An improved benders decomposition applied to a multi-layer network design problem

Bernard Fortz, Michael Poss

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 Wessäly [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.

*Corresponding author. Email: mposs@ulb.ac.be. Adress: Boulevard du Triomphe CP 210 / 01, B-1050 Brussels, Belgium.

Kubilinskas 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.

Höller 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 [14] 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 [14]. 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 3, 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 4 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 $G^{virt} = (V, F)$ and $G^{phys} = (V, E)$ constructed on the same node set V. Commodities k to be routed in the virtual layer belong to the set $\mathcal{K}$ 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_e (resp. b_f) of the x_e (resp. y_f) 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 $\mathcal{P}^k$ in the virtual layer are indexed by the commodity $k \in \mathcal{K}$ to which they refer. Hence, variables u_p^k specify the portion of the demand d_k routed on path $p \in \mathcal{P}^k$.

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

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

$$
\begin{aligned}
(AP) \qquad \min \quad & \sum_{e \in E} a_e x_e + \sum_{f \in F} b_f y_f && && (1)\\
\text{s.t.} \quad & \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}^k : p \ni f} u_p^k \le C y_f && \forall f \in F && (2)\\
& \sum_{p \in \mathcal{P}^k} u_p^k = d_k && \forall k \in \mathcal{K} && (3)\\
& \sum_{f \in F} \sum_{q \in \mathcal{Q}^f : q \ni e} v_q^f \le D x_e && \forall e \in E && (4)\\
& \sum_{q \in \mathcal{Q}^f} v_q^f = C y_f && \forall f \in F && (5)\\
& v_q^f,\ u_p^k \ge 0 && && (6)\\
& x_e,\ y_f \in \mathbb{Z}_+. && && (7)
\end{aligned}
$$

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 (7) force capacities to be installed by modules. Finally, because routing variables are continuous (6), 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].

$$
\begin{array}{lll}
 & \min & \sum_{e\in E} a_e x_e + \sum_{f\in F} b_f y_f \\
(CP) & \text{s.t.} & x \in X_y \\
 & & y \in Y,
\end{array}
$$

where sets X_y and Y are defined by metric inequalities:

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

and

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

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

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

This is one of the many applications of Benders decomposition to network design problems (see [7]). 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 $\gcd(C\lambda)$ be the greatest common divisor of those integers. Hence, dividing both sides of (10) by $\gcd(C\lambda)$ and rounding up $d/\gcd(C\lambda)$, we get the stronger cut

$$\sum_{f\in F} \frac{C\lambda_f}{div} y_f \geq \left\lceil \frac{d}{\gcd(C\lambda)} \right\rceil. \tag{11}$$

We show in Section 4 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

$$\begin{array}{lll}
w := \min & \sum_{e\in E} a_e x_e + \sum_{f\in F} b_f y_f \\
(MP) \qquad \text{s.t.} & x_e,\ y_f \in \mathbb{Z}_+.
\end{array}$$

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

$$\begin{array}{lll}
Sat(z,t) := \min & \sum_{i<j} \lambda_{ij} z_{ij} - \sum_{i<j} \lambda_{ij} t_{ij} \\
\text{s.t.} & \sum_{1\leq i<j\leq n} \lambda_{ij} = 1, \qquad (12) \\
 & \lambda \in M_n
\end{array}$$

for any vectors $z, t \in \mathbb{R}^{n(n-1)/2}$. Constraint (12) bounds the LP. If $Sat(z, t) < 0$, the solution λ^* 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 $Sat(Cy^*, d)$ and $Sat(Dx^*, Cy^*)$ are non-negative, capacities x^* and y^* are feasible for problem (CP). This general procedure is described in Algorithm 1.

Algorithm 1 Cutting Plane Algorithm

```
Initial cut pool P is empty.
repeat
  Solve (MP) augmented with cuts in P. Let (x*, y*) be an optimal solution.
  Compute s1 = Sat(Cy*, d) and s2 = Sat(Dx*, Cy*).
  if s1 < 0 or s2 < 0 then
    Add the corresponding cut(s) to P.
    Optional: Add problem-specific cuts to P.
  end if
until s1 ≥ 0 and s2 ≥ 0
return (x*, y*)
```

In Algorithm 1, the solution time of (MP) is usually much higher than the solution time of 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 $Sat(Dx^*, d)$:

$$D \sum_{i<j} \lambda^*_{ij} x_{ij} - \sum_{i<j} \lambda^*_{ij} d_{ij} \geq 0. \tag{14}$$

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.

(MC) Aiming to reduce the iterations of Algorithm 1 even further, they introduce *multiple constraint generation*. This algorithm adds the same cuts as SC plus a few bipartition inequalities violated by (x^*, y^*), at each iteration.

Another situation occurs when the subproblems are separable, i.e., they can be decomposed into several problems, the solution of which results in the addition of a cut to (MP). See, for example, the multi-cut L-shaped algorithm for stochastic programming problems with recourse [5].

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 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 $o' \in T$.
    $T := T \backslash \{o'\}$
    Solve o'. Let (x^*, y^*) be an optimal solution and w^* the optimal cost.
    if $w^* < \overline{w}$ **then**
      if (x^*, y^*) is fractional **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 if
  end while
  return $(\overline{x}, \overline{y})$

4 Computational Experiments

In this section, we compare resolution times and number of constraints generated for cutting plane algorithms SC and MC, and the branch-and-cut algorithm B&C. The solution times given for B&C contain the generation of the cut pool by Algorithm 1 plus the time spent in the branch-and-cut from Algorithm 2. Even though an important fraction of the cuts are generated by Algorithm 1, its duration is much shorter than the duration of Algorithm 2, except for instances with 8 nodes for which both durations are of the same order.

We also compare these results with an arcs-nodes formulation solved by the standard MIP solver of CPLEX 11, which we denote by AN-CPLEX (or just CPLEX) in what follows. Such a formulation is very similar to that used for single-layer networks, that is

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

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 $\mathcal{P}^k$ and $\mathcal{Q}^f$ contain all possible paths for each commodity k and virtual edge f, so that (AN) and (AP) solve the same problem [8]. 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.

Instances			Time/Gap	Time			Cuts generated			Iterations		Explored nodes		Times ratio
in	n	e	CPLEX	SC	MC	B&C	SC	MC	B&C	SC	MC	CPLEX	B&C	min(SC, MC)/B&C
1	8	14	433	32.7	13.1	0.7	76	155	84	52	38	110345	1911	18.7
2	8	14	*1.95%*	9.2	9.1	0.6	76	177	85	46	33	926934	1886	15.1
3	8	14	3508	29.9	31.8	4.1	101	168	115	48	32	668218	13259	7.3
4	8	14	387	33.1	27.5	0.8	78	202	84	45	37	83482	1989	**34.4**
5	8	14	183	4.9	4.2	0.2	63	144	71	34	23	37388	76	21
6	8	16	984	54.1	37.9	1.2	88	183	92	54	30	184589	3282	31.2
7	8	16	508	33.8	15.4	0.8	83	187	88	39	24	118575	2310	19.3
8	8	16	2434	21.2	34.4	4.9	86	186	89	63	45	659259	26834	4.3
9	8	16	856	104.2	66.6	3.0	121	189	118	58	29	118797	8862	22.2
10	8	16	3487	33.3	16.2	1.8	90	171	99	54	26	803442	6099	9
11	9	16	2002	363.8	170.0	12.3	119	303	128	87	50	231606	42675	13.8
12	9	16	3063	217.8	244.5	15.5	115	310	186	68	43	284648	36745	14.1
13	9	16	538	226.2	264.0	14.3	149	272	156	95	39	58767	38257	15.8
14	9	16	*4.16%*	1639.2	450.1	25.3	127	275	156	85	46	232556	74618	17.8
15	9	16	*0.72%*	125.5	67.6	4.7	112	315	104	73	42	318587	14759	14.4
16	9	18	*2.93%*	190.5	143.3	66.1	149	328	164	103	47	420776	241323	**2.2**
17	9	18	*3.94%*	529.6	272.2	14.8	129	299	147	72	40	340148	39818	18.4
18	9	18	*1.54%*	109.3	55.2	8.4	111	261	154	66	45	293697	19807	6.6
19	9	18	539	60.0	21.9	0.9	92	225	114	56	28	55293	1461	24.3
20	9	18	*0.63%*	425.6	224.1	78.0	160	286	192	79	41	265740	143706	2.9
21	9	20	*1.96%*	67.2	53.3	13.7	100	261	105	60	40	313350	58887	3.9
22	9	20	*3.35%*	415.0	201.2	62.8	131	341	165	83	48	282078	209571	3.2
23	9	20	*6.07%*	293.8	67.7	28.3	130	266	155	83	36	283205	91849	2.4
24	9	20	*3.1%*	–	730.7	66.4	–	290	187	–	47	266132	174850	11.0
25	9	20	*2.32%*	193.2	217.3	12.0	113	227	121	72	41	371485	44940	16.1

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

CPLEX chose to use the dynamic search for both cutting plane algorithms and for AN-CPLEX. Note that we also solved AN-CPLEX using the standard enumeration. This resulted in larger gaps for hard instances, whereas the easy ones were solved in more or less the same amount of time.

Although CPLEX 11 solves B&C as well, we use CutCallback, IncumbentCallback and BranchCallback to implement the different steps of Algorithm 2, which suppresses the dynamic search. Then we keep default parameters for node selections, branching rules and generation of cuts, unless a cut (13) is added, in which case the cut is added as a global cut and as a branching constraint. These programs were run on a HP Compaq 6510b with an Intel Core 2 Duo processor at 2.40 GHz and 2 GB of RAM memory.

4.2 Instances

The first set of 35 instances are randomly generated and share the next features: $C = 64$, $D = 128$, $G^{up} = (V, F)$ is a complete graph. Demands are random integers uniformly generated between 0 and 64 for each pair of nodes and the cost of any edge $e \in E \cup F$ is based on the distance between the extremities of e.

The following six instances are based on networks from *SNDlib* [19], which have been taken as physical layers; virtual layers are complete graphs. The matrix demand taken from SNDlib contains a demand for each pair of nodes in all instances but pdh and di-yuan.

In both sets of instances, the costs linked to both layers are of the same order of magnitude and no upper bounds are imposed on the capacities.

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.

Instances		Initial cuts		3600 seconds		18000 seconds	
in	e	NC	RC	NC	RC	NC	RC
26	20	111	145	197	129	67	–
27	20	105	145	265	240	12	–
28	20	142	204	305	266	40	19
29	20	105	162	373	322	14	6
30	20	124	165	267	120	–	–
31	25	144	161	544	368	30	37
32	25	127	222	398	332	7	22
33	25	131	165	231	242	8	11
34	25	122	179	391	216	7	–
35	25	129	187	347	225	37	70

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.

Inst	Time/Gap (limit 3600s)			Time/Gap (limit 18000s)		
	CPLEX	NC	RC	CPLEX	NC	RC
26	*3.43%*	*0.53%*	745	*2.71%*	4109	–
27	*4.33%*	*1.22%*	1965.7	*2.2%*	77801	–
28	*4.88%*	*5.09%*	*2.86%*	*4.06%*	*4.72%*	*0.55%*
29	*1.19%*	*4.23%*	*1.37%*	7802	*2.77%*	9355
30	*2.76%*	368.9	79	*1.41%*	–	–
31	*6.43%*	*3.66%*	*4.54%*	*3.96%*	*2.47%*	*4.1%*
32	*2.82%*	*2.69%*	*0.79%*	*2.03%*	*2.37%*	5032
33	*5.13%*	*0.65%*	*1.09%*	*3.98%*	4768	5922
34	*1.16%*	*1.05 %*	1447	8301	5076	–
35	*3.3%*	*2.5%*	*1.37%*	*2.35%*	*1.98%*	9963

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

Inst	3600 seconds			18000 seconds		
	CPLEX	NC	RC	CPLEX	NC	RC
26	148455	5352019	1214254	747087	6377687	–
27	139125	5722520	2716054	714667	11484670	–
28	122756	2846870	23400446	648448	5337499	13068541
29	144445	284670	2866228	375985	5337499	8042525
30	148609	408489	103577	820049	–	–
31	105340	3112489	3618044	575839	5179420	5469716
32	115136	3471580	3174756	657367	5552061	4553309
33	125850	3917977	4222550	701442	5563211	7104511
34	150276	2676776	1248257	404154	3979981	–
35	147617	3514051	4155845	783877	6664584	11265256

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

Instances			T/G (limit 3600s)			T/G (limit 18000s)		
in	n	e	CPLEX	NC	RC	CPLEX	NC	RC
pdh	11	34	*3.07%*	*0.84%*	3343.4	*2.53%*	7230	–
di-yuan	11	42	*1.38%*	*1.87%*	*1.9%*	10450	8710	9586
dfn-gwin	11	47	*3.77%*	*1.13%*	*1.18%*	*3.36%*	*0.54%*	*0.98%*
polska	12	18	*3.66%*	*0.32%*	*0.47%*	*2.45%*	*0.32%*	*0.23%*
nobel-us	14	21	*8.02%*	*8.01%*	*6.79%*	*3.92%*	*1.03%*	*1.51%*
atlanta	15	22	*3.92%*	314	707.9	*0.09%*	–	–

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

Acknowledgements

This research is supported by an "Actions de Recherche Concertées" (ARC) projet of the "Communautée 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.

References

[1] P. Avella, S. Mattia, and A. Sassano, *Metric inequalities and the network loading problem*, Discrete Optimization **4** (2007), 103–114.

[2] G. Baier, T. Engel, A. M. C. A. Koster, S. Orlowski, C. Raack, and R. Wessäly, *Single-layer cuts for multi-layer network design problems*, ZIB Report ZR-07-21, August 2007.

[3] P. Belotti, A. Capone, G. Carello, F. Malucelli, F. Senaldi, and A. Totaro, *Mpls over transport network: Two layers approach to network design with statistical multiplexing*, Conference on Next Generation Internet Design and Engineering (NGI 2006), Valencia (Spain), April 2006.

[4] P. Belotti and F. Malucelli, *Row-column generation for multilayer network design*, Proceedings, International Network Optimization Conference, 2005, Lisbon, Portugal, March 2005.

[5] J. R. Birge and F. V. Louveaux, *Introduction to stochastic programming (2nd edition)*, Springer Verlag, New-York, 2008.

[6] A. Capone, G. Carello, and R. Matera, *Multi-layer network design with multicast traffic and statistical multiplexing*, IEEE GLOBECOM 2007, Washington DC, USA, December 2007.

[7] A. M. Costa, *A survey on benders decomposition applied to fixed-charge network design problems*, Comput. Oper. Res. **32** (2005), no. 6, 1429–1450.

[8] T. G. Crainic and B. Gendron, *Relaxations for multicommodity capacitated network design problems*, Tech. Report publication CRT-965, Centre de recherche sur les transports, Université de Montréal, 1994.

[9] G. Dahl, A. Martin, and Mechthild Stoer, *Routing through virtual paths in layered telecommunication networks*, Oper. Res. **47** (1999), no. 5, 693–702.

[10] M. Deza and M. Laurent, *Geometry of cuts and metrics*, vol. 15, Springer, 1997.

[11] V. Gabrel, A. Knippel, and M. Minoux, *Exact solution of multicommodity network optimization problems with general step cost functions*, Operations Research Letters **25** (August 1999), 15–23(9).

[12] J. Geffard, B. Lardeux, and D. Nace, *Multiperiod network design with incremental routing*, Netw. **50** (2007), no. 1, 109–117.

[13] H. Holler and S. Voss, *A heuristic approach for combined equipment-planning and routing in multilayer sdh/wdm networks*, European Journal of Operational Research **127** (2006), no. 3, 787–796.

[14] A. Knippel and B. Lardeux, *The multi-layered network design problem*, European Journal of Operational Research **127** (2007), no. 1, 87–99.

[15] A. Knippel, B. Lardeux, and J. Geffard, *Efficient algorithms for solving the 2-layered network design problem*, Proceedings of INOC, Paris, 2003.

[16] A. M. C. A. Koster, S. Orlowski, C. Raack, and R. Wessäly, *Two-layer network design by branch-and-cut featuring MIP-based heuristics*, Proceedings of INOC, Spa, Belgium, April 2007.

[17] A. M. C. A. Koster, S. Orlowski, C. Raack, and R. Wessäly, *Capacitated network design using general flow-cutset inequalities*, Submitted to Networks. ZIB Report 07-14, 2007.

[18] E. Kubilinskas and M. Pióro, *An ip/mpls over wdm network design problem*, In Proceedings of INOC, Lisbon, vol. 3, 2005.

[19] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, *SNDlib 1.0–Survivable Network Design Library*, Proceedings of INOC, Spa, Belgium, April 2007, http://sndlib.zib.de.

[20] S. Orlowski and R. Wessäly, *An integer programming model for multi-layer network design*, ZIB Preprint ZR-04-49, December 2004.

[21] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*, Elsevier, 2004.

[22] D. Yuan, *An annotated bibliography in communication network design and routing*, Ph.D. thesis, Institute of Technology, Linköpings Universitet, 2001.