Institut für Informatik
Universität zu Köln

Technical Report
March 2014

Single-commodity Robust Network Design Problem: Complexity, Instances and Heuristic Solutions[*]

Eduardo Álvarez-Miranda[1], Valentina Cacchiani[2], Andrea Lodi[2], Tiziano Parriani[2], and Daniel R. Schmidt[3]

[1] DMGI, Universidad de Talca, Merced 437, Curicó, Chile, e.alvarez@unibo.it
[2]DEI, University of Bologna, Viale Risorgimento 2, I-40136, Bologna, Italy,
{valentina.cacchiani,andrea.lodi,tiziano.parriani}@unibo.it
[3]Institut für Informatik, Universität zu Köln, Albertus-Magnus-Platz, 50923 Köln, Germany,
schmidt@informatik.uni-koeln.de

March 24th, 2014
This preprint also appeared as technical report OR-13-14 at the University of Bologna.

We study a single-commodity Robust Network Design problem (RND) in which an undirected graph with edge costs is given together with a discrete set of balance matrices, representing different supply/demand scenarios. In each scenario, a subset of the nodes is exchanging flow. The goal is to determine the minimum cost installation of capacities on the edges such that the flow exchange is feasible for every scenario. Previously conducted computational investigations on the problem motivated the study of the complexity of some special cases and we present complexity results on them, including hypercubes. In turn, these results lead to the definition of new instances (random graphs with {-1,0,1} balances) that are computationally hard for the natural flow formulation. These instances are then solved by means of a new heuristic algorithm for RND, which consists of three phases. In the first phase the graph representing the network is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase consists of a neighborhood search on the reduced graph based on a Mixed-Integer (Linear) Programming (MIP) flow model. Finally, the third phase applies a *proximity search* approach to further improve the solution, taking into account the original graph. The heuristic is tested on the new instances, and the comparison with the solutions obtained by Cplex on a natural flow formulation shows the effectiveness of the proposed method.

1 Introduction

Network design problems arise in many different areas, such as transportation and telecommunication. Recently, the class of *robust network design* problems has received increasing attention. The term robust can represent the capability of the network to cope with disruptions or to deal with different traffic scenarios in different times of the day, as is the case of our work.

In this paper, we study the *single-commodity* Robust Network Design problem (RND) defined as follows. We are given an undirected graph $G = (V,E)$, a cost vector (c_e) $(e \in E)$ and an integer *balance* matrix $B = (b_i^q)$ $(i \in V, q = 1,\dots,K)$. The q-th row b^q of B is called the q-th *scenario*.

[*]We thank the Ateneo Italo-Tedesco VIGONI programme 2011-2012 for its financial support. This work was supported by a STSM Grant from COST Action TD1207 (grant 16810).

For a given scenario, we call a node with nonzero balance a *terminal*. More specifically, a node i with positive balance is called a *source* and we call the balance of i its *supply*. A node with negative balance is called a *sink* and its balance is called *demand*.

Let us denote by (i,j) and (j,i) the arcs (directed from i to j and from j to i, respectively) corresponding to edge $e=\{i,j\}\in E$. In addition, let us call $f^q_{i,j}\in\mathbb{Z}_+$ the integral amount of flow that is sent along arc (i,j) from i to j in scenario q and by f^q the corresponding flow vector.

RND calls for determining integer capacities $(u_e)\in\mathbb{Z}^{|E|}_+$ $(e\in E)$ with minimal costs c^Tu such that, for each q $(q=1,\ldots,K)$, 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. In particular, the flow f^q $(q=1,\ldots,K)$ must fulfill the following constraints:

1. $f^q_{i,j}+f^q_{j,i}\le u_e$ for all edges $e=\{i,j\}\in E$, which imposes that the sum of the flows going along every edge (in both directions) must respect the installed edge capacity, for every scenario,

2. $\sum_{\{i,j\}\in E}(f^q_{i,j}-f^q_{j,i})=b^q_i$ for all nodes $i\in V$, which implies that the flow must satisfy the required integer balances.

An overall natural model for RND reads as follows

$$\min \sum_{\{i,j\}\in E} c_{ij}u_{ij} \tag{1}$$

$$\sum_{j:\{j,i\}\in E} f^q_{ji} - \sum_{j:\{i,j\}\in E} f^q_{ij} = b^q_i \qquad \forall i\in V,\ q=1,\ldots,K \tag{2}$$

$$f^q_{ij}+f^q_{ji}\le u_{ij} \qquad \forall\{i,j\}\in E,\ q=1,\ldots,K \tag{3}$$

$$f^q_{ij}\ge 0 \qquad \forall\{i,j\}\in E,\ q=1,\ldots,K \tag{4}$$

$$u_{ij}\in\mathbb{Z}_+, \qquad \forall\{i,j\}\in E \tag{5}$$

where the objective function (1) is to minimize the total cost of the installed capacities. Constraints (2) ensure flow-conservation in each scenario and impose to satisfy the required balances. Constraints (3) model 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 balances are integral [17].
As described in [9], an example of a practical application of the considered problem is the following: some clients wish to download some program stored on several servers. For a client, it is not important which server he or she is downloading from, as long as the demand is satisfied. In other words, we consider servers that store identical data: examples are video on demand or large datacentre in which one mirrors his data over several locations. This is opposed to multi-commodity network design, in which point-to-point connections are considered, i.e. each client requests a specific server. In addition, we consider the robust version of the problem: at different times of the day, the demands may change (e.g. different clients show up), and the goal is to design a network that is able to route all flow in all different scenarios. In particular, we consider a finite list of demands, i.e. we sample different times of the day.
Contribution of the paper. Preliminary computational investigations have been performed on classical graphs from the literature with random balances [9] and on special hypercubes with $\{-1,0,1\}$ balances [2]. The results in both papers have shown that the former instances are surprisingly easy for a general-purpose Mixed-Integer Programming (MIP) solver on the natural flow-formulation (1)–(5), while the latter instances are structurally difficult. The first contribution of the paper is in studying the complexity of some RND special cases[1] associated with the above instances and enlightening the reasons of the observed computational behavior. Second, based on the complexity results, we propose a new family of randomly generated RND instances that are computationally challenging for the natural flow formulation already for $|V|=50$ and $K=10$. Third, motivated by those instances (available upon request from the authors), we propose new and general heuristic approaches that provide high-quality approximated solutions for large graphs (tests are reported for $|V|$ up to 500) in short computing times[2].

[1]The RND problem is strongly NP-hard [27].

[2]A preliminary version of the heuristic approaches described here was introduced in [2] where the first phase of the investigation on RND, which was the topic of the “Vigoni 2011-2012” project between the University of Köln and the University of Bologna, was summarized.

Organization of the paper. Section 2 reviews the (vast) related literature by pointing out differences and similarities. In Section 3 we present the complexity results we achieved on special classes of instances, while Section 4 describes the proposed heuristic algorithm and its performance is reported in Section 5. Finally, in Section 6 we draw conclusions and describe ideas for future research.

2 Related Literature

The work on classical (i.e., non-robust) network design goes back as far as the early 1960s where it was studied by Chien [11] and Gomory and Hu [16, 15]. Since then, network design has evolved to a vast field of research which we cannot fully discuss in the scope of this article. We rather refer to [10] for a complete overview and restrict ourselves to a few exemplary related works that are of direct importance for us here.

The common theme of network design problems is installing optimum-cost capacities in a given network topology such that a set of traffic requests can be routed through the network. In practice, however, the traffic requests are not exactly known in advance. This can be due to measuring errors or simply because they cannot be predicted [6]. Here, the robustness comes in: Following an idea by Soyster [29], Ben-Tal and Nemirowski [5] coined the term of an *uncertainty set* that is added to the model and contains all possible (or likely) scenarios against which the robustness should protect. Since then, robust network design has been very actively studied. The notions of *network topology*, *cost*, *capacity*, *traffic request* and *routing* can vary – as well as the exact way in which the problem is *robustified*.

In this paper, we study a *worst-case* robust model in the sense of [5]. This means that our solutions must be feasible for all the scenarios from the uncertainty set. The uncertainty set is *finite* and *explicitly given* as part of the input (an idea that goes back to [23]). We use an *undirected* graph as the network topology and allow *dynamic routing* (each scenario may be routed on different paths). Furthermore, we assume *linear costs* for the capacities and integer multiples of a unit capacity may be installed on each edge. Each node specifies its traffic request by a scalar number that gives its supply or demand and each such traffic request may be routed on an arbitrary number of paths (the routing is *splittable*) as long as each edge carries an integer amount of flow in total. Therefore, the underlying flow model is a standard *single-commodity*, splittable network flow in our case.

To the best of our knowledge, only two prior publications on this specific problem exist. The problem was first studied by Buchheim, Liers and Sanità [9]. They gave an exact branch-and-cut algorithm that solves a flow-model MIP through sophisticated general-purpose cutting planes. Lately, Álvarez-Miranda et al. [2] introduced a capacity-based MIP-model, and discussed a preliminary set of results of the biennial "Vigoni 2011-2012" between the universities of Köln and Bologna.

Atamtürk [3] considers a variant of the non-robust single-commodity network design problem where integer multiples of a facility with fixed capacity can be installed on each arc. Ortega and Wolsey [24] report on the performance of general MIP solvers on various network design problems and develop an exact algorithm for the single-commodity fixed-charge network design problem (all arcs may be bought at a fixed-charge and then be used at full capacity).

A close variant of single-commodity RND is the *multi-commodity* robust network design problem. Here, the traffic requests specify the amount d_{ij} of flow that should be exchanged among all pairs of nodes i and j. In particular, this defines fixed source/sink pairs – which is not the case in our problem. Also, each commodity has a single source (or sink). While this condition can also be established in the single-commodity case, it requires the use of fixed-capacity edges and therefore, our single-commodity variant is not a true special case of the multi-commodity problem. Sanità showed in her doctoral thesis [27] that the multi-commodity variant is NP-hard even if there are only three scenarios, all scenarios use a unique source node and all demands are from $\{-1,0,1\}$. This immediately implies that the single-commodity variant is NP-hard as well. The thesis contains many further complexity results; among others Sanità gives a $O(\log|V|)$-approximation for the multi-commodity robust network design problem with unsplittable routing and shows that removing the integrality constraint from the capacities makes the problem polynomial time solvable. This is also true for the single-commodity RND. The multi-commodity RND was also first considered as a classical (non-robust) problem [8].

A vast variety of problems exists in the multi-commodity case. The case where the uncertainty set is finite was studied by Minoux [23], though fractional capacities are assumed in [23], and in Labbé et al. [20]. Duffield et al. [12] introduced the Hose uncertainty model in which the uncertainty set is defined by inflow and outflow bounds on all nodes. Ben-Ameur and Kerivin [4] observed that this type of uncertainty set is a polytope and developed an exact approach that additionally assumes *static routing* (i.e., in all scenarios, the flow must be routed along the same subset of paths). This configuration is also known as the *Virtual Private Network* problem. An exact approach for this problem was given in [1] under the additional constraint that each commodity may only use a single path (unsplittable routing).

In the case of dynamic routing, an exact approach by Mattia [21] exists. Bertsimas and Sim [7] introduced Γ-robustness as a general model for robustification. Exact approaches that apply this type of robustness to multicommodity network design are presented in [19].

Finally, one of the most basic network design problems, the *Steiner Tree* problem, is the special case of the single-commodity robust network design problem where for each pair i, j of Steiner nodes, there exists a scenario in which exactly i and j are terminals with supply/demand of $1/-1$. If not all the Steiner node scenarios are present, the single-commodity RND instance is instead a special case of the survivable network design problem. Note, however, that, in general, RND does not consider the requirement of disjointness that is in Survivable Network Design. We refer the reader to [18] for an extensive survey on this subject.

3 Complexity

In this section, we characterize the complexity of some RND special cases. The RND case in which we have a single scenario ($K = 1$) corresponds to a standard polynomial time minimum cost flow problem. Already for $K = 3$, RND is NP-hard (see [27]): the reduction comes from the 3-Dimensional Matching Problem for the special case of RND in which there is the same source in each scenario and balances are $\{-1, 0, 1\}$.

Motivated by the computational investigations in [9, 2], in the following, we analyze some special cases:

- RND with balances different from 1 and -1;
- RND on hypercubes with all balances equal to 1, 0, or -1;
- RND on hypercubes with all balances equal to r, 0, or $-r$, with r integer and > 1.

The analysis is intended to show some classes of hard instances and some classes of easier instances. According to the results that we present in the following subsections, we are able to get a better understanding of empirical results in [9, 2], and we propose a family of randomly generated instances that are challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$.

3.1 All balances different from 1 and -1

Because instances defined on random graphs with random integer balances on the (randomly chosen) terminals turn out to be surprisingly easy for a general-purpose MIP solver on the natural flow-formulation (1)–(5), a natural question to ask is if this special case remains NP-hard. The following theorem answers positively through a reduction from Hamiltonian cycle ([28]).

In order to prove that RND, defined on graph $G = (V, E)$ ($|V| \geq 3$), with balances different from 1 and -1, is NP-hard, let us define the following RND instance I_R. We use G without modification and install a cost of 1 on each edge. We choose some arbitrary numbering of the nodes. We install $|V| - 1$ scenarios. In scenario i, only nodes 1 and $i + 1$ are terminals; the node 1 gets a balance of 2 while the node $i + 1$ has a balance of -2.

Theorem 1. *A graph $G = (V, E)$ (with $|V| \geq 3$) has a Hamiltonian cycle C if and only if the described RND instance I_R has a solution with cost equal to $|V|$.*

Proof. If G has a Hamiltonian cycle C, we build a feasible solution for I_R by installing a capacity of 1 on each edge of C. In each scenario i, both unique terminals 1 and $i + 1$ lie on C. The node $i + 1$ decomposes C into two paths P_1, P_2 from 1 to $i + 1$ (one clockwise, one counterclockwise). We can route one unit of flow on P_1 and one unit of flow on P_2, satisfying the demands of scenario i. Thus, our solution for I_R is feasible and additionally, it has cost of $|C| = |V|$.

On the other hand, suppose we have a solution for I_R of cost $|V|$. By our choice of scenarios (we have a single source at node 1 and all other nodes are terminals in some scenario), each node must be connected to node 1. Therefore, any feasible solution for I_R must have a support S that induces a connected component of G containing all nodes. S must contain at least $|V| - 1$ edges, otherwise it cannot be connected. If S contains exactly $|V| - 1$ edges, a capacity of 2 must be installed on each edge in S in order to route all demands. However, such a solution has cost of $2 \cdot |V| - 2 > |V|$ and therefore S must contain at least $|V|$ edges. If some node in $G[S]$ has a degree of 1, then we must install a capacity of 2 on its unique incident edge. By the same argument as before, the remaining nodes $|V|-1$ nodes must be connected by at least $|V|-1$ edges. Then again, the cost of the solution is at least $|V|-1+2>|V|$. Therefore, all nodes in $G[S]$ must have a degree of at least 2 and because we can have at most $|V|$ edges in S, each node must have exactly degree 2. Together with our observation that $G[S]$ is connected and contains all nodes, we have a Hamiltonian cycle. □

3.2 Hypercubes

The authors defined a structurally difficult class of instances in [2], based on d-dimensional hypercubes. In the following we repeat the construction.

Definition 2. *A* d-dimensional hypercube $\mathscr{H}_d$ *is the result of the following recursive construction:* $\mathscr{H}_0$ *is the graph that consists of a single node. For* $d>0$, $\mathscr{H}_d$ *is obtained by duplicating the nodes and edges of* $\mathscr{H}_{d-1}$ *and connecting each node* v *to its copy* v' *with an additional edge* $\{v,v'\}$.

Definition 3. *We say that two nodes* v,w *are* diagonally opposite *on* $\mathscr{H}_d$ *iff the shortest path from* v *to* w *in* $\mathscr{H}_d$ *has maximum length, i. e., length* d.

Notice that for every node v in $\mathscr{H}_d$ there is exactly one node v^o that is diagonally opposite to v. It is well-known that $\mathscr{H}_d$ has $N_d := 2^d$ nodes and $M_d := d\cdot 2^{d-1}$ edges.

We can now define a class of instances on d-dimensional hypercubes as follows. For $d\in\mathbb{Z}_+$, consider the following instance I_d of the RND problem on $\mathscr{H}_d$. Observe that $\mathscr{H}_d$ is composed of two hypercubes H^s,H^t of dimension $d-1$. Add 2^{d-1} scenarios to $\mathscr{H}_d$. In scenario $1\le q\le 2^{d-1}$, assign a supply of 1 to the q-th node v_q (in some fixed numbering) of H^s and a demand of -1 to its diagonally opposite node v_q^o which lies in H^t 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.

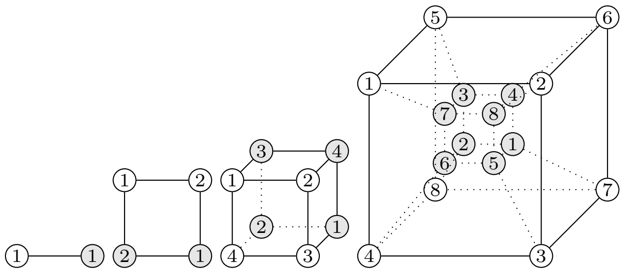

Figure 1: The hypercubes in 1, 2, 3 and 4 dimensions. Copied nodes are displayed in gray. The node numbering refers to the scenarios.

We denote the instance obtained in this way by $\mathscr{H}_d^1$. Scaling all balances in $\mathscr{H}_d^1$ by $r\in\mathbb{Z}_+$, we obtain the instance $\mathscr{H}_d^r$.

3.2.1 All balances equal to 1, 0, or -1

It is shown in [2] that this class of instances is difficult for MIP-based solution approaches as the integrality gap (i.e., the ratio of an optimum integral solution value and an optimum fractional solution value) of $\mathscr{H}_d^1$ converges to 2 as $d\to\infty$. We refer the reader to [2] for details.

3.2.2 All balances equal to r, 0, or $-r$, r integer and >1

We characterize the integrality gap for $r>1$. The optimum values for integer and fractional solutions are the same, i.e. the integrality gap is 1. We need a series of Lemmata to prove this result, stated and proven at the end of this section.

It is a well-known fact that $\mathscr{H}_d$ is hamiltonian for any $d\ge 2$ and we shall use this fact on several occasions. In particular, we can obtain a feasible integer solution for $\mathscr{H}_d^2$ by installing a capacity of 1 on each edge of a Hamiltonian cycle in $\mathscr{H}_d^2$.

Lemma 4. *For any $d \geq 2$, there is a feasible integer solution for $\mathscr{H}_d^2$ with costs 2^d.*

To derive the cost of this solution, recall that $\mathscr{H}_d$ has 2^d nodes. Similarly, we can state a feasible integer solution for $\mathscr{H}_d^3$.

Lemma 5. *For any $d \geq 3$, there is a feasible integer solution for $\mathscr{H}_d^3$ with costs $3 \cdot 2^{d-1}$.*

Proof. Let $d \geq 3$. Then $\mathscr{H}_d$ decomposes into two copies H_1, H_2 of $\mathscr{H}_{d-1}$ and a set of edges F connecting H_1 and H_2. We install a capacity of 1 on each edge in F. Since $d-1 \geq 2$, we find Hamiltonian cycles C_1, C_2 in H_1 and H_2, respectively, and install a capacity of 1 on each edge of C_1 and of C_2.

This solution is feasible: For any scenario $i \in \{1, \ldots, q\}$, let s_i, t_i be the corresponding terminal pair. We need to route three units of flow from s_i to t_i. To do that, let $s_i' \in H_2$ and $t_i' \in H_1$ be the unique nodes such that $e_1 = \{s_i, s_i'\} \in F$ and $e_2 = \{t_i', t_i\} \in F$. Also, let $e_3 = \{u, v\} \in F$ with $u \in H_1$ and $v \in H_2$ be an arbitrary connecting edge that is different from e_1 and e_2. Mark here that F contains at least four edges because $d \geq 3$. Figure 2 shows an example for the situation on $\mathscr{H}_4^3$. Now, by sending one unit of flow over each of e_1, e_2, e_3, we have reduced the instance to two instances on $\mathscr{H}_{d-1}$: The first instance is defined on H_1; here, s_i has a balance of 2 and both u and t_i' have a balance of -1. However, these balances can be routed along the Hamiltonian C_1. In the second instance, which is defined on H_2, the sink t_i has a balance of -2 and both s_i' and v have a balance of 1. Again, these balances can be routed along the Hamiltonian cycle C_2.

Both C_1 and C_2 contain exactly 2^{d-1} edges, each with capacity 1. There are 2^{d-1} edges in F, all of them having capacity 1. This gives a total cost of $3 \cdot 2^{d-1}$. □

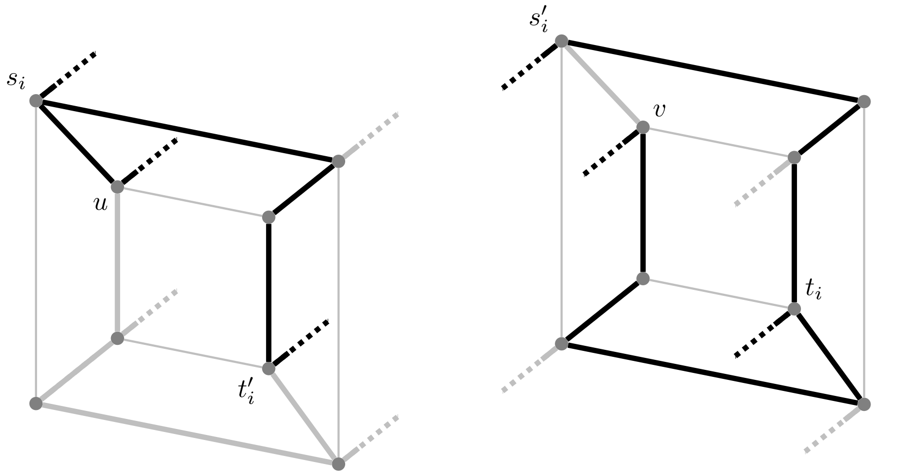

Figure 2: An example for $\mathscr{H}_4^3$.

We show next that we can construct an integer feasible solution for any $\mathscr{H}_d^r$ using the two previous ones.

Lemma 6. *Let $d \geq 2$ and let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0, 1\}$. If there exists an integer feasible solution for $\mathscr{H}_d^2$ with cost at most c_2 and an integer feasible solution for $\mathscr{H}_d^3$ with cost at most c_3, then there exist an integer feasible solution for $\mathscr{H}_d^r$ with cost at most*

$$m \cdot c_2 + n \cdot c_3.$$

Proof. We can decompose $\mathscr{H}_d^r$ into m copies of $\mathscr{H}_d^2$ and, if r is odd, a single copy of $\mathscr{H}_d^3$. The copies have costs of c_2 and c_3 each, respectively. For the i-th copy and $i = 1, \ldots, m+n$, we have an integer capacity vector u_i that allows for routing all scenarios. Then, $u = \sum_{i=1}^{m+n} u_i$ is an integer capacity vector that admits a routing of all scenarios of $\mathscr{H}_d^r$ and has exactly cost $mc_2 + nc_3$. □

To calculate the integrality gap for our solutions, we also need the value of an optimum fractional solution. Such a solution can be obtained by installing r/d units of capacity on each edge of $\mathscr{H}_d$ and since $\mathscr{H}_d$ has $d \cdot 2^{d-1}$ edges, this gives the following result.

Lemma 7. *An optimum fractional solution for $\mathscr{H}_d^r$ has a value of $r \cdot 2^{d-1}$.*

Proof. Check that a solution that installs r/d units of capacity on each edge satisfies the complementary slackness optimality conditions of the *cut-set formulation* [2] for the problem. The corresponding dual solution has a variable ξ_S for all meaningful cuts $S \subseteq V[\mathscr{H}_d]$. It sets $\xi_S := 1/2$ if $|\delta(S)| = d$ and $\xi_S = 0$. The remaining part of the proof is straight-forward if we observe that for $d \geq 3$, we have $|\delta(S)| = d$ if and only if $|S| = 1$. The full proof is shown in the Appendix. □

We can now prove that the optimum values for integer and fractional solutions are the same:

Theorem 8. *For $d \geq 3$ and $r \geq 2$, an optimum integer solution for $\mathscr{H}_d^r$ has value $r \cdot 2^{d-1}$. In particular, the integrality gap for $\mathscr{H}_d^r$ is 1.*

Proof. Let $r = 2m + 3n$ with $m \in \mathbb{Z}_+$ and $n \in \{0,1\}$. Putting together Lemma 6 with Lemma 4 and Lemma 5, we obtain that there is an integer solution for $\mathscr{H}_d^r$ with value $c_r := m \cdot 2^d + n \cdot 3 \cdot 2^{d-1}$. If r is even, we have $n = 0$ and $m = r/2$. Therefore, $c_r = r \cdot 2^{d-1}$. On the other hand, if r is odd, we have $n = 1$ and $m = (r-3)/2$. Then, $c_r = (r-3)/2 \cdot 2^d + 3 \cdot 2^{d-1} = r \cdot 2^{d-1} - 3 \cdot 2^{d-1} + 3 \cdot 2^{d-1} = r \cdot 2^{d-1}$. By Lemma 7, this is optimal. □

3.3 Challenging Instances

In the previous sections we have shown that, although *computationally easy* [9, 2], RND instances defined on random graphs with random balances are difficult in theory. The explanation of this is suggested by the fact that structurally hard instances like those defined on hypercubes and $\{-1,0,1\}$ balances become *theoretically easy* when balances are in r, 0, or $-r$, with r integer and $r > 1$. have an integrality gap of value one. Building on top of those results, we concentrate on instances on random graphs with balances $\{-1,0,1\}$ that turn out to be computationally challenging for the natural flow formulation already for $|V| = 50$ and $K = 10$. An effective heuristic approach for this family is described and computationally evaluated in Section 4 and Section 5, respectively.

4 Heuristic Algorithm

In this section, we present our heuristic algorithm, which, although general, is designed having in mind the class of hard instances introduced in the precious section, i.e., random graphs with balances of $\{-1,0,1\}$. It consists of three phases. In the first phase (*constructive phase*, CP), the graph is reduced by heuristically deleting a subset of the arcs, and a feasible solution is built. The second phase (*neighborhood search phase*, NSP) consists of a neighborhood search on the reduced graph in order to improve the solution found: in particular, the MIP flow-formulation is solved, within a time limit, by the general-purpose MIP solver Cplex. Finally, the third phase (*proximity search phase*, PSP) consists of iteratively applying a local search (by solving a carefully constructed MIP) to further improve the solution, taking into account the original graph, and is based on the recent work [13].

In the following, we describe the three phases in detail.

4.1 Constructive Phase

Initially, the graph we are dealing with is reduced, and then a solution is built. Our goal is to reduce the graph so that we are able to quickly compute a feasible solution, and we can warm start the NSP described in 4.2. At the same time, the graph reduction should not be too "aggressive", because the NSP should be able to improve the solution found. In other words, we need to find a trade-off between reducing the computing time and reducing the solution space. Note that, since the (nonzero) balances are 1 or -1 in our problem, it is not common to have large capacity values installed on the edges. Therefore, solutions differ mainly because of the different set of edges on which capacity is installed. Our goal is to select a "large enough" set of edges for our reduced graph.

The following steps are executed in the CP:

1. Consider the scenarios from 1 to K and multiply all balances by a given constant F;
2. construct a feasible solution for the new obtained RND instance (see Section 4.1.1);
3. reduce the graph by deleting all the edges that are not used in the solution found (and the nodes such that they do not have any incident edge after edge deletion) and obtain graph $\overline{G} = (\overline{V}, \overline{E})$;

4. set back the balances to 1 and -1, and construct a feasible solution (see Section 4.1.1) for the original RND instance on the reduce graph $\overline{G}$.

Step 1 is used to define the search space that we want to use in the NSP. Indeed, by increasing the absolute value of the balances, more edges are likely to be used in the solution computed in step 2 and they constitute the neighborhood of the solution computed in step 4. The next section describes how to compute a feasible solution for an RND instance.

4.1.1 Construction of a Feasible Solution

In the case of a single scenario, an algorithm for the Minimum Cost Flow (MCF) problem can be used to solve RND as follows: we define a directed graph having the same set of nodes as G and two arcs for each edge of G (one for each direction) with infinite upper bounds on the capacities. The flows that we obtain by solving the MCF problem on the defined graph determine the edge capacities, i.e., the RND solution.

In the case of K scenarios, ordered from 1 to K, in a generic scenario q we can use for free the capacities that have already been installed on the edges in scenarios $1,\dots,q-1$. A straightforward heuristic algorithm consists of iteratively solving a MCF problem for each scenario (in the order from 1 to K), updating the capacities that can be used for free after each MCF execution. In particular, we define an auxiliary directed graph $G_{dir} = (V,A)$ having the same set of nodes of G and the set of arcs defined as follows. For each edge $e = \{i, j\} \in E$, 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. Two arcs are needed for each direction in order to take into account, in a generic scenario q, the previous scenarios $1,\dots,q-1$: one arc has an upper bound on its capacity equal to the capacity already installed on the corresponding edge in the previous scenarios $1,\dots,q-1$ and has zero cost; the other arc has an infinite upper bound on its capacity and has cost equal to the cost of the corresponding edge. More precisely, for each arc $a \in A$, we initialize the upper bounds UB_a on the capacities and the costs c_a as $UB_{a_1^e} := \infty$, $UB_{a_2^e} := 0$, $UB_{a_3^e} := \infty$ and $UB_{a_4^e} := 0$; $c_{a_1^e} := c_e$, $c_{a_2^e} := 0$, $c_{a_3^e} := c_e$, $c_{a_4^e} := 0$. A MCF problem is then solved for each scenario and the upper bounds are updated according to the capacities installed on each arc.

The described algorithm follows a greedy approach. It would be useful if, when solving scenario q, we could know what happens in the next scenarios $q+1,\dots,K$ so that we could choose accordingly the best capacity installation. In addition, a MCF solution for a generic scenario q that installs capacity on more edges (at the same cost) should be preferred: indeed, it is more likely that free capacity can be used in scenarios $q+1,\dots,K$. Therefore, a MCF solution with integer flows split over disjoint paths should be preferred with respect to a MCF solution that sends flows along a single path.

Based on these two observations, we derive an improvement of the described heuristic algorithm. We apply a preprocessing in which we divide each scenario $q = 1,\dots,K$ in R sub-scenarios $g_1^q,\dots,g_R^q$, where R is an integer positive number. We consider the sub-scenarios 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 scenarios $1,\dots,K$. The balances are defined as follows: $b_v^{g_1^q} = \lfloor b_v^q/R \rfloor$, $b_v^{g_2^q} = \lfloor b_v^q/(R-1) \rfloor$, up to $b_v^{g_R^q} = b_v^q$, $v \in V$. This means that the complete MCF solution of a generic scenario q will more likely have a split integer flow over disjoint paths, because each sub-scenario might use different subsets of arcs.

The improved heuristic algorithm iteratively solves a MCF problem for each sub-scenario g_l^q ($l = 1,\dots,R$, $q = 1,\dots,K$). Let us call u^{RND} the RND solution that we compute with the improved heuristic algorithm. At $h = 0$, u^{RND} is initialized to be the zero vector. Let f^{h*} be the MCF solution obtained at iteration h corresponding to sub-scenario g_l^q. The flows in f^{h*} along the arcs with infinite upper bound determine the additional capacities that must be installed on the corresponding edges: for each $e = \{i, j\} \in E$, $u_e^{RND} = u_e^{RND} + f_{a_1^e}^{h*} + f_{a_3^e}^{h*}$. Note that, in each sub-scenario, there will always be an optimal solution using, for each edge, only arcs in one of the two directions: it is a single commodity flow, so we could simply do flow cancellation on cycles. In addition, the values u^{RND} are used to update the upper bounds on the capacities, before considering the following sub-scenario: $UB_{a_2^e} := u_e^{RND}$ and $UB_{a_4^e} := u_e^{RND}$. When all the sub-scenarios have been considered, the algorithm returns the solution found u^{RND}.

Note that in step 2 the described algorithm is used to define the reduced graph $\overline{G}$: all the edges such that $u^{RND} = 0$ are deleted from G and the nodes that do not have anymore incident edges are removed as well. Step 4 is instead used to obtain a first feasible solution to our problem and is executed on the reduced graph $\overline{G}$. Let us call u^{CP} the solution obtained at the end of the constructive phase.

4.2 Neighborhood Search Phase

This phase consists of solving an MIP flow-formulation for RND (1)–(5) on the reduced graph $\overline{G}$ defined in the previous section.

Then, NSP explores the neighborhood of solution u^{CP} by allowing the use of different edges belonging to $\overline{G}$ and by allowing the installation of different capacities on the edges. The neighborhood is explored by solving the proposed model, initialized with u^{CP}, by Cplex within a given time limit. Let us call u^{NSP} the obtained improved solution and c^{NSP} its cost. Since we consider the reduced graph, this phase is able to quickly obtain an improved solution, as it will be seen in Section 5.

4.3 Proximity Search Phase

Recently, Fischetti and Monaci [13] investigated the effects of replacing the objective function of a 0-1 Mixed-Integer Convex Programming problem with a "proximity" one, i.e., with minimizing the distance from a feasible solution of the problem, with the aim of enhancing the heuristic behavior of a black-box solver. In particular, they consider the Hamming distance:

$$\Delta(x,\tilde{x}) := \sum_{j\in J:\tilde{x}=0} x_j + \sum_{j\in J:\tilde{x}=1} (1-x_j), \tag{6}$$

where $x_j \in \{0,1\}$, $\forall j \in J$, and $\tilde{x}$ is a feasible solution to the considered problem. The idea consists of starting with an initial feasible solution $\tilde{x}$ with cost $f(\tilde{x})$, and iteratively searching for an improved solution by adding a cutoff constraint that imposes the cost of the improved solution to be smaller than $f(\tilde{x})$ by at least a quantity θ. The search is performed by solving with a black-box solver the new model with objective function that minimizes the Hamming distance from $\tilde{x}$, until a termination condition is reached, namely, until the first improved solution has been found. If no improved solution is found, θ is reduced. The process is then iterated by using the improved solution found as new $\tilde{x}$. The algorithm is terminated when a given time limit is reached. The method can be enhanced by providing an incumbent solution to each iteration of proximity search. This is obtained by adding an auxiliary continuous variable z which is used to keep the cutoff constraint feasible:

$$f(x) \le f(\tilde{x}) - \theta + z \tag{7}$$

and has a large cost M in the objective function. In this way, $\tilde{x}$ is a (very costly) feasible solution for the MIP it defines. As soon as z becomes 0, an improved solution is found.

We apply this idea to RND, i.e. we deal with an MIP. We start with initial solution u^{NSP} and we consider the original graph G (instead of the reduced one) in order to have a higher probability of improving u^{NSP}. Since capacities assume integer (and not only binary) values, we need to modify the definition of distance presented in [13]. Instead of expressing the distance as $|u-u^{NSP}|$, u_{ij} integer $\forall\{i,j\}\in E$, we fix upper bounds on the capacity variables, based on the values of u_{ij}^{NSP}, as follows. For each edge $\{i,j\}\in\overline{E}$ such that $u_{ij}^{NSP}>0$, the upper bound is set to u_{ij}^{NSP}. For all the remaining edges the upper bound is the set to be infinite. The distance is then defined as

$$\sum_{\{i,j\}\in E:u_{ij}^{NSP}=0} u_{ij} + \sum_{\{i,j\}\in E:u_{ij}^{NSP}>0} (u_{ij}^{NSP}-u_{ij}). \tag{8}$$

By imposing upper bounds on the capacity variable, we limit the search space and, consequently, the computing time, by using the solution found u^{NSP}. At the same time we leave the possibility of installing capacity on edges that were not used in the previous solution. Note that, by imposing upper bounds on the capacities, the proximity search becomes a heuristic method for RND. Given this distance measure definition, we iteratively solve the following proximity search model

$$\min \sum_{\{i,j\}\in E: u_{ij}^{NSP}=0} u_{ij} - \sum_{\{i,j\}\in E: u_{ij}^{NSP}>0} u_{ij} + Mz \tag{9}$$

$$\sum_{\{i,j\}\in E} c_{ij}u_{ij} - z\theta \le c^{NSP} - \theta, \tag{10}$$

$$\sum_{j:\{j,i\}\in E} f_{ji}^q - \sum_{j:\{i,j\}\in E} f_{ij}^q = b_i^q \qquad \forall i \in V,\ q = 1,\dots,K \tag{11}$$

$$f_{ij}^q + f_{ji}^q \le u_{ij} \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{12}$$

$$u_{ij} \le u_{ij}^{NSP} \qquad \forall \{i,j\} \in \overline{E} : u_{ij}^{NSP} > 0 \tag{13}$$

$$f_{ij}^q \ge 0 \qquad \forall \{i,j\} \in E,\ q = 1,\dots,K \tag{14}$$

$$u_{ij} \in \mathbb{Z}_+ \qquad \forall \{i,j\} \in E \tag{15}$$

$$z \in \{0,1\}, \tag{16}$$

where the auxiliary variable z is to guarantee feasibility of u^{NSP}. The objective function (9) calls for minimizing the distance from the previous solution u^{NSP} and for obtaining a solution with $z = 0$, i.e., an improved solution that respects the cutoff constraint (10). Constraint (10) imposes to obtain a reduction in the cost of the improved solution of at least θ. Constraints (11) and (12) correspond to the RND problem constraints. Constraints (13) impose the upper bounds on the capacity variables. Finally, constraints (14)–(16) impose variables' bounds. Note that z is defined as binary as it turned out in our computational experiments that it is very effective to impose branching priority on z, in order to quickly obtain a solution with $z = 0$.

Model (9)-(16) is solved by Cplex until the first feasible solution with $z = 0$ is obtained. In our experiments θ was set to 1. Therefore, if $z = 1$ the process is stopped. On the first feasible solution found, Cplex *polishing* (see, Rothberg [25]) is applied until the first improved solution is found. Formulation (9)–(16) is then solved again by replacing u^{NSP} with the improved solution. The proximity search phase is executed until a given time limit is reached. When the time limit is reached, PS returns the best solution found u^{PSP}.

5 Computational Results

In this section, we report the computational results that we achieved on instances generated on random graphs with balances $\{-1,0,1\}$. Instances are generated as follows: n nodes are randomly located in a unit Euclidean square. Two nodes are connected with an edge if the Euclidean distance is less than $\alpha/\sqrt{n}$ where α is a parameter set to 2 in our generator. The edge cost for capacity installation is proportional to the Euclidean distance. For each scenario, 25%, 50% or 100% of the nodes are randomly selected to be terminals. We consider 5 or 10 scenarios.

The heuristic was developed in C language, and Cplex version 12.5 with 4 threads was used as a general purpose solver. The tests were executed on a PC 1.73 GHz, 6 GB Ram. The computing times are expressed in seconds. The algorithm CS2 by Goldberg [14] was used for solving the Minimum Cost Flow. The following parameter setting is used for the heuristic: a time limit of 300 seconds is given to NSP and a time limit of 600 seconds is given to PSP. The total time limit for the heuristic is fixed to 900 seconds, because the computation time of the CP is negligible. We fix $F = 100$, $R = 10$, $\theta = 1$ and $M = 100u^{NSP}$, based on parameter tuning.

An important feature of our heuristic algorithm is that it is robust to parameter setting, i.e. the efficacy of the algorithm is not really dependent on the specific parameter values, as long as balances are increased and scenarios are split in sub-scenarios (i.e., $F > 1$ and $R > 1$). In particular, the difference between average gaps, computed with respect to the solutions obtained by Cplex 5h, for different combinations of F and R (with $F > 1$ and $R > 1$) are negligible (below 1%). Among all combinations, the one that has the best balance between gap and standard deviation is given by $R = 10$ and $F = 100$. The other parameter used in our heuristic algorithm is θ. We choose $\theta = 1$ as a conservative value, i.e. a value that allows us to obtain good solutions on average on all the instances. In particular, we observed that, on the small instances (with 50 to 100 nodes), larger values for θ do not produce high quality solutions. When the instances get larger, a more aggressive policy (e.g. with $\theta = 100$) can give better results. We decided to keep a conservative value, in order to avoid parameter overtuning.

In Figure 3, we show the results obtained, with a time limit of 900 seconds, by the proposed method after each of the three phases described in Section 4. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented with respect to the solutions obtained after the constructive phase and we show in black the percentage improvement of the solutions obtained after the neighborhood search phase (NSP) and in gray the percentage improvement of the solutions obtained after the proximity search phase (PSP). On the right of the figure, we also show the average and standard deviation for each class of instances. As it can be seen, both the NSP and the PSP are effective in obtaining improvements for all instances but six, on which only PSP is able to improve the solution found u^{CP} after the constructive phase. The detailed results are reported in Table 2 in the Appendix.

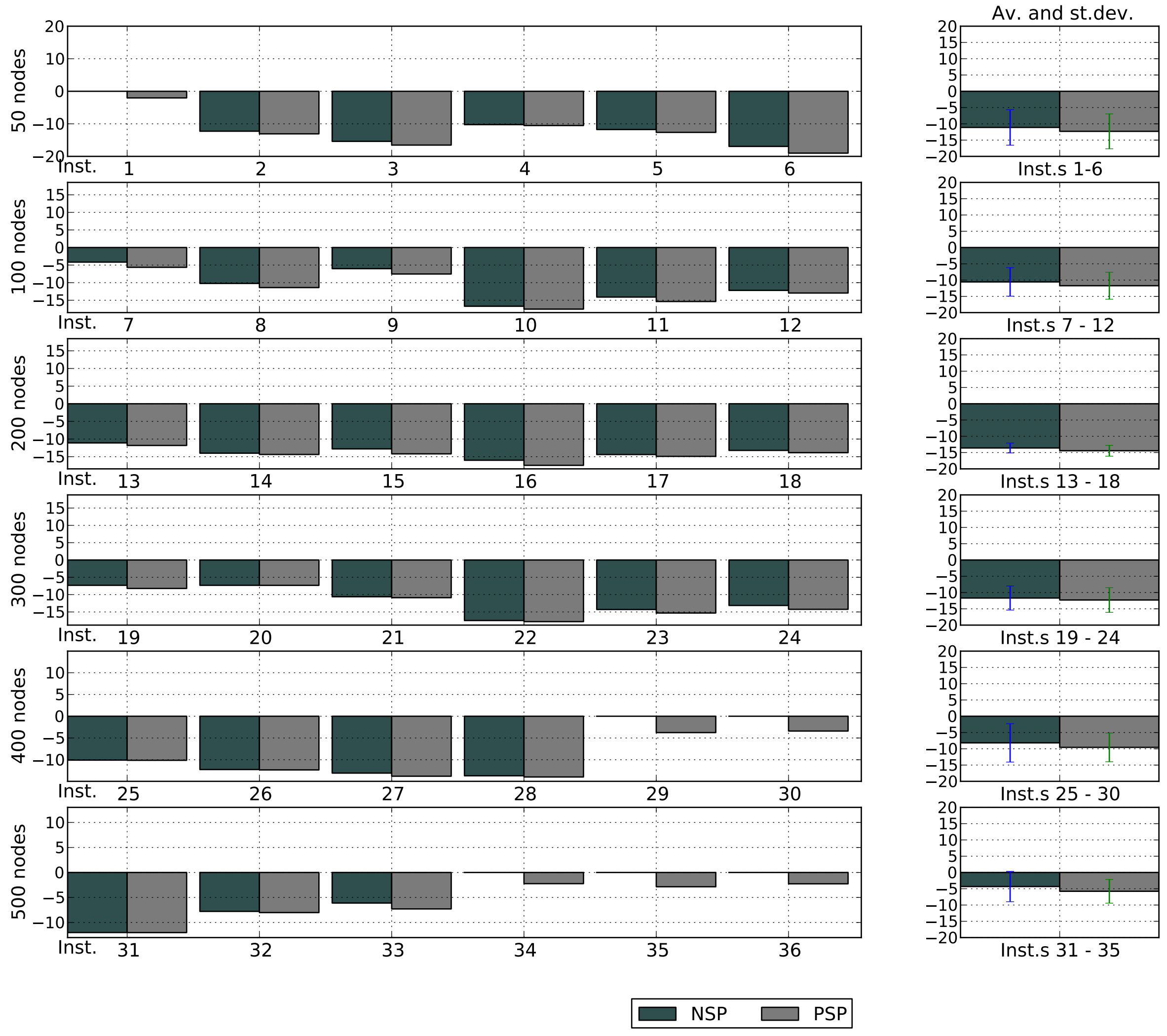

Figure 3: Comparison of the results obtained after NSP and after PSP with those obtained after CP.

In the following, we present a comparison of the results obtained by the proposed heuristic (indicated as RND Heur.) with those obtained by Cplex applied to the MIP flow model (1)-(5) on the original graph G. In particular, we show the results obtained when Cplex is run with a time limit of five hours (Cplex 5h) in default setting, and the results obtained with Cplex in the effective heuristic configurations suggested in [13] (Cplex Pol. 900s), i.e., solution *polishing* is applied, and the time limit is set to 900 seconds. The proposed method, Cplex 5h and Cplex Pol. 900s are initialized with the solution u^{CP} constructed as explained in Section 4.1.

In Table 1, we report the results obtained by Cplex 5h, which will be used as our benchmark for comparison. In particular, we report the data on the instances, the solution (u^{CP}) obtained at the end of the constructive phase and used for initializing each of the methods, the best lower bound (LB) and the best upper bound (UB) obtained by Cplex 5h, the duality gap (Gap%), the number of branch and bound nodes (BBn), and the computing time. As it can be seen from Table 1, the time limit is reached for all instances but the three smallest ones for which Cplex is able to prove optimality. For the remaining instances, the duality gaps are often quite large and for seven instances Cplex is not even able to improve the initial solution.

					Cplex 5h				
Inst.	n	$t\%$	K	u^{CP}	LB	UB	Gap%	BBn	Time
1	50	25	5	22904	22438	22438	0.00	1977	6
2	50	50	5	60921	52947	52952	0.01	602223	2666
3	50	100	5	79487	66334	66340	0.01	968741	4634
4	50	25	10	52835	44129	47272	6.65	464679	18000
5	50	50	10	66781	55277	57861	4.47	374536	18000
6	50	100	10	88323	70085	71526	2.01	544735	18000
7	100	25	5	39255	36741	37031	0.78	960936	18000
8	100	50	5	89264	76498	78702	2.80	430334	18000
9	100	100	5	81126	74331	74822	0.66	675507	18000
10	100	25	10	86929	67148	71192	5.68	63827	18000
11	100	50	10	115437	92265	97246	5.12	44262	18000
12	100	100	10	132233	112062	114624	2.24	76558	18000
13	200	25	5	98497	77288	85676	9.79	67461	18000
14	200	50	5	142509	113158	122062	7.29	53440	18000
15	200	100	5	169962	139302	144508	3.60	75979	18000
16	200	25	10	134999	98358	114995	14.47	11630	18000
17	200	50	10	173335	133819	148087	9.63	9406	18000
18	200	100	10	219903	175660	184992	5.04	14684	18000
19	300	25	5	92259	73805	83302	11.40	28125	18000
20	300	50	5	139954	115164	128296	10.24	22212	18000
21	300	100	5	183689	150048	162860	7.87	22284	18000
22	300	25	10	148349	103953	148349	29.93	2927	18000
23	300	50	10	201301	151200	199456	24.19	2198	18000
24	300	100	10	271340	214577	268072	19.96	2603	18000
25	400	25	5	109241	87877	98297	10.60	14881	18000
26	400	50	5	217300	175286	190328	7.90	12671	18000
27	400	100	5	291469	234266	252987	7.40	18336	18000
28	400	25	10	158033	117143	158033	25.87	1191	18000
29	400	50	10	253648	191242	253648	24.60	1239	18000
30	400	100	10	325512	255769	325512	21.43	1278	18000
31	500	25	5	106191	75197	98778	23.87	7576	18000
32	500	50	5	189269	159465	177572	10.20	10186	18000
33	500	100	5	261922	216832	241684	10.28	8584	18000
34	500	25	10	214149	153247	214149	28.44	325	18000
35	500	50	10	262379	196930	262379	24.94	315	18000
36	500	100	10	323275	249201	323275	22.91	564	18000

Table 1: Results obtained with Cplex on the MIP flow formulation in five hours of time limit.

In order to measure the performance of the proposed method, we show in Figure 4 the comparison between the results we obtain in 900 seconds of time limit and the results obtained by Cplex 5h and by Cplex Pol. 900s. The detailed results are reported in Table 3 in the Appendix. In particular, we show one graphic for each class of instances (from 50 nodes to 500 nodes). In this graphic, the comparison is presented with respect to the solutions obtained by Cplex 5h and we show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur. On the right of the figure, we also show the average and standard deviation for each class of instances.

As it is evident from Figure 4, the three methods obtain comparable results for instances with up to 100 nodes. However, as the instances get larger, the proposed method becomes more effective than the other ones, and it is able to improve the results obtained by the other two methods. In particular, compared to Cplex Pol. 900s that has the same time limit, the proposed method always obtains better solutions for instances with at least 300 nodes. It obtains solutions with a cost less or equal than those obtained by Cplex Pol. 900s for 27 out of 36 instances, and is at most 1.05% worse for a single instance. The improvement is significant (between 3% and more than 14%) for 14 out of 36 instances. Even compared to Cplex run for five hours, the proposed method performs on average better on instances with at least 200 nodes, especially when we have 10 scenarios. It is able

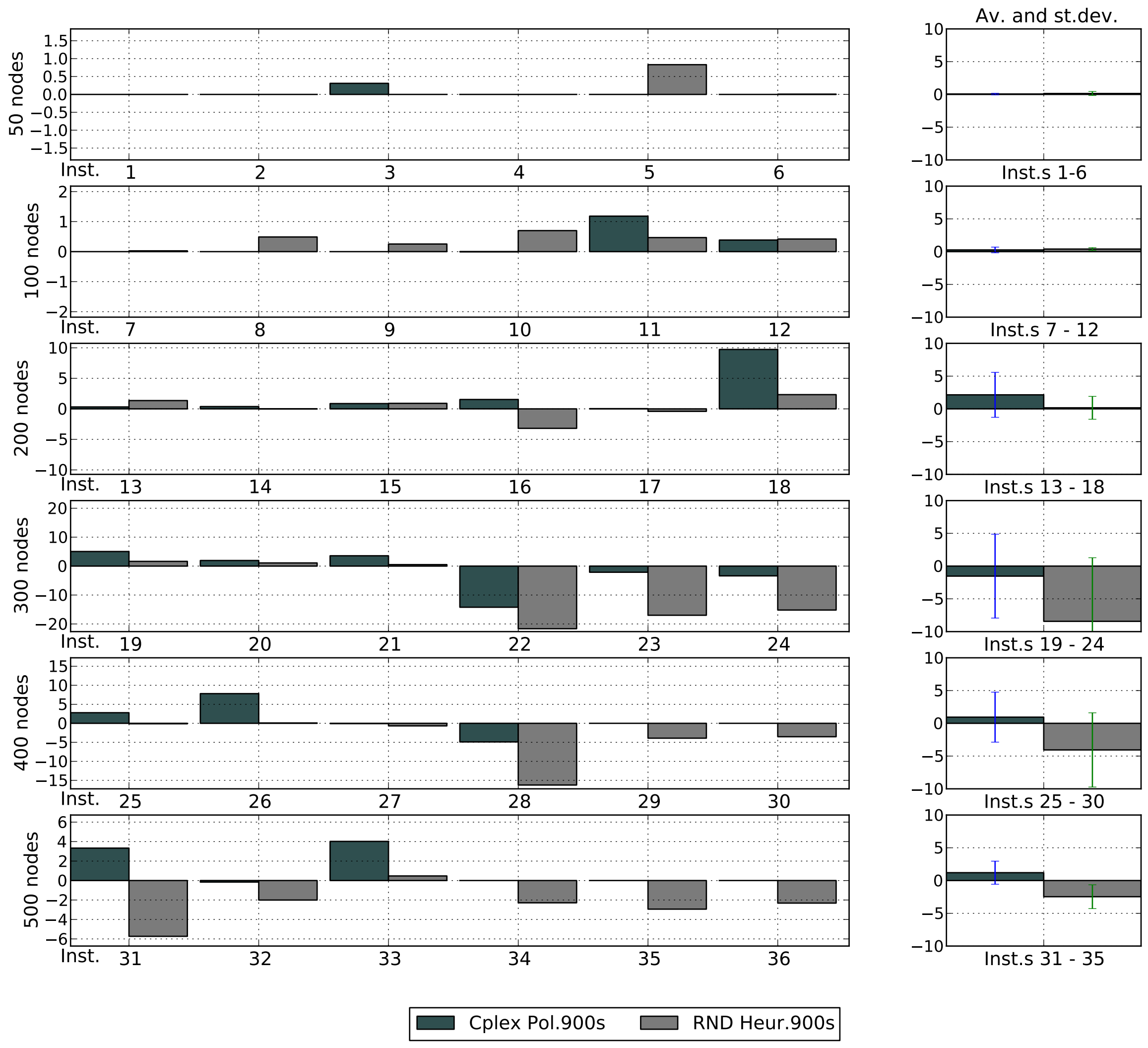

Figure 4: Comparison of the proposed heuristic RND (time limit of 15 minutes) with Cplex Pol. 900s and with Cplex 5h.

to obtain better or equal solutions for 20 out of 36 instances. The average percentage improvement with respect to Cplex 5h and Cplex Pol. 900s is 2.38% and 2.90%, respectively.

In order to further validate the results presented in Figure 4, we performed extensive computational experiments on instances with $n = 300$ and $n = 400$ nodes. In particular, we considered five instances for each sub-class, defined by selecting $t \in \{25\%, 50\%, 100\%\}$ and $k \in \{5, 10\}$. This gives a total testbed of 60 instances. In Figure 5, we show the comparison between the three methods. The comparison is presented with respect to the solutions obtained by Cplex in five hours. We show in black the percentage gap of the solutions obtained by Cplex Pol. 900s and in gray the percentage gap of the solutions obtained by RND Heur in 900s. Compared to Cplex Pol. 900s, that has the same time limit, the proposed method always obtains better solutions, and, compared to Cplex 5h, performs better on all instances with 10 scenarios, confirming the effectiveness of the proposed approach.

6 Conclusions and Future Research

We have presented a single-commodity robust network design problem and we have shown complexity results for special classes of instances, including hypercubes. By the complexity analysis, we have shown that instances with random integer balances different from 1 and -1 are NP-hard, even if computationally easy ([9, 2]). In order to explain why, we have shown that instances defined on hypercubes with balances in $\{-r, 0, r\}$ (r integer, $r > 1$) are theoretically easy, while instances defined on hypercubes with balances in $\{-1, 0, 1\}$ are structurally

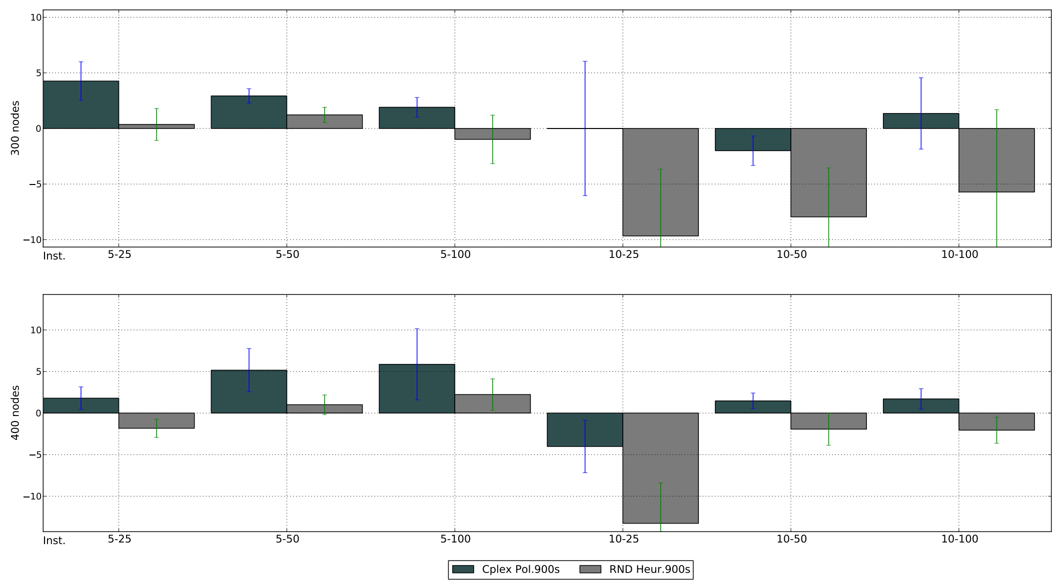

Figure 5: Comparison of the three methods on additional instances with 300 and 400 nodes.

hard. This has motivated us to study instances (defined on random graphs) with balances of $\{-1,0,1\}$. We have developed a heuristic algorithm composed of three phases. The first one reduces the instance graph and constructs a feasible solution, the second one solves an MIP flow-formulation of the problem on the reduced graph for a given time limit, in order to improve the solution found, and the last phase applies a modified version of the recent technique of proximity search to further improve the solution. We have tested the proposed method on randomly generated instances with balances of $\{-1,0,1\}$, and we have compared the obtained results with those obtained by Cplex both in 5 hours (default version) or by using the *polishing* algorithm to enhance its heuristic behavior (for 900 seconds). The results show that our method is comparable with the other ones for instances with up to 100 nodes, but obtains better solutions for larger instances. Future research can be devoted to extend the proposed algorithm to the multi-commodity case. In addition, the proposed method takes into account the balances of all the scenarios, but a less conservative approach could be considered, for example, by taking into account the probability of each scenario. Other extensions could be to tackle related variants of robust network design, such as Survivable Network Design: mostly the constructive phase needs to be modified, as long as a good MIP formulation exists. Additional parameter tuning might be necessary as well.

Acknowledgments

Financial support is acknowledged from the Ateneo Italo-Tedesco (VIGONI programme 2011-2012). The authors would like to thank Mike Jünger, Frauke Liers and Laura Sanità for their valuable comments. The third author acknowledges the support by MIUR under the PRIN2009 grant.

References

[1] A. Altin, E. Amaldi, P. Belotti, and M.C. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49:100–115, 2007.

[2] E. Álvarez-Miranda, V. Cacchiani, T. Dorneth, M. Jünger, F. Liers, A. Lodi, T. Parriani, and D. Schmidt. Models and algorithms for robust network design with several traffic scenarios. In A. R. Mahjoub et al., editor, *ISCO 2012*, volume 7422 of *Lecture Notes in Computer Science*, pages 261–272. Springer, 2012.

[3] A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92:425–437, 2000.

[4] W. Ben-Ameur and H. Kerivin. Routing of uncertain traffic demands. *Optimization and Engineering*, 6:283–313, 2005.

[5] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[6] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 2000.

[7] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[8] D. Bienstock, S. Chopra, O. Günlük, and C.H. Tsai. Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81(2):177–199, 1998.

[9] C. Buchheim, F. Liers, and L. Sanità. An exact algorithm for robust network design. In J. Pahl, T. Reiners, and S. Voß, editors, *INOC*, volume 6701 of *Lecture Notes in Computer Science*, pages 7–17. Springer, 2011.

[10] C. Chekuri. Routing and network design with robustness to changing or uncertain traffic demands. *SIGACT News*, 38:106–128, 2007.

[11] R. T. Chien. Synthesis of a communication net. *IBM Journal of Research and Development*, 4(3):311–320, 1960.

[12] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J.E. van der Merwe. A flexible model for resource management in virtual private networks. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 95–108. ACM, 1999.

[13] M. Fischetti and M. Monaci. Proximity search for 0-1 mixed-integer convex programming. Technical report, DEI, University of Padova, Italy, 2013.

[14] A.V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms*, 22(1):1–29, 1997.

[15] R. Gomory and T. Hu. An application of generalized linear programming to network flows. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):260–283, 1962.

[16] R.E Gomory and T.C. Hu. Multi-terminal network flow. *SIAM Journal on Applied Mathematics*, 9:551–570, 1961.

[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] H. Kerivin and A.R. Mahjoub. Design of survivable networks: A survey. In *In Networks*, pages 1–21, 2005.

[19] A.M.C.A. Koster, M. Kutschka, and C. Raack. Robust network design: Formulations, valid inequalities, and computations. *Networks*, 61(2):128–149, 2013.

[20] M. Labbé, R. Séguin, P. Soriano, and C. Wynants. Network synthesis with non-simultaneous multicommodity flow requirements: Bounds and heuristics, 1999.

[21] S. Mattia. The robust network loading problem with dynamic routing. *Computational Optimization and Applications*, 54:619–643, 2013.

[22] Karl Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10(1):96–115, 1927.

[23] M. Minoux. Optimum synthesis of a network with non-simultaneous multicommodity flow requirements. In *Annals of Discrete Mathematics (11) Studies on Graphs and Discrete Programming*, volume 59, pages 269 – 277. North-Holland, 1981.

[24] F. Ortega and L.A. Wolsey. A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem. *Networks*, 41(3):143–158, 2003.

[25] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, 19:534–541, 2007.

[26] Youcef Saad and Martin H. Schultz. Topological properties of hypercubes. Technical Report YALEU/DCS/TR389, Yale University, 1985.

[27] L. Sanità. *Robust Network Design*. Ph.D. Thesis. Università La Sapienza, Roma, 2009.

[28] L. Sanità. Private communication. 2013.

[29] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

Appendix

Proof of Lemma 7

We now provide a full version of the proof of Lemma 7. Let us denote by V^d and E^d the set of nodes and edges of $\mathscr{H}_d$, respectively.

Proof of Lemma 7. If we define the set

$$\mathscr{S} := \left\{ S \subset V^d \mid S \text{ is connected and separates at least one } v_q \text{ from its partner } v_q^o \right\},$$

we can find an optimum fractional solution for $\mathscr{H}_d^r$ with the following linear program [2].

$$\begin{aligned} \min \sum_{e \in E^d} u_e & \\ \sum_{e \in \delta(S)} u_e \geq r & \quad \text{for all } S \in \mathscr{S} \\ u_e \geq 0 & \quad \text{for all } e \in E \end{aligned} \tag{CAP}$$

If $d = 2$, it holds that $|S| = d = 2$ for all $S \in \mathscr{S}$. Consequently, if we set $u_e = r/2$ for all $e \in E^d$, all primal constraints are satisfied with equality and the solution is optimal. If $d \geq 3$, we introduce dual variables ξ_S for all $S \in \mathscr{S}$ and obtain the following dual program:

$$\begin{aligned} \max \sum_{S \in \mathscr{S}} r \cdot \xi_S & \\ \sum_{\substack{S \in \mathscr{S}: \\ \{i,j\} \in S}} \xi_S \leq 1 & \quad \text{for all } \{i,j\} \in E^d \\ \xi_S \geq 0 & \end{aligned} \tag{CAP^*}$$

We consider the following pair of primal and dual solutions:

$$u_e := r/d \qquad \text{for all } e \in E^d \qquad\qquad \xi_S := \begin{cases} 0, & \text{if } |\delta(S)| > d \\ 1/2, & \text{if } |\delta(S)| = d \end{cases} \qquad \text{for all } S \in \mathscr{S}.$$

To prove our claim, we need to show that u and ξ are feasible and satisfy complementary slackness. Feasibility of u follows by the first part of Lemma 9: For all $S \in \mathscr{S}$, we have $|\delta(S)| \geq d$ and thus $\sum_{e \in \delta(S)} u_e = |\delta(S)|(r/d) \geq r$. Observe that by the second part of Lemma 9 equality holds if and only if $|\delta(S)| = d$. Thus, we have $(\sum_{e \in \delta(S)} u_e - r) \cdot \xi_S = 0$ for all $S \in \mathscr{S}$, yielding primal complementary slackness. To see why ξ is feasible for (CAP^*) we need to show that

$$\sum_{\substack{S \in \mathscr{S}: \\ \{i,j\} \in S}} \xi_S = \sum_{\substack{S \in \mathscr{S}: \\ |\delta(S)| = d \\ \{i,j\} \in S}} \xi_S \leq 1 \qquad \text{for all } \{i,j\} \in E^d.$$

By applying Lemma 9, we can rewrite this as

$$\sum_{\substack{S \in \mathscr{S}: \\ |\delta(S)| = d \\ \{i,j\} \in S}} \xi_S = \sum_{\substack{S \in \mathscr{S}: \\ |S| = 1 \\ \{i,j\} \in S}} \xi_S = \xi_{\{i\}} + \xi_{\{j\}} = 1 \qquad \text{for all } \{i,j\} \in E^d$$

which also yields that $(\sum_{S \in \mathscr{S}: e \in S} \xi_S - 1) \cdot u_e = 0$ for all $e \in E^d$, i.e., we have dual complementary slackness. Finally, both solutions yield the desired objective value of $\sum_{e \in E^d} r/d = d \cdot 2^{d-1} \cdot (r/d) = r \cdot 2^{d-1}$. □

The following lemma provides the missing piece for the above proof.

Lemma 9. *Let* $d \geq 3$. *Then in* $\mathscr{H}_d$, $|\delta(S)| \geq d$ *for all* $\emptyset \subsetneq S \subsetneq V^d$. *Moreover, equality is attained if and only if* $|S| = 1$ *or* $|S| = |V^d| - 1$.

Proof. The first part of the lemma is well-known: Saad and Schultz [26, Propositions 3.2 and 3.3] proved that for any two nodes i, j of a d-dimensional hypercube, there are at least d node disjoint paths between i and j. By Menger's Theorem [22], this implies that $|\delta(S)| \geq d$ for all $\emptyset \subsetneq S \subsetneq V^d$. Also, if S contains a single node i, then $|\delta(S)| = |\delta(i)| = d$. It remains to show that the inequality is strict if $2 \leq |S| \leq |V^d| - 2$. Without loss of generality, we can assume that $|S| \leq \frac{1}{2}|V^d|$ since $\delta(S) = \delta(V \setminus S)$.

Now, choose an arbitrary decomposition of $\mathscr{H}_d$ into two $(d-1)$-dimensional hypercubes $H_1 = (V_1, E_1), H_2 = (V_2, E_2)$ such that neither of $S_1 := S \cap V_1$ and $S_2 := S \cap V_2$ is empty. This is possible because S contains at least two and at most $|V|/2$ nodes. It also implies that neither $S_1 = V_1$ nor $S_2 = V_2$, as otherwise S_2 or S_1 would be empty, respectively.

For $i = 1, 2$, the node set S_i defines a cut $\delta_i(S_i)$ in H_i. Since $S_i \neq \emptyset$ and $S_i \neq V_i$, we know that $|\delta_i(S_i)| \geq d - 1$, since H_i is a $(d-1)$-dimensional hypercube. Also, $\delta_1(S_1), \delta_2(S_2) \subseteq \delta(S)$ and therefore $|\delta(S)| \geq 2 \cdot (d-1) > d$ for $d \geq 3$. □

Tables

In Table 2, we show the results obtained by the proposed method after each of the three phases described in Section 4. In particular, we show the instance name (Inst.), the number n of nodes in the graph G, the percentage $t\%$ of nodes that are terminals, the number K of considered scenarios, the solution (u^{CP}) obtained at the end of the constructive phase, the solution u^{NSP} obtained after the neighborhood search phase (and the corresponding percentage improvement $Impr_{u^{CP}}\%$ with respect to u^{CP}) and the final solution u^{PSP} provided by our method by applying proximity search (and the corresponding percentage improvement $Impr_{u^{NSP}}\%$ with respect to u^{CP}). We do not report the computing times, as the time limit of 900 seconds is reached for all instances.

In Table 3, we report the value of the best solution obtained by each method, and, for Cplex Pol. 900s and for RND Heur., we show the percentage gap Gap$_{C^{5h}}$% to the best upper bound computed by Cplex 5h. In the last column, we also show the percentage gap Gap$_{C^{900s}}$% between the solutions obtained by RND Heur. and Cplex Pol. 900s. Finally, in the last rows of the table, we show the average (Avg.), the median (Median) and the standard deviation (StDev.) of the percentage gaps, as well as the minimum (Min) and the maximum (Max) percentage gap.

Inst.	n	$t\%$	K	u^{CP}	u^{NSP}	$Impr_{u^{CP}}\%$	u^{PSP}	$Impr_{u^{NSP}}\%$
1	50	25	5	22904	22904	0.00	22438	-2.03
2	50	50	5	60921	53443	-12.27	52952	-13.08
3	50	100	5	79487	67250	-15.39	66340	-16.54
4	50	25	10	52835	47419	-10.25	47272	-10.53
5	50	50	10	66781	58928	-11.76	58346	-12.63
6	50	100	10	88323	73352	-16.95	71530	-19.01
7	100	25	5	39255	37624	-4.15	37041	-5.64
8	100	50	5	89264	80139	-10.22	79088	-11.40
9	100	100	5	81126	76247	-6.01	75012	-7.54
10	100	25	10	86929	72399	-16.71	71694	-17.53
11	100	50	10	115437	99155	-14.10	97703	-15.36
12	100	100	10	132233	116100	-12.20	115107	-12.95
13	200	25	5	98497	87562	-11.10	86855	-11.82
14	200	50	5	142509	122543	-14.01	122032	-14.37
15	200	100	5	169962	148214	-12.80	145826	-14.20
16	200	25	10	134999	113380	-16.01	111439	-17.45
17	200	50	10	173335	148397	-14.39	147487	-14.91
18	200	100	10	219903	190824	-13.22	189406	-13.87
19	300	25	5	92259	85518	-7.31	84681	-8.21
20	300	50	5	139954	129723	-7.31	129709	-7.32
21	300	100	5	183689	164206	-10.61	163699	-10.88
22	300	25	10	148349	122424	-17.48	121953	-17.79
23	300	50	10	201301	172539	-14.29	170487	-15.31
24	300	100	10	271340	235728	-13.12	232706	-14.24
25	400	25	5	109241	98219	-10.09	98176	-10.13
26	400	50	5	217300	190677	-12.25	190492	-12.34
27	400	100	5	291469	253378	-13.07	251291	-13.78
28	400	25	10	158033	136413	-13.68	135968	-13.96
29	400	50	10	253648	253648	0.00	244109	-3.76
30	400	100	10	325512	325512	0.00	314428	-3.41
31	500	25	5	106191	93433	-12.01	93425	-12.02
32	500	50	5	189269	174540	-7.78	174082	-8.02
33	500	100	5	261922	245907	-6.11	242828	-7.29
34	500	25	10	214149	214149	0.00	209360	-2.24
35	500	50	10	262379	262379	0.00	254891	-2.85
36	500	100	10	323275	323275	0.00	315955	-2.26
Avg.						-9.91		-11.02

Table 2: Results obtained by the proposed method within 900 seconds of time limit.

				Cplex 5h	Cplex Pol. 900s		RND Heur. 900s		
Inst.	n	$t\%$	K	UB	UB	Gap$_{C^{5h}}\%$	u^{PSP}	Gap$_{C^{5h}}\%$	Gap$_{C^{900s}}\%$
1	50	25	5	22438	22438	0.00	22438	0.00	0.00
2	50	50	5	52952	52952	0.00	52952	0.00	0.00
3	50	100	5	66340	66546	0.31	66340	0.00	-0.31
4	50	25	10	47272	47272	0.00	47272	0.00	0.00
5	50	50	10	57861	57861	0.00	58346	0.83	0.83
6	50	100	10	71526	71526	0.00	71530	0.01	0.01
7	100	25	5	37031	37031	0.00	37041	0.03	0.03
8	100	50	5	78702	78702	0.00	79088	0.49	0.49
9	100	100	5	74822	74822	0.00	75012	0.25	0.25
10	100	25	10	71192	71189	0.00	71694	0.70	0.70
11	100	50	10	97246	98409	1.18	97703	0.47	-0.72
12	100	100	10	114624	115068	0.39	115107	0.42	0.03
13	200	25	5	85676	85947	0.32	86855	1.36	1.05
14	200	50	5	122062	122522	0.38	122032	-0.02	-0.40
15	200	100	5	144508	145770	0.87	145826	0.90	0.04
16	200	25	10	114995	116786	1.53	111439	-3.19	-4.80
17	200	50	10	148087	148138	0.03	147487	-0.41	-0.44
18	200	100	10	184992	204936	9.73	189406	2.33	-8.20
19	300	25	5	83302	87723	5.04	84681	1.63	-3.59
20	300	50	5	128296	130825	1.93	129709	1.09	-0.86
21	300	100	5	162860	168882	3.57	163699	0.51	-3.17
22	300	25	10	148349	129877	-14.22	121953	-21.64	-6.50
23	300	50	10	199456	195300	-2.13	170487	-16.99	-14.55
24	300	100	10	268072	259317	-3.38	232706	-15.20	-11.44
25	400	25	5	98297	101115	2.79	98176	-0.12	-2.99
26	400	50	5	190328	206445	7.81	190492	0.09	-8.37
27	400	100	5	252987	252842	-0.06	251291	-0.67	-0.62
28	400	25	10	158033	150661	-4.89	135968	-16.23	-10.81
29	400	50	10	253648	253648	0.00	244109	-3.91	-3.91
30	400	100	10	325512	325512	0.00	314428	-3.53	-3.53
31	500	25	5	98778	102182	3.33	93425	-5.73	-9.37
32	500	50	5	177572	177292	-0.16	174082	-2.00	-1.84
33	500	100	5	241684	251807	4.02	242828	0.47	-3.70
34	500	25	10	214149	214149	0.00	209360	-2.29	-2.29
35	500	50	10	262379	262379	0.00	254891	-2.94	-2.94
36	500	100	10	323275	323275	0.00	315955	-2.32	-2.32
Avg.						0.51		-2.38	-2.90
Median						0.00		0.00	-1.35
StDev.						3.67		5.75	3.96
Min						-14.22		-21.64	-14.55
Max						9.73		2.33	1.05

Table 3: Comparison of the proposed heuristic (time limit of 15 minutes) with Cplex (time limit of 5 hours or 15 minutes).