.00 | 0.00 |
| | 5 | 306 | 15 | 75.82 | 0.01 | 17 | 46.73 | 0.00 | 17 | 0.00 | 0.00 |
| | 6 | 345 | 17 | 73.91 | 0.02 | 19 | 43.48 | 0.00 | 19 | 0.00 | 0.00 |
| | 7 | 381 | 23 | 76.38 | 0.02 | 21 | 47.77 | 0.00 | 21 | 8.40 | 0.01 |
| | 8 | 413 | 24 | 78.21 | 0.02 | 23 | 48.18 | 0.00 | 23 | 6.78 | 0.01 |
| | 9 | 429 | 30 | 79.02 | 0.04 | 30 | 37.06 | 0.00 | 30 | 1.40 | 0.00 |
| | 10 | 451 | 35 | 80.04 | 0.05 | 38 | 36.59 | 0.01 | 38 | 0.00 | 0.01 |
| Random15-60 | 1 | 86 | 5 | 0.00 | 0.00 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.00 |
| | 2 | 163 | 8 | 0.00 | 0.01 | 8 | 0.00 | 0.00 | 8 | 0.00 | 0.00 |
| | 3 | 221 | 10 | 55.24 | 0.01 | 11 | 85.52 | 0.00 | 11 | 16.74 | 0.00 |
| | 4 | 248 | 13 | 87.50 | 0.01 | 18 | 56.04 | 0.01 | 18 | 10.01 | 0.00 |
| | 5 | 268 | 16 | 57.49 | 0.02 | 20 | 51.49 | 0.00 | 20 | 5.97 | 0.00 |
| | 6 | 287 | 18 | 93.38 | 0.02 | 22 | 50.52 | 0.01 | 22 | 6.62 | 0.00 |
| | 7 | 295 | 20 | 85.05 | 0.02 | 23 | 46.44 | 0.01 | 23 | 13.90 | 0.00 |
| | 8 | 301 | 19 | 59.47 | 0.01 | 21 | 46.84 | 0.00 | 21 | 17.94 | 0.00 |
| | 9 | 306 | 18 | 60.13 | 0.01 | 18 | 39.87 | 0.00 | 18 | 19.93 | 0.00 |
| Random20-140 | 1 | 81 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 |
| | 2 | 158 | 7 | 0.00 | 0.02 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.00 |
| | 3 | 228 | 10 | 0.00 | 0.02 | 10 | 0.00 | 0.00 | 10 | 0.00 | 0.00 |
| | 4 | 253 | 12 | 82.48 | 0.03 | 13 | 0.00 | 0.00 | 13 | 0.00 | 0.01 |
| | 5 | 274 | 16 | 84.69 | 0.04 | 16 | 1.46 | 0.01 | 16 | 0.00 | 0.01 |
| | 6 | 294 | 18 | 84.69 | 0.04 | 24 | 69.05 | 0.01 | 24 | 3.40 | 0.01 |
| | 9 | 325 | 22 | 86.15 | 0.04 | 24 | 44.92 | 0.01 | 24 | 0.31 | 0.01 |
| | 10 | 327 | 21 | 86.24 | 0.01 | 19 | 3.67 | 0.00 | 19 | 3.67 | 0.00 |
| | 11 | 327 | 20 | 86.24 | 0.01 | 19 | 0.61 | 0.00 | 19 | 0.61 | 0.00 |
| Sum | | | 14 | 49.40 | 0.01 | 15 | 22.29 | <0.01 | 15 | 3.21 | <0.01 |

Table 5: Results from solving the single-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only.

| | | | 3HEUR | | | 2HEUR | | | 2HEUR+FM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Problem** | **k** | **z** | **iter.** | **gap** | **time** | **iter.** | **gap** | **time** | **iter.** | **gap** | **time** |
| tg10-2 | 1 | 389 | 5 | 0.00 | 0.00 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 |
| | 2 | 557 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.00 |
| | 3 | 716 | 9 | 0.00 | 0.00 | 10 | 15.22 | 0.00 | 10 | 0.00 | 0.00 |
| | 4 | 815 | 8 | 0.00 | 0.00 | 8 | 15.83 | 0.01 | 8 | 0.00 | 0.00 |
| | 5 | 815 | 8 | 0.00 | 0.00 | 8 | 0.00 | 0.00 | 8 | 0.00 | 0.00 |
| tg20-2 | 1 | 385 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 | 4 | 0.00 | 0.00 |
| | 2 | 643 | 5 | 0.00 | 0.00 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.00 |
| | 3 | 832 | 11 | 18.03 | 0.00 | 10 | 0.01 | 0.00 | 10 | 0.00 | 0.01 |
| | 4 | 853 | 10 | 0.00 | 0.01 | 10 | 0.00 | 0.00 | 10 | 0.00 | 0.00 |
| tg40-1 | 1 | 517 | 5 | 0.00 | 0.01 | 5 | 0.00 | 0.01 | 5 | 0.00 | 0.01 |
| | 2 | 750 | 7 | 72.13 | 0.01 | 9 | 61.33 | 0.01 | 9 | 0.00 | 0.01 |
| tg40-5 | 1 | 487 | 8 | 0.00 | 0.00 | 6 | 0.00 | 0.00 | 6 | 0.00 | 0.01 |
| tg80-1 | 1 | 549 | 7 | 0.00 | 0.04 | 6 | 0.00 | 0.02 | 6 | 0.00 | 0.02 |
| | 2 | 984 | 11 | 52.74 | 0.14 | 14 | 14.63 | 0.06 | 14 | 0.00 | 0.06 |
| tg100-2 | 1 | 530 | 7 | 0.00 | 0.07 | 6 | 0.00 | 0.02 | 6 | 0.00 | 0.02 |
| | 2 | 1007 | 8 | 28.10 | 0.15 | 8 | 0.00 | 0.04 | 8 | 0.00 | 0.03 |
| Sum | | | 7 | 10.69 | 0.03 | 8 | 6.69 | 0.01 | 8 | 0.00 | 0.01 |

Table 6: Results from solving the `tg` instances heuristically, where each algorithm terminates after having evaluated the root node only.

| | | | 3HEUR | | | 2HEUR | | | 2HEUR+FM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Problem | k | z | iter. | gap | time | iter. | gap | time | iter. | gap | time |
| Random10-40 | 1 | 110 | 6 | 31.82 | 0.00 | 5 | 20.00 | 0.01 | 5 | 17.27 | 0.00 |
| | 2 | 194 | 6 | 0.00 | 0.00 | 9 | 60.82 | 0.00 | 9 | 0.00 | 0.00 |
| | 3 | 258 | 11 | 80.62 | 0.01 | 11 | 69.38 | 0.00 | 11 | 6.59 | 0.00 |
| | 4 | 293 | 14 | 66.21 | 0.02 | 15 | 36.52 | 0.01 | 15 | 7.17 | 0.01 |
| | 5 | 309 | 16 | 67.96 | 0.02 | 19 | 34.95 | 0.00 | 19 | 8.41 | 0.01 |
| | 6 | 318 | 21 | 68.89 | 0.03 | 25 | 33.02 | 0.00 | 25 | 5.97 | 0.01 |
| | 7 | 321 | 17 | 84.42 | 0.02 | 21 | 24.61 | 0.00 | 21 | 1.56 | 0.01 |
| | 8 | 323 | 21 | 84.52 | 0.01 | 20 | 22.29 | 0.00 | 20 | 4.34 | 0.00 |
| | 9 | 323 | 20 | 84.52 | 0.01 | 21 | 9.29 | 0.00 | 21 | 0.00 | 0.00 |
| Random11-42 | 1 | 291 | 6 | 6.19 | 0.01 | 6 | 6.19 | 0.00 | 6 | 0.00 | 0.01 |
| | 2 | 343 | 4 | 0.00 | 0.00 | 5 | 19.53 | 0.00 | 5 | 4.08 | 0.00 |
| | 3 | 344 | 4 | 0.00 | 0.00 | 5 | 0.00 | 0.01 | 5 | 0.00 | 0.00 |
| | 4 | 344 | 4 | 0.00 | 0.00 | 5 | 0.00 | 0.00 | 5 | 0.00 | 0.00 |
| Random20-80 | 1 | 347 | 6 | 25.36 | 0.01 | 6 | 16.14 | 0.01 | 6 | 0.00 | 0.01 |
| | 2 | 553 | 7 | 35.80 | 0.02 | 7 | 15.91 | 0.01 | 7 | 0.00 | 0.01 |
| | 3 | 584 | 9 | - | 0.02 | 9 | 7.53 | 0.00 | 9 | 0.00 | 0.01 |
| | 4 | 601 | 12 | - | 0.03 | 12 | 7.65 | 0.01 | 12 | 0.00 | 0.01 |
| | 5 | 617 | 14 | - | 0.04 | 16 | 4.05 | 0.02 | 16 | 2.27 | 0.00 |
| | 6 | 621 | 12 | 58.29 | 0.03 | 14 | 0.64 | 0.01 | 14 | 0.00 | 0.01 |
| | 7 | 626 | 12 | 58.63 | 0.03 | 14 | 0.96 | 0.01 | 14 | 0.80 | 0.01 |
| | 8 | 626 | 12 | 0.00 | 0.03 | 14 | 0.00 | 0.01 | 14 | 0.00 | 0.01 |
| Random22-56 | 1 | 365 | 6 | 0.00 | 0.01 | 5 | 0.00 | 0.00 | 5 | 0.00 | 0.00 |
| | 2 | 389 | 6 | 1.54 | 0.00 | 5 | 1.54 | 0.00 | 5 | 0.00 | 0.00 |
| | 3 | 407 | 6 | 0.00 | 0.01 | 5 | 0.00 | 0.00 | 5 | 0.00 | 0.01 |
| | 4 | 407 | 6 | 0.00 | 0.00 | 5 | 0.00 | 0.01 | 5 | 0.00 | 0.01 |
| Sum | | | 9 | 34.31 | 0.01 | 11 | 15.64 | <0.01 | 11 | 2.34 | <0.01 |

Table 7: Results from solving the multi-commodity `Random` instances heuristically, where each algorithm terminates after having evaluated the root node only. Sum only sums over the instances where all heuristics found a feasible solution.

forbids the use of certain paths (2BP+B). The latter also adds branching cuts to the master problem.

The 2BP and 2BP+B algorithms are implemented and compared with a leading exact algorithm from the literature denoted 3BP. Results show that the 2BP+B algorithm performs significantly better than the 2BP and the 3BP algorithms both with respect to the number of solved instances and with respect to the time usage. The main reason is that using the 2BP+B algorithm produces smaller search trees, reduces the number of variables in the master problem, and eliminates some of the symmetry in the solution space.

Terminating the computations after having evaluated the root node transforms the 3BP and the 2BP/2BP+B algorithms into heuristics denoted 3HEUR and 2HEUR, respectively. Because no branching occurs in this heuristic approach, the 2BP and the 2BP+B algorithms become identical. Test results for this approach show that the 3HEUR does not perform well, with the majority of the solution values having gaps of up to 94%. The 2HEUR algorithm, however, shows very promising performance when including a flow-moving approach, which transforms some fractional solutions into feasible solutions. In most cases optimal solutions are found and the average solution gaps never exceed 4%. Both heuristics terminate in less than a second for all tested instances.

All algorithms suffer from weak formulations for bounding the number of used paths per commodity. We believe that future work should concentrate on tightening these constraints. The problem could for example be reformulated such that the number of used paths per commodity is implicitly satisfied through pricing or primal convexification techniques. We believe that the focus should be on cuts violated in the edge-based model or the original master problem. Future work could also concentrate on finding better branching strategies for the 2-index formulation in order to further reduce the size of the search tree.

**Acknowledgement**
We would like to thank GlobalConnect A/S for their support of this work.

## References

[1] F. Alvelos and J. M. V. de Carvalho. Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem. In *International Network Optimization Conference*, pages 7–12, 2003.

[2] G. Baier, E. Kohler, and M. Skutella. On the $k$-splittable flow problem. *Algorithmica*, 42:231–248, 2005.

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

[4] M. Caramia and A. Sgalambro. An exact approach for the maximum concurrent k-splittable flow problem. *Optimization Letters*, 2:251–265, 2008.

[5] M. Caramia and A. Sgalambro. A fast heuristic algorithm for the maximum concurrent k-splittable flow problem. *Optimization Letters*, 4:37–55, 2010.

[6] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[7] J. W. Evans and C. Filsfils. *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice*. Morgan Kaufmann, 2007.

[8] M. Gamst, P. N. Jensen, D. Pisinger, and C. E. M. Plum. Two- and three-index formulations of the minimum cost multicommodity $k$-splittable flow problem. *European Journal of Operations Research*, 202(1):82–89, 2010.

[9] J. Kleinberg. Single-source unsplittable flow. In *37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, 1996.

[10] R. Koch, M. Skutella, and I. Spenke. Approximation and complexity of $k$-splittable flows. In *Approximation and Online Algorithms, Third International Workshop, WAOA*, pages 244–257, 2005.

[11] R. Koch, M. Skutella, and I. Spenke. Maximum $k$-splittable s, t -flows. *Theory of Computing Systems*, 43(1):1432–4350, 2008.

[12] S. G. Kolliopoulos. Minimum-cost single-source 2-splittable flow. *Information Processing Letters*, 94(1):15–18, 2005.

[13] R. Lougee-Heimer. The common optimization interface for operations research. *IBM Journal of Research and Development*, 47:57–66, 2003.

[14] M. Martens and M. Skutella. Flows on few paths: algorithms and lower bounds. *Networks*, 48(2):68–76, 2006.

[15] F. Salazar and M. Skutella. Single-source k-splittable min-cost flows. *Operations research letters*, 37:71–74, 2009.

[16] J. Truffot and C. Duhamel. A branch and price algorithm for the $k$-splittable maximum flow problem. *Discrete Optimization*, 5(3):629–646, 2008.

[17] J. Truffot, C. Duhamel, and P. Mahey. Using branch-and-price to solve multicommodity $k$-splittable flow problems. In *Proceedings of International Network Optimization Conference (INOC)*, 2005.

[18] J. Truffot, C. Duhamel, and P. Mahey. k-splittable delay constrained routing problem: A branch-and-price approach. *Networks*, 55(1):33–45, 2010.

[19] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.