Models and Algorithms for Robust Network Design with Several Traffic Scenarios

Eduardo Álvarez-Miranda[1], Valentina Cacchiani[1], Tim Dorneth[2], Michael Jünger[2], Frauke Liers[2], Andrea Lodi[1], Tiziano Parriani[1], and Daniel R. Schmidt[2]

[1] DEIS, University of Bologna, Viale Risorgimento 2, I-40136, Bologna, Italy
[2] Institut für Informatik, Universität zu Köln, Pohligstrasse 1, 50969 Köln, Germany
{e.alvarez,valentina.cacchiani,andrea.lodi,tiziano.parriani}@unibo.it
{dorneth,liers,mjuenger,schmidt}@informatik.uni-koeln.de

Abstract. We consider a robust network design problem in which optimum integral capacities need to be installed on the edges of a network such that the supplies and demands in each of the explicitly known traffic scenarios are satisfied by a single-commodity flow. In Buchheim et al. (LNCS 6701, 7 - 17 (2011)), an integer-programming (IP) formulation of polynomial size was given that uses both flow and capacity variables. In this work, we introduce an IP formulation that only uses capacity variables and exponentially many constraints that can be separated in polynomial time. We argue that the latter formulation has advantageous features when used within branch and cut and evaluate preliminary computational results for the bounds in the root node. We introduce a class of instances that is difficult for IP-based solution approaches. We design and implement a heuristic solution approach based on the definition and exploration of large neighborhoods of carefully selected size. The performance of the heuristic is evaluated on the difficult class of instances. The results are encouraging, with a good understanding of the trade-off between solution quality and neighborhood size.

Keywords: robust network design, cut-set inequalities, separation, large neighborhood search

1 Introduction

Due to their importance in modern life, network design problems have recently received increased attention. In particular, the class of *robust* network design problems has many applications and is currently studied intensively, see, e.g., [2, 1, 8, 6, 9]. For a survey, see Chekuri [5]. In this class of problems, we are given the nodes and edges of a graph together with non-negative edge costs. Furthermore, supplies and demands are explicitely or implicitely given for a set of scenarios. The task is to determine, at minimum cost, the edge capacities such that the supplies/demands of all scenarios are satisfied.

In this work, we consider the following optimization problem. We are given an undirected graph $G = (V, E)$, a cost vector $(c_e)_{e \in E}$ and an integer *balance* matrix $B = (b_v^q)_{v \in V}^{q=1,\dots,K}$. The q-th row b^q of B is called the q-th *scenario*. The Robust Network Design (RND) problem asks for integer capacities $(u_e)_{e \in E} \in \mathbb{Z}_{\geq 0}^{|E|}$ with minimal costs $c^T u$ such that for each $q = 1, \dots, K$, there is a directed network flow f^q in G that is feasible with respect to the capacities and the balances of the q-th scenario, i. e., that fulfills $f_{u,v}^q + f_{v,u}^q \leq u_e$ for all edges $e \in E$ and $\sum_{\{u,v\} \in E} f_{u,v}^q - f_{v,u}^q = b_v^q$ for all nodes $v \in V$. Here, we denote by $f_{u,v} \in \mathbb{Z}_{\geq 0}$ the integral amount of flow that is sent along the arc (u, v) from u to v in scenario q and by f^q we denote the corresponding flow vector.

For a given scenario, we call a node with nonzero balance a *terminal*. More specifically, a node v with positive balance is called a *source* and we call the balance of v its *supply*. A node with negative balance is called a *sink* and its balance is called *demand*. Whereas for $K = 1$ the RND problem is a standard polynomial-time minimum-cost flow problem, it is NP-hard already for $K = 3$ [21]. In [4], an exact branch-and-cut algorithm was introduced for RND. It is based on a flow formulation strengthened by the so-called local cuts [3]. We will show an alternative formulation for RND using inequalities of *cut-set* type. Related formulations are known for the *survivable* network design problems and its special case, the *steiner tree* problem. They can lead to strengthened formulations and approximation algorithms, see, e.g., [19] for a polyhedral study and [16, 11] for approximation algorithms.

Other related problems have been studied in the literature. Ben-Ameur and Kerivin [2] have introduced a widely-used model for robust network design in which the considered traffic scenarios are not explicitely given but belong to a polyhedron. A specific polyhedral set of traffic scenarios, the *hose-model* introduced in [6, 9], is the basis of the Virtual Private Network Design problem [14, 13]. For robust network design with a polyhedral set of scenarios, exact methods (see, e.g., [1, 8]) and approximation algorithms (see, e.g., [7, 10, 15]) exist.

If the scenarios are given as a finite list, the problem is a network synthesis problem [18] with non-simultaneous (and usually multi-commodity) flows. As an application, suppose some clients want to download some amount of data that is stored at several servers. As the clients' demands might change over time, we wish to install edge capacities that are large enough in order to satisfy the demands at all times.

In this work, we introduce a cut-set formulation for RND together with a polynomial-time separation routine for the cut-set inequalities. It turns out that the polytope that corresponds to the flow-formulation from [4] can be viewed as an extended formulation of the new model introduced here. We then introduce a class of instances that is difficult for IP-based solution approaches. We propose a heuristic algorithm for solving RND and evaluate it on the class of difficult instances. It turns out that it yields solutions of high quality within relatively short computing time.

2 Integer Linear Programming Models

In this section we introduce two possible IP formulations for RND (together with separation algorithms), and we discuss the relation between them.

2.1 Flow Formulation of RND

In [4], an integer-programming (IP) formulation for RND is given that uses flow variables. The capacity that needs to be installed on an edge $\{i,j\}$ equals the maximum amount of flow routed along (i,j) over all scenarios. Minimizing the total costs thus yields a non-linear cost-function with integrality constraints that make the problem NP-hard for the general case. Using capacity variables, it can be linearized trivially, yielding the model ($\mathrm{RND}_{\mathrm{flow}}$) as

$$
\begin{aligned}
&\min \textstyle\sum_{\{i,j\}\in E} c_{ij}u_{ij} \\
&\textstyle\sum_{j:\{j,i\}\in E} f^q_{ji} - \sum_{j:\{i,j\}\in E} f^q_{ij} = b^q_i && \forall i \in V,\ q = 1,\dots,K \\
&u_{ij} \geq f^q_{ij} + f^q_{ji} && \forall \{i,j\} \in E,\ q = 1,\dots,K \\
&f^q_{ij} \geq 0 && \forall \{i,j\} \in E,\ q = 1,\dots,K \\
&u_{ij} \in \mathbb{Z}_{\geq 0} && \forall \{i,j\} \in E
\end{aligned}
\qquad (\mathrm{RND}_{\mathrm{flow}})
$$

The first set of constraints ensures flow-conservation in each scenario. The second set models that the capacity of an edge is at least as large as the flow it carries. Integral flows are enforced through integrality of the capacity variables, as all supplies and demands are integral.

We denote by P_{flow} the polytope that consists of the convex hull of all integral solutions feasible for ($\mathrm{RND}_{\mathrm{flow}}$).

2.2 Cut-Set Formulation of RND

Let us now introduce an alternative formulation of the RND problem that only uses capacity variables. In more detail, we denote by $P_{\text{cut-set}}$ the convex hull of all integer capacity vectors $u \in \mathbb{Z}^{|E|}_{\geq 0}$ that permit sending a feasible flow on $G = (V,E)$ for each scenario in B. Next, we examine the structure of $P_{\text{cut-set}}$.

Lemma 1. *For a node set $S \subseteq V$, the cut-set inequalities*

$$\sum_{e\in\delta(S)} u_e \geq \max_{q=1\dots K} \left| \sum_{i\in S} b^q_i \right| \qquad (1)$$

are valid for $P_{\text{cut-set}}$.

Proof. The right-hand side of a cut-set inequality is exactly the amount of supply/demand that cannot be satisfied by the nodes inside S. This amount of flow has to be sent along the cut $\delta(S)$ from S to $V \setminus S$, or vice versa. Thus, for each scenario q, the capacity of the cut has to be at least as large as $|\sum_{i\in S} b^q_i|$.

We can now show that the cut-set-inequalities exactly characterize every integer point in $P_{\text{cut-set}}$.

Theorem 1. *Let $u \in \mathbb{Z}_{\geq 0}^{|E|}$. Then $u \in P_{\text{cut-set}}$ if and only if u satisfies the corresponding cut-set inequality for all $S \subseteq V$ (1).*

Proof. Validity is shown in Lemma 1. We need to prove that a vector $u \in \mathbb{Z}_{\geq 0}$ that satisfies all cut-set inequalities is contained in $P_{\text{cut-set}}$. Assume that for some vector $u \in \mathbb{Z}_{\geq 0}$ the supplies/demands of some scenario q cannot be met even though u satisfies (1) for all $W \subseteq V$.

Then, let us run the algorithm by Ford and Fulkerson [17] (in a suitably adapted version for undirected networks) on G. Denote by f the (directed) flow that the algorithm computed and let N_f be the residual network corresponding to f. Observe that there is no augmenting path in N_f. Let $S := \{s \in V \mid b_s^q > 0\}$ and $R := \{u \in V \mid \text{there exists a path in } N_f \text{ from some node in } S \text{ to } u\}$.

By construction, all outgoing edges from R must be saturated as otherwise the endnode of an unsaturated outgoing edge belongs to R. Also, the flow on all incoming edges of R is zero since with an easy modification of Ford and Fulkerson's algorithm we can w.l.o.g. assume that flow is sent along an edge in only one direction. Thus,

$$\sum_{e \in \delta(R)} u_e = \sum_{\{u,v\} \in \delta(R)} (f_{u,v} + f_{v,u}) = \sum_{\substack{\{u,v\} \in \delta(R) \\ u \in R}} f_{u,v} \overset{\text{Lemma 2}}{<} \left| \sum_{u \in R} b_u^q \right|$$

Here, we get strict inequality from Lemma 2 as by our assumption, f does not satisfy all supplies and demands and yet there is no augmenting path. On the other hand,

$$\left| \sum_{u \in R} b_u^q \right| \leq \max_{k=1,\ldots,K} \left| \sum_{u \in R} b_u^k \right| \leq \sum_{e \in \delta(R)} u_e$$

by the cut-set-inequality for R, yielding a contradiction. □

The following Lemma provides the missing piece in the above proof. It is proven in the Appendix.

Lemma 2. *For an undirected graph $G = (V, E)$ let $b \in \mathbb{Z}^{|V|}$ be a balance vector that satisfies $\sum_{i=1}^{|V|} b_i = 0$. Let f be a (directed) flow on G that satisfies some but not all supplies and demands in b. Let N_f be the residual network corresponding to f. Then it holds that*

$$\sum_{\substack{\{u,v\} \in \delta(R) \\ u \in R}} f_{u,v} \leq \left| \sum_{u \in R} b_u \right| \tag{2}$$

where $R := \{u \in V \mid \text{there exists a path from a source node } s \text{ to } u \text{ in } N_f\}$. If there is no augmenting path in N_f, then (2) holds with strict inequality.

We thus have the following IP formulation for the RND problem:

$$(\text{RND}_{\text{cut-set}}) \qquad \begin{array}{rll} \min & \sum_{e \in E} c_e u_e & \\ & \sum_{e \in \delta(S)} u_e \geq \max_{q=1,\ldots,K} \left| \sum_{v \in S} b_v^q \right| & \forall\, S \subseteq V \\ & u_e \in \mathbb{Z}_{\geq 0} & \forall\, e \in E \end{array}$$

2.3 Separation of Cut-Set Inequalities

Given a (possibly fractional) vector $u^* \in \mathbb{R}^{|E|}_{\geq 0}$, the separation procedure answers whether all cut-set inequalities are satisfied. If not, it returns a violated cut-set inequality of form (1).

The cut-set inequalities can be separated independently for each scenario. Indeed, if for some $S \subseteq V$ and some scenario k it holds $\sum_{e \in \delta(S)} u^*_e < \left|\sum_{v \in S} b^k_v\right|$, then this implies a violated cut-set inequality

$$\sum_{e \in \delta(S)} u^*_e < \left|\sum_{v \in S} b^k_v\right| \leq \max_{q=1,\dots,K} \left|\sum_{v \in S} b^q_v\right|.$$

For separating the cut-set inequalities for some scenario k, we consider the graph $G' = (V', E')$ that arises from G by inserting an additional node s. For each node τ that is a terminal in scenario k, the weight of the edge $\{s, \tau\}$ in G' is set to $-b^k_\tau$. All edges that originally appear in G receive a weight of u^*_e in G'.

Consider a node set S in G' with $s \in S$. If the weight of the cut $\delta(S)$ is smaller than zero, the corresponding cut-set inequality is violated. Indeed, the weight $w(\delta(S))$ can be written as

$$w(\delta(S)) := \sum_{e \in \delta(S)} u^*_e = \sum_{\substack{e \in \delta(S) \\ e \in E}} u^*_e - \sum_{\substack{e \in \delta(S) \\ e = \{s,v\}}} b^k_v.$$

If $w(\delta(S)) < 0$, this is equivalent to

$$\sum_{\substack{e \in \delta(S) \\ e \in E}} u^*_e < \sum_{\substack{e \in \delta(S) \\ e = \{s,v\}}} b^k_v \quad \text{and thus} \quad \sum_{\substack{e \in \delta(S) \\ e \in E}} u^*_e < \Big|\sum_{\substack{e \in \delta(S) \\ e = \{s,v\}}} b^k_v\Big|.$$

If $\delta(S)$ represents a minimum cut in G', the corresponding cut-set inequality is one with maximum violation. In general, the determination of cuts with minimum or maximum weight in a graph with arbitrary weights is NP-hard. However, in our case, edges with negative weights only appear as edges that are incident to the single node s. McCormick et al. call such graphs *star negative* in [20]. They show that minimum cuts in star negative graphs can be found in polynomial time by determining a minimum s-t-cut in certain a network. We summarize our findings in the following lemma.

Lemma 3. *The cut-set inequalities (1) can be separated in polynomial time.*

2.4 Comparison of the Flow Formulation with the Cut-Set Formulation

There is a close connection between the polytopes P_{flow} and $P_{\text{cut-set}}$. In fact, an orthogonal projection of P_{flow} to the capacity variables yields a polytope that is isomorphic to $P_{\text{cut-set}}$. Conversely, P_{flow} can be seen as an extended formulation of $P_{\text{cut-set}}$.

We argue next that from a practical point of view, it is advantageous to work with $P_{\text{cut-set}}$ within a branch-and-cut method. In [4], general separation routines have been used for solving (RND_{flow}). A way to improve over this method is to exploit problem-specific polyhedral knowledge as well. A disadvantage of formulation (RND_{flow}) is that the polytope P_{flow} is different for each graph, for each number and each choice of scenarios. It turns out that a theoretic understanding of P_{flow} is difficult already for small instances. In contrast, formulation ($\text{RND}_{\text{cut-set}}$) simply defines a polytope for every graph, independently of the number and choice of scenarios. Polyhedral investigations of $P_{\text{cut-set}}$ are thus considerably easier. Furthermore, we can optimize over the LP-relaxation of ($\text{RND}_{\text{cut-set}}$) within polynomial time, see Section 2.3. Therefore, we will use formulation ($\text{RND}_{\text{cut-set}}$) within branch-and-cut methods. However, for the heuristics in Section 4, it is favorable to work with flows and the formulation (RND_{flow}).

For an instance and some relaxation, let us define the *integrality gap* as the value of an optimum integral solution, divided by the optimum value of the relaxation. Generally, the size of the integrality gap at the root node can serve as a rough estimate for an instance's computational difficulty. Usually, the larger the gap is, the more difficult it is to solve with IP-based methods. In order to see what performance can be expected from such an approach, we experimentally evaluate the size of the integrality gap for a set of difficult instances. In Sections 3 and 4 we will present a class of instances on d-dimensional hypercubes, both with uniform and random scenarios. On the same set of instances, we use the cut-set based relaxation and initialize the LP with all cut-set inequalities, i. e., no separation is used. It turns out that the linear programming relaxation determines an optimum integral solution on all random instances, except for the instance `A2` where the gap is about 7%. This shows that our relaxation yields very strong bounds. For the uniform instances, we get a bound of 50%, 75% and 75% for $d = 2, 3, 4$, respectively. If $d > 4$, the formulation is too large to be used without separation.

3 Class of Instances with Large Integrality Gap

Next, we present a class of instances that is difficult for IP-based solution approaches. The instances are defined on a d-dimensional hypercube (for $d \in \mathbb{Z}_{>0}$) and have a large integrality gap. More specifically, the ratio of an optimum integral solution and an optimum fractional solution converges to 2 as $d \to \infty$.

Definition 1. *A* d-dimensional hypercube $\mathcal{H}_d$ *is the result of the following recursive construction:* $\mathcal{H}_0$ *is the graph that consists of a single node. For* $d > 0$*,* $\mathcal{H}_d$ *is obtained by duplicating the nodes and edges of* $\mathcal{H}_{d-1}$ *and connecting each node* v *to its copy* v' *with an additional edge* $\{v, v'\}$*.*

We say that two nodes v, w are *diagonally opposite* on $\mathcal{H}_d$ iff the shortest path from v to w in $\mathcal{H}_d$ has maximum length, i. e., length d. Notice that for every node v in $\mathcal{H}_d$ there is exactly one node v^o that is diagonally opposite to v. It is well-known that $\mathcal{H}_d$ has $N_d := 2^d$ nodes and $M_d := d \cdot 2^{d-1}$ edges.

For $d \in \mathbb{Z}_{>0}$, consider the following instance I_d of the RND problem on $\mathcal{H}_d$. First, observe that $\mathcal{H}_d$ is composed of two hypercubes $\mathcal{H}^s_{d-1}, \mathcal{H}^t_{d-1}$ of dimension $d-1$. Then, add 2^{d-1} scenarios to $\mathcal{H}_d$: In scenario $1 \leq q \leq 2^{d-1}$, assign a supply of 1 to the q-th node v_q (in some fixed numbering) of $\mathcal{H}^s_{d-1}$ and a demand of -1 to its diagonally opposite node v^o_q which lies in $\mathcal{H}^t_{d-1}$ by our construction. Set all other balances of scenario q to zero and set the costs for each edge to 1. Figure 1 shows the construction.

If we allow fractional capacities, we obtain an optimum LP-solution u^F for I_d by setting all capacities to $1/d$. This solution is feasible as for any pair v, v^o of diagonally opposite nodes, there are d disjoint paths from v to v^o in $\mathcal{H}_d$. In scenario q, we need to send one unit of flow from v_q to v^o_q. By splitting this unit equally over d disjoint v_q-v^o_q-paths, we can send it while respecting the capacities.

For a lower bound on an integer solution u^I, we show that any connected component of the support graph of a feasible solution contains at least $2d$ nodes: Each connected component C of u^I must contain one source s and its corresponding sink t. Yet, since the shortest path between any source-sink-pair in $\mathcal{H}_d$ has length d, $d-1$ additional nodes $A \subseteq V$ must be contained in C. Each node in A is a terminal in some scenario, as all nodes are terminals. However, for no source or sink in A the corresponding terminal can lie in A as otherwise the shortest path between such a terminal pair has length $< d$. Thus, as u^I is feasible, another $d-1$ nodes need to be contained in C. This gives $d+1+d-1 = 2d$ nodes in total. Therefore, no feasible solution can contain more than $Z := \lfloor N_d/(2d) \rfloor = \lfloor 2^d/(2d) \rfloor$ connected components. However, in order to have at most Z connected components, the solution must contain at least $N_d - Z$ edges. Thus, we can bound the integrality gap $GAP(I_d)$ as follows:

$$GAP(I_d) \geq \frac{|u^I|}{|u^F|} \geq \frac{N_d - 2^d/(2d)}{1/d \cdot M_d} = \frac{2^d - 2^{d-1}/d}{1/d \cdot d \cdot 2^{d-1}} = 2 - \frac{1}{d} \xrightarrow{d\to\infty} 2$$

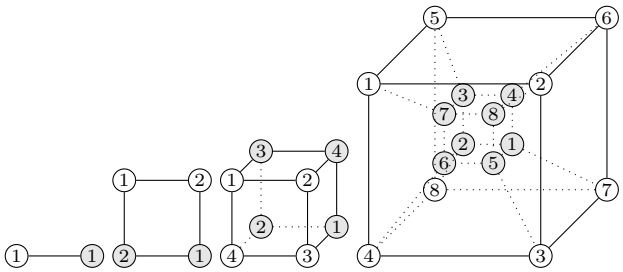

Fig. 1: The hypercube instances in 1,2,3 and 4 dimensions. Copied nodes are displayed in gray. The node numbering refers to the scenarios: The source-sink pair for scenario i is depicted with number i.

.

4 Heuristic Algorithm

In this section, we present our heuristic algorithm that consists of a *forward phase* (FP) and a *backward phase* (BP).

4.1 Forward Phase

FP computes a feasible solution for the RND problem by solving a sequence of Minimum Cost Flow (MCF) problems, one for each scenario $q = 1, \dots, K$ in the directed graph $G_{dir} = (V, A)$ defined as follows. It has the same set of nodes of G and for each $e = \{i, j\} \in E$, we introduce four arcs a_1^e, a_2^e, a_3^e and a_4^e: a_1^e and a_2^e are directed from i to j, while a_3^e and a_4^e are directed from j to i. For each arc $a \in A$, let UB_a be its upper bound on the capacity and c_a its cost. When solving the MCF problems, edge costs and bounds are set such that a scenario can use the capacities for free that are already installed. Algorithm 1 illustrates the general structure of the FP procedure for the construction of an RND-solution. Let u^{FP*} be the solution that we obtain by applying FP.

foreach $e \in E$ **do**
  set $UB_{a_1^e} := \infty$, $UB_{a_2^e} := 0$, $UB_{a_3^e} := \infty$ and $UB_{a_4^e} := 0$;
  set $c_{a_1^e} := c_e$, $c_{a_2^e} := 0$, $c_{a_3^e} := c_e$ and $c_{a_4^e} := 0$;
  set $u^{FP*} := 0$;
foreach *scenario* $q = 1, \dots, K$ **do**
  solve MCF for scenario q;
  obtain solution f^{q*};
  foreach $e \in E$ **do**
    $u_e^* := f_{a_1^e}^{q*} + f_{a_3^e}^{q*}$;
    $UB_{a_2^e} := UB_{a_2^e} + u_e^*$ and $UB_{a_4^e} := UB_{a_4^e} + u_e^*$;
    $u_e^{FP*} := u_e^{FP*} + u_e^*$
return u^{FP*}

Algorithm 1: FP procedure.

It is more likely that we can use some capacity "for free" later if we have installed capacities on a larger set of edges. Thus, we apply a preprocessing at the beginning of FP. We divide each scenario $q = 1, \dots, K$ in R sub-scenarios $g_1^q, \dots, g_R^q$ having balances $b_v^{g_1^q} = \lfloor b_v^q / R \rfloor$, $b_v^{g_2^q} = \lfloor b_v^q / (R-1) \rfloor$, up to $b_v^{g_R^q} = b_v^q$, $v \in V$, where R is an integer positive number. Each sub-scenario is then dealt with as an original scenario and we apply the FP procedure in the order g_1^q, $(q = 1, \dots, K)$, g_2^q, $(q = 1, \dots, K)$, up to g_R^q, $(q = 1, \dots, K)$. In this way, the generic sub-scenario g_l^q of scenario q can already take into account the partial solution computed for all the other scenarios $q = 1, \dots, K$. This also means that more likely the complete MCF-solution of a generic scenario q will have a split integer flow, because each scenario might use different subsets of arcs.

4.2 Backward Phase

To improve the solution found in FP, BP uses a modified version of the compact (RND_{flow}) model, described in Section 2.1 Indeed, the described compact formulation appears suitable for performing a large neighborhood search while keeping

the computing time short. BP starts from the solution u^{FP*} found by FP. First, edges $\{i,j\}$ with $u_{ij}^{FP*}=0$ are removed from G which leads to a reduced graph $\bar{G}=(V,\bar{E})$. In addition, the derived capacities u_{ij}^{FP*} are used to impose upper bounds on the capacity variables u_{ij} of (RND_{flow}). More precisely, in order to perform a large neighborhood search, we allow an increase in the capacities with respect to the solution u^{FP*} up to a maximum total value T. This is obtained by adding continuous variables $w_{ij}\geq 0$, for $\{i,j\}\in\bar{E}$ and the constraints

$$u_{ij}\leq u_{ij}^{FP*}+w_{ij},\quad \forall\{i,j\}\in\bar{E} \tag{3}$$

$$\sum_{\{i,j\}\in\bar{E}} w_{ij}\leq T, \tag{4}$$

with a parameter T whose size has to be determined. It is not necessary to impose the integrality of the w_{ij} variables because u_{ij} are imposed to be integer. Parameter T is used to control the size of the neighborhood of the FP-solution that we want to consider. If T is set to 0, then we are imposing the values u_{ij}^{FP*} as upper bounds of the capacity variables. If T is set to $+\infty$, then we get the (RND_{flow}) model on the reduced graph. The neighborhood is explored by solving the proposed model for a fixed T to optimality.

4.3 Computational Results

We tested our heuristic on a set of d-dimensional hypercubes (see Section 3), with randomly generated integer scenarios. We considered both the case of unit demand and a more general uniform distribution in $[1,10]$. Each instance has 2^d nodes and 2^{d-1} scenarios. Instead, because of space limitation, we did not report results on instances like those in [4] that turn out to be much easier to solve.

Forward and backward phases were coded in C language and Cplex 12.3 was used. The tests were executed on a PC Intel(R) Core(TM) i7 CPU, 64 bit, 1.73 GHz, 6 Gb RAM, running Windows 7. Computing times are expressed in seconds. In FP, the code CS2 by Goldberg [12] for solving MCF is used.

Unit demands. It is easy to see that for hypercubes with unit demands, the capacity installed on any edge cannot be larger than 1 in an optimal solution: indeed, we have one source-sink pair in each scenario. Thus, removing the edges not used in FP cannot lead to an improvement of the solution because BP, as described in the previous section, can only increase the capacity of used edges. To overcome this difficulty, we randomly selected a percentage P of demands and temporarily increased them to D before running FP, so as to enlarge the set of edges in input to BP, which is then executed with $T=0$ (BP_0). We tested this variant of the heuristic for $d=\{3,4,5,6\}$, and compared it with the (RND_{flow}) model initialized with the FP solution (RND_{init}). Parameter P is adaptively set to produce a reduced graph of manageable size for BP, namely $P=100\%$ for $d=\{3,4,5\}$ and $P=20\%$ for $d=6$, while we set $D=2$ in the experiments. Both RND_{init} and BP_0 obtain the optimal solution for $d\in\{3,4\}$. For $d=5$, BP_0 improves the FP solution of value 31 decreasing it to 30 in around 3 minutes (and finishes optimizing the neighborhood in 733.67 seconds),

while RND_{init} only obtains this value slightly before reaching the time limit of 7200 seconds. Finally, for $d = 6$ BP_0 decreases to 62 the FP solution of value 63 in less than 2000 seconds and then reaches the time limit of 7200, while RND_{init} does not improve in the same time limit. As expected, these instances turn out to be extremely difficult and other modification of the proposed heuristic are under investigation.

Uniform demands. For the more general instances with demand uniformly distributed in $[1, 10]$ we considered $d = \{3, \dots, 7\}$, and 5 instances for each value of d. After parameter tuning, the R parameter of FP was set to 10. We tested BP with different values of T. As a trade-off between solution quality and computing time we selected $T = 25$ (BP_{25} in the tables) and compared it with solving (RND_{flow}) and to BP with $T = +\infty$ (BP_{∞} in the tables). The latter case corresponds to the (RND_{flow}) model on the reduced graph. All models, including (RND_{flow}), receive on input the solution computed by FP. For this reason, model (RND_{flow}) is indicated in the following as RND_{init}. The aim of this comparison is twofold. On the one side, testing the complete (RND_{flow}) model shows the difficulty of these instances for Cplex. Its comparison with the approaches using a backward phase highlights the speedup that can be obtained by exploring only a portion of the solution space. On the other side, comparing BP_{∞} and BP_{25} allows to grasp the relation between the quality of the heuristic solution and the size of the neighborhood T. For $d \in \{3, 4, 5\}$, RND_{init} computes the optimal solution very fast, the computing times of BP_{∞} and BP_{25} are negligible, and their solutions are very accurate. Thus, we do not report the corresponding results and focus on instances with $d \in \{6, 7\}$. The results of the comparison among RND_{init}, BP_{∞} and BP_{25} are reported in Table 1. More precisely, the table reports, for each instance, its name (Inst., with the number representing its size d), the best lower bound obtained by solving the RND_{init} (LB), the best upper bound computed by the three methods (UB), and, for each of the algorithms, the computing times (time), the total number of branch-and-bound nodes (nodes), and the percentage gap between its solution value (sol) and LB (namely, $100 \cdot (\text{sol} - LB)/LB$). In addition, Table 1 reports computing times and percentage gaps for FP. The results are obtained by imposing a time limit of 7200 seconds, and TL indicates that the time limit was reached. Instances reaching the time limit count with 7200 seconds in the averages. Finally, the average results over the 5 instances with $d = 6$ and with $d = 7$ are reported.

The results in Table 1 show that FP is very fast in computing a feasible solution, even if the quality of this solution is not very good. However, after applying BP, the percentage gaps reduce significantly. This means that, on the one side, FP is able to identify how to reduce the graph, and, on the other hand, the neighborhood considered in BP is explored effectively. For instances with $d = 6$, the computing times and the percentage gaps of BP_{∞} and BP_{25} are almost the same. Compared to RND_{init}, both BP methods are more than one order of magnitude faster to explore their neighborhood, while the average percentage gap is acceptable (1.02% for both). The effectiveness of the proposed heuristic is more evident on instances with $d = 7$. This time RND_{init} reaches the

		FP		RND_{init}			BP_∞			BP_{25}		
Inst.	LB UB	time	%gap	time	nodes	%gap	time	nodes	%gap	time	nodes	%gap
A6	160 162	0.05	28.75	TL	44216	1.25	34.37	212	3.13	47.10	593	3.13
B6	212 212	0.06	16.04	119.95	407	0.00	5.29	0	0.00	4.98	0	0.00
C6	210 210	0.05	11.90	18.27	0	0.00	3.29	0	0.95	21.14	79	0.95
D6	192 192	0.08	9.38	106.11	162	0.00	3.04	0	1.04	2.81	0	1.04
E6	194 194	0.06	14.43	223.64	1100	0.00	63.13	682	0.00	5.43	0	0.00
Av.s		0.06	16.10	1533.59	9177	0.25	21.82	179	1.02	16.29	135	1.02
A7	376 383	0.35	27.66	TL	11	5.32	3549.62	551	1.86	1200.60	455	3.99
B7	374 383	0.26	25.94	TL	10	25.94	2587.78	489	2.41	78.52	0	6.68
C7	373 381	0.16	24.40	TL	10	7.24	TL	774	2.14	76.04	0	3.22
D7	357 364	0.24	22.69	TL	8	6.44	TL	558	1.96	473.92	45	3.64
E7	351 359	0.19	28.21	TL	6	8.55	2873.80	604	2.28	414.81	104	2.56
Av.s		0.24	25.78	7200.00	9	10.70	3003.73	596	2.13	448.77	121	4.02

Table 1: Comparison among RND_{init}, BP_∞ and BP_{25}.

time limit for all instances. The average percentage gap of BP_∞ is 2.13% and it finds the best solution in all cases, while the gap of BP_{25} is 4.02%. In terms of efficiency, BP_∞ is able to fully explore its neighborhood in 3/5 cases in about 3000 CPU seconds on average, while for the 2 instances on which it reaches the time limit the solution value is much better than the one of RND_{init}. Instead, BP_{25} is much faster in exploring its (smaller) neighborhood, with an average CPU time of 448.77 seconds In that concern, BP_{25} seems to provide a good compromise between quality of the solution and speed. However, it is interesting to note that even very small neighborhoods like $T = 5$ and $T = 1$ are somehow effective: for $d = 7$, the average FP percentage gap of 25.78% reduces to 8.75% and to 9.9%, respectively, with average computing times of 40 and 6 seconds, respectively. This means that neighborhoods of small size could be iteratively explored by primal heuristics in a branch-and-cut algorithm.

5 Conclusions and Future Research

In this work, we have introduced a cut-set formulation for the RND problem. It turns out that the LP-bounds are very strong. Moreover, we have proposed a two-phase heuristic algorithm that explores large neighborhoods whose size can be carefully controlled. Thus, the use of this heuristic framework can be foreseen both as stand-alone algorithm and as primal heuristic within a branch-and-cut approach, which represents the next step of our work. Furthermore, we intend to investigate the polyhedral structure of $P_{\text{cut-set}}$ in detail.

Acknowledgments

Financial support is acknowledged from the German Science Foundation (contract Li 1675/1) and the Ateneo Italo-Tedesco (VIGONI programme 2011-2012).

References

1. A. Altin, E. Amaldi, P. Belotti, and M.C. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1):100–115, 2007.
2. W. Ben-Ameur and H. Kerivin. Routing of uncertain demands. *Optimization and Engineering*, 3:283–313, 2005.
3. C. Buchheim, F. Liers, and M. Oswald. Local cuts revisited. *Operations Research Letters*, 36(4):430–433, 2008.
4. C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *INOC*, volume 6701 of *Lecture Notes in Computer Science*, pages 7–17. Springer, 2011.
5. C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38(3):106–128, 2007.
6. N.G. Duffield, P. Goyal, A.G. Greenberg, P.P. Mishra, K.K. Ramakrishnan, and J.E. van der Merwe. A flexible model for resource management in virtual private networks. *Proceedings of SIGCOMM*, 29:95–108, 1999.
7. F. Eisenbrand, F. Grandoni, G. Oriolo, and M. Skutella. New approaches for virtual private network design. *SIAM Journal on Computing*, pages 706–721, 2007.
8. T. Erlebach and M. Rüegg. Optimal bandwidth reservation in hose-model VPNs with multi-path routing. *Proceedings of INFOCOM*, 4:2275–2282, 2004.
9. J.A. Fingerhut, S. Suri, and J.S. Turner. Designing least-cost nonblocking broadband networks. *Journal of Algorithms*, 24(2):287–309, 1997.
10. S. Fiorini, G. Oriolo, L. Sanità, and D.O. Theis. The VPN problem with concave costs. *SIAM Journal on Discrete Mathematics*, pages 1080–1090, 2010.
11. H. N. Gabow, M. X. Goemans, and D. P. Williamson. An efficient approximation algorithm for the survivable network design problem. *Math. Program.*, 82:13–40, 1998.
12. A.V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.
13. N. Goyal, N. Olver, and B. Shepherd. The VPN conjecture is true. *Proceedings of STOC*, pages 443–450, 2008.
14. A. Gupta, J. Kleinberg, R. Rastogi, and B. Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. *Proceedings of STOC*, pages 389–398, 2001.
15. A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. *Proceedings of STOC*, pages 365–372, 2003.
16. K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
17. L. R. Ford Jr. and D. R. Fulkerson. A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal of Mathematics*, 9:210–218, 1957.
18. M. Labbé, R. Séguin, P. Soriano, and C. Wynants. Network synthesis with non-simultaneous multicommodity flow requirements: Bounds and heuristics, 1999.
19. T.L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45:61–79, 2005.
20. S. T. McCormick, M. R. Rao, and G. Rinaldi. Easy and difficult objective functions for max-cut. *Math. Program.*, 94(2-3, Ser. B):459–466, 2003.
21. L. Sanità. *Robust Network Design.* Ph.D. Thesis. Università La Sapienza, Roma, 2009.

Appendix

Proof (of Lemma 2). We have $\sum_{u\in R} b_u \geq 0$, since $\sum_{v\in V} b_v = 0$ and there can only be nodes with non-positive balance outside of R. If we let $\overline{b}_u := \sum_{v\in\delta(u)} f_{u,v} - f_{v,u}$ be the residual balance of node u, then we get by the same argument that $\sum_{u\in R} \overline{b}^q_u \geq 0$. Also, we can assume w.l.o.g. that all flow entering R is zero, as R contains all sources and hence all incoming flow must originate from R and can be suitably rerouted. Thus,

$$\begin{aligned}
\Big|\sum_{u\in R} b_u\Big| &= \sum_{u\in R} b_u \\
&= \sum_{u\in R}\left[\overline{b}_u + \sum_{v\in\delta(u)} (f_{u,v} - f_{v,u})\right] \\
&\geq 0 + \sum_{u\in R}\sum_{\substack{v\in\delta(u)\\ v\in R}} (f_{u,v} - f_{v,u}) + \sum_{u\in R}\sum_{\substack{v\in\delta(u)\\ v\in V\setminus R}} (f_{u,v} - f_{v,u}) \\
&= 0 + \sum_{u\in R}\sum_{\substack{v\in\delta(u)\\ v\in V\setminus R}} (f_{u,v} - f_{v,u}) \\
&= \sum_{u\in R}\sum_{\substack{v\in\delta(u)\\ v\in V\setminus R}} f_{u,v} \\
&= \sum_{\substack{\{u,v\}\in\delta(R)\\ u\in R}} f_{u,v}
\end{aligned}$$

By our assumption, there is a source s and a sink t with residual balance $\overline{b}_s > 0$ and $\overline{b}_t < 0$, respectively. Now, if we suppose that there is no augmenting path in N_f, then $s \in R$ by definition and thus $t \in V \setminus R$. Hence, $\sum_{u\in R} \overline{b}_u > 0$ and we get strict inequality in the above calculation. □