RAIRO-Oper. Res. 49 (2015) 569–588
DOI: 10.1051/ro/2014056

RAIRO Operations Research
www.rairo-ro.org

VIRTUAL PRIVATE NETWORK DESIGN OVER THE FIRST CHVÁTAL CLOSURE

Ahmad Moradi[1], Andrea Lodi[2] and S. Mehdi Hashemi[1]

Abstract. In this paper we consider the virtual private network (VPN) design problem. Given upper bounds on the amount of traffic that an endpoint could send or receive, the problem needs to reserve enough capacities in such a way that any demand matrix that respects the upper bound could be routed without exceeding the reserved capacities and the total reservation cost is minimized. In *On the difficulty of virtual private network instances* (*Networks* **63** (2014) 327–333), we argued that the computational investigation on exact mathematical programming approaches for VPN needs to be revised after that challenging instances have been exposed. To that end, we consider the VPN design problem over the first Chvátal closure and demonstrate that tight solutions could be found for the VPN design problem only by optimizing over the closure. First, we perform theoretical investigation on adding rank-1 Chvátal–Gomory cuts to the problem. Along the way, an important property for such cuts is proved that omits a large number of redundant rank-1 cuts. We then provide interesting insights about the problem and reduce the existing MIP formulations to a binary one. On the computational side, we investigate the idea of adding rank-1 cuts more aggressively in order to computationally evaluate tightness of the first Chvátal closure for the VPN design problem. Here, the binary reduction plays an important role allowing the use of special cuts of the first closure, namely the zero-half cuts. We show that, almost all the success of the first Chvátal closure of the VPN design problem in raising dual bound is due to zero-half cuts. Our experiments on the benchmark instances in this article show that a state-of-the-art IP solver without using zero-half cuts could not even hit the challenging benchmarks. As a results a cut-and-branch framework that aggressively

Received June 26, 2014. Accepted November 5, 2014.

[1] Department of Mathematics and Computer Science, Amirkabir University of Technology, No. 424, Hafez Avenue, Tehran, Iran. moradi@aut.ac.ir; hashemi@aut.ac.ir

[2] DEI, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. andrea.lodi@unibo.it

Article published by EDP Sciences © EDP Sciences, ROADEF, SMAI 2015

adds such cuts at the root could solve the challenging VPN instances to the extent of zero or small integrality gap in a reasonable amount of time.

Keywords. Binary formulation, Chvátal-Gomory closure, cutting planes, computational analysis, VPN design problem.

Mathematics Subject Classification. 90C10, 90C11, 90C57.

1. Introduction

As a business grows, it might expand to geographically diverse sites, offices and remote employees that need to share information in a fast and secure way. One popular technology to address this requirement is to use a virtual private network. By means of this technology distant parts of the business could be connected into a common network and communicate via secured tunnels.

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer to send and receive data across shared or public networks as if it were directly connected to the private network, while benefiting from the functionality, security and management policies of the private network [22]. The base motivation for VPNs lies in the economics of communications. Nowadays communications systems typically exhibit the characteristic of a high fixed-cost component, and smaller variable-cost components that vary with the transport capacity, or bandwidth, of the system. Within this economic environment, it is generally financially attractive to bundle numerous discrete communications services onto a common, high-capacity communications platform, allowing the high fixed-cost component associated with the platform to be amortized over a larger number of network clients [12].

In this paper, we consider a real-world network design problem that occurs while provisioning virtual private networks. In a basic network design problem, we are usually given a (traffic) demand matrix and we want to determine how much capacity is needed and how to economically distribute it in the network to address some routing/flow constraints [26]. Here, it is usually assumed that the demand matrix is fixed and known. This simplification comes at a price as, in real world, traffic demands may be uncertain or rapidly changing. A practical way to address this issue is to overestimate demand values, at the price of network resources such as the installed capacities being largely wasted. Fortunately, there is still other more flexible way to address this issue through the so-called *hose model*. The hose model is originally introduced in [9] in the context of virtual private networks. In this model only upper bounds on the amount of traffic arriving/leaving each endpoint is considered to be known in advance. Let us call the upper bounds *hose thresholds*. The model allows for changes in demand matrix as the traffic on an endpoint could be arbitrarily distributed over the other endpoints. Then, the capacity reserved on the network must support any demand matrix that respects the hose thresholds.

In the VPN design problem, given the hose thresholds on some endpoints, we need to reserve enough capacities in such a way that any demand matrix that respects the thresholds could be routed without exceeding the reserved capacities and the total reservation cost is minimized. As customary in network design, the VPN design problem is considered in many variants when different routing restrictions (splittable, unsplittable) are imposed or different cost structures (linear, nonlinear) are considered. An interested reader is referred to [24] for a structured review on different variants of the VPN design problem. The version we refer to in this paper, called *asymmetric* VPN design problem, is formally defined below.

In the following we use the same notation used in [23]. In an instance of the asymmetric VPN design problem, we are given an undirected graph $G = (V, E)$ and a nonempty terminal set $T \subset V$. Each edge $\{i, j\} \in E$ has a per-unit capacity reservation cost $c_{ij} \geq 0$ and each terminal node $s \in T$ has associated hose nonnegative thresholds b_s^+ and b_s^- specifying the maximum amount of traffic that the terminal node may send and receive, respectively. Let $S = \{(s, t) : s, t \in T, s \neq t\}$ be the set of all ordered pairs of distinct terminal nodes and d_{st} be a nonnegative demand assigned to a terminal pair (s, t).

- Let $d = \{d_{st} \geq 0 : \forall (s, t) \in S\}$ be a *traffic assignment* over the underlying network. Such a traffic assignment is called valid (*e.g.*, [18]) if it respects hose thresholds, *i.e.*,

$$\sum_{(s,t)\in S} d_{st} \leq b_s^+, \sum_{(t,s)\in S} d_{ts} \leq b_s^-, \quad \forall s \in T. \tag{1.1}$$

- For every $(s, t) \in S$, the traffic d_{st} is allowed to be routed only on a simple path from s to t, say P_{st}. The set of such paths, $P = \{P_{st} : \forall (s, t) \in S\}$, is called a *path assignment* over the network. When multiple path routing is allowed such set is called *multiple path assignment*. In this paper we consider the problem under single path routing restriction. The problem in the presence of multiple path routing is polynomially solvable [24].
- Let $x = \{x_{ij}\}_{\{i,j\}\in E}$ be a *capacity reservation* over the underlying network. Having x installed, we say a traffic assignment d could be *feasibly* routed along P, if the amount of capacity needed on an edge $\{i, j\}$ after sending traffic demands (each along its corresponding path) does not exceed x_{ij}.

Given as input the underlying network G, the terminal set T and the capacity reservation cost vector c, the task in the VPN design problem (*e.g.*, [11]) is to find a capacity reservation x and a path assignment P such that all possible valid traffic assignments can be feasibly routed along P and the total cost, given by $\sum_{\{i,j\}\in E} c_{ij}x_{ij}$, is minimized.

The asymmetric VPN design problem is NP-hard (*e.g.*, [18]). We will omit the world asymmetric whenever it causes no ambiguity. Most of the research on the problem has been dedicated to design approximation algorithms among which we mention [17]. As noted in [23], very little attempt has been made to tackle the problem with exact mathematical programming approaches. The only computational attempt to solve the VPN design problem is carried out in [1], where a compact mixed-integer linear programming (MIP) model is developed and claimed, through computational experiments, to be practically tight. This perception had left moderate interest in the research community for exact mathematical programming approaches to VPN until a challenging family of benchmark instances with large integrality gap and computationally hard for current mixed-integer programming solvers has been proposed in [23].

Paper Contribution. In the present study, we are interested in rank-1 cutting planes, *i.e.*, cuts that can be derived applying a separation procedure to a single constraint, often called *base inequality*, obtained by combining only the constraints in the original formulation (*e.g.*, [6] for formal definitions and details). More precisely, we are interested in the aggressive application of rank-1 cuts so as to close a large fraction of integrality gap as shown in [3, 4, 8, 14]. In particular, we demonstrate that tight solutions could be found for the VPN design problem only by optimizing over the first Chvátal closure [6, 15]. More precisely, we first study the so-called Chvátal–Gomory (CG) separation problem formulated in [4, 14] as a mixed-integer linear programming problem. Having the existing compact formulation at hand, we then draw out an important property of a valid rank-1 Chvátal–Gomory cut for the VPN design problem. As a result of this property we show that the associated CG separation problem of the existing compact formulation can only produce redundant valid rank-1 inequalities. Finally, we tighten the compact formulation and reduce it to a binary formulation. On that new formulation CG separation is instead very effective and we perform a computational investigation on the idea of adding rank-1 cuts more aggressively within a pure cutting plane method. The computational investigation clearly shows that the binary reduction plays an important role by allowing MIP solvers to effectively separate and use special cuts of the first closure, namely the zero-half cuts [5]. Even more, we show that when a huge number of rank-1 cuts are allowed to be added to the VPN binary formulation, it is enough to only add zero-half ones. As a result a cut-and-branch framework that aggressively adds rank-1 cuts at the root can successfully solve the challenging VPN instances to the extent of zero or small integrality gap in a reasonable amount of time.

Paper Organization. In Section 2 the existing compact formulation for the VPN design problem is reviewed and the CG separation problem for it is discussed. In Section 3 we show that the compact MIP formulation can be reduced to a pure binary programming problem. This reduction makes the use of CG cutting planes viable and leads to a generic cutting plane method for the VPN design problem. In Section 4 we discuss adding rank-1 cuts more aggressively and finally in Section 5 we computationally evaluate the strength of the first Chvátal closure for the VPN design problem.

2. Compact MIP formulation and CG separation

Following [1] the VPN design problem can be modeled as a mixed-integer linear programming problem by using directed multicommodity flows. Let us orient the undirected graph $G = (V, E)$ by defining two arcs (i, j) and (j, i) for each edge $\{i, j\} \in E$. Also, define the arc set $A = \{(i, j), (j, i) : \{i, j\} \in E\}$ and let $y = \{y_{ij}^{st} : \forall (s, t) \in S,\ \forall (i, j) \in A\}$ be a set of binary variables where

$$y_{ij}^{st} = \begin{cases} 1 & \text{if arc } (i,j) \in A \text{ is used to route demand } d_{st} \\ 0 & \text{otherwise.} \end{cases}$$

Note that any path assignment P is equivalent to a unique y satisfying the flow conservation constraints

$$\sum_{j:(i,j)\in A} y_{ij}^{st} - \sum_{j:(j,i)\in A} y_{ji}^{st} = \begin{cases} +1 & i = s \\ -1 & i = t \\ 0 & \text{otherwise.} \end{cases} \quad \forall i \in V, \forall (s,t) \in S \tag{2.1}$$

Imposing such constrains will ensure demand satisfaction by imposing a unit flow between each oriented pair of terminals. Once a path assignment has been selected, *i.e.*, values for the corresponding y vector have been set, the required capacity x_{ij} on an edge $\{i, j\}$ could be computed by means of the linear program

$$x_{ij} = \max \Bigg\{ \sum_{(s,t)\in S} d_{st} \left(y_{ij}^{st} + y_{ji}^{st}\right) :$$
$$\sum_{(s,t)\in S} d_{st} \le b_s^+, \sum_{(t,s)\in S} d_{ts} \le b_s^- \quad \forall s \in T \quad \text{and} \quad d_{st} \ge 0 \quad \forall (s,t) \in S \Bigg\}. \tag{2.2}$$

Then, the capacity x_{ij} installed on the edge $\{i, j\}$ is sufficient to carry the total traffic flowing through it. By strong duality this is equivalent to

$$x_{ij} = \min \sum_{s\in T} \left(b_s^+ \omega_{ij}^{s+} + b_s^- \omega_{ij}^{s-}\right) \tag{2.3}$$

$$y_{ij}^{st} + y_{ji}^{st} - \omega_{ij}^{s+} - \omega_{ij}^{t-} \le 0 \quad \forall (s,t) \in S, \tag{2.4}$$

where $\omega_{ij}^{s+}, \omega_{ij}^{s-}$ are the dual variables associated with constraints (1.1). Putting all together, we have the following compact formulation for the VPN design problem. (The reader is referred to [1] for more details.)

$$\text{(VPN)} \min \sum_{\{i,j\}\in E} c_{ij} \sum_{s\in T} \left(b_s^+ \omega_{ij}^{s+} + b_s^- \omega_{ij}^{s-}\right) \tag{2.5}$$

$$\sum_{(i,j)\in A} y_{ij}^{st} - \sum_{(j,i)\in A} y_{ji}^{st} = \begin{cases} +1 & i = s \\ -1 & i = t \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in V, \forall (s,t) \in S. \tag{2.6}$$

$$y_{ij}^{st} + y_{ji}^{st} - \omega_{ij}^{s+} - \omega_{ij}^{t-} \le 0 \quad \forall \{i,j\} \in E,\ \forall (s,t) \in S \tag{2.7}$$

$$y_{ij}^{st} \in \{0, 1\} \quad \forall (i,j) \in A,\ \forall (s,t) \in S \tag{2.8}$$

$$\omega_{ij}^{s+}, \omega_{ij}^{s-} \ge 0 \quad \forall \{i,j\} \in E,\ \forall s \in T. \tag{2.9}$$

Some effort (with limited success) to derive problem-specific inequalities to improve the linear programming (LP) relaxation of the above model has been attempted in [1]. In this situation where problem specific investigations have failed, general mixed-integer programming techniques might still be helpful. One such general and effective ideas, as discussed in [14], suggests optimizing a problem over the first Chvátal closure. The closure could be obtained by adding all rank-1 Chvátal–Gomory cuts [6, 15] to the LP relaxation of the problem. This approach was then extended to a mixed-integer program by projecting onto the space of integer variables and then deriving CG cuts for the projected problem [4]. This way the generated cuts are called *projected* Chvátal–Gomory (pro-CG) cuts. Experiments reported in [14] for pure integer linear programming problems (IPs) and in [4] for MIPs show that the first Chvátal closure often gives a surprisingly tight approximation of the (mixed-)integer hull. Because the above formulation of VPN is a MIP, we then concentrate on pro-CG cuts but, where no ambiguity exists, we will use both CG and pro-CG indifferently.

Optimizing over the first Chvátal closure is equivalent to a separation problem called CG-separation problem [14]. Given a point $(\dot{y}, \dot{\omega})$, the separation problem asks for a CG cut that is (maximally) violated by $(\dot{y}, \dot{\omega})$ or a certificate that the point is in the closure. In our case, separating any point in the LP relaxation is enough since the LP relaxation is compact and any other point could be separated by simple enumeration. In the following we will study the possibility of finding effective rank-1 CG cuts by solving the CG-separation problem.

Let $\mathscr{P}$ be the feasible solution space of the above (VPN) MIP formulation. Also let $\mathscr{P}'$ be the polytope obtained by projecting $\mathscr{P}$ onto the space of integer variables. Then, any inequality valid for $\mathscr{P}'$ is also valid for $\mathscr{P}$. Following the general framework in [4, 14], one can add rank-1 CG cuts for the polytope $\mathscr{P}'$ by solving the CG-separation problem (modeled) as a mixed-integer linear program. It will be shown in the following that useful rank-1 inequalities for $\mathscr{P}'$ must use ω variables unless they will be redundant. As a result, pro-CG cuts that have null coefficients on the continuous ω variables are redundant for the classical MIP formulation of VPN. To show that, first observe the following.

Lemma 2.1. *$\mathscr{P}'$ is in an integral polyhedron.*

Proof. It suffices to show that constraints (2.7) do not have any impact on the projection of $\mathscr{P}$ onto the space of y variables. In fact, if y is taken as to satisfy (2.6) and (2.8), then one can always choose ω as to satisfy (2.7) and (2.9) by just taking ω big enough. This implies that the projection of $\mathscr{P}$ onto the y-space coincides with (2.6) and (2.8). On the other hand, constraints from (2.6) defines a totally unimodular system (it is simply a flow polytope). Then, the polytope defined by (2.6) and nonnegativity is integral. □

As an important consequence of the above discussion, we have the following fundamental property of pro-CG cuts, which shows that they are useless for VPN.

Corollary 2.2. *A valid rank-1 CG-cut for the VPN design problem that only uses flow variables is redundant.*

The discussion in this section clearly shows that the classical (compact) MIP formulation for the VPN design problem does not benefit from pro-CG cuts, *i.e.*, the pro-CG closure does not close any integrality gap. In the next section we prove that the (VPN) MIP formulation can be reduced to a pure IP one, thus being in a position of reconsidering CG cuts. We further reduce the IP formulation to a binary programming problem, which largely benefits from rank-1 cuts, and especially zero-half cuts [5].

3. Binary formulation

In this section we study the (VPN) model and prove that it could be reduced to a pure IP. More precisely, by substituting (2.9) with

$$\omega_{ij}^{s+}, \omega_{ij}^{s-} \in \{0,1\}, \quad \forall \{i,j\} \in E,\ \forall s \in T \tag{3.1}$$

we will still contain any feasible solution. In the following we first prove that the (VPN) model could be reduced to a pure *integer* program and then we show that ω variables in this IP model could be further restricted to be in $\{0,1\}$.

Let us start with an insightful and intuitive discussion. A feasible solution $(\dot{y}, \dot{\omega})$ to the (asymmetric) VPN design problem is a set of simple paths each of which corresponds exactly to an existing terminal pair. Now consider an edge $\{i,j\}$ selected by the solution, *i.e.*, the capacity installed on the edge is strictly greater than zero. Since the solution is feasible, the minimum amount of capacity installed on it will be given by

$$(\textsc{Capacity}) \max \left\{ \sum_{(s,t)\in S} d_{st} \left\{ \dot{y}_{ij}^{st} + \dot{y}_{ji}^{st} \right\} : (1.1) \ \text{ and } \ \forall (s,t) \in S \ d_{st} \geq 0 \right\} \tag{3.2}$$

with which we maximize the amount of capacity needed on the edge when a valid traffic is being routed. Note that the model (Capacity) above is written for a fixed edge $\{i,j\} \in E$. The above model is a transportation problem and indeed has totally unimodular coefficient matrix. Let h_{ij} be the cost vector of the (Capacity) model, *i.e.*, $h_{ij}^{st} = \dot{y}_{ij}^{st} + \dot{y}_{ji}^{st}$, $\forall (s,t) \in S$, which is an integer vector. The LP given by the (Capacity) model is bounded and then from linear programming theory, optimal dual multipliers of the LP could be computed as

$$\omega_{ij} = (h)_B B^{-1}, \tag{3.3}$$

where B is an optimal basis of the LP. By unimodularity (of Capacity), B^{-1} is an integer matrix and so does the optimal multipliers vector. Furthermore, we know that those multipliers are (by definition of dual) always nonnegative. Now by setting the value of $\dot{\omega} := (\omega_{ij})$ the new solution is still feasible in (VPN) and costs no more than the original solution.

By means of the above discussion, (VPN) model reduces to a pure integer program as the capacity needed by the path assignment (given by the vector of y variables) could be computed only using an integer vector of ω variables. This observation is formally demonstrated below.

Lemma 3.1. *The* (VPN) *MIP formulation could be reduced to a pure integer formulation.*

Proof. For a fixed integral $\dot{y}$, the system in the ω variables is totally unimodular as it could be simply observed by using the classical Ghouila-Houri technique [19], where one puts all ω^+ variables in one set, and all ω^- in the other. Totally unimodularity of the system in turn shows that for the given integral $\dot{y}$, the optimum could always be achieved at an integral $\dot{\omega}$. $\square$

Now, we could state the main result.

Theorem 3.2. *The model (*VPN*) could be reduced to a* $0-1$ *integer program where* $\omega \in \{0,1\}$*.*

Proof. Let the IP version of (VPN) model be denoted by (VPN-I). Consider a feasible solution $(\dot{y}, \dot{\omega})$ to the (VPN-I) formulation, then $\dot{\omega}$ is an integer vector and it remains only to prove that $\dot{\omega} \in \{0,1\}$. We observe that the vector y represents a path assignment containing only simple paths. Thus,

$$\dot{y}_{ij}^{st} + \dot{y}_{ji}^{st} \in \{0,1\}, \quad \forall \{i,j\} \in E,\ \forall (s,t) \in S \tag{3.4}$$

and, because of (2.7), any entry of $\dot{\omega}$ with value more than 1 could be decreased to 1 without destroying feasibility. This operation does not also increase the cost as the hose thresholds are assumed to be nonnegative. $\square$

As a simple consequence, integrality of ω variables (as we proved through Lem. 3.1 and Thm. 3.2) together with (2.3) simply shows that when the hose thresholds are integer, the optimal solution reserves integer capacities on any edge. This proves parts of the observations in [1].

Corollary 3.3. *When hose thresholds are integer, the VPN design problem always admits an integer optimal capacity reservation.*

The binary formulation of the VPN problem is stronger than the previous MIP formulation because the domain of ω variables is reduced from $[0, \infty]$ to $\{0,1\}$. Then, the linear programming relaxation of the binary formulation is (strictly) included in the linear relaxation of the original MIP formulation. However, note that any solution of the LP relaxation of the MIP formulation corresponds to a multiple path assignment. In the presence of multiple path routing, the right-hand-side of (3.4) (the term $\dot{y}_{ij}^{st} + \dot{y}_{ji}^{st}$) still remains less than or equal to 1. Now, by using (2.7) one can simply reduce a solution of the LP relaxation of the MIP formulation to a solution of the LP relaxation of the binary formulation at no more cost. Therefore, the LP relaxation of both MIP and BP formulations have the same optimal value.

We note that theoretical results in this paper are heavily depended to unimodularity property inside the VPN design problem. To the best of our knowledge, this is the first time that the role of totally unimodularity for the VPN design problem is addressed. This naturally calls for further attempts to specially concentrate on this property and its consequences for the VPN design problem.

Let the binary formulation of the VPN problem be denoted by (VPN-B). Here, given a point $(\dot{y}^t, \dot{\omega}^t)$ to be separated, feasible space of the CG-separation MIP (see [14]) associated with the (VPN-B) model will contain valid inequalities like $\alpha\dot{y}+\beta\dot{\omega} \leq \lambda$ that could have nonzero β coefficients. Then, for the (VPN-B) formulation, solving the CG-separation MIP could generate valid inequalities and tighten the LP relaxation of the formulation. This fact will be immediately translated into the generic pure cutting plane algorithm given below.

Algorithm 1. The algorithm (iteratively) adds rank-1 valid CG-cuts to the initial LP relaxation of the (VPN-B) problem in order to eliminate non-integral optimal solutions of the LP relaxation. More precisely, after solving the underlying LP relaxation, if the optimal is integral, then an optimal solution was found and we are done. Otherwise, the CG separation problem is solved to find an additional inequality (defining a hyperplane) that could cut the non-integral optimal solution off and tighten the relaxation. Then, the inequality is added to the LP relaxation and the procedure is continued. In case the CG-separation problem could not find such an inequality, the algorithm terminates without finding an integral solution for the (VPN-B) problem and outputs an improved formulation. The improved formulation can be the input to a branch-and-bound algorithm. The stopping criterion in this algorithm is set to an overall time limit.

Exactly solving the separation subproblem in Algorithm 1 would ensure generating more violated cuts, potentially leading to tighter bounds in a smaller number of iterations. However, as the subproblem needs to solve a MIP each time, the algorithm is computationally very expensive in practice. To reduce separation overhead in Algorithm 1, one could limit the MIP in each subproblem to be solved to a pre-specified number of branching node explored. This criterion could even be dynamically updated. Following [14], the MIP solving process in a separation problem can be stopped when a pre-specified number of branching node has been explored after the last update of an incumbent solution. Note that any MIP feasible solution corresponds to a violated cutting plane.

4. More aggressive rank-1 cut separation

As we discussed above, separating CG cuts through a MIP is very time consuming. Accordingly, Algorithm 1 based on CG cuts will not provide a strong tool to computationally investigate how tight the first closure is for the VPN design problem instances. Fortunately, there are good heuristic algorithms in the literature for generating general rank-1 as well as zero-half cuts. Two of those heuristics are used in our computational evaluation in Section 5 and are described in the following.

Rank-1 GMI heuristic [7]

A heuristic to generate general rank-1 Gomory mixed-integer cuts (GMIs, [16]) was recently designed in [7]. The heuristic has the ability to quickly generate a large number of rank-1 GMI cuts from many tableaus. More precisely, in a separation round the LP relaxation of the current formulation (likely augmented by rank-1 GMI cuts) is solved to obtain an optimal solution as well as an optimal LP basis. This optimal basis is most likely not a basis for the LP relaxation of the initial formulation. Then, the heuristic looks for a basis of the initial LP that is "close enough" to the current optimal basis. It is always possible to find such a basis of the initial LP from the current optimal basis by removing all the columns that are nonbasic with respect to the current optimal solution and restricting the new basis of the initial LP relaxation to be a submatrix of current optimal basis. (Of course, the more modifications/cancelations are done, the more the new basis is "far" from the starting one.) Then, rank-1 cuts associated with the new basis of the initial LP are generated in the hope of cutting the current optimal solution.

Rank-1 zero-half heuristic [2]

The second heuristic we employ was originally described in [2] and it is designed to separate zero-half cuts for pure IPs. The separation heuristic is based on the property that, for any IP, zero-half cuts can be separated in polynomial time if any row of the IP has at most two odd coefficients. For an IP with this property separating zero-half cuts is equivalent to finding a special Eulerian cycle on a large multigraph, called *separation graph*, in which nodes (resp. edges) correspond to columns (resp. rows) of the given IP. More precisely, given a point $\dot{x}$ to be separated, the heuristic first builds a relaxation of the given IP to get the property of having at most two odd coefficient on each row. The relaxation together with the given point $\dot{x}$ is then used to build the separation graph. Finally, the heuristic finds proper Eulerian cycles on the separation graph and transforms them into zero-half cuts. Generated cuts are then added to the original IP and the process iterated.

In general, the separation graph could have any structure and in the heuristic in [2] it is treated as a complete graph. Then, for middle to large VPN instances, the heuristic needs a huge amount of memory while building the separation graph in each round. However, in the VPN binary program a variable appears on at most $|T|$ rows. This fact is used in the computation of the next section to effectively bound the degree of the corresponding node in the separation graph and therefore reduce the memory needed in each separation round. Moreover, in the original implementation of the heuristic finding proper Eulerian cycles is not diversified. Precisely, the heuristic simply considers nodes of the separation graph (*i.e.*, IP variables) in the order of their index and tries to find as many proper Eulerian cycles, containing the node, as possible. Then, as soon as "enough" Eulerian cycle (*i.e.*, enough cuts) are found, the algorithm stops the search and outputs the cuts. We added a simple mechanism that limits the number of cycles found that contain a special node. This mechanism diversifies the cycles (*i.e.*, the cuts) generated and has a nice computational effect on the dual bound improvement.

5. Computational study

In this section, the goal is to investigate how beneficial is, in practice, to optimize over the (first) Chvátal closure associated with the problem. Here, we perform extensive experiments and computationally demonstrate that the CG closure of the binary formulation is effective and that the most effective way to solve the problem is by adding special cuts of the Chvátal closure, namely (non-necessarily rank-1) zero-half cuts.

Test problems considered in this section are the challenging instances introduced in [23]. We performed all experiments on a single core of an Intel Core i5 with 2.53 GHz processor, under Linux with 4 GB of RAM. We coded cut-and-branch algorithms in C++, and Cplex 12.5 has been used for directly solving mixed-integer or binary formulations as well as solving LP relaxations. We also set a time limit of 10 000 s for solving each benchmark instances. For all instances that cannot be solved to optimality within the time limit, we report the gap between the best known upper bound and the lower bound obtained. In this section, we use the term MIP (BP) instances to refer to the VPN problem instances modeled by the Mixed integer (Binary) formulation.

In the computational experiments we shed light on two main issues.

- First, we consider the effect of the reformulation of VPN problem as a binary IP with respect to the classical MIP formulation. This is done by both running Cplex as a black-box solver and by giving special emphasis to the separation of rank-1 cuts for both formulations by measuring the percentage gap closed. This experiment is discussed in Section 5.1.
- Second, we compare the impact of the two heuristics discussed in the previous section so as to state how much zero-half cuts alone are enough to close the gap within a cutting plane approach with respect to the larger and stronger family of GMI cuts. This experiment is discussed in Section 5.2.

We report computational experiments of the following various methods.

(A) MIP: The MIP formulation, (VPN), is solved with Cplex as a black-box solver.

(B) MIP Agg. GMI: The MIP formulation is solved with Cplex, where all built-in cutting planes are turned off except GMIs, which are separated aggressively as they are the only cutting planes appearing to be useful in experiments with (A).

(C) MIP Agg. ZH: The MIP formulation is solved with Cplex, where all built-in cutting planes are turned off except zero-half (ZH) cuts, which are separated aggressively.

(D) BP: The BP formulation, (VPN-B), is solved with Cplex as a black-box solver.

(E) BP ZH: The BP formulation is solved with Cplex, where all built-in cutting planes are turned off except zero-half (ZH) cuts as they are the most effective cuts in the experiments with (D).

(F) BP Agg. ZH: The BP formulation is solved with Cplex, where all built-in cutting planes are turned off except zero-half cutting planes, which are separated aggressively.

(G) MIP Heur.1: The MIP formulation is solved with the heuristic in [7]. The heuristic is employed in a cut-and-branch framework, *i.e.*, the heuristic keeps iterating at the root node of the search tree until a specific condition is met. Then, (if necessary) branching starts on the tightened formulation.

(H) BP Heur.1: The BP formulation is solved with the heuristic in [7]. The heuristic is employed as in (G).

While using the rank-1 heuristic in [7] a dynamic termination condition is imposed. We forced the heuristic to stop cutting at the root after a specified number N_h of consecutive separation rounds with less than ϵ_h percent improvement in the dual bound. For the heuristic, we also let M_h denote the maximum number of cuts generated per separation round. We set $M_h = 1000$, $N_h = 5$ and $\epsilon_h = 0.005$ because preliminary testing and tuning have shown that the heuristic achieves its best performance on the VPN challenging instances under this settings. In the experiments reported for Heur.1, GMI cuts are generated at the root node and then, before branching, we ask Cplex to choose among the cuts. Our experiments show that in this way Cplex is always able to quickly reach almost the same dual bound by adding considerably less number of generated cuts. However, in case the cutting phase takes all the given time limit, we report dual bounds obtained by adding all the violated cuts. Here, the number of cuts reported is that of all the violated ones found by the heuristic in [7].

Finally, the zero-half heuristic in [2] is not used in MIP versus BP comparison as it is intended to only work with pure IPs. We employ the heuristic, denoted by Heur.2, to perform a comparison between general rank-1 CG cuts and zero-half cuts over the BP instances and to prove our claim on competitiveness of zero-half cuts in Section 5.2.

5.1. MIP versus binary formulations

Tables 1 and 2 summarize the results of running the methods discussed above at the root node, so as to test the impact of cutting planes on both MIP and BP formulations. For each of the methods (A)–(H) the following information is reported.

- gap_i: Initial percentage gap that is the gap between the LP solution value (LP) and the best known integer solution value (BN) of an instance computed as $(BN - LB)/BN \times 100$.
- gap_r: Percentage gap at the root node. It represents the gap between the values of the best known integer solution and the linear relaxation strengthened by cuts.
- *# cuts*: The number of cuts applied. We also briefly provide types of the cuts as "GMI" and "ZH". Whenever other kinds of cuts appear we report them as "Other".
- t_r: Time spent on the root.

According to the results in Tables 1 and 2, we note that reducing the original compact formulation to a binary program significantly enhances effectiveness of a black-box solver. For all the ring instances, this effect can be seen by comparing column $\%gap_r$ obtained by methods (A) and (B) with the same results reported in (D). Comparing the gaps obtained by methods (D) and the gaps obtained by (E) and (F) also shows that the reason behind superiority of the binary formulation is only due to generating a special kind of rank-1 cuts, *i.e.*, zero-half cuts. Gaps obtained by methods (G) and (H) also show that adding general rank-1 cuts is not much affected by the binary reduction.

Table 1. Results on the root for MIP instances.

			(A) MIP		(B) MIP: Agg. GMI			(C) MIP: Agg. ZH			(G) MIP: Heur.1		
Name	$\%gap_i$	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r
			(GMI, Other)			GMI			ZH			Rank1	
ring-20-15	47.32	34.89	(58, 0)	2.15	17.05	90	4.40	47.32	0	1.56	1.08	3021	161.08
ring-20-20	61.52	53.85	(22, 0)	6.95	37.14	48	19.36	61.52	0	4.16	9.29	2928	699.79
ring-50-10	59.90	54.52	(42, 0)	3.16	42.25	70	5.47	59.90	0	1.9	14.38	7749	147.00
ring-50-15	49.69	47.50	(31, 0)	10.83	40.56	71	39.42	49.69	0	10.01	8.86	5070	1078.49
ring-50-20	35.36	32.48	(79, 0)	32.56	27.60	97	76.32	35.36	0	39.05	8.96	5966	2375.92
ring-50-25	29.63	27.11	(79, 0)	84.65	24.37	112	293.86	29.63	0	60.02	4.48	3395	7838.73
ring-50-30	25.88	23.71	(96, 0)	192.77	19.63	104	434.78	25.88	0	236.78	4.01	14608	Time Limit
ring-80-10	24.81	23.64	(18, 0)	3.88	18.39	46	9.33	24.81	0	4.47	3.08	7222	340.48
ring-80-15	49.74	46.63	(40, 0)	23.19	39.64	85	40.26	49.74	0	28.96	11.25	7467	2807.80
ring-80-20	49.59	45.82	(31, 0)	97.81	43.01	79	127.60	49.59	0	73.16	6.59	5458	6348.31
ring-80-25	32.26	31.29	(92, 0)	217.62	28.55	115	636.08	32.26	0	179.72	8.09	17500	Time Limit
ring-80-30	28.42	27.13	(67, 0)	491.00	23.42	111	877.51	28.42	0	554.46	8.19	12816	Time Limit
ring-100-7	49.83	47.83	(10, 0)	1.71	39.25	44	2.93	49.83	0	1.04	12.20	19226	163.06
ring-100-10	40.86	38.11	(33, 0)	6.36	31.55	49	11.54	40.86	0	4.7	8.16	14384	915.29
ring-100-14	27.34	26.30	(16, 0)	23.26	22.86	54	70.31	27.34	0	17.62	4.40	8232	2448.54
ring-100-20	61.62	60.03	(52, 0)	216.84	55.28	156	409.98	61.62	0	183.98	27.32	25691	Time Limit
ring-100-25	60.28	58.70	(78, 0)	459.33	55.59	97	1440.88	60.28	0	421.19	33.91	14954	Time Limit
ring-100-30	50.38	49.11	(21, 0)	751.18	47.17	84	2403.20	50.38	0	558.08	47.58	1998	Time Limit
Average	43.58	40.48	(50.75, 0)	144.02	34.07	86.12	383.05	43.58	0	132.27	12.32	9871.39	4740.25

TABLE 2. Results on the root for BP instances.

			(D) BP			(E) BP: ZH			(F) BP: Agg. ZH			(H) BP: Heur.1	
Name	$\%gap_i$	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r	$\%gap_r$	$\#cuts$	t_r
			(ZH, Other)			ZH			ZH			Rank1	
ring-20-15	47.32	0.00	(562, 23)	23.08	0.00	473	18.91	0.00	473	16.19	1.86	3059	167.78
ring-20-20	61.52	4.17	(1555, 33)	172.23	4.31	756	180.46	4.31	2510	164.84	10.51	2975	690.67
ring-50-10	59.90	4.91	(991, 3)	152.18	4.02	636	131.64	4.02	1219	113.28	14.15	7749	366.80
ring-50-15	49.69	2.09	(1220, 17)	547.04	2.23	1000	580.55	2.83	1615	490.46	11.41	5271	964.00
ring-50-20	35.36	0.69	(1015, 34)	610.53	0.00	1228	613.10	0.00	1189	401.89	5.27	4086	2973.47
ring-50-25	29.63	0.00	(2005, 43)	797.27	0.00	2330	726.43	0.00	2330	750.70	3.49	3585	5533.70
ring-50-30	25.88	0.00	(3291, 63)	1212.00	0.00	2938	1127.49	0.00	2938	1114.03	4.56	4353	9383.32
ring-80-10	24.81	0.00	(1044, 13)	67.33	0.00	1231	73.48	0.00	1231	40.39	4.41	11560	568.76
ring-80-15	49.74	0.00	(1520, 19)	1526.59	0.00	1534	1733.79	0.00	1678	1245.99	11.22	8796	3097.23
ring-80-20	49.59	0.00	(3050, 56)	2815.62	0.00	2523	2557.76	0.00	2249	2151.88	6.93	15837	Time Limit
ring-80-25	32.26	0.00	(2465, 30)	4234.54	0.00	2120	3437.78	0.00	3363	3165.45	12.63	6218	Time Limit
ring-80-30	28.42	0.09	(4333, 90)	5097.17	0.13	2992	6754.11	0.00	3788	5392.32	11.89	7926	Time Limit
ring-100-7	49.83	1.69	(923, 1)	150.32	0.00	1032	149.64	0.25	864	88.21	12.89	22149	585.94
ring-100-10	40.86	0.65	(1323, 4)	466.60	0.00	1297	394.73	0.00	1306	228.25	10.78	18773	1092.54
ring-100-14	27.34	0.00	(3121, 12)	343.21	0.00	3446	310.00	0.00	3446	281.03	5.12	9668	2818.87
ring-100-20	61.62	6.49	(4362, 38)	Time Limit	6.16	5580	Time Limit	6.14	5243	Time Limit	31.43	9415	8749.57
ring-100-25	60.28	18.68	(2504, 32)	Time Limit	18.10	2464	Time Limit	18.10	2464	Time Limit	40.56	5534	9172.74
ring-100-30	50.38	18.77	(6852, 42)	Time Limit	18.60	6547	Time Limit	18.60	6547	Time Limit	33.48	8987	Time Limit
Average	43.58	3.23	(2340.89, 30.72)	2678.65	2.97	2229.28	2710.55	3.01	2469.61	2535.83	12.92	8663.39	4786.97

Comparing gap obtained by (G) and (H) with the gap reached by (D) and (E) clearly points out the effectiveness of zero-half cuts. However, note that since zero-half cuts generated by Cplex are not necessarily of rank-1, we need to externally examine the effect of rank-1 zero-half cuts and compare them with general rank-1 CG cuts. This motivates the experiments reported in Section 5.2.

In terms of the number of cuts applied the binary program allows a black-box solver to apply much more (effective) cuts being able to close most of the integrality gap only at the root. Comparing the number of cuts applied by the methods (E) and (F) with the same numbers obtained by (D) and (G) one could observe that limiting to only generate zero-half cuts carefully guides the cutting phase to produce more effective cuts at no more computational costs. Here, almost the same amount of gap (on average) could be closed with less (but more effective) zero-half cuts. An instance by instance comparison of the results obtained by (D) and (E) also emphasizes this fact. Furthermore, aggressive use of zero-half cuts, as in (F), improves time spent on the root in average. We also mention that Cplex does not separate zero-half cuts in the MIP case as it could be seen from results obtained by (A), (B) and (C).

In terms of the time spent on the root, methods (A) and (B) spend much less time at the root because in the MIP case none of the separation procedures seems to be effective. On the MIP instances, a black-box solver tends to leave almost all the computational burden to branching phase resulting in a weak performance. This fact is exaggerated in (C) as it will do nothing on the root and only performs pure branching. In BP case, on the other hand, extensive but careless use of rank-1 cuts might be partly effective but not efficient. Here, using zero-half cuts at the root clearly addresses the trade-off as revealed by results of the methods (D), (E) and (F).

The results on running the full cut-and-branch approach are summarized in Tables 3 and 4. That is, we allow an algorithm to continue solving the tightened MIP/BP instances by performing standard branch and bound. For each of the methods the following information is displayed.

- gap_f: Percentage gap after time limit. It represents the gap between the integer optimal solution and the final dual bound.
- n_{BB}: The number of branch and bound nodes explored.
- t_f: Total time spent.

Note that, appearance of the character "M" in Table 4 means a memory limit happened while branching (see Tab. 3). Then, the corresponding number reported there is the time spent by the method before the memory limit is reached.

Similarly to the results at the root, we observe that methods (E) and (F) also outperform other methods within the time limit both in terms of final gap reached and the total time spent as it is clear from comparing results in Tables 3 and in 4.

In terms of number of branch-and-bound nodes explored, comparing results over MIP instances and results over BP instances one can observe that the less cutting performed at the root, the more branch-and-bound nodes needed to be explored. On the other hand, aggressive cutting on the root could have side effects

TABLE 3. Results after time limit for MIP instances.

		(A) MIP			(B) MIP: Agg. GF			(C) MIP: Agg. ZH (pure branching)			(G) MIP: Heur.1		
Name	$\%gap_i$	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f
ring-20-15	47.32	0.00	1897	62.90	0.00	3237	94.29	0.00	13699	359.96	0.00	6	164.79
ring-20-20	61.52	10.86	71662	2937.56(M)	7.70	62441	2217.71 (M)	23.28	98488	4203.32(M)	0.00	9769	6683.90
ring-50-10	59.90	0.00	26803	443.48	0.00	40760	1019.93	0.00	26816	626.22	0.00	3395	597.96
ring-50-15	49.69	0.00	86124	9129.21	0.00	32942	3580.60	0.00	30079	4014.85	0.00	729	2573.10
ring-50-20	35.36	6.23	11658	Time Limit	0.00	5483	5215.49	3.69	14359	Time Limit	0.00	876	6487.41
ring-50-25	29.63	9.33	7181	Time Limit	8.35	3435	Time Limit	14.96	17225	Time Limit	4.20	111	Time Limit
ring-50-30	25.88	14.03	1073	Time Limit	10.49	2138	Time Limit	19.49	1709	Time Limit	4.01	1	Time Limit
ring-80-10	24.81	0.00	330	22.80	0.00	480	31.64	0.00	170	18.71	0.00	22	353.83
ring-80-15	49.74	2.07	30061	Time Limit	4.78	35459	Time Limit	20.37	48959	6995.77(M)	3.89	2676	Time Limit
ring-80-20	49.59	23.94	25025	7082.69(M)	10.14	12384	Time Limit	12.34	12394	Time Limit	6.18	428	Time Limit
ring-80-25	32.26	20.05	2798	Time Limit	17.88	1370	Time Limit	23.08	7710	Time Limit	8.09	1	Time Limit
ring-80-30	28.42	19.03	979	Time Limit	16.36	981	Time Limit	16.48	1154	Time Limit	8.19	1	Time Limit
ring-100-7	49.83	0.00	630	11.94	0.00	847	18.70	0.00	1106	17.76	0.00	190	225.62
ring-100-10	40.86	0.00	20093	1241.53	0.00	1373	101.90	0.00	6828	567.16	0.00	720	1560.72
ring-100-14	27.34	0.00	1000	511.25	0.00	1230	948.96	0.00	1052	681.11	0.00	392	4634.15
ring-100-20	61.62	49.15	8890	Time Limit	44.25	10107	Time Limit	51.94	15798	Time Limit	27.32	1	Time Limit
ring-100-25	60.28	52.69	1850	Time Limit	47.96	1122	Time Limit	55.89	4088	Time Limit	33.91	1	Time Limit
ring-100-30	50.38	44.63	497	Time Limit	42.58	1304	Time Limit	45.59	570	Time Limit	47.58	1	Time Limit
Average	43.58	14.00	16586.17	6191.30	11.69	12060.72	5734.96	15.95	16789.11	5971.38	7.97	1073.33	6293.41

TABLE 4. Results after time limit for BP instances.

		(D) BP			(E) BP: ZH			(F) BP: Agg. ZH			(H) BP: Heur.1		
Name	$\%gap_i$	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f	$\%gap_f$	n_{BB}	t_f
ring-20-15	47.32	0.00	1	23.08	0.00	1	18.91	0.00	1	16.19	0.00	30	187.44
ring-20-20	61.52	0.00	1491	1728.72	0.00	1720	1455.79	0.00	594	1090.61	1.06	10519	Time Limit
ring-50-10	59.90	0.00	209	370.10	0.00	122	247.70	0.00	69	202.52	0.00	1219	736.09
ring-50-15	49.69	0.00	35	905.23	0.00	46	943.36	0.00	16	719.62	0.00	1340	3220.94
ring-50-20	35.36	0.00	4	670.78	0.00	1	613.10	0.00	1	401.89	0.00	235	4688.10
ring-50-25	29.63	0.00	1	797.27	0.00	1	726.43	0.00	1	750.70	1.27	585	Time Limit
ring-50-30	25.88	0.00	1	1212.00	0.00	1	1127.49	0.00	1	1114.03	4.08	7	Time Limit
ring-80-10	24.81	0.00	1	67.33	0.00	1	73.48	0.00	1	40.39	0.00	249	920.46
ring-80-15	49.74	0.00	1	1526.59	0.00	1	1733.79	0.00	1	1245.99	2.86	1817	Time Limit
ring-80-20	49.59	0.00	1	2815.62	0.00	1	2557.76	0.00	1	2151.88	0.00	1	Time Limit
ring-80-25	32.26	0.00	1	4234.54	0.00	1	3437.78	0.00	1	3165.45	0.00	1	Time Limit
ring-80-30	28.42	0.00	2	5177.11	0.00	2	6798.46	0.00	1	5392.32	0.00	1	Time Limit
ring-100-7	49.83	0.00	10	165.35	0.00	1	149.64	0.00	2	90.11	0.00	323	808.92
ring-100-10	40.86	0.00	8	498.99	0.00	1	394.73	0.00	1	228.25	0.00	664	2083.69
ring-100-14	27.34	0.00	1	343.21	0.00	1	310.00	0.00	1	281.03	0.00	179	4011.45
ring-100-20	61.62	6.49	1	Time Limit	6.16	1	Time Limit	6.14	1	Time Limit	30.34	12	Time Limit
ring-100-25	60.28	18.68	1	Time Limit	18.10	1	Time Limit	18.10	1	Time Limit	40.12	5	Time Limit
ring-100-30	50.38	18.77	1	Time Limit	18.60	1	Time Limit	18.60	1	Time Limit	33.48	1	Time Limit
Average	43.58	2.44	98.33	2807.55	2.38	105.78	2810.47	2.38	38.61	2605.05	6.29	954.89	6480.95

as it usually makes branching subproblems more difficult. This could be seen by comparing average time needed per subproblem over the MIP instances with the same value over the BP instances.

5.2. General rank-1 GMIs *vs.* zero-half cuts

In this section we perform experiments to show the fact that when a huge number of rank-1 CG cuts are allowed to be added to the VPN Binary Program, it is enough to only add zero-half ones. Note that results obtained by (D), (E) and (F) in Table 2 could not clearly address this fact as zero-half cuts generated by Cplex are not necessarily of rank-1. To externally examine rank-1 zero-half cuts against general rank-1 CG cuts we run both Heur.1 and Heur.2 in the following two settings. Here no time limit/dynamic termination condition is imposed.

- set.1 We run both heuristics for a maximum number of 10 separation rounds with maximum number of 1000 cuts generated and added per separation round. The setting is considered as Heur.1 has its best performance with this number of cuts generated per round.
- set.2 We run both heuristics for a maximum number of 10 separation rounds with maximum number of 5000 cuts generated and added per separation round. The motivation behind this setting is that by adding a large number of cuts one could assure that good cuts of each type (general rank-1 or zero-half) are generated and added to the underlying IP.

Table 5 reports results obtained by Heur.1 and Heur.2 over the above two settings on the BP instances.

The effectiveness of zero-half cuts is clear by the results. An instance by instance comparison shows that, when enough of rank-1 CG cuts are allowed to be generated at the root node of VPN instances, almost all power of such cuts in tightening the LP relaxation and reducing the integrality gap can be obtained by adding zero-half cuts.

For large instances, adding zero-half cuts aggressively makes the underlying LP more and more difficult. In this case it is not needed to add all the generated zero-half cuts to the LP. For the VPN instances, some preliminary experiments we performed show that if we only add "sparse" cuts, as also suggested in [2], we would generally be able to reach good dual bound in smaller amount of time.

We also note that, results obtained by methods that only add zero-half cuts are strong enough to close most of the integrality gap on the root. This naturally suggests that additional investigation on the facial properties of zero-half cuts for VPN could lead to a much better understanding of the structure of the associated polyhedron as it already happened for several combinatorial optimization problems [10, 13, 20, 21, 25].

Acknowledgements. We are indebted to Matteo Fischetti and Sanjeed Dash for providing the computer codes implementing the heuristics in [2] and [7], respectively. The second author acknowledges the support of MIUR under the PRIN2012 grant. We are also indebted to two anonymous referees for very useful comments that led to an elegant simplification of some of the theoretical results in the paper.

TABLE 5. General Rank-1 CG *vs.* Zero-Half cuts.

		set.1: 1000 cuts per rounds				set.2: 5000 cuts per rounds			
		BP: Heur.1		BP: Heur.2		BP: Heur.1		BP: Heur.2	
Name	$\%gap_i$	$\%gap_r$	t_r	$\%gap_r$	t_r	$\%gap_r$	t_r	$\%gap_r$	t_r
ring-20-15	47.32	7.85	52	2.75	76	6.91	243	2.75	346
ring-20-20	61.52	18.74	153	13.19	147	16.02	463	9.91	1216
ring-50-10	59.90	28.46	42	9.51	166	30.22	238	7.03	693
ring-50-15	49.69	20.43	147	10.10	616	21.06	652	1.73	1738
ring-50-20	35.36	10.70	736	9.10	1308	9.32	1911	3.28	1517
ring-50-25	29.63	12.19	72	0.81	250	10.34	350	0.00	185
ring-50-30	25.88	27.44	419	24.60	937	28.56	815	4.79	2723
ring-80-10	24.81	26.64	1079	30.32	2046	23.41	1646	15.57	3354
ring-80-15	49.74	36.20	25	20.77	86	35.79	99	14.17	247
ring-80-20	49.59	26.32	134	15.71	285	26.03	172	0.00	1241
ring-80-25	32.26	13.69	418	11.44	937	13.18	796	0.00	1486
ring-80-30	28.42	38.32	1956	47.35	2868	39.69	3725	30.32	13148
ring-100-7	49.83	6.13	2040	16.88	2039	6.05	4116	2.23	2871
ring-100-10	40.86	4.92	4734	16.21	3323	5.38	14178	0.00	6077
ring-100-14	27.34	15.12	4374	23.62	4012	12.15	5445	8.44	13100
ring-100-20	61.62	11.90	10819	22.13	8065	9.61	12333	(M)	–
ring-100-25	60.28	40.36	7964	52.03	6255	37.29	5860	39.13	13455
ring-100-30	50.38	31.88	9276	44.17	12505	47.31	12932	(M)	–
Average	43.58	20.96	2468.88	20.59	2551.16	21.02	3665.36	8.71	3962.39

References

[1] A. Alın, E. Amaldi, P. Belotti and M.C. Pınar, Provisioning virtual private networks under traffic uncertainty. *Networks* **49** (2007) 100–115.

[2] G. Andreello, A. Caprara and M. Fischetti, Embedding cuts in a branch and cut framework: a computational study with $\{0, \frac{1}{2}\}$-cuts. *INFORMS J. Comput.* **19** (2007) 229–238.

[3] E. Balas and A. Saxena, Optimizing over the split closure. *Math. Program.* **113** (2008) 219–240.

[4] P. Bonami, G. Cornuéjols, S. Dash, M. Fischetti and A. Lodi, Projected Chvátal-Gomory cuts for mixed integer linear programs. *Math. Program.* **113** (2008) 241–257.

[5] A. Caprara and M. Fischetti, $\{0, \frac{1}{2}\}$-Chvátal–Gomory cuts. *Math. Program.* **74** (1996) 221–235.

[6] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems. *Disc. Math.* **4** (1973) 305–337.

[7] S. Dash and M. Goycoolea, A heuristic to generate rank-1 GMI cuts. *Math. Program. Comput.* **2** (2010) 231–257.

[8] S. Dash, O. Günlük and A. Lodi, MIR closures of polyhedral sets. *Math. Program.* **121** (2010) 33–60.

[9] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan and J.E. van der Merive, A flexible model for resource management in Virtual Private Networks, in *Proc. of ACM Special Interest Group on Data Communication* (SIGCOMM), Cambridge, MA (1999) 95–108.

[10] J. Edmonds and E.L. Johnson, Matching: a well-solved class of integer linear programs. In *Combinatorial Structures and Their Applications*, edited by R.K. Guy, H. Hanani, N. Sauer. Gordon and Breach, New York (1970) 80–92.

[11] F. Eisenbrand, F. Grandoni, G. Oriolo and M. Skutella, New approaches for virtual private network design. *SIAM J. Comput.* **37** (2007) 706–721.

[12] P. Ferguson and G. Huston, What Is a VPN? – Part I. *The Internet Protocol J.* **1** (1998).

[13] S. Fiorini, $\{0, \frac{1}{2}\}$-cuts and the linear ordering problem: Surfaces that define facets. *SIAM J. Disc. Math.* **20** (2006) 893–912.

[14] M. Fischetti and A. Lodi, Optimizing over the first Chvátal closure. *Math. Program.* **110** (2007) 3–20.

[15] R.E. Gomory, Outline of an algorithm for integer solutions to linear programs. *Bull. Amer. Math. Soc.* **64** (1958) 275–278.

[16] R.E. Gomory, An algorithm for integer solutions to linear programs. In *Recent Advances in Mathematical Programming*, edited by R.L. Graves, P. Wolfe. McGraw-Hill, New York (1963) 269–302.

[17] F. Grandoni, T. Rothvoß and L. Sanitá, From uncertainty to nonlinearity: Solving virtual private network via single-sink buy-at-bulk. *Math. Oper. Res.* **36** (2011) 185–204.

[18] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi and B. Yener, Provisioning a virtual private network: a network design problem for multicommodity flow, in *Proc. of the 33rd Annual ACM Symposium on Theory of Computing* (STOC) (2001) 389–398.

[19] B. Korte and J. Vygen, Combinatorial optimization: Theory and algorithms, 4th ed. Springer Publishing Company, Incorporated (2007).

[20] A.M.C.A. Koster and A. Zymolka, Stable multi-sets. *Math. Meth. Oper. Res.* **56** (2002) 45–65.

[21] A.M.C.A. Koster and A. Zymolka, On cycles and the stable multi-set polytope. *Discret. Optim.* **2** (2005) 241–255.

[22] A.G. Mason, Cisco secure virtual private network. *Cisco Press* **7** (2002).

[23] A. Moradi, A. Lodi and S.M. Hashemi, On the difficulty of virtual private network instances. *Networks* **63** (2014) 327–333.

[24] N.K. Olver, *Robust Network Design.* Ph.D. thesis, McGill University (2010).

[25] M. Padberg, On the facial structure of set packing polyhedra. *Math. Program.* **5** (1973) 199–215.

[26] M. Pioro and D. Medhi, *Routing, flow and capacity design in communication and computer networks.* San Francisco, CA, Morgan Kaufmann (2004).