An improved benders decomposition applied to a multi-layer network design problem
Bernard Fortz, Michael Poss

To cite this version:
hal-01255545

HAL Id: hal-01255545
https://hal.science/hal-01255545
Submitted on 30 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
An improved Benders decomposition applied to a multi-layer network design problem

B. Fortz
Department of Computer Science, Université Libre de Bruxelles, Brussels, Belgium.
CORE, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

M. Poss
Department of Computer Science, Université Libre de Bruxelles, Brussels, Belgium.

July 18, 2013

Abstract

Benders decomposition has been widely used for solving network design problems. In this paper, we use a branch-and-cut algorithm to improve the separation procedure of Gabrel et al. and Knippel et al. for capacitated network design. We detail experiments on bi-layer networks, comparing with Knippel’s previous results.

Keywords: multi-layer network design, metric inequalities, branch-and-cut.

1 Introduction

Today, telecommunication networks are designed with a layered structure, according to different technologies. For instance, one could consider a virtual layer over a physical layer, also called transport layer. This leads to bi-layer network design problems. In those problems, demands are usually given in the virtual layer. They have to be routed in the virtual layer, leading to the installation of “virtual capacities” (which are routers or other devices). Virtual capacities define demands for the transport layer, leading to the installation of capacities (optical fiber, copper link, ...), in the physical layer. Therefore, when a demand is routed through a path in the virtual layer (composed of many virtual edges), each edge corresponds to a path in the layer underneath (also called a ”grooming path”).

Technically, each layer has its own technology [21], for instance:

- MPLS: Multi-Protocol Label Switching,
- WDM: Wavelength-Division-Multiplexing,
- SDH: Synchronous Digital Hierarchy.

As the single-layer capacitated network design problem is complicated enough, most approaches for the bi-layer problem consider each layer separately:

- First, a network design problem is solved for the virtual layer only.
- Then, virtual capacities to be installed in the virtual layer define demands for another network design problem, for the transport layer this time.

Though much easier to solve, this relaxed approach might provide solutions far from the optimal solution of the problem. Therefore, an integrated approach should be considered.

Network design has been widely studied for many years [22]. However, the interest in multi-layered network design is more recent and can be traced back to a paper by Dahl and Stoer [9]. They assume given physical capacities and aim to select virtual edges (called pipes in the paper) and to configure the routing in both layers. A polyhedral study is made resulting in a cutting-plane algorithm. Since then, the interest in this field has rapidly grown and different approaches have been suggested to address these problems.

Orlowski and Wessel [20] begin by giving a good introduction to multi-layered networks where they offer technical examples and develop a model considering many technical constraints. However, they do not propose a specific solution method. In further papers, Koster et al. develop different branch-and-cut approaches. Extending previous work by Belotti et al. [4], they briefly describe a cut-and-branch-and-price algorithm. They then solve an integer formulation using a branch-and-cut framework [16], where they introduce efficient heuristics. Finally, they address a more complex formulation, taking node hardware and survivability into account [2]. They also extend and test different sorts of cuts coming from mono-layer models [17].

Capone et al. [3, 6] study multi-layered design with statistical multiplexing, the motivation being that routing different commodities on the same capacity results in less variation of the flow on the capacity. They compute a lower bound through a Lagrangian relaxation and use heuristics to find good upper bounds.
Kubiliunas and Pioro [18] address the problem of maximizing the profit of satisfying demands in a bi-layer (MPLS over WDM) situation. They present an iterative procedure to solve their complex mixed-integer problem. This procedure consists of splitting the problems into two stages, one for each layer, where the solution of the first layer defines demands in the second one. Then the routing solution in the physical layer leads to cost modification for edges in the first layer and the whole problem is solved again.

Hölter and Voß [13] propose an integer programming formulation for two layer networks consisting of SDH over WDM. Strictly speaking, this is not a multi-layer problem in the sense that demands are routed through either SDH links or WDM links. They solve the problem using two different heuristics.

Gabrel et al. introduce a constraint generation procedure based on a Benders decomposition for capacitated network design problems [11]. Knippel and Lardeux extend this work to multi-layered networks [13] and multi-period time scheduling [12]. They introduce the metric cone, which eases cut generation. In [15], they improve this method using the knapsack-like structure of the master problem to facilitate its resolution.

In this paper, we improve the constraint generation method used by Knippel and Lardeux [13]. Namely, we develop a branch-and-cut algorithm to solve the Benders decomposition of the problem. This speeds up the resolution times by a factor of 10 on average. Also, we obtain bounds for difficult problems, whereas Knippel’s cutting plane is unable to compute upper bounds. This framework could easily be extended to improve results on the multi-period network design problem solved by Lardeux [12].

In the next section, we describe the model and its reformulation using a Benders decomposition. In Section 2, we describe different algorithms to solve the problem: the cutting plane algorithm used by Knippel and Lardeux and a branch-and-cut algorithm. Finally, section 3 presents our computational results, comparing CPLEX 11, the cutting plane algorithms and the branch-and-cut algorithm. In addition to better solution times, the branch-and-cut algorithm provides an optimality gap for difficult instances. We also show the improvement obtained by strengthening the metric inequalities.

2 Problem statement

2.1 Model description

The model described here aims to minimize the cost of capacities installed in both layers. There is no cost associated with the routing. First, we must route commodities given by the demand matrix in the virtual layer. This results in the installation of some capacities in that layer. Then, each virtual edge defines a commodity in the physical layer with a demand equal to the capacity installed on the virtual edge.

Hence, there is a strong interaction between the two layers. Two feasible solutions with the same virtual layer cost can have different overall costs since the cost of physical capacities can differ. This model is therefore more complex than the single layer capacitated network design model.

The two layers are represented by undirected graphs $G^{\text{virt}} = (V, F)$ and $G^{\text{phys}} = (V, E)$ constructed on the same node set $V$. Commodities $k$ to be routed in the virtual layer belong to the set $K$ and their values are denoted by $d_k$.

Our model uses an arc-path formulation for each layer. The objective (1) is to minimize the sum of the costs $a \kappa (\text{resp. } b \kappa)$ of the $x \kappa$ (resp. $y \kappa$) modules that are installed in the physical layer (resp. virtual layer), with modular capacity $D$ (resp. $C$).

As in the single-layer case, the sets of paths $P^\kappa$ in the virtual layer are indexed by the commodity $\kappa \in K$ to which they refer. Hence, variables $u^\kappa_p$ specify the portion of the demand $d_k$ routed on path $p \in P^\kappa$.

Recall that commodities to be routed on the physical layer are given by capacities installed in the virtual layer. Therefore, sets of paths $Q^f$ in the physical layers are indexed by virtual edges $f \in F$. Variable $v^f_q$ specifies the fraction of capacity $C y_f$, installed on link $f \in F$, routed on path $q \in Q^f$.

With this set of variables, the problem can be formulated as:

$$\begin{align*}
\text{min} \quad & \sum_{e \in E} a \kappa x \kappa + \sum_{f \in F} b \kappa y \kappa \\
\text{s.t.} \quad & \sum_{\kappa \in K} \sum_{p \in P^\kappa, p \ni f} u^\kappa_p \leq C y_f \quad \forall f \in F \quad (2) \\
& \sum_{p \in P^\kappa} u^\kappa_p = d_k \quad \forall \kappa \in K \quad (3) \\
& \sum_{f \in F} \sum_{q \in Q^f, q \ni e} v^f_q \leq D x \kappa \quad \forall e \in E \quad (4) \\
& \sum_{q \in Q^f} v^f_q = C y_f \quad \forall f \in F \quad (5) \\
& v^f_q, u^\kappa_p \geq 0 \\
& x \kappa, y \kappa \in \mathbb{Z}_+. \quad (6) \\
\end{align*}$$

Constraints (2) and (4) impose that the total flow on an edge is less than the capacity installed on that edge, whereas (3) and (5) ensure that all the demands are routed on the network. Integrality constraints (6) force capacities to be installed by modules. Finally, because routing variables are continuous (4), each commodity can be split among an arbitrary number of paths in each layer.

2.2 Benders decomposition

When facing a complex optimization problem, a classical approach is to project out complicating variables. This projection results in the addition of many additional constraints to the problem. For the network design model (AP), the result is the so-called capacitated formulation [14].
\[
\min \sum_{e \in E} a_e x_e + \sum_{f \in F} b_f y_f \\
(CP) \quad \text{s.t.} \quad x \in X_y, \quad y \in Y,
\]

where sets \(X_y\) and \(Y\) are defined by metric inequalities:

\[
X_y = \{x \in \mathbb{Z}^{|E|}_+ \mid \forall \lambda \in M_n, \sum_{(ij) \in E} \lambda_{ij} x_{ij} \geq C \sum_{(ij) \in F} \lambda_{ij} y_{ij}\} \tag{8}
\]

and

\[
Y = \{y \in \mathbb{Z}^{|F|}_+ \mid \forall \lambda \in M_n, \sum_{(ij) \in F} \lambda_{ij} y_{ij} \geq \sum_{i<j} \lambda_{ij} d_{ij}\}. \tag{9}
\]

where the metric cone (see [10]) \(M_n\) is defined by

\[
M_n = \{\lambda \in \mathbb{R}^{n(n-1)/2} \mid \lambda_{ij} \leq \lambda_{il} + \lambda_{lj}, \forall 1 \leq i < j \leq n, \forall 1 \leq l \leq n, j \neq l \neq i\}.
\]

This is one of the many applications of Benders decomposition to network design problems (see [2]). Note that the absence of costs for routing variables simplifies this decomposition. In the general case, projecting out a group of variables results in the addition of optimality constraints to the feasibility constraints of (8) and (9).

Metric inequalities (8) and (9) are weak when capacities are modular. A simple way of strengthening them without increasing the complexity of the separation algorithm is to round coefficients for constraints in (9)

\[
\sum_{f \in F} C \lambda_f y_f \geq d. \tag{10}
\]

If \(C \lambda_f\) is integer for each \(f \in F\), let \(\text{gcd}(C \lambda)\) be the greatest common divisor of those integers. Hence, dividing both sides of (10) by \(\text{gcd}(C \lambda)\) and rounding up \(d/\text{gcd}(C \lambda)\), we get the stronger cut

\[
\sum_{f \in F} C \lambda_f^{\ast} y_f \geq \left\lfloor \frac{d}{\text{gcd}(C \lambda)} \right\rfloor. \tag{11}
\]

We show in Section [2] the effect of these stronger cuts.

Note that Avella et al. [1] introduced the Tight Metric Inequalities, which completely describe \(Y\). However, since they are NP-hard to separate, we do not consider them in this paper.

### 3 Algorithms

This section describes two ways of managing the huge number of constraints introduced in (8) and (9). First, we recall a simple cutting plane framework. While easy to implement, this algorithm suffers the requirement of solving many integer programs to optimality. We then outline two improvements developed by Knippel and Lardeux [14], and following this, we present a new branch-and-cut algorithm for the problem. Comparative results of the three algorithms are given in Section [4].

#### 3.1 cutting plane approach

First, we get rid of the metric inequalities in (CP), resulting in the relaxed master problem

\[
w := \min \sum_{e \in E} a_e x_e + \sum_{f \in F} b_f y_f \\
(MP) \quad \text{s.t.} \quad x_e, y_f \in \mathbb{Z}_+.
\]

We can test whether a given integer vector \((x^\ast, y^\ast)\) is feasible for (CP) by solving the separation LP \(\text{Sat}(Cy^\ast, d)\) and \(\text{Sat}(Dx^\ast, Cy^\ast)\), with \(\text{Sat}(z, t)\) defined by

\[
\text{Sat}(z, t) := \min \sum_{i<j} \lambda_{ij} z_{ij} - \sum_{i<j} \lambda_{ij} t_{ij} \\
\text{s.t.} \sum_{1 \leq i < j \leq n} \lambda_{ij} = 1, \lambda \in M_n
\]

for any vectors \(z, t \in \mathbb{R}^{n(n-1)/2}\). Constraint (12) bounds the LP. If \(\text{Sat}(z, t) < 0\), the solution \(\lambda^\ast\) leads to a metric inequality violated by \((z, t)\):

\[
\sum_{i<j} \lambda_{ij} z_{ij} - \sum_{i<j} \lambda_{ij} t_{ij} \geq 0. \tag{13}
\]

On the other hand, if both \(\text{Sat}(Cy^\ast, d)\) and \(\text{Sat}(Dx^\ast, Cy^\ast)\) are non-negative, capacities \(x^\ast\) and \(y^\ast\) are feasible for problem (CP). This general procedure is described in Algorithm [1].

Algorithm 1 Cutting Plane Algorithm

\begin{algorithm}
\begin{algorithmic}
\State Initial cut pool \(P\) is empty.
\Repeat
\State \textbf{repeat}
\State Solve \((MP)\) augmented with cuts in \(P\). Let \((x^\ast, y^\ast)\) be an optimal solution.
\State Compute \(s_1 = \text{Sat}(Cy^\ast, d)\) and \(s_2 = \text{Sat}(Dx^\ast, Cy^\ast)\).
\State \If {\(s_1 < 0\) \text{ or } \(s_2 < 0\)}
\State \quad Add the corresponding cut(s) to \(P\).
\State \EndIf
\EndRepeat
\State \textbf{until} \(s_1 \geq 0\) and \(s_2 \geq 0\)
\State \Return \((x^\ast, y^\ast)\)
\end{algorithmic}
\end{algorithm}

In Algorithm [1] the solution time of \((MP)\) is usually much higher than the solution time of \(\text{Sat}\) because of the integrality restrictions in \((MP)\). Therefore, a common trend is to reduce the number of \((MP)\) solved. Following this observation, Knippel and Lardeux implemented two cutting plane algorithms based on Algorithm [1] (SC) Their single constraint generation adds up to three cuts per iteration. Besides the ones described in Algorithm [1] they also consider cuts coming from subproblem \(\text{Sat}(Dx^\ast, d)\):

\[
D \sum_{i<j} \lambda_{ij}^{\ast} x_{ij} - \sum_{i<j} \lambda_{ij}^{\ast} d_{ij} \geq 0. \tag{14}
\]

Although cuts (14) are not needed to ensure feasibility, they help to reduce the number of required iterations by forcing \(x\) to take sensible values, especially in the first few iterations.
3.2 Branch-and-cut approach

An alternative strategy is to solve only one (MP). We aim to embed the generation of violated feasibility cuts (13) into the branch-and-cut framework for solving (MP). This is detailed in Algorithm 2. Before starting the branch-and-cut, we need to set up a cut pool $P$.

It is important to add many cuts early in the tree to avoid exploration of too many infeasible nodes. For instance, some tests have been made starting with an empty cut pool $P$. This resulted in a very slow branch-and-cut because of some infeasible nodes being fathomed only late in the search. However, adding too many unnecessary cuts would slow down the LP relaxation at each node. A good starting cut pool is obtained by solving the LP relaxation of (MP), with the cutting planes described in Algorithm 1; then setting up $P$ contains all constraints added to solve the LP relaxation. Note that, in contrast to SC and MC, experiments have shown that using cuts of type (14) increases the total resolution time. Thus, we do not generate cuts (14) during the cutting plane setting up $P$, nor during Algorithm 2.

In Algorithm 2 solving a node $o' \in T$ means solving the LP relaxation of (MP), augmented with branching constraints of $o'$ and cuts in pool $P$.

Algorithm 2 Branch-and-cut Framework (B&C) 

**Require**: A starting cut pool $P$.

*Initialize the tree: $T = \{ o \}$ where $o$ has no branching constraints; $\overline{w} := +\infty$.*

**while** $T$ is nonempty **do**

Select a node $o' \in T$.

$T := T \setminus \{ o' \}$

Solve $o'$. Let $(x^*, y^*)$ be an optimal solution and $w^*$ the optimal cost.

if $w^* < \overline{w}$ **then**

Branch, resulting in nodes $o^*$ and $o^{**}$, $T := T \cup \{ o^*, o^{**} \}$.

else

Compute $s_1 := Sat(Cy^*, d)$ and $s_2 := Sat(Dx^*, Cy^*)$.

if $s_1 < 0$ or $s_2 < 0$ **then**

Add the corresponding cut(s) to $P$.

$T := T \cup \{ o' \}$

else

Define a new upper bound $\overline{w} := w^*$ and save current solution, $(\overline{x}, \overline{y}) := (x^*, y^*)$.

**end if**

**end if**

**end while**

**return** $(\overline{x}, \overline{y})$

for single-layer networks, that is

\[
\begin{align*}
\text{min} & \quad a^t x + b^t y \\
\text{s.t.} & \quad \sum_{k \in K} u_k^+ + u_k^- \leq Cy \\
& \quad B(u_k^+ - u_k^-) = \tilde{d}_k, \quad \forall k \in K \\
& \quad \sum_{f \in F} v_f^+ + v_f^- \leq Dx \\
& \quad A(v_f^+ - v_f^-) = C\tilde{y}_f, \quad \forall f \in F,
\end{align*}
\]

where $x$ and $y$ are the capacity variables as before, $u$ and $v$ the flow variables on each arc in both directions and for each commodity, and $A$ and $B$ are the arc-node incidence matrices for each layer. $\tilde{d}_k$ (resp. $\tilde{y}_f$) take values $d_k$, $-d_k$ or 0 (resp. $y_f$, $-y_f$ or 0), depending on whether the considered node is one of the extremes of demand. This formulation considers implicitly that sets $P^*$ and $Q^*$ contain all possible paths for each commodity $k$ and virtual edge $f$, so that (AN) and (AP) solve the same problem [3]. However, some tests have proven the MIP solver of CPLEX 11 to solve (AN) faster than (AP) considering all paths.

Finally, we show on harder instances the improvement obtained using strengthened cuts (11) instead of standard metric inequalities.

4.1 Implementation details

All models have been written in JAVA, and the CPLEX MIP solver is used with default settings both for solving (MP) in Algorithm 1 and $Sat$ in both algorithms.
<table>
<thead>
<tr>
<th>Instances</th>
<th>Time/Gap</th>
<th>Time</th>
<th>Cuts generated</th>
<th>Iterations</th>
<th>Explored nodes</th>
<th>Times ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 8 14</td>
<td>1.95%</td>
<td>9.2</td>
<td>9.1</td>
<td>0.6</td>
<td>76</td>
<td>155</td>
</tr>
<tr>
<td>2 8 14</td>
<td>26%</td>
<td>3.9</td>
<td>3.8</td>
<td>3.9</td>
<td>76</td>
<td>177</td>
</tr>
<tr>
<td>3 8 14</td>
<td>4%</td>
<td>3.2</td>
<td>3.1</td>
<td>3.1</td>
<td>101</td>
<td>168</td>
</tr>
<tr>
<td>4 8 14</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>78</td>
<td>202</td>
</tr>
<tr>
<td>5 8 14</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>63</td>
<td>144</td>
</tr>
<tr>
<td>6 8 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>88</td>
<td>183</td>
</tr>
<tr>
<td>7 8 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>83</td>
<td>187</td>
</tr>
<tr>
<td>8 8 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>86</td>
<td>186</td>
</tr>
<tr>
<td>9 8 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>121</td>
<td>189</td>
</tr>
<tr>
<td>10 8 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>90</td>
<td>171</td>
</tr>
<tr>
<td>11 9 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>119</td>
<td>303</td>
</tr>
<tr>
<td>12 9 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>115</td>
<td>310</td>
</tr>
<tr>
<td>13 9 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>149</td>
<td>272</td>
</tr>
<tr>
<td>14 9 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>127</td>
<td>275</td>
</tr>
<tr>
<td>15 9 16</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>112</td>
<td>315</td>
</tr>
<tr>
<td>16 9 18</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>149</td>
<td>328</td>
</tr>
<tr>
<td>17 9 18</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>129</td>
<td>299</td>
</tr>
<tr>
<td>18 9 18</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>111</td>
<td>261</td>
</tr>
<tr>
<td>19 9 18</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>92</td>
<td>225</td>
</tr>
<tr>
<td>20 9 18</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>160</td>
<td>286</td>
</tr>
<tr>
<td>21 9 20</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>100</td>
<td>261</td>
</tr>
<tr>
<td>22 9 20</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>131</td>
<td>341</td>
</tr>
<tr>
<td>23 9 20</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>130</td>
<td>266</td>
</tr>
<tr>
<td>24 9 20</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>–</td>
<td>290</td>
</tr>
<tr>
<td>25 9 20</td>
<td>18.6%</td>
<td>18.5</td>
<td>18.5</td>
<td>0.2</td>
<td>113</td>
<td>227</td>
</tr>
</tbody>
</table>

Table 1: Results of CPLEX, SC, MC and B&C on randomly generated instances with 8 and 9 nodes.

4.3 Results

We fix a time limit of 3600 seconds for instances with 8 and 9 nodes. The corresponding Time/Gap column gives either the solution time in seconds or the gap when the time limit is reached. For instances 26–35 and the ones from SNDlib, we allow up to 18000 seconds, reporting the status after 3600 seconds. The reported solution times are for the whole durations. Underlined gaps indicate memory overflows.

We can see in Table 1 that SC, MC and B&C outperform CPLEX by far. B&C is always faster than both SC and MC. The ratio between the solution time of B&C and the one of the faster cutting plane algorithm ranges from 2.2 to 34.4 with a geometric average of 10.7. This is explained by the high number of iterations performed by both cutting plane algorithms, where each iteration is required to solve an IP to optimality. However, the ratio is still far from the number of iterations, since many of the iterations contain only a few cuts.

B&C usually generates more cuts than SC, even though SC generates cuts of type \([14]\). Thus, many of these cuts are not needed to ensure the feasibility of the solution. Hence more efficient management of the cut pool, eliminating the non active cuts, may improve Algorithm 2.

The relative performance of SC and MC is as expected: MC adds many more cuts than SC, resulting in fewer iterations and shorter solution times. See \([14]\) for a more detailed comparison of SC and MC.

Note that the cutting plane algorithms were unable to solve any of the larger instances within 18000 seconds. Hence, in Tables 3, 4 and 5 we compare CPLEX and B&C with normal and rounded cuts \((10)\) and \((11)\), respectively) for those instances. Although NC and RC beat CPLEX for most instances, the difference is much
smaller than it is for easier instances from Table 1.

Results from Table 4 show that CPLEX explores hundreds of thousands of nodes, whereas both NC and RC explore millions of them. Note that the number of nodes explored by B&C grows rapidly with the problem size. CPLEX, however, manages to compute good bounds for hard instances, while exploring a relatively small tree.

Table 2: Number of cuts generated by B&C with normal and rounded cuts (NC and RC respectively) at the different steps of Algorithm 2, on randomly generated instances with 10 nodes.

<table>
<thead>
<tr>
<th>Inst</th>
<th>Initial cuts</th>
<th>3600 seconds</th>
<th>18000 seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NC</td>
<td>RC</td>
<td>NC</td>
</tr>
<tr>
<td>26</td>
<td>111</td>
<td>145</td>
<td>197</td>
</tr>
<tr>
<td>27</td>
<td>130</td>
<td>165</td>
<td>265</td>
</tr>
<tr>
<td>28</td>
<td>142</td>
<td>204</td>
<td>305</td>
</tr>
<tr>
<td>29</td>
<td>162</td>
<td>222</td>
<td>187</td>
</tr>
<tr>
<td>30</td>
<td>124</td>
<td>165</td>
<td>267</td>
</tr>
<tr>
<td>31</td>
<td>144</td>
<td>161</td>
<td>544</td>
</tr>
<tr>
<td>32</td>
<td>127</td>
<td>222</td>
<td>398</td>
</tr>
<tr>
<td>33</td>
<td>131</td>
<td>165</td>
<td>231</td>
</tr>
<tr>
<td>34</td>
<td>122</td>
<td>179</td>
<td>391</td>
</tr>
<tr>
<td>35</td>
<td>129</td>
<td>187</td>
<td>347</td>
</tr>
</tbody>
</table>

Table 3: Solution times for CPLEX, NC and RC.

<table>
<thead>
<tr>
<th>Inst</th>
<th>CPLEX 3600 seconds</th>
<th>CPLEX 18000 seconds</th>
<th>NC</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>745</td>
<td>2512</td>
<td>745</td>
<td>2512</td>
</tr>
<tr>
<td>27</td>
<td>77801</td>
<td>4109</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>1144670</td>
<td>712226</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>29</td>
<td>533749</td>
<td>77801</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>375385</td>
<td>7802</td>
<td>7777</td>
<td>9355</td>
</tr>
<tr>
<td>31</td>
<td>1144670</td>
<td>77801</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td>533749</td>
<td>7802</td>
<td>7777</td>
<td>9355</td>
</tr>
<tr>
<td>33</td>
<td>144441</td>
<td>77801</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>34</td>
<td>1144670</td>
<td>77801</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>533749</td>
<td>7802</td>
<td>7777</td>
<td>9355</td>
</tr>
</tbody>
</table>

Table 4: Number of explored nodes by CPLEX, NC and RC.

<table>
<thead>
<tr>
<th>Inst</th>
<th>CPLEX 3600 seconds</th>
<th>CPLEX 18000 seconds</th>
<th>NC</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>111</td>
<td>145</td>
<td>197</td>
<td>259</td>
</tr>
<tr>
<td>27</td>
<td>120</td>
<td>165</td>
<td>265</td>
<td>334</td>
</tr>
<tr>
<td>28</td>
<td>132</td>
<td>204</td>
<td>305</td>
<td>366</td>
</tr>
<tr>
<td>29</td>
<td>160</td>
<td>222</td>
<td>187</td>
<td>321</td>
</tr>
<tr>
<td>30</td>
<td>122</td>
<td>165</td>
<td>267</td>
<td>328</td>
</tr>
<tr>
<td>31</td>
<td>142</td>
<td>161</td>
<td>544</td>
<td>368</td>
</tr>
<tr>
<td>32</td>
<td>127</td>
<td>222</td>
<td>398</td>
<td>332</td>
</tr>
<tr>
<td>33</td>
<td>131</td>
<td>165</td>
<td>231</td>
<td>242</td>
</tr>
<tr>
<td>34</td>
<td>122</td>
<td>179</td>
<td>391</td>
<td>216</td>
</tr>
<tr>
<td>35</td>
<td>129</td>
<td>187</td>
<td>347</td>
<td>225</td>
</tr>
</tbody>
</table>

Table 5: Solution times for CPLEX, NC and RC on instances based on networks from SNDlib.

<table>
<thead>
<tr>
<th>Inst</th>
<th>Time/Gap (limit 3600s)</th>
<th>Time/Gap (limit 18000s)</th>
<th>NC</th>
<th>RC</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>3.43%</td>
<td>0.53%</td>
<td>745</td>
<td>2512</td>
</tr>
<tr>
<td>27</td>
<td>4.33%</td>
<td>1.22%</td>
<td>1965.7</td>
<td>77801</td>
</tr>
<tr>
<td>28</td>
<td>4.88%</td>
<td>5.09%</td>
<td>2324.6</td>
<td>7802</td>
</tr>
<tr>
<td>29</td>
<td>1.19%</td>
<td>4.23%</td>
<td>7802</td>
<td>7777</td>
</tr>
<tr>
<td>30</td>
<td>2.76%</td>
<td>3.66%</td>
<td>3.96%</td>
<td>9355</td>
</tr>
<tr>
<td>31</td>
<td>2.82%</td>
<td>2.69%</td>
<td>2.63%</td>
<td>9355</td>
</tr>
<tr>
<td>32</td>
<td>5.13%</td>
<td>6.5%</td>
<td>3.98%</td>
<td>9355</td>
</tr>
<tr>
<td>33</td>
<td>1.16%</td>
<td>1.05%</td>
<td>8301</td>
<td>5076</td>
</tr>
<tr>
<td>34</td>
<td>3.3%</td>
<td>2.5%</td>
<td>3.35%</td>
<td>9963</td>
</tr>
<tr>
<td>35</td>
<td>2.77%</td>
<td>1.37%</td>
<td>3.35%</td>
<td>9963</td>
</tr>
</tbody>
</table>

References


Acknowledgements

This research is supported by an “Actions de Recherche Concertées” (ARC) projet of the “Communauté française de Belgique”. Michael Poss is a research fellow of the “Fonds pour la Formation à la Recherche dans l’Industrie et dans l’Agriculture” (FRIA). The authors also acknowledge two anonymous referees for their constructive comments on a first version of the paper.


