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

B. Fortz† and M. Poss‡

November 10, 2008

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 in a layered manner, according to different technologies. For instance, one could consider a virtual layer (IP/MPLS) over a physical layer (WDM), 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 that layer (virtual), leading to the installation of "virtual capacities" (which are routers or other devices). Now, these capacities define demands for the transport layer. This leads again to installation of capacities (optical fiber, copper link, ...), in the physical layer this time. 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 "grooming path").

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

- A first network design problem is solved for the virtual layer only.
- Then, virtual capacities installed in the virtual layer define demands for another network design problem, for the transport layer this time.

Even though much easier to solve, this relaxed approach might provide solutions far from the optimal solution of the problem. Therefore it is more and more thought that an integrated approach should be considered.

There has been a huge number of papers on network design [23] since many years. However the interest in multi-layered network design is more recent, starting with a paper of Dahl and Stoer [10]. This paper assumes given physical capacities and aims 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 that paper, the interest in this field has rapidly grown and different approaches have been proposed for those problems.

Orlowski and Wessäly [22] begin by a good introduction to multi-layered networks where they give technical examples and develop a very complete model. However no precise solution method is given. In further papers with Koster *et al.*, they develop different branch-and-cut approaches. First a cut-and-branch-and-price is briefly described [4], extending a paper of Belotti *et al.* [5]. Then an integer formulation is solved through a branch-and-cut framework [18], where they introduce efficient heuristics. Finally, they address a more complex formulation, taking node hardware and survivability into account [2]. They extend and test different sorts of cuts coming from mono-layer models [19].

Capone *et al.* [3, 7] have studied multi-layered design with statistical multiplexing: routing different commodities on the same capacities results in cancellations of those demands variations . They compute a lower bound through a lagrangian relaxation, and use heuristics to find good upper bounds.

Kubilinskas and Pioro [20] address a problem of maximizing the profit of satisfying demands in a two layers (IP/MPLS over WDM) situation. They present then an iterative procedure to solve their complex MIP. This procedure consists in splitting the problems into two stages, one for each layer, where the solution of the first layer defines demands in the second one and where the routing solution in the physical layer leads to cost modification for edges in the first layer.

Höller and Voß [14] propose an integer programming formulation for two-layers network consisting in 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 this problem using two different heuristics.

Gabrel *et al.* introduce a constraint-generation procedure based on a Benders decomposition for capaci-

*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 á la Recherche dans l'Industrie et dans l'Agriculture" (FRIA).

†Department of Computer Science, Université Libre de Bruxelles, Brussels, Belgium and CORE, Université catholique de Louvain, Louvain-la-Neuve, Belgium. Email: bfortz@ulb.ac.be

‡Department of Computer Science, Université Libre de Bruxelles, Brussels, Belgium. Email: mposs@ulb.ac.be

tated network design problems [12]. Then, Knippel and Lardeux extend this work to multi-layered networks [16] and multi-period time scheduling [13]. Besides they introduce the metric cone, which eases cuts generation. Then, they improve this method, using the knapsack-like structure of the master problem to facilitate its resolution [17]. Finally, Knippel proposes improvements for the separation problem and tests his improvements on single layer networks [15].

In this paper, we improve the constraints generation method used by Knippel and Lardeux [16]. Namely, we develop a branch-and-cut to solve the Benders decomposition of the problem. This speeds up the resolution times by a factor of 10 on average. Besides, we get bounds for hard problems, whereas the cutting plane from Knippel does never compute upper bounds. Finally, our work presents only a branch-and-cut framework which is compatible with problem-specific cuts. This framework could easily be extended to improve results on the multi-period network design problem solved by Lardeux [13].

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 planes algorithm used by Knippel and Lardeux, and a branch-and-cut algorithm. Finally, computational results are given in Section 4, comparing CPLEX 11, the cutting planes algorithms and the branch-and-cut. Besides better solution times, the branch-and-cut algorithm provides an optimality gap for hard instances, whereas cutting planes algorithms cannot compute upper bounds. We also show the improvement obtained by strengthening metric inequalities.

2 Problem statement

2.1 Model description

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

Hence, there is a strong interaction between both layers. Two feasible solutions whose virtual layer costs are equal can have different overall cost, because the cost of physical capacities are different. This model is therefore more complex that 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.

The model given hereby presents 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).

Like in the single layer case, set 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 portion of the capacity Cy_f installed on link $f \in F$ routed on path $q \in \mathcal{Q}^f$.

In the following model, (2) and (4) imply that the total flow on one edge is less than the capacity installed on that edge, whereas (3) and (5) ensure that the whole demand of each commodity is routed among its paths. Integrality constraints (7) force capacities to be installed by modules. Finally, because routing variables are real (6), each commodity can be split among an arbitrary number of paths, in each layer.

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

2.2 Benders decomposition

When facing a complex optimization problem, a classical approach is to project out complicating variables. This results adding many additional constraints to the problem. For the network design model (AP) this results in the so-called capacitated formulation [16], introducing metric inequalities in (8) and (9):

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

where sets X_y and Y are defined as follows:

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

and

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

where the metric cone (see [11]) M_n is defined by

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

This is one of the many applications of Benders decomposition to network design problems, see [8]. Note that the absence of costs for routing variables simplifies this decomposition. In the general case, projecting out a group a variables results in addition of optimality constraints besides feasibility constraints from (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 div be the greatest common divisor of those integers. Hence, dividing both sides of (10) by div and rounding up d/div, we get a stronger cut

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

We show in Tables 2 and 3 the effects of these stronger cuts.

Note that Avella *et al.* [1] introduced the *Tight Metric Inequalities* which completely describe Y. However, there are NP-hard to separate so that 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 planes framework. Easy to implement, this algorithm suffers from solving many integer programs to optimality. Thus, we cite two improvements developed by Knippel and Lardeux [16]: SC and MC. Then we describe a new branch-and-cut algorithm for the problem, B&C. Comparative results of the three algorithms are given in Section 4.

3.1 Cutting planes approach

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

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

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

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

for any vectors $z, t \in \mathbb{R}^{n(n-1)/2}$. Constraint (12) bounds the LP. Then, if $Sat(z,t) < 0$, the solution λ^* gives a metric inequality violated by (z,t):

$$\sum_{i<j} \lambda^*_{ij} z_{ij} - \sum_{i<j} \lambda^*_{ij} t_{ij} \geq 0. \tag{13}$$

In the other hand, if both $Sat(Cy^*, d)$ and $Sat(Dx^*, Cy^*)$ are positive or zero, capacities x^* and y^* are feasible for problem (CP). This general procedure is described on Algorithm 1.

Algorithm 1 Cutting planes algorithm

```
Initial cut pool $P$ is empty.
repeat
    Solve $(MP)$ augmented with cuts in $P$. Let $(x^*, y^*)$ be an optimal solution.
    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 $P$.
        Optional: Add problem-specific cuts to $P$.
    end if
until $s_1 \geq 0$ and $s_2 \geq 0$
return $(x^*, y^*)$
```

In Algorithm 1, the solution time of (MP) is usually much bigger than those 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 planes algorithms based on Algorithm 1.

(SC) Their *single constraint generation* adds up to three cuts per iteration. Besides those described on Algorithm 1, they consider also 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 iterations required by forcing x to take sensible values, especially in the first iterations.

(MC) Aiming to reduce even more the number of iterations of Algorithm 1, they introduce the *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. Hence, they are decomposed into many problems, and the solution of each of them results in adding a cut to (MP), see for example the multi-cut L-shaped for stochastic programming problems with recourse [6].

3.2 Branch-and-cut approach

An alternative strategy is to solve one (MP) only. We want to imbed the generation of violated feasibility cuts (13) into the branch-and-cut framework 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, 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 described next. First, solve the LP relaxation of (MP) with the cutting planes described in Algorithm 1. Then P would contain all constraints added to solve the LP relaxation. Note that, in opposition to SC and MC, experiments have shown that using cuts of type (14) increases the total resolution time. Thus we do not generate constraints of type (14) neither during the cutting plane, 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.
  while $T$ is nonempty do
    Select a node $o' \in T$.
    $T := T/\{o'\}$
    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 among cutting planes SC and MC, and branch-and-cut B&C.

We also compare those 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 the one used for single-layer networks, that is

$$
(AN) \quad \begin{array}{lll}
\min & a^t x + b^t y & \\
\text{s.t.} & \sum_{k \in \mathcal{K}} |u_k| \le Cy & \\
& Bu_k = \tilde{d}_k, & \forall k \in \mathcal{K} \\
& \sum_{f \in F} |v_f| \le Dx & \\
& Av_f = C\tilde{y}_f, & \forall f \in F,
\end{array}
$$

where x and y are the capacity variables as before, u and v the flow variables on each arc, for each commodity, A and B are the arc-node incidence matrix 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 extremities of the 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 [9]. However, some tests have proved the MIP solver of CPLEX 11 to solve (AN) faster than (AP).

Finally, we show the improvement obtained by strengthened cuts (11).

4.1 Implementation details

All models have been written in JAVA, and CPLEX 11 MIP is used for solving (MP) in Algorithm 1, and *Sat* in both algorithms. The dynamic search is activated for both cutting planes and for AN-CPLEX.

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

4.2 Instances

The first 35 instances set have been randomly generated and share the following 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 node.

Then six instances are based on networks from *SNDlib* [21] 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 set of instances, costs of 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 for all instances and algorithms. The corresponding Time/Gap column give ei-

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

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

ther the resolution time in seconds or the gap when the time limit of 3600 seconds is reached.

We can see on Table 1 that SC, MC and B&C outperform CPLEX by far. Then, we see that B&C is always faster than both SC and MC, the ratio between the resolution time of B&C and the one of the faster cutting planes 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 planes algorithms, where each iteration requires to solve an IP to optimality. However the ratio is still far from the number of iterations, because many of the iterations contain only a few cuts so that their solution time is very short.

We see that B&C generate usually more cuts than SC, even though SC generate cuts of type (14). Thus, many of those cuts are not needed to ensure the feasibility of the solution. Hence managing more efficiently the cut pool, getting rid of the non active cuts might improve Algorithm 2.

The relative performance of SC and MC is as expected: MC adds much more cuts than SC, resulting in fewer iterations and smaller solution time. Indeed, all instances but 3,8,12,13 and 25 are solved faster by MC than SC. Moreover, SC cannot solve the instance 24 whereas MC solves it in 730.7 seconds. See [16] for a more detailed comparison of those algorithms.

Now, both of the cutting planes algorithm can not solve any of the bigger instances within 3600 seconds. Hence, on Tables 2 and 3 we compare CPLEX and B&C with normal and rounded cuts ((10) and (11) respectively). Although CPLEX is still outperformed by both B&C, the difference is less significative than on the smaller instances. Specifically, on instance 29 and dfn-gwin, CPLEX gets a better gap that both B&C, in instance 34 and atlanta CPLEX beats NC only. This may be due to the dynamic search, known to be more efficient on harder problems. Also we see that both B&C generate much more cuts than in the small instances.

Finally, we see also that RC gets better gaps than NC on all instances but *di-yuan*, *polska* and *nobel-us*. For instance, NC solves to optimality only instances 26, 30, *pdh* and *di-yuan* whereas RC solves also 27 and 33 to optimality.

Instances			Time/Gap			Cuts	
numb	n	e	CPLEX	NC	RC	NC	RC
26	10	20	*2.86%*	3419.4	675.2	284	318
27	10	20	*3.82%*	*0.55%*	3556.7	414	237
28	10	20	*4.86%*	*4.38%*	*2.28%*	339	443
29	10	20	*1.16%*	*4.01%*	*2.05%*	404	504
30	10	20	*2.75%*	212.5	87.1	326	334
31	10	25	*6.43%*	*4.23%*	*3.12%*	695	423
32	10	25	*2.79%*	*2.76%*	*0.56%*	500	574
33	10	25	*5.11%*	*1.16%*	3083.1	394	353
34	10	25	*1.17%*	*1.49%*	*0.9%*	503	426
35	10	25	*3.3%*	*2.7%*	*1.64%*	531	528

Table 2: Results of B&C with normal and rounded cuts (NC and RC respectively) and CPLEX on randomly generated instances.

Acknowledgements

Two anonymous referee are gratefully acknowledged for their constructive comments on a first version of the pa-

Instances			Time/Gap			Cuts	
name	n	e	CPLEX	NC	RC	NC	RC
pdh	11	34	*13.97%*	1891.5	1730.4	477	359
di-yuan	11	42	63	10.8	10.8	111	111
dfn-gwin	11	47	*9.93%*	*10.7%*	*10.68%*	616	627
polska	12	18	*4.97%*	*1.56%*	*1.81%*	827	1132
nobel-us	14	21	*12.59%*	*0.3%*	*0.31%*	1706	1697
atlanta	15	22	*4.58%*	*5.31%*	*2.33%*	1721	2117

Table 3: Results of B&C with normal and rounded cuts (NC and RC respectively) and CPLEX on instances based on networks from SNDlib.

per.

References

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

[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, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 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, A. M. C. A. Koster, and S. Orlowski, *A cut-and-branch-and-price approach to two-layer network design*, Proceedings, The Eighth INFORMS Telecommunications Conference, Dallas, Texas, 2006.

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

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

[7] 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.

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

[9] 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.

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

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

[12] 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).

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

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

[15] A. Knippel, *Accelerating a bender's method for network design*, Actes de MCO 2004, Metz, 1-3 juillet 2004, 2004.

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

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

[18] 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 the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium, April 2007.

[19] 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, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2007.

[20] E. Kubilinskas and M. Pióro, *An ip/mpls over wdm network design problem*, In Proceedings of the Second International Network Optimization Conference (INOC 2005), Lisbon, vol. 3, 2005.

[21] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, *SNDlib 1.0–Survivable Network Design Library*, Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium, April 2007, http://sndlib.zib.de (English).

[22] S. Orlowski and R. Wessäly, *An integer programming model for multi-layer network design*, ZIB Preprint ZR-04-49, Konrad-Zuse-Zentrum für Informationstechnik Berlin, December 2004.

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