Two- and Three-index formulations of the Minimum Cost Multicommodity k-splittable Flow Problem

Gamst, Mette; Jensen, Peter Neergaard; Pisinger, David; Plum, Christian Edinger Munk

Published in: International Network Optimization Conference

Publication date: 2009

Document Version: Publisher's PDF, also known as Version of record

Link back to DTU Orbit

Two- and Three-index formulations of the Minimum Cost Multicommodity $k$-splittable Flow Problem

Mette Gamst  Peter Neergaard Jensen  David Pisinger  Christian Plum

Department of Computer Science, University of Copenhagen
Universitetsparken 1, 2100 Copenhagen Ø, Denmark
{gamst, peergaard, pisinger, plum}@diku.dk

Abstract

The Multicommodity Flow Problem (MCFP) considers the efficient routing of commodities from their origins to their destinations, subject to capacity restrictions and edge costs. This paper studies the $\mathcal{NP}$-hard Minimum Cost Multicommodity $k$-splittable Flow Problem in which each commodity may use at most $k$ paths between its origin and its destination. The problem has applications in transportation problems where a number of commodities must be routed, using a limited number of transportation units at each destination. Based on a three-index formulation from (Truffot et al., 2005) we present a new two-index formulation for the problem, and solve both formulations through branch-and-price. The three-index algorithm by Truffot et al. is improved by introducing a heuristic method to reach a feasible solution by eliminating some symmetry. A novel branching strategy for the two-index formulation is presented, forbidding subpaths in the branching children. The proposed heuristic for the three-index algorithm improves the performance, but the three-index algorithm is still outperformed by the two-index algorithm, both with respect to running time and to the number of solved test instances.

Keywords: mixed integer programming, $k$-splittable, Multicommodity Flow, Branch-and-Price

1 Introduction

We consider the $\mathcal{NP}$-hard Minimum Cost, $k$-splittable variant of the Multicommodity Flow Problem (MCFP). A MCFP consists of a network with capacitated edges, and of a set of commodities, and the goal is to either minimize the total cost of sending all flow of the commodities, or to maximize the total amount of flow sent for all the commodities. The MCFP can be formulated as a linear programming problem and is thus polynomial [1]. Often, however, extra conditions must be satisfied, making the problem $\mathcal{NP}$-hard. An example of such a condition is, that all flow for each commodity must be sent via just one path. This problem is denoted the Unsplittable MCFP, and was introduced and proved $\mathcal{NP}$-hard by Kleinberg [5]. Yet another practically relevant condition is an upper bound on the number of paths used by a commodity. This is called the Multicommodity $k$-splittable Flow Problem (MC$k$FP). We consider the Minimum Cost MC$k$FP, which for instance is relevant in the transportation sector or in telecommunication context.

Barnhart et al. [4] considered the Minimum Cost Unsplittable MCFP. They presented a branch-and-price-and-cut algorithm with a new branching rule allowing new columns to be generated effectively. The Multicommodity $k$-splittable Flow Problem (MC$k$FP) was introduced and proved to be $\mathcal{NP}$-hard by Baier et al. [3], who also presented approximation algorithms for the Single- and Multicommodity $k$-splittable Flow Problems. Truffot et al. [8, 10] used branch-and-price to solve the Maximum MC$k$FP. The pricing problem is a shortest path problem solvable in polynomial time. Truffot et al. [9] also introduced the Minimum Cost MC$k$FP. A three-index model for the problem was solved using a branch-and-price algorithm. The algorithm is closely related to the one presented in [10].
Three-index model

Let \( P^l \) be the set of possible paths for commodity \( l \). The variable \( x_{p}^{hl} \) denotes the amount of flow on path \( p \) for the \( h \)'th path of commodity \( l \). The binary variable \( y_{p}^{hl} \) decides whether path \( p \) for the \( h \)'th path of commodity \( l \) is to be used or not. The model is:

\[
\begin{align*}
\min & \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} c_p x_{p}^{hl} \\
\text{s.t.} & \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} \delta_{e}^{hp} x_{p}^{hl} \leq u_e \quad \forall e \in E \\
& \quad x_{p}^{hl} - u_p y_{p}^{hl} \leq 0 \quad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l \\
& \quad \sum_{p \in P^l} y_{p}^{hl} \leq 1 \quad \forall l \in L, h = 1, \ldots, k^l \\
& \quad \sum_{h=1}^{k^l} \sum_{p \in P^l} x_{p}^{hl} = F^l \quad \forall l \in L \\
& \quad x_{p}^{hl} \geq 0 \quad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l \\
& \quad y_{p}^{hl} \in \{0, 1\} \quad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l
\end{align*}
\]

(MIP1)

The objective function minimizes the total cost. Constraint (1) is a capacity constraint, in which \( \delta_{e}^{hp} \) indicates whether or not edge \( e \) is used by path \( p \). In (2), \( u_p \) denotes the capacity constraint on path \( p \), which is defined as \( u_p = \min \{ u_e | e \in p \} \), hence (2) forces every decision variable, \( y_{p}^{hl} \), to be set, if there is flow on the corresponding path, \( x_{p}^{hl} \). Constraint (3) ensures, that at most one path is used as the \( h \)'th path of a commodity \( l \), and finally (4) ensures that all commodities are shipped.

The model is relaxed into an LP-model: first the binary variables \( y_{p}^{hl} \) are LP-relaxed to \( 0 \leq y_{p}^{hl} \leq 1 \). From (2) and (3) we have that: \( x_{p}^{hl} / u_p \leq y_{p}^{hl} \leq 1 \). We set \( y_{p}^{hl} \) to its lower bound, which leaves the following model:

\[
\begin{align*}
\min & \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} c_p x_{p}^{hl} \\
\text{s.t.} & \quad \sum_{l \in L} \sum_{h=1}^{k^l} \sum_{p \in P^l} \delta_{e}^{hp} x_{p}^{hl} \leq u_e \quad \forall e \in E \\
& \quad \sum_{p \in P^l} x_{p}^{hl} \leq 1 \quad \forall l \in L, h = 1, \ldots, k^l \\
& \quad \sum_{h=1}^{k^l} \sum_{p \in P^l} x_{p}^{hl} = F^l \quad \forall l \in L \\
& \quad x_{p}^{hl} \geq 0 \quad \forall l \in L, h = 1, \ldots, k^l, \forall p \in P^l
\end{align*}
\]

(LP2)

Model (LP2) causes symmetry in the solution space, as the \( h \)-index may result in equivalent solutions being treated as different solutions. To eliminate some of this symmetry, the following variable ordering constraint is added to
Model is similar to commodity. Each path is assigned a unique value of \( h \). To reach a feasible solution faster, the heuristic seeks to eliminate these. For a commodity, several identical paths have different values of \( h \) and \( l \). The remaining variables have the same meaning as in the three-index model. The model is similar to \((MIP1)\), only constraint (12) has been added to limit the number of used paths for commodity \( l \).
to $k^l$. The problem is relaxed in the same manner as the three-index model, i.e. we replace $y_p^l$ with $x_p^l/u_p$ getting:

\[
(LP4) \quad \min \sum \sum c_p x_p^l \\
\text{s.t.} \sum \sum \delta_{he}^p x_p^l \leq u_e \quad \forall e \in E \tag{14}
\]

\[
= \sum_{p \in P} u_p \leq k^l \quad \forall l \in L \tag{15}
\]

\[
= \sum_{p \in P} x_p^l = F^l \quad \forall l \in L \tag{16}
\]

\[= x_p^l \geq 0 \quad \forall l \in L, \forall p \in P^l
\]

Let $\pi_e$, $\lambda^l$ and $\sigma^l$ be the dual variables for equations (14), (15) and (16) in (LP4). The reduced cost for a path $p \in P^l$ for a commodity $l \in L$ is given by:

\[
\sum_{e \in E} \delta_{he}^p (c_e - \pi_e) - \frac{\lambda^l}{u_p} + \sigma^l \tag{17}
\]

The reduced cost is similar to that for the three-index model, hence the pricing problem is solved using Dijkstra’s algorithm for the shortest path problem for each pair of values $(h, l)$ and for each of the at most $|E|$ values of $u_p$.

The branching strategy for the three-index model cannot be reused, because the $h$-indices are omitted. A new strategy is developed, which considers the paths emanating from the first divergence node for each commodity. If the number of emanating paths is greater than $k^l$, then branching is necessary. The number of edges with positive flow going out of the divergence node may be smaller than the number of paths emanating from the node. Thus, it does not suffice to forbid the use of an edge. Rather, the branching strategy must forbid the use of a subpath. For each emanating path, the strategy finds the smallest sequence of edges, which makes the path unique. That is, the strategy seeks to minimize the size of the forbidden subpath. The number of branching children is $k^l + 1$, in which forbidden edge sequences are evenly distributed such that each branching child contains at least one forbidden edge sequence. No feasible solution is omitted from the combined solution space of the branching children. A feasible solution can use at most $k^l$ of the subpaths we consider in a branch and each of these is forbidden in exactly one of the $k^l + 1$ branching children. Any valid solution will therefore be valid in at least one of these branching children, where its $k^l$ used subpaths are forbidden in the remaining $k^l$ branching children. The solution space of the branching children is not necessarily disjoint, which may result in degeneracy problems, since a solution can exist in several branching children, which must thus be explored.

The branching strategy necessitates some changes to 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 using a modified $k$-shortest path algorithm [11].

### 4 Computational Results

The described branch-and-price algorithms for the two models are tested on an Intel Pentium 4, 3.00 GHz machine with 2 GB RAM. The algorithms have been implemented using the framework COIN [7] with ILOG CPLEX 9.1 as LP-solver. We have through preliminary results decided to use strong branching, where all possible branching candidates are generated. A best-first search strategy is used in the branch-and-bound tree. Computations regarding selection of branching candidate and branching child are handled by COIN. The number of paths priced in per iteration to 0.5 · $|L|$ · $k$ for the three-index algorithm and to 0.5 · $|L|$ for the two-index algorithm.

The algorithms are tested on four types of problems: The randomly generated Carbin instances ($b$ and $bs$) [2], the grid instances formed as grids and planar instances simulating problems arising in telecommunication [6]. Three different values for $k$ has been tested: $k = 2, 3$ and $10$. When $k = 1$ the problem becomes the unsplittable MCFP, where more specialized algorithms are developed [4]. For large values of $k$, the problem becomes the linear MCFP.
Table 1: Results for the three-index algorithm with and without the proposed heuristic. The column H. Time denotes the time spent in the heuristic.

<table>
<thead>
<tr>
<th>Problem, k</th>
<th>Heur.</th>
<th>Time</th>
<th>Tree size</th>
<th>Depth</th>
<th>Gap</th>
<th>UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>bl03, 2</td>
<td>no</td>
<td>225.06</td>
<td>&gt;34000</td>
<td>75</td>
<td>0.23</td>
<td>15836.0</td>
</tr>
<tr>
<td>bl03, 2</td>
<td>yes</td>
<td>225.38</td>
<td>&gt;34000</td>
<td>75</td>
<td>0.23</td>
<td>15836.0</td>
</tr>
<tr>
<td>bl03, 3</td>
<td>no</td>
<td>2.98</td>
<td>317</td>
<td>49</td>
<td>0.00</td>
<td>15799.0</td>
</tr>
<tr>
<td>bl03, 3</td>
<td>yes</td>
<td>0.44</td>
<td>1</td>
<td>0</td>
<td>0.00</td>
<td>15799.0</td>
</tr>
<tr>
<td>bl03, 10</td>
<td>no</td>
<td>2.17</td>
<td>63</td>
<td>31</td>
<td>0.00</td>
<td>15799.0</td>
</tr>
<tr>
<td>bl03, 10</td>
<td>yes</td>
<td>0.13</td>
<td>1</td>
<td>0</td>
<td>0.00</td>
<td>15799.0</td>
</tr>
<tr>
<td>bs03, 2</td>
<td>no</td>
<td>0.59</td>
<td>125</td>
<td>28</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
<tr>
<td>bs03, 2</td>
<td>yes</td>
<td>0.47</td>
<td>97</td>
<td>25</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
<tr>
<td>bs03, 3</td>
<td>no</td>
<td>0.17</td>
<td>29</td>
<td>14</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
<tr>
<td>bs03, 3</td>
<td>yes</td>
<td>0.02</td>
<td>1</td>
<td>0</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
<tr>
<td>bs03, 10</td>
<td>no</td>
<td>1.31</td>
<td>61</td>
<td>25</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
<tr>
<td>bs03, 10</td>
<td>yes</td>
<td>0.08</td>
<td>1</td>
<td>0</td>
<td>0.00</td>
<td>16488.0</td>
</tr>
</tbody>
</table>

Table 2: The number of test instances solved to optimality with the 3-index and 2-index algorithms, for various k values. A.Mean is the average mean time in seconds calculated over those instances solved to optimality by both algorithms.

<table>
<thead>
<tr>
<th>Name</th>
<th>k</th>
<th># instances</th>
<th>3-index A.Mean</th>
<th>Opt.</th>
<th>2-index A.Mean</th>
<th>Opt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bl</td>
<td>2</td>
<td>11</td>
<td>5.06</td>
<td>6/11</td>
<td>1.90</td>
<td>11/11</td>
</tr>
<tr>
<td>bl</td>
<td>3</td>
<td>11</td>
<td>0.43</td>
<td>10/11</td>
<td>0.21</td>
<td>11/11</td>
</tr>
<tr>
<td>bl</td>
<td>10</td>
<td>11</td>
<td>0.87</td>
<td>11/11</td>
<td>0.22</td>
<td>11/11</td>
</tr>
<tr>
<td>bs</td>
<td>2</td>
<td>11</td>
<td>41.66</td>
<td>3/11</td>
<td>0.32</td>
<td>9/11</td>
</tr>
<tr>
<td>bs</td>
<td>3</td>
<td>11</td>
<td>37.95</td>
<td>8/11</td>
<td>0.32</td>
<td>11/11</td>
</tr>
<tr>
<td>bs</td>
<td>10</td>
<td>11</td>
<td>1.08</td>
<td>11/11</td>
<td>0.27</td>
<td>11/11</td>
</tr>
<tr>
<td>planar</td>
<td>2</td>
<td>5</td>
<td>117.92</td>
<td>4/5</td>
<td>3.09</td>
<td>5/5</td>
</tr>
<tr>
<td>planar</td>
<td>3</td>
<td>5</td>
<td>2.58</td>
<td>4/5</td>
<td>2.75</td>
<td>5/5</td>
</tr>
<tr>
<td>planar</td>
<td>10</td>
<td>5</td>
<td>267.40</td>
<td>5/5</td>
<td>15.13</td>
<td>5/5</td>
</tr>
<tr>
<td>grid</td>
<td>2</td>
<td>7</td>
<td>1.40</td>
<td>4/7</td>
<td>0.24</td>
<td>5/7</td>
</tr>
<tr>
<td>grid</td>
<td>3</td>
<td>7</td>
<td>0.09</td>
<td>5/7</td>
<td>0.73</td>
<td>7/7</td>
</tr>
<tr>
<td>grid</td>
<td>10</td>
<td>7</td>
<td>7.00</td>
<td>7/7</td>
<td>1.31</td>
<td>7/7</td>
</tr>
</tbody>
</table>

First off, results of computational evaluations of the branch-and-price algorithm for the three-index model with and without the heuristic are showed in Table 1. The running times are improved significantly by including the proposed heuristic, as this gives a smaller search tree. Hence, the heuristic is included in the remaining tests.

Next, we compare the two branch-and-price algorithms with each other, see Table 2. For $k = 2$, the three-index algorithm shows difficulty in solving many instances, whereas the two-index algorithm has much greater success. The latter also has better running times for instances, both algorithms can solve. Both algorithms, however, fail for larger instances. For $k = 3$ and $k = 10$ both algorithms perform well with respect to the number of solved instances, but the branch-and-price algorithm for the two-index model has better running times.

The running times reflect the complexity of the corresponding problem instances and used algorithms. Whenever the value of $k$ exceeds some threshold value, the running time for solving the instance decreases. The reason for this is that at some point, $k$ does not impose a constraint on the problem, i.e., the instance corresponds to the linear MCFP. The value of $k$ has greater impact on the three-index algorithm. When $k$ takes on a value greater than the mentioned threshold, the running time of the three-index algorithm increases, because columns are generated for each $i = 1, \ldots, k$, and are priced into the master problem. Generating columns and solving a larger master problem is time consuming. The same is obviously not the case for the two-index algorithm.

The three-index algorithm is capable of solving instances with up to 2239 commodities, 850 edges and 150 nodes (planar150), and 400 commodities, 1520 edges and 400 nodes (grid400:1520:400) for $k = 10$, and instances with up to 532 commodities, 1085 edges and 100 nodes (planar100) for $k = 2$. The two-index algorithm solves instances with up to 2239 commodities, 850 edges and 150 nodes (planar150) and 400 commodities, 1520 edges
and 400 nodes (grid\textsubscript{400:1520:400}) for \( k = 10 \), and instances with up to 2239 commodities, 850 edges and 150 nodes (planar\textsubscript{150}) for \( k = 2 \). Also, the three-index algorithm is capable of solving about 76\% of the test instances to optimality, while the two-index has solved just over 96\% of the test instances to optimality. Hence, for the far majority of the problem instances the two-index algorithm outperforms the three-index algorithm, both with respect to time spent and to the number of instances solved to optimality. We conclude, that this is partly due to the extra \( h \)-index in the three-index model causing symmetry in the solution space, and partly due to the three-index algorithm having \( k \) times as many variables as the two-index algorithm.

5 Conclusions

In this paper we have presented a branch-and-price algorithm for the MCMC\textsubscript{k}FP, which outperforms existing methods. The new branch-and-price algorithm is based on a mathematical formulation, which unlike previous formulations omits a symmetry inducing index for each of the \( k \) paths per commodity. Hence, we have named our formulation the two-index model, while the existing model is a three-index model. The two-index model has parallelly been suggested for the Maximum Flow MCKFP by Truffot et al. [9], but they discarded the model because it complicates branching. We have presented a branching strategy for the model, which ensures that the pricing problem can be solved efficiently. The branching strategy and the algorithm for the resulting pricing problem can be directly used on the Maximum Flow problem.

Furthermore, we have introduced a heuristic for the three-index branch-and-price algorithm which transforms certain fractional solutions into feasible solutions. Though the heuristic boosts the performance of the three-index algorithm, it is still outperformed by the two-index algorithm both with respect to time usage and to the number of solved instances. The three-index algorithm including the proposed heuristic has solved 76\% of the problem instances to optimality, where the two-index has solved 96\% of the problem instances to optimality.

Acknowledgement

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

References