TWO-CONNECTED NETWORKS WITH RINGS OF BOUNDED CARDINALITY

B. FORTZ
Institut d'Administration et de Gestion, Universite´ Catholique de Louvain, Place des Doyens 1, B-1348 Louvain-la-Neuve, Belgium

M. LABBE
Institut de Statistique et de Recherche Ope´rationnelle, SMG, CP 210/01, Univ. Libre de Bruxelles, Bddu Triomphe, B-1050 Bruxelles, Belgium

Abstract. We study the problem of designing at minimum cost a two-connected network such that each edge belongs to a cycle using at most K edges. This problem is a particular case of the two-connected networks with bounded meshes problem studied by Fortz, Labbé and Maffioli (Operations Research, vol. 48, no. 6, pp. 866–877, 2000).

In this paper, we compute a lower bound on the number of edges in a feasible solution, we show that the problem is strongly NP-complete for any fixed K, and we derive a new class of facet defining inequalities. Numerical results obtained with a branch-and-cut algorithm using these inequalities show their effectiveness for solving the problem.

Keywords: network design, combinatorial optimization, branch-and-cut

1. Introduction

In the last decade, telecommunication network planning has become an important problem area for developing and applying optimization models. Telephone companies have initiated extensive modeling and planning efforts to expand and upgrade their transmission facilities.

Recently, Fortz et al. [8] introduced a new model for the topological design of backbone telecommunication networks. The two-connected network with bounded rings (or meshes) problem (2CNBR) consists in designing a minimum cost network N with the following constraints:

1. N contains at least two node-disjoint paths between every pair of nodes (*2-connectivity constraints*), and
2. each edge of N must belong to at least one cycle whose length is bounded by a given constant K (*ring constraints*).

In the most common mathematical model for topological network design, a graph $G = (V, E)$ is considered where V is the set of nodes that have to be connected and E is the set of edges, that is the set of potential links between nodes. Each edge e of E has a fixed nonnegative cost ce and the objective is to find the subset F of E of minimum total cost, such that the resulting network $N = (V, F)$ satisfies some survivability requirement.

This requirement can be that the network be either *k -edge-connected* or *k-node-connected*, which means that the removal of any (k — 1) or fewer edges (respectively, nodes) leaves G connected.

In most cases, two-connected networks have been found to provide a sufficient level of survivability. Hence, a considerable amount of research has focused on so-called *low- connectivity constrained* network design problems, i.e., problems for which each node j is characterized by a requirement $rj \in \{0, 1, 2\}$ and $\min\{r_v, r_w\}$ node-disjoint paths between every pair of distinct nodes v, w are required. Work on this kind of problem goes from the early contributions of Steiglitz et al. [16] to the more recent articles of Grötschel and Monma [11], Boyd and Hao [3], Monma and Shallcross [14], Gro¨ tschel et al. [12, 13], and others. For in depth surveys in this area the reader is referred to Fortz [5] and Grötschel et al. [13].

The minimum-cost two-connected network is often a Hamiltonian cycle. Therefore, any edge failure would require to reroute the flow that passed through that edge, using all the edges of the network, an obviously undesirable feature. Fortz et al. [8] proposed to add ring constraints to limit the region of influence of the traffic which is necessary to reroute if a connection is broken. They implemented a first branch-and-cut algorithm to solve the problem as well as several constructive heuristics. More recently, Fortz and Labbé [7] studied the structure of the underlying polyhedra, deriving new classes of facet-defining inequalities.

An important application of ring constraints appears in topologies using the recent technology of *self-healing rings*. Self-healing rings are cycles in the network equipped in such away that any link failure in the ring is automatically detected and the traffic rerouted by the alternative path in the cycle. Due to technological constraints, the length of self-healing rings must be limited. This is equivalent to set a bound on the length of the shortest cycle including each edge. In practice, the length of the ring is computed as the number of *hops*, i.e., the number of nodes that compose the ring. This corresponds to the particular case of 2CNBR that arises when a unit length is given to each edge. Our model is only a first step in solving the self-healing ring network design problem, as we only ensure the presence of feasible rings in the network. The next step is dimensioning the rings, taking into account the demands and the additional cost for inter-ring transfer. A heuristic for the self-healing ring network design problem was proposed by Fortz et al. [9].

In this paper, we derive additional properties for this particular case. The next section introduces some notation and a mathematical formulation for our model. In Section 3, we compute a lower bound on the number of edges in any feasible solution. This bound is used in Section 4 to show that the problem is NP-complete even for K fixed, and is then extended to a new class of valid inequalities in Section 5. The separation problem for these inequalities is studied in Section 6, and a branch-and-cut algorithm is outlined in Section 8. Numerical results with this algorithm are presented in Section 9.

2. Notation and model

As mentioned before, we represent the given set of nodes and possible cable connections by an undirected graph $G = (V, E)$. We suppose that E does not contain parallel edges.
Throughout this paper, $n := |V|$ and $m := |E|$ will denote the number of nodes and edges of G.

Given the graph $G = (V, E)$ and $W \subset V$, the edge set

$$\delta(W) := \{ij \in E \mid i \in W, j \in V \setminus W\}$$

is called the *cut* induced by W. We write $\delta_G(W)$ to make clear—in case of possible ambiguities—with respect to which graph the cut induced by W is considered. The *degree* of a node v is the cardinality of $\delta(v)$. The set

$$E(W) := \{ij \in E \mid i \in W, j \in W\}$$

is the set of edges having both end nodes in W. We denote by $G(W) = (W, E(W))$ the subgraph induced by edges having both end nodes in W. If $E(W)$ is empty, W is an *independent set*. G/W is the graph obtained from G by contracting the nodes in W to a new node w (retaining parallel edges).

We denote by $V - z := V \setminus \{z\}$ and $E - e := E \setminus \{e\}$ the subsets obtained by removing one node or one edge from the set of nodes or edges, and $G - z$ denotes the graph $(V - z, E \setminus \delta(\{z\}))$, i.e., the graph obtained by removing a node z and its incident edges from G. This is extended to a subset $Z \subset V$ of nodes by the notation $G - Z := (V \setminus Z, E \setminus (\delta(Z) \cup E(Z)))$.

Each edge $e := ij \in E$, has a *fixed cost* c_e representing the cost of establishing the direct link connection. The cost of a network $N = (V, F)$ where $F \subseteq E$ is a subset of possible

edges is denoted by $c(F) := \sum_{e \in F} c_e$. The *distance* between two nodes i and j in this network is denoted by $d_F(i, j)$ and is given by the minimum number of edges in a path linking these two nodes in F.

A useful tool to analyze feasible solutions of 2CNBR is the *restriction of a graph to bounded rings*. Given a graph $G = (V, E)$ and a constant $K > 0$, we define for each subset of edges $F \subseteq E$ its restriction to bounded rings F_K as

$$F_K := \left\{ e \in F : \begin{array}{l} e \text{ belongs to at least one cycle} \\ \text{of length less than or equal to } K \text{ in } F \end{array} \right\}.$$

The subgraph $G_K = (V, E_K)$ is the *restriction of G to bounded rings*. A cycle of length less than or equal to *K* is also called a *feasible cycle*. Remark that an edge $e \in E \setminus E_K$ will never belong to a feasible solution of 2CNBR.

In order to formulate the 2CNBR problem, we associate with every subset $F \subseteq E$ an *incidence vector* $\mathbf{x}^F = (x_e^F)_{e \in E} \in \{0, 1\}^{|E|}$ by setting

$$x_e^F := \begin{cases} 1 & \text{if } e \in F, \\ 0 & \text{otherwise.} \end{cases}$$

Conversely, each vector $\mathbf{x} \in \{0, 1\}^{|E|}$ induces a subset $F^{\mathbf{x}} := \{e \in E \mid x_e = 1\}$.

Further we denote by $\mathcal{D}_{G,K}$ the set of incidence vectors x^F with $F \subseteq E$ such that

1. *F* is two-connected,
2. *F* = *FK* .

Then, the 2CNBR problem consists in

$$\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathcal{D}_{G,K} \right\}.$$

Since all costs *ce* , *e* ∈ *E* are assumed to be nonnegative, there always exists an optimal solution of 2CNBR whose induced graph is minimal with respect to inclusion. Hence, 2CNBR can be equivalently formulated as

$$\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathbb{Z}^+ \text{ and there exists } y \in \mathcal{D}_{G,K} : y \leq x \right\}.$$

For any subset of edges *F* ⊆ *E* we define

$$x(F) := \sum_{e \in F} x_e.$$

We also denote by *ei* the *i* -th unit vector in $\mathbb{R}^n$.

If a subset of edges *S* ⊆ *E* is such that (*G* - *S*)*K* is not two-connected, then *G* - *S* does not contain a feasible solution, and therefore each feasible solution contains at least one edge from *S*. As we are only interested in minimal feasible solutions, this is sufficient to formulate the 2CNBR problem as the following integer linear program:

$$\begin{array}{llll}
\min & \sum_{e \in E} c_e x_e & & \\
\text{s.t.} & x(S) \geq 1 & S \subseteq E, (G - S)_K \text{ is not two-connected}, & (1) \\
& x_e \geq 0 & e \in E, & (2) \\
& x_e \text{ integer} & e \in E. & (3)
\end{array}$$

Constraints (1) are called *subset constraints* and provide a set covering formulation of the 2CNBR problem. This formulation was introduced by Fortz and Labbe´ [7]. They also characterized which subset constraints are facet-defining for

$$\mathcal{P}_{G,K} := \text{conv}\left\{x \in \mathbb{R}^{|E|} : x \text{ satisfies (1)–(3)}\right\},$$

the polyhedron associated to the 2CNBR problem. This polyhedron is full dimensional, and is the dominant of conv($\mathcal{D}_{G,K}$).

3. A lower bound on the number of edges

In this section, we compute a lower bound on the number of edges in any feasible solution of 2CNBR. This result is useful for showing that the problem is NP-complete for any fixed $K > 3$ and for deriving new valid inequalities.

We first need the following definitions and properties from graph theory. Let $G = (V, E)$ be a connected graph with $n = |V|$ nodes and $m = |E|$ edges, and let T be a spanning tree in G . The addition of an edge to T creates exactly one cycle, called a *fundamental cycle*. Since there are $m — n + 1$ chords in G, there are exactly $m — n + 1$ fundamental cycles associated with each spanning tree.

Now, suppose the edges of G are ordered as $e_1, \ldots, e_m$, and we have a set of cycles σ_i, $i = 1,\ldots, k$. Then, we can define a *cycle matrix* $C = (c_{ij})_{k \times m}$ in which

$$c_{ij} = \begin{cases} 1 & \text{if } \sigma_i \text{ contains edge } e_i, \\ 0 & \text{otherwise.} \end{cases}$$

If the cycles σ_i are the fundamental cycles associated to a spanning tree T, this matrix is called *fundamental cycle matrix*. Moreover, if edges in G are numbered starting from the edges not belonging to T , and the fundamental cycles are numbered accordingly, then the fundamental cycle matrix has the following form:

$$C = (I_\mu ; C_t),$$

where I_μ is an identity matrix of dimension $\mu = m — n + 1$, and C_t is the remaining matrix of dimension μ x $(n — 1)$ corresponding to the edges of T.

It is clear that the rank of the fundamental cycle matrix C is $\mu = m — n + 1$. With respect to scalars in $\{0, 1\}$ and the addition modulo 2 (that we denote by φ), the fundamental cycles are independent, and we can define a vector space for which the fundamental cycles form a basis. This vector space is called the *cycle space*, and all the other cycles in G can be obtained as a linear combination of rows representing the fundamental cycles in C. The dimension μ of the cycle space is also called the *cyclomatic number* of the graph.

More details about fundamental cycles and the cycle space can be found, e.g., in Berge [2].

We can now prove the first important result of this section.

Theorem 1. *Let $G = (V, E)$ be a two-edge-connected network with $n = |V|$ nodes and $m = |E|$ edges. If there exists a covering of the edges of the network by feasible cycles, then there exists such a covering using at most $\mu = m — n + 1$ independent cycles.*

Proof: Since G is two-edge-connected, each edge belongs to a cycle, which can be obtained by a linear combination of the elements of a basis of the cycle space. This means that each edge must belong to at least one cycle in a basis, or, in other words, that a basis of the cycle space covers the edges of the network. Moreover, each cycle matrix of rank μ defines a basis of the cycle space. This implies that the set of cycles defined by any cycle matrix of rank μ—and in particular, fundamental cycles—covers the edges of the network.

Let C be a fundamental cycle matrix for G. If a subset of the fundamental cycles that use at most K nodes covers the edges, it forms the requested covering. Otherwise, we transform C in a cycle matrix C' of rank μ such that the subset of feasible cycles of C' covers the network.

Suppose C is a cycle matrix of rank μ, defining $r(r \le \mu)$ feasible cycles. In order to obtain the desired transformation, we now show that if the r feasible cycles do not cover the set of edges E, C can be transformed into a cycle matrix C' of rank μ such that the feasible cycles in C' cover one more edge. By applying this construction iteratively to the fundamental cycle matrix C, we create a sequence of matrices of rank μ, with an increasing number of edges covered by feasible cycles. We stop when the feasible cycles cover the network. This occurs in a finite number of steps, since in the worst case, we end with a matrix containing μ feasible cycles that cover all the edges.

Let C_1 be a $\mu \times m$ cycle matrix of rank μ and suppose there are r feasible cycles in C_1. Without loss of generality, we can suppose these cycles form the r first rows of $C1$, i.e., C_1 is of the form

$$C_1 = \begin{pmatrix} C_r \\ C_t \end{pmatrix},$$

where C_r is a $r \times m$ cycle matrix such that each cycle defined by C_r uses at most K nodes and Ct is a $(\mu — r) \times m$

cycle matrix such that each cycle defined by Ct uses at least $K + 1$ nodes.

Suppose cycles in C_r do not cover the network, i.e., there exists an edge $e \in E$ which does not belong to any cycle in C_r. This edge must belong to a cycle σ using at most K nodes. Since C_1 is of rank μ, σ can be obtained by a linear combination of its rows. Moreover, since no cycle in Cr uses e, the linear combination uses at least one row of Ct . Replacing this row of Ct by the linear combination of rows defining σ, we obtain a new cycle matrix $C2$ of same rank μ and covering one more edge.

Interestingly, Theorem 1 still holds if the graph G contains parallel edges. Note that the cycles in the cover do not necessarily correspond to a spanning tree.

Theorem 1 is the key to establish the main result of this section.

Theorem 2. *Let $G = (V, E)$ be a two-connected network with $n = |V|$ nodes and $m = |E|$ edges, such that there exists a covering of the network by cycles using at most K nodes. Then,*

$$m \geq M(n, K) := n + \min\left(\left\lceil \frac{n-K}{K-2} \right\rceil, \left\lceil \frac{n}{K-1} \right\rceil\right). \tag{4}$$

i.e., G contains at least $M(n, K)$ edges.

Proof: From Theorem 1, there exists a covering of G by at most $m — n + 1$ independent cycles using at most K nodes. We consider two disjoint cases:

1. *There exists a covering satisfying Theorem 1 and using at most $m — n$ cycles.*
   In this case, the sum of the number of edges used in each cycle is greater than or equal to m, since the cycles cover the network. Moreover, since each cycle uses at most K edges, we have

   $$m \leq (m-n)K,$$

   and the integrality of m allows to conclude that

   $$m \geq n + \left\lceil \frac{n}{K-1} \right\rceil. \tag{5}$$

2. *All coverings satisfying Theorem 1 use exactly $m — n + 1$ cycles.*
   It means that there exists such a covering which is also a basis of the cycle space. We first show that each cycle in this covering shares at least one edge with the others. Suppose it is not the case for some cycle σ in the covering. We define the *boundary* of σ as the subset of nodes of σ adjacent to nodes that do not belong to σ . Since the network is two-connected, there exist two nodes u and v in the boundary of σ that are linked by a path P with no edge in common with σ,asdepicted in figure 1. Two other paths $P1$ and P_2 between u and$v$ form σ . Combining P with P_1 and P_2 respectively, we obtain

   two new cycles$\sigma_1 = P \oplus P_1$ and$\sigma_2 = P \oplus P_2$. Since the covering defines a basis, $\sigma1$ and $\sigma2$ can be obtained by a linear combination of the cycles in the covering.
   If we suppose the edges are ordered starting from those in $P1$ then $P2$, and finally the remaining edges, the vector corresponding to σ has the form

   $$(\underbrace{1\ldots1}_{P_1}\underbrace{1\ldots1}_{P_2}0\ldots0)$$

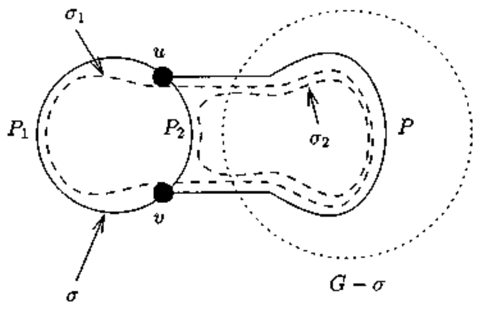
   while the other cycles in the basis have the form

   $$(\underbrace{0\ldots0}_{P_1}\underbrace{0\ldots0}_{P_2}\times\ldots\times)$$

   since we supposed σ had no edge in common with any other cycle in the covering. Moreover, the vector corresponding to σ_1 has the form

$$(\underbrace{1\ldots1}_{P_1}\underbrace{0\ldots0}_{P_2}\times\ldots\times).$$

***Figure 1*. Combinations of a cycle *σ* and an outside path *P* to generate cycles *σ*1 and *σ*2.**



It is clear from the form of those vectors that σ1 cannot be obtained by a linear combination of cycles in the covering, which leads to a contradiction.

So each cycle in the basis shares at least one edge with some other cycle in the basis. As there are $m - n + 1$ cycles, at least $m - n$ repetitions of edge occur when we count the edges in all cycles. Moreover, each edge is covered by at least one cycle and the sum of the number of edges used in each cycle is thus greater than or equal to $m + (m - n) = 2m - n$. This sum is also less than or equal to $(m - n + 1)K$ since each cycle uses at most K edges. We can conclude that

$$2m - n \le (m - n + 1)K.$$

or, since m is integral,

$$m \ge n + \left\lceil \frac{n-K}{K-2} \right\rceil. \qquad (6)$$

Since one of these two cases must occur, (5) or (6) is satisfied, thus the number of edges m satisfies (4).

Theorem 2 provides a lower bound on the number of edges in a feasible solution. This bound is tight, as we will see in the proof of Theorem 6, and it is useful for showing that the problem is NP-complete, which is done in the next section. Moreover, a similar application of Theorem 1 leads to some classes of strong valid inequalities that are described in Section 5.

4. Complexity

In this section, we show that the recognition version of the 2CNBR problem is NP-complete for any fixed value of the bound K .

Problem 3 (R2CNBR). Let $G = (V, E)$ be a graph, $K \ge 3$ a given constant and $B \ge 0$ an integer. To each edge $e \in E$ is associated a cost c_e and a unit length $d_e = 1$. Does there exists a subset $F \subseteq E$ of edges such that FK is two-connected and $c(F) \le B$?

Theorem 4. *R2CNBR is NP-complete for any $K \ge$* **3**.

Proof: It is easy to see that R2CNBR belongs to NP. We show that the Hamiltonian cycle problem reduces to R2CNBR, for any fixed $K \ge 3$. Let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges, and suppose $V = \{v_1, \ldots, v_n\}$. The Hamiltonian cycle problem consists in determining if there exists a Hamiltonian cycle in G.
This problem is NP-complete and it can be transformed into R2CNBR in the following way.

If $K > n$, finding a Hamiltonian cycle is equivalent to finding a two-connected network of cost less than or equal to n, for the same graph, with unit edge costs. But $K > n$ implies that R2CNBR is equivalent to the two-connected network problem, since the constraints on the rings are redundant. Thus the Hamiltonian cycle problem reduces to R2CNBR if $K > n$.

If $K < n$, define a graph $G' = (V', E')$ with

$$V' = \bigcup_{k=0}^{K-2} V^k, \quad \text{where } V^k = \{v_1^k, \ldots, v_n^k\}, \quad k = 0, \ldots, K-2,$$

and

$$E' = \bigcup_{i,j: v_i v_j \in E} \{v_i^0 v_j^0, v_i^0 v_j^{K-2}, v_j^0 v_i^{K-2}\} \cup \bigcup_{k=0}^{K-3} \bigcup_{i=0}^{n} \{v_i^k v_i^{k+1}\}$$

A unit cost is again assigned to each edge. This transformation is illustrated in figure 2 for $K = 5$.

G' is composed of $n' = (K - 1)n$ nodes and $m' = 3m + (K - 2)n$ edges. The transformation can thus be performed in polynomial time and space. We will show that there exists a feasible solution of 2CNBR of cost less than or equal to Kn in G' if and only if there exists a Hamiltonian cycle in G , and conversely.

Suppose there exists a solution F of 2CNBR of cost less than or equal to Kn in G'. Since edge costs are unitary, the number of edges in F is equal to the cost of the solution, $c(F)$. By Theorem 2, we thus have

$$c(F) \geq n' + \min\left(\left\lceil \frac{n'-K}{K-2} \right\rceil, \left\lceil \frac{n'}{K-1} \right\rceil\right).$$

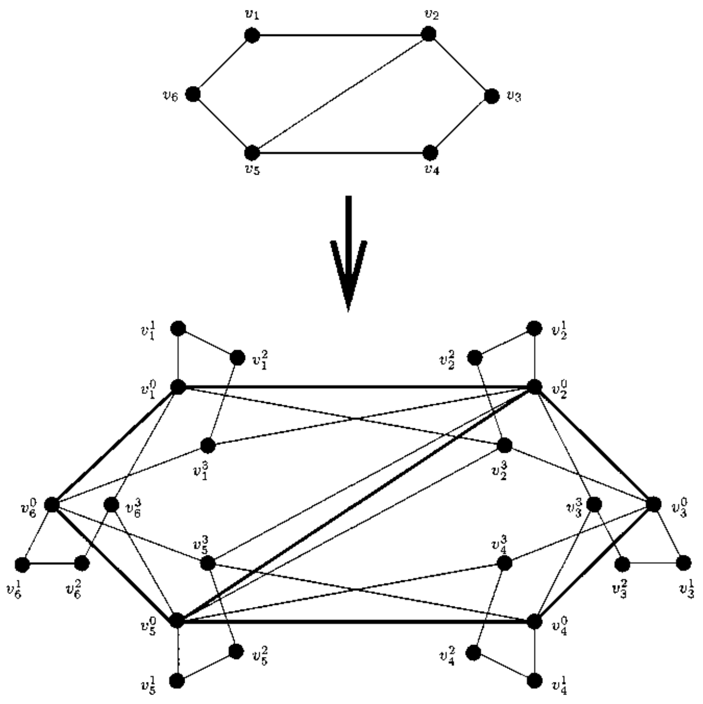
But $n' = (K - 1)n$, so we have

$$\frac{n'}{K-1} = n$$

and

$$\frac{n'-K}{K-2} = \frac{(K-1)n-K}{K-2} = n + \frac{n-K}{K-2} > n = \frac{n'}{K-1},$$

***Figure 2*. Transformation of Hamiltonian cycle into 2CNBR.**



since $n > K$ and $K \geq 3$. Thus,

$$Kn \geq c(F) \geq n' + \left\lceil \frac{n'}{K-1} \right\rceil = (K-1)n + n = Kn.$$

and F contains exactly Kn edges.

Since F is two-connected, the degree of each node in this solution is at least two. Moreover, for each v^k_i, $i = 1,..., n$, $k = 1, ..., K — 3$, there are only two possible edges, and these edges must belong to any feasible solution. There are $(K — 2)n$ such edges that form n paths of length $K — 2$, $(v_i^0, v_i^1, ..., v_i^{k-2})$, as illustrated in bold in figure 3. Since each edge in these paths must belong to a cycle using at most K nodes, two edges (v^{K-2}, w) and (w, v_i^0) must be in the solution to close each path and form a feasible cycle. It is clear that the only possible choice for w is a node $v_j^0 \in V^0$.

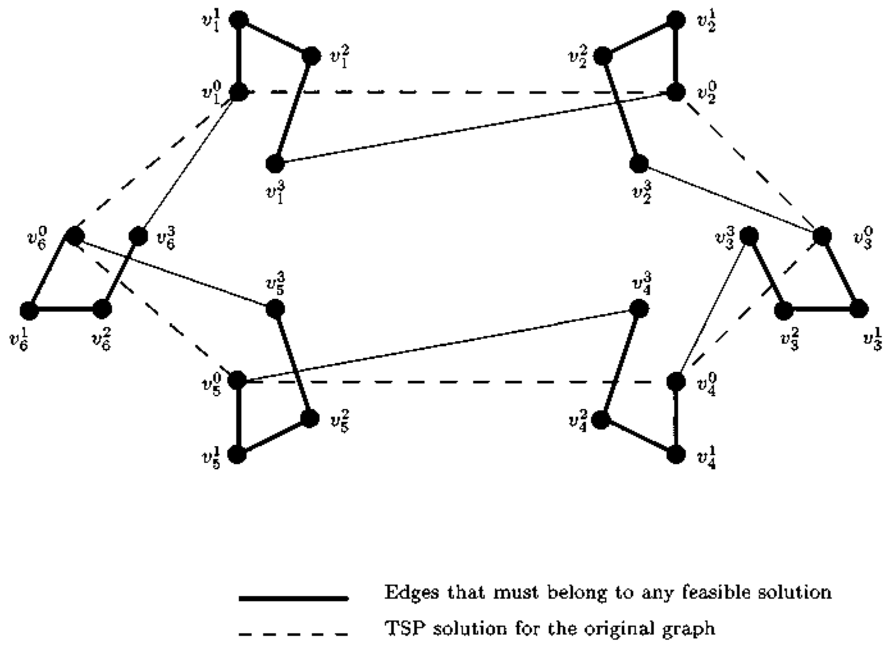
In this way, we get a subgraph (V', F) with Kn edges, with exactly n edges in $F(V^0)$, corresponding to edges in E. We show that these edges form a Hamiltonian cycle in G . Since there are exactly n edges, it is sufficient to show that these induce a two-connected subgraph of G, i.e., that $F(V^0)$ is two-connected. Let v_i^0 and v^0_j be any two nodes in V^0.In F, there are two node-disjoint paths from v_i^0, to v^0_j, since F is two-connected. If these paths use only nodes in V^0, the expected result holds. Otherwise, each node of these paths not belonging to V^0 must appear in a sequence of the form v_i^0, v_k^1,..., v^{K-2}, v_l^0 By our previous construction, this imply that (v_k^0 , v_l^0) is also in the solution. Replacing each sequence of the form v_k^0, v_k^1, ... v_k^{K-2}, v_l^0 using nodes outside V^0 by the corresponding shortcut (v_k^0, v_l^0) leads to two disjoint paths from v_i^0 to v_j^0 in $E(V^0)$.

So, $F(V^0)$ is two-connected and the n edges in $F(V^0)$ correspond to a Hamiltonian cycle in G.

Conversely, it is easy to see that we can complete an edge set $F(V^0)$ corresponding to a Hamiltonian cycle in G , defined by the sequence of nodes $v_{i_1}, v_{i_2}, \ldots, v_{i_n}, v_{i_{n+1}} = v_{i_1}$, into a feasible solution F of 2CNBR with Kn edges by adding the paths $v^0_{i_k}, v^1_{i_k}, \ldots, v^{K-2}_{i_k}, v^0_{i_{k+1}}$ for $k = 1,..., n$ to the edges already in $F(V_0)$.

This concludes the proof of Theorem 4.

***Figure 3*. A solution of R2CNBR and the corresponding Hamiltonian cycle.**



5. Cyclomatic inequalities

From the lower bound on the number of edges in a feasible solution obtained in Section 3, we can conclude that

$$\sum_{e\in E} x_e \geq M(n, K) = n + \min\left(\left\lceil\frac{n-K}{K-2}\right\rceil, \left\lceil\frac{n}{K-1}\right\rceil\right)$$

is a valid inequality for $P_{G,K}$. In this section, we extend this result to new classes of valid inequalities for $P_{G,K}$.
Our first result is an extension of Theorem 2 to a partition of V .

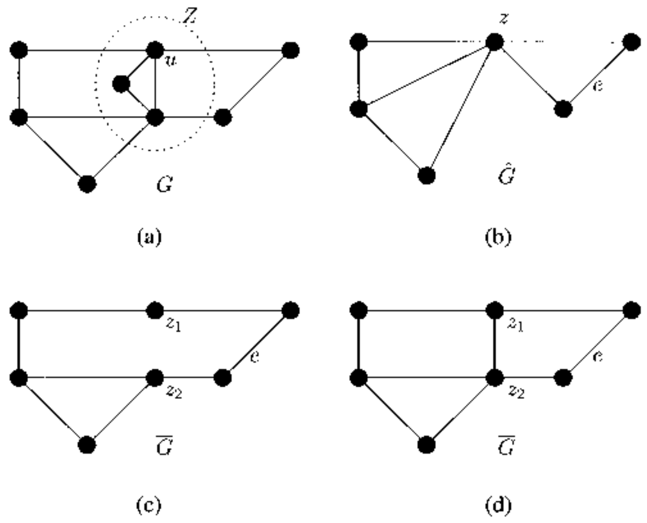
Proposition 5. *Let* $G = (V, E)$ *be a graph with* $n = |V|$ *nodes,* $K \geq 3$ *a given constant, and* $W1, W2, ..., W_p$ ($p \geq 2$) *a partition of V. Then*

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta(W_i)) \geq M(p, K) \qquad (7)$$

is a valid inequality for PG, K .

Proof: Let F be a feasible solution to the 2CNBR problem and let $\hat{G}$ denote the contracted graph $(V, F)/ W1 / ... / W_p$, and $\hat{m}$ the number of edges in $\hat{G}$. We show that $\hat{m} \geq M(p, K)$.
It is easy to see that $\hat{G}$ is two-edge-connected and that each edge in $\hat{G}$ belongs to a cycle using at most K edges.

***Figure 4*. Agraph *G* , the contracted graph *G* and the two steps of *G* construction, with a bound *K* = 4.**



If $\hat{G}$ is two-connected, Theorem 2 holds for $\hat{G}$, i.e. $\hat{m} \geq M(p, k)$. Otherwise, $\hat{G}$ contains $q \geq 1$ articulation points. Let z be one of these nodes, obtained after the contraction of a subset $Z \subseteq V$ of nodes in G (figure 4(a) and (b)). Since F is two-connected, the boundary of Z contains at least two nodes. Let u be one of these nodes. Replacing z by two nodes $z1$ and $z2$, and replacing each edge incident to z by an edge connected to $z1$ if the corresponding edge in G was connected to u, and by an edge connected to z_2 otherwise, we obtain a graph with one articulation point less, as illustrated in Figure 4(c). However, it is possible that this new graph contains some edges not belonging to a cycle using at most K edges. In this case, let e be one of these edges. Before replacing z , e belonged to at least one feasible cycle. Each of these cycles has been replaced by a path from z_1 to z_2. It is easy to see that one of these paths must contain at most $K — 1$ edges, otherwise the corresponding edge in F cannot belong to a feasible cycle. Therefore, adding an edge linking z_1 and z_2 is sufficient to create a feasible cycle containing e (figure 4(d)). Repeating this construction for each articulation point, we obtain a two-connected graph $\bar{G}$ such that each edge in $\bar{G}$ belongs to a cycle using at most K edges. Since $\bar{G}$ has $p+q$ nodes, it contains at least $M(p + q, K)$ edges by Theorem 2, and by our construction, it contains at most $m + q$ edges. Therefore, and we have proved Proposition 5.

Inequalities (7) are called *cyclomatic inequalities*. The next theorem shows that the inequality bounding the total number of edges (i.e., $p = n$) is facet-defining for complete graphs.

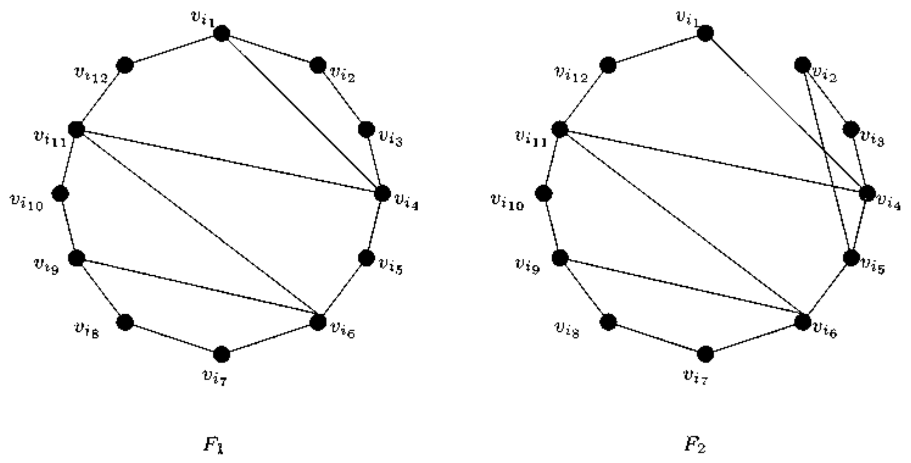
Theorem 6. *Let $G = (V, E)$ be a complete graphwith $V = \{v_1 , . . . , v_n\}$, and $3 \leq K < n$ a given constant. Then $x(E) > M(n, K)$ defines a facet of $P_{G,k}$.*

Proof: We consider separately the two cases corresponding to the two possible values of $M(n, K)$.
$\mathbf{M(n, K) = n + \lceil \frac{n-K}{K-2} \rceil}$. Let $b^T x \geq \beta$ be a facet-defining inequality such that the face induced by $x(E) > M(n, k)$ in $P_{G,K}$ is contained in the facet F_b induced by $b^T x \geq \beta$.

Our aim is to show that *be* has the same value for all $e \in E$.
Consider a permutation $\{i_1,...,i_n\}$of $\{1, ..., n\}$, and the two following sets of edges:

$$F_1 = \bigcup_{j=1}^{n-1} \{v_{i_j} v_{i_{j+1}}\} \cup \{v_{i_1} v_{i_n}\} \cup \{v_{i_1} v_{i_K}\} \cup \bigcup_{j=1}^{l_1} \{v_{i_{K+(j-1)(K-2)}} v_{i_{n+1-j(K-2)}}\}$$
$$\cup \bigcup_{j=1}^{l_2} \{v_{i_{n+1-j(K-2)}} v_{i_{K+j(K-2)}}\},$$
$$F_2 = \bigcup_{j=2}^{n-1} \{v_{i_j} v_{i_{j+1}}\} \cup \{v_{i_1} v_{i_n}\} \cup \{v_{i_1} v_{i_K}\} \cup \{v_{i_2} v_{i_{K+1}}\} \cup \bigcup_{j=1}^{l_1} \{v_{i_{K+(j-1)(K-2)}} v_{i_{n+1-j(K-2)}}\}$$
$$\cup \bigcup_{j=1}^{l_2} \{v_{i_{n+1-j(K-2)}} v_{i_{K+j(K-2)}}\},$$

Figure 5**. **F**1 **and** **F**2 **for** **n** **= 12 and** **K** **= 4.



where $l_1 = \lceil \frac{n-2}{2(K-2)} \rceil - 1$ and $l_2 = \lceil \frac{n-K}{2(K-2)} \rceil - 1$. These constructions are illustrated in figure 5 for n = 12 and K = 4. It is easy to see that F_1 and $F2$ define feasible solutions of 2CNBR, and that $|F_11 = |F_2| = M(n, K)$. Therefore, the incidence vectors of F_1 and F_2 lie in F_b. Since F_2 is obtained from F_1 by replacing edge $v_{i\,1}\,v_{i2}$ by $v_{i2}\,v_iK_{+1}$, we obtain that $b_{vi\,1}\,Vi_2 = bv_{i2}\,v_{K+1}$.

Consider three distinct nodes vi,vj and vk. Since the permutation defined above was arbitrary, any permutation such that $i\,_1 = i$, $i_2 = j$ and $i_{K+1} = k$ leads to $b_vi\,_{vj} = b_{vj}\,_vk$. If v_i, $v\,j$, v_k and v_l are four distinct nodes, we obtain by transitivity that $b_{vi\;vj} = b_{vj\;vk} = b_{vk\;vl}$, and therefore be has the same value for all $e \in E$.

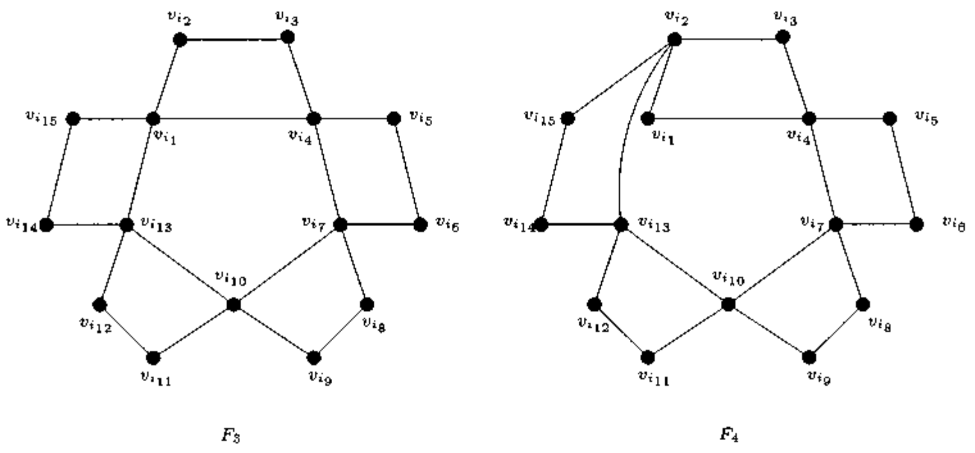
$\mathbf{M(n, K) = n + \lceil \frac{n}{K-1} \rceil}$. Let $b^T x \geq \beta$ be a facet-defining inequality such that the face induced by $x(E) \geq M(n, k)$ in $P_{G,K}$ is contained in the facet F_b induced by $b^T x \geq \beta$. Our aim is to show that b_e has the same value for all $e \in E$.

Consider a permutation $\{i\,_1,..., i_n\}$ of $\{1, ..., n\}$, and the two following sets of edges:

$$F_3 = \{v_{i_1} v_{i_n}\} \cup \{v_{i_1} v_{i_{n-K+2}}\} \cup \bigcup_{j=1}^{n-1} \{v_{i_j} v_{i_{j+1}}\} \cup \bigcup_{j=1}^{\lceil \frac{n}{K-1} \rceil - 1} \{v_{i_{1+(j-1)(K-1)}} v_{i_{1+j(K-1)}}\},$$
$$F_4 = \{v_{i_2} v_{i_n}\} \cup \{v_{i_2} v_{i_{n-K+2}}\} \cup \bigcup_{j=1}^{n-1} \{v_{i_j} v_{i_{j+1}}\} \cup \bigcup_{j=1}^{\lceil \frac{n}{K-1} \rceil - 1} \{v_{i_{1+(j-1)(K-1)}} v_{i_{1+j(K-1)}}\}.$$

These constructions are illustrated in figure 6 for n = 15and K = 4. It is easy to seethat $F3$ and F_4 define feasible solutions of 2CNBR, and that \ $F3$\ = \ F_4\ = $M(n, K)$. Therefore, the incidence vectors of $F3$ and $F4$ lie in Fb . Since $F4$ is obtained from $F3$ by replacing edges $v_{i_1} v_{i_n}$ and $v_{i_1} v_{i_{n-K+2}}$ by $v_{i_2} v_{i_n}$ and $v_{i_2} v_{i_{n-K+2}}$, we obtain that $b_{v_{i_1} v_{i_n}} + b_{v_{i_1} v_{i_{n-K+2}}} = b_{v_{i_2} v_{i_n}} + b_{v_{i_2} v_{i_{n-K+2}}}$.

Figure 6. ***F*3 and *F*4 for *n* = 15 and *K* = 4.**



Consider four distinct nodes v_i ,v_j,v_k and v_l. Since the permutation defined above was arbitrary, any permutation such that $i_1 = i$, $i_2 = j$, $i_n = k$ and $i_{n-K+2} = l$ leads to

$$b_{v_i v_k} + b_{v_i v_l} = b_{v_j v_k} + b_{v_j v_l}.$$

Similarly, any permutation such that $i_1 = k$, $i_2 = l$, $i_n = i$ and $i_{n-K+2} = j$ leads to

$$b_{v_i v_k} + b_{v_j v_k} = b_{v_i v_l} + b_{v_j v_l}.$$

This leads to $b_{v_i v_k} = b_{v_j v_l}$.

It remains to be shown that two edges having a common endpoint have the same coefficient. If v_i ,v_j,v_k,v_l and v_m are five distinct nodes, we obtain by transitivity that $b_{v_i v_j} = b_{v_l v_m} = b_{v_i v_k}$, and therefore b_e has the same value for all $e \in E$.

6. Separation of cyclomatic inequalities

The main difficulty that appears when trying to separate cyclomatic inequalities is the fact that the right-hand-side of the inequality is not linear in the number p of subsets that define the partition.

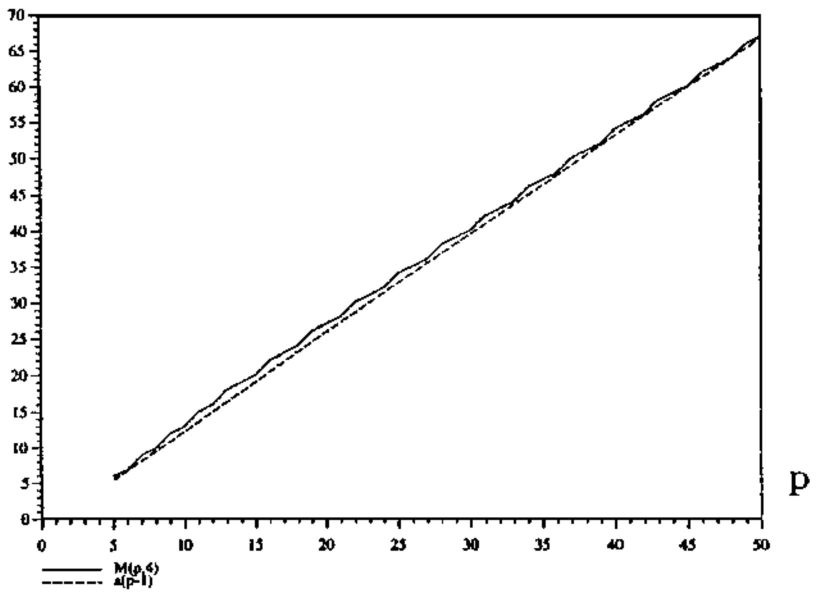
To override this difficulty, we decided to approximate $M(p, K)$ by a linear function of the form $a(p - 1)$. Therefore, we try to find a most violated inequality of the form

$$\frac{1}{2}\sum_{i=1}^{p}\sum_{e\in\delta(W_i)}\frac{x_e}{a} \geq p-1,$$

which can be performed in polynomial time using Barahona's algorithm [1] for partition inequalities. If such a violated inequality is found, itis sufficient to check that the cyclomatic inequality defined by the same partition is also violated.

Figure 7. **Approximation of *M*(*p*, 4) for *n* = 50.**

Published in : Computational Optimization and Applications, 27, 123–148, 2004
Status : Postprint (Author's version)



The effectiveness of this procedure depends heavily on the choice of *a* in order to have a good approximation of *M*(*p*, *K*). Our choice was to take

$$a = \frac{nK}{(n-1)(K-1)}$$

which leads to $a(p-1) = n + \frac{n}{K-1}$ for *p* = *n*. Figure 7 illustrates this approximation for *K* = 4 and *n* = 50. We can observe that the approximation is quite accurate, and gives good results in practice.

7. Other valid inequalities

In this section, we briefly present the other valid inequalities used in our Branch-and-Cut algorithm for the 2CNBR problem. All these inequalities come from [7, 8].

7.1. Cut constraints

Classical inequalities used to impose that a network is two-edge-connected are cut constraints. These contraints are widely used to formulate the traveling salesman problem—in this case, they are equivalent to subtour elimination constraints—orthe minimum-cost two-connected network problem. Given a subset of nodes $W \subseteq V, \emptyset \neq W \neq V$, the cut constraint imposes that there are at least two edges leaving *W*, i.e.,

$$x(\delta(W)) \geq 2. \tag{8}$$

The separation of cut constraints can be carried out by computing a minimum cut in the graph, with capacities given by the current LP solution. This can be done in polynomial time, e.g. by the Gomory-Hu algorithm [10] that requires *n* — 1 maximum flow computations.

7.2. Metric inequalities

Metric inequalities arise from the projection of a flow formulation of the 2CNBR problem (Fortz et al. [8]).

Let $F \subseteq E$ be a feasible solution defined by the $(x_e)_{e \in E}$ variables, and consider an edge $e := ij \in F$. This edge belongs to a feasible cycle of length less than or equal to K. This means that the problem

$$\min \quad w(e) = \sum_{kl \in E} d(k,l)(u_{kl} + u_{lk})$$

$$\begin{aligned}
\text{s.t.} \quad & \sum_{l:kl \in E} (u_{kl} - u_{lk}) = \begin{cases} 2x_{ij} & \text{if } k = i \\ -2x_{ij} & \text{if } k = j \\ 0 & \text{otherwise} \end{cases} \quad && k \in V. && (9) \\
& u_{kl} + u_{lk} \leq x_{kl} && kl \in E. && (10) \\
& u_{kl} \geq 0 && kl \in E. && (11) \\
& u_{lk} \geq 0 && kl \in E && (12)
\end{aligned}$$

has an optimal solution value $w^*(e)$ such that $w^*(e) \leq K$.

Using duality, the following class of valid inequalities was obtained in [8]. Consider an edge $e := ij \in E$ and a set of node potentials $(\alpha_k)_{k \in V}$ satisfying

$$\alpha_i - \alpha_j > K - d(i,j).$$

Then

$$\sum_{f \in E \setminus \{e\}} v_f x_f \geq x_e \qquad (13)$$

is a valid inequality for 2CNBR (*metric inequality*) where

$$v_f = \min\left(1, \max\left(0, \frac{|\alpha_l - \alpha_k| - d(k,l)}{\alpha_i - \alpha_j + d(i,j) - K}\right)\right) \qquad (14)$$

for all $f := kl \in E \setminus \{e\}$.

Algorithm 1 was developed by Fortz et al. [8] and provides a heuristic for solving the corresponding separation problem.

Algorithm 1: Separation of metric inequalities

```
Data: a graph $G = (V, E)$, a real number $K > 0$, a vector $\tilde{\mathbf{x}} := (\tilde{x}_f)_{f \in E}$, an edge
    $e := ij \in E$.
 1: Compute a minimum cost flow of value $\tilde{x}_e$ between $i$ and $j$ with capacities
      $(\tilde{x}_f)_{f \in E \setminus \{e\}}$ in $G - e$.
 2: Let $(\pi_k)_{k \in V}$ be the corresponding node potentials.
 3: if $\pi_j + K - d_e \geq 0$ then
 4:   stop {all metric inequalities having $x_e$ as right-hand side are satisfied}
 5: for all $k \in V$ do
 6:   if $k = j$ then
 7:     $\alpha_k := \pi_j$
 8:   else
 9:     $\alpha_k := 0$
10:   label($k$) := false
11: while $\exists k \in V$ : label($k$) = false do
12:   $l :=$ arg min$\{\alpha_k : k \in V$, label($k$) =false$\}$
13:   for all $k \in V : kl \in E \setminus \{e\}$ do
14:     if $(\alpha_k > \alpha_l + d(k, l))$ and $(\alpha_l + d(k, l) \geq \pi_k)$ then
15:       $\alpha_k := \alpha_l + d(k, l)$
16:   label($l$) := true
17: for all $f := kl \in E \setminus \{e\}$ do
18:   $v_f := \min\left(1, \max\left(0, \frac{|\alpha_l - \alpha_k| - d(k,l)}{\alpha_i - \alpha_j + d(i,j) - K}\right)\right)$
19: if $\sum_{f \in E \setminus \{e\}} v_f \tilde{x}_f < \tilde{x}_e$ then
20:   {$(\alpha_k)_{k \in V}$ define a violated metric inequality}
21: else
22:   {no violated inequality was found}
```

7.3. Subset inequalities

Subset inequalities (1) were used to formulate the 2CNBR problem. The subset inequality induced by a subset of edges $S \subseteq E$ such that $(G — S)K$ is not two-connected is $x(S) > 1$.

We rely on a greedy heuristic proposed in [7] to separate these inequalities. This heuristic is summarized in Algorithm 2.

The main drawback of the greedy heuristic is that it often fails to find a suitable F .Itis easy to adapt the algorithm to perform the exact separation by a backtracking procedure, enumerating all the subsets F such that $x(E \backslash F) < 1$. However, there is an exponential number of such subsets, and complete enumeration is not efficient in practice. An intermediate approach is to allow a fixed number of backtracking steps. Our computational experiments show that performing 10 backtracking steps is a good tradeoff between computing time and quality of the separation.

Another way of deriving violated subset inequalities comes from the separation of metric inequalities. Let $\sum_{f \in E \setminus \{e\}} v_f x_f \geq x_e$ be a metric inequality, and let $S := \{f \in E \setminus \{e\}:$ $v_f > 0\}$. If $(G — S)K$ is not two-connected and $x(S) < 1$, then S defines a violated subset inequality. This situation is often met in practice. Moreover, if $vf = 1$ for all $f \in S$, then the subset inequality is stronger than the metric inequality.

Algorithm 2: Separation of subset inequalities

```
Data: a graph $G = (V, E)$, a real number $K > 0$, a vector $\tilde{\mathbf{x}} := (\tilde{x}_f)_{f \in E}$.
1:  let $F := \{e \in E : \tilde{x}_f > 0\}$
2:  if $F_K$ is two-connected then
3:      sort edges of $F$ in increasing order of $\tilde{x}_f$,
        such that $F = \{j(1), \dots, j(|F|)\}$, $\tilde{x}_{j(i)} \le \tilde{x}_{j(k)}$ for all $i < k$.
4:      $i := 1$
5:      $v := 0$
6:      while $v + \tilde{x}_{j(i)} < 1$ and $i \le |F|$ and $F_K$ is two-connected do
7:          $F := F \backslash \{j(i)\}$
8:          if $F_K$ is two-connected then
9:              $v := v + \tilde{x}_{j(i)}$
10:         i := i + 1
11: if $F_K$ is not two-connected then
12:     $S := \emptyset$
13:     for all $e \in E \backslash F$ do
14:         if $(F \cup \{e\})_K$ is two-connected then
15:             $S := S \cup \{e\}$
16:         else
17:             $F := F \cup \{e\}$
18:         {$x(S) \ge 1$ is a violated facet-defining subset inequality}
19: else
20:     {no violated subset was inequality found}
```

7.4. *Ring-cut inequalities*

The cut defined by a subset $W \subseteq V$ of nodes in a two-connected network must contain at least two edges, leading to cut constraints (8). To determine if a cut constraint is facetdefining, it is useful to know if there exists a vector of $\mathcal{P}_{G,K}$ lying in the face $x(\delta(W)) = 2$ with $x_e = xf = 1$ for a pair of edges $e, f \in \delta(W)$. This is the case if and only if the incidence vector of $C_{e,f} := E(W) \cup E(V \backslash W) \cup \{e, f\}$

belongs to $\mathcal{P}_{G,K}$, i.e. if $(C_{e,f})_K$ is two-connected. A useful tool to represent and analyze the vectors belonging to the face defined by a cut constraint is the *ring-cut graph* defined below.

Definition 7 (Ring-cut graph). Let $G = (V, E)$ be a graph, $K > 0$ a given constant, and $W \subseteq V$ a subset of nodes, $\emptyset \neq W \neq V$.

The ring-cut graph $RCG_{W,K} := (\delta(W), RCE_{W,K})$ induced by W is the graph defined by associating one node to each edge in δ (W) and by the set of edges $RCE_{W,K} = \{ef \subseteq \delta(W) : (C_{e,f})_K \text{ is two-connected}\}$.

Using the ring-cut graph, we derived a new classes of valid inequalities called *ring-cut inequalities* (see [6]). Given a subset of nodes $W \subseteq V, \emptyset \neq W \neq V$ and an independent subset $S \subseteq \delta(W)$ in the ring-cut graph $RCG_{W,K}$, the ring-cut inequality is given by

$$x(S) + 2x(\delta(W) \backslash S) \ge 3. \qquad (15)$$

The separation problem for ring-cut inequalities is NP-complete [7]. We solve it using a greedy heuristic. As we use the Gomory-Hu algorithm to separate cut constraints, we benefit from this information in the separation of ring-cut inequalities, considering the n — 1 minimum cuts provided by the Gomory-Hu tree as good candidates. To each cut of capacity less than 3 in the tree, we apply the greedy heuristic described in Algorithm 3 to determine an independent subset in the ring-cut graph corresponding to the cut.

Algorithm 3: Separation of ring-cut inequalities

```
Data: a graph $G = (V, E)$, a real number $K > 0$, a vector $\tilde{\mathbf{x}} := (\tilde{x}_f)_{f \in E}$,
      a subset $W \subseteq V, \emptyset \neq W \neq V$ such that $\tilde{x}(\delta(W)) < 3$.
1:  $v := \tilde{x}(\delta(W))$
2:  $S := \emptyset$
3:  for all $e \in \delta(W)$ do
4:      $mark(e) :=$ false
5:  while $(v < 3)$ and $(\exists e \in \delta(W) : mark(e) =$ false$)$ do
6:      $e := \arg\max\{\tilde{x}_f : f \in \delta(W), mark(f) =$ false$\}$
7:      for all $f \in \delta(W) : mark(f) =$ false do
8:          if $(C_{e,f})_K$ is two-connected then
9:              {edge $ef$ belongs to the ring-cut graph}
10:             $v := v + \tilde{x}_f$
11:             $mark(f) :=$ true
12:     $mark(e) :=$ true
13:     $S := S \cup \{e\}$
14: if $v < 3$ then
15:     {$x(S) + 2x(\delta(W)\backslash S) \geq 3$ is a violated ring-cut inequality}
```

7.5. Node-partition inequalities

The polyhedron of connected networks is completely described by *partition inequalities* (Grotschel and Monma [11]). Given a partition $W_1, W_2, \ldots, W_p (p > 2)$ of V into p nonempty subsets, the corresponding partition inequality is

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta(W_i)) \geq p - 1 .$$

Since a two-connected network remains connected when a node is removed,

$$\frac{1}{2}\sum_{i=1}^{p} x(\delta_{G-z}(W_i)) \geq p - 1$$

is a valid inequality for the polyhedron of two-connected networks, and therefore for 2CNBR, where $W_1, W_2, \ldots, W_p (p > 2)$ is a partition of $V\backslash\{z\}$. These inequalities are called *node-partition inequalities*.

The first separation algorithm for these inequalities was given by Cunningham [4] and requires $|E|$ min-cut computations. Barahona [1] reduced this computing time to $|V|$ mincut computations.

Separating node-partition inequalities can thus be done for each node $z \in V$ by applying Barahona's algorithm to $G - z$. This requires $|V|^2$ min-cut computations. In order to reduce this number, note that if G is a two-edge-connected network, any articulation point in G has a degree at least equal to 4. Therefore, we decided to apply Barahona's algorithm to $G - z$ only if $x(\delta(z)) > 3$, which leads to a much faster separation procedure.

8. Implementation of the Branch-and-Cut algorithm

In this section, we describe some strategic choices that were made in the implementation of our Branch-and-Cut algorithm for the 2CNBR problem. Our aim here is not to describe in detail the general Branch-and-Cut framework, but to emphasize the problem-specific aspects of our algorithm. For a general introduction to Branch-and-Cut, we refer the reader to Thienel [17] or to Nemhauser and Wolsey [15] for a more complete survey on combinatorial optimization and polyhedral theory.

The algorithm was implemented in C++, using version 2.0 of the ABACUS library (Thienel [17, 18]), and CPLEX 4.0 as LP-solver.

The initial linear program is defined by degree constraints $x(\delta(v)) \geq 2$ for all $v \in V$ and the lower bound on the number of edges $x(E) \geq M(n, K)$.

An important issue in the effectiveness of a Branch-and-Cut algorithm is the computation of good upper bounds. Due to the effectiveness of the Tabu Search heuristic presented in [5], we perform 600 interations of Tabu Search in parallel with the Branch-and-Cut algorithm to obtain a good upper bound.

Moreover, we try to transform each LP-solution obtained in the Branch-and-Cut to a feasible solution by rounding up to 1 all the variables with fractional value.

The pool used to store generated inequalities is the standard pool in ABACUS. We start with a pool size equal to 100 times the number of nodes in the network, and we allow this size to be increased dynamically if necessary. All the generated inequalities are put in the pool and are dynamic, i.e., they are removed from the current LP when they are not active. The separation of valid inequalities is performed as follows. We first separate inequalities from the standard pool. If all the inequalities in the pool are satisfied by the current LP-solution, we separate inequalities in the following order:

1. cyclomatic inequalities;
2. subset inequalities;
3. cut constraints;
4. ring-cut inequalities;
5. node-partition inequalities;
6. metric inequalities.

The order of separation was chosen after a series of numerical experiments, the choice of inequalities separated first seemingly being the best trade-off between separation time and efficiency of the cuts.

Moreover, we go to the next class of inequalities only if the number of generated cutting planes is less than 50. Otherwise, we solve the LP again and restart the separation procedure.

All inequalities are global (i.e., valid in all the tree), except ring-cut inequalities that are valid locally.

9. Computational results

We present in this section numerical results obtained for the 2CNBR problem with the Branch-and-Cut algorithm.

***Table 1*. List of abbreviations.**

|*V*|	number of nodes in the graph
|*E*|	number of edges after preprocessing
K	bound on the length of cycles
p/o	for random problems, total number of problems/number of problems solved to optimality
# ineq.	number of inequalities generated
#B&B nodes	number of Branch-and-Bound nodes examined (including the root node)
LB (root)	lower bound obtained at the root node of the Branch-and-Bound tree
LB (final)	global lower bound at the end of the optimization
UB	best upper bound found
Gap	gap between the final upper and lower bounds: $\text{gap} = \frac{100(UB-LB)}{LB}$
CPU time	time spent in the Branch-and-Cut (without the Tabu Search)

***Table 2.* Branch-and-Cut results, real applications, unit edge lengths.**

|V|	|E|	*K*	# ineq.	#B&B nodes	LB (root)	LB (final)	UB	Gap (root)	Gap (final)	CPU time (hh:mm:ss)
12	39	3	179	31	740	794	794	7.3	0.0	0:00:09
12	39	4	288	105	647	681	681	5.3	0.0	0:00:18
12	39	5	73	7	598	606	606	1.3	0.0	0:00:01
12	39	6	53	5	563	568	568	0.9	0.0	0:00:01

12	39	8	39	13	521	537	537	3.1	0.0	0:00:01
12	39	10	27	7	517	521	521	0.8	0.0	0:00:01
12	39	16	24	1	496	496	496	0.0	0.0	0:00:01
17	88	3	1570	3521	1034	1100	1100	6.4	0.0	0:18:29
17	88	4	2549	5381	907	966	966	6.5	0.0	0:29:37
17	88	5	270	61	837	855	855	2.2	0.0	0:00:26
17	88	6	75	1	797	797	797	0.0	0.0	0:00:02
17	88	8	222	105	752	766	766	1.9	0.0	0:00:22
17	88	10	36	1	715	715	715	0.0	0.0	0:00:01
17	88	16	13	1	711	711	711	0.0	0.0	0:00:01
30	200	3	8481	38011	1310	1417	1431	9.2	1.0	10:00:00
30	200	4	23545	33887	1171	1227	1326	13.2	8.1	10:00:00
30	200	5	22860	35693	1078	1127	1176	9.1	4.3	10:00:00
30	200	6	2683	2721	1018	1055	1055	3.6	0.0	0:45:22
30	200	8	1957	1049	930	956	956	2.8	0.0	0:18:21
30	200	10	315	9	895	901	901	0.7	0.0	0:00:41
30	200	16	1650	781	843	861	861	2.1	0.0	0:08:31
52	622	3	4143	11115	1630	1674	1870	14.7	11.7	10:00:00
52	622	4	5129	5783	1411	1457	1663	17.9	14.1	10:00:00
52	622	5	6474	3781	1303	1354	1478	13.4	9.2	10:00:00
52	622	6	5233	4099	1256	1292	1362	8.4	5.4	10:00:00
52	622	8	5974	3049	1178	1209	1246	5.8	3.1	10:00:00
52	622	10	7535	3933	1136	1160	1208	6.3	4.1	10:00:00
52	622	16	10117	4693	1074	1091	1104	2.8	1.2	10:00:00

The Branch-and-Cut was implemented using ABACUS 2.0 and CPLEX 4.0, and tested on a SUN Sparc Ultra 1 workstation with a 166 Mhz processor and 128 MB RAM. We fixed the maximum CPU time to 10 hours, except for randomly generated problems with 40 and 50 nodes, where it was limited to 3 hours, due to the large number of problems to solve. Moreover, for these large problems, we noticed that the bounds did not improve much after 3 hours.

We consider costs equal to the rounded Euclidean distance. Tests were made for different values of the bound, for instances coming from real applications, with 12, 17, 30 and 52 nodes, and for random problems with nodes uniformly generated in a square of size 250 × 250. Random problems with 10 to 50 nodes were generated, and we tested five instances of each size. Data on the randomly generated test problems are available at the Web page http://www.poms.ucl.ac.be/staff/bf/en/2cnbm/data.html. The CPU times reported do not include the Tabu Search procedure, as it was run in parallel on another processor.

Table 2 reports results obtained for problems coming from real applications, while Table 4 reports the average results obtained for randomly generated problems. Abbreviations used in the tables are summarized in Table 1.

For 20 nodes or less, all problems could be solved to optimality. For larger problems, we remark that the problems with a small value of K are much harder, the lower bound at the root of the Branch-and-Bound tree being far from the optimum.

Problems with a large value of K are easier to solve due to the fact that these problems are closer to the two-connected network problem (without ring constraints), which can be solved efficiently using cut and node-partition inequalities (for the instances we considered). The most difficult cases seem to be for K between 3 and 5, and especially for K = 4.

These instances were not tested in [8] and [7]. Our main new contribution is the introduction of cyclomatic inequalities. To test their impact on the efficiency of our algorithm, we tested the small instances coming from real application (12 and 17 nodes) without the cyclomatic inequalities. The maximum CPU time was set to 30 minutes since all the instances were solved to optimality within this time limit using the cyclomatic inequalities. Results are reported in Table 3 and clearly show that the use of cyclomatic inequalities considerably improves the lower bounds and decreases the computing times.

Table 3. **Branch-and-Cut results for unit edge lengths without cyclomatic inequalities.**

|V|	|E|	*K*	# ineq.	# B&B	LB	LB	UB	Gap	Gap	CPU time
				nodes	(root)	(final)		(root)	(final)	(hh:mm:ss)

12	39	3	566	371	689	794	794	15.2	0.0	0:01:14
12	39	4	1203	1311	606	681	681	12.4	0.0	0:02:54
12	39	5	542	265	565	606	606	7.3	0.0	0:00:30
12	39	6	221	67	544	568	568	4.4	0.0	0:00:08
12	39	8	105	33	513	537	537	4.7	0.0	0:00:02
12	39	10	94	19	501	521	521	4.0	0.0	0:00:01
12	39	16	24	1	496	496	496	0.0	0.0	0:00:01
17	88	3	2956	5449	965	1057	1108	14.8	4.8	0:30:00
17	88	4	6015	5135	853	911	966	13.2	6.0	0:30:00
17	88	5	1960	1773	798	855	855	7.1	0.0	0:08:52
17	88	6	784	189	772	797	797	3.2	0.0	0:01:17
17	88	8	1134	567	731	766	766	4.8	0.0	0:02:19
17	88	10	71	9	709	715	715	0.8	0.0	0:00:02
17	88	16	217	113	693	711	711	2.6	0.0	0:00:13

Table 4. **Branch-and-Cut results, random networks, unit edge lengths.**

					#B&B	Gap	Gap	CPU time
|*V*|	|*E* |	*K*	p/o	# ineq.	nodes	(root)	(final)	(hh:mm:ss)
10	25.6	3	5/5	177.8	64.2	5.7	0.0	0:00:08
10	25.6	4	5/5	93.6	35.0	4.7	0.0	0:00:03
10	25.6	5	5/5	72.0	20.6	2.5	0.0	0:00:02
10	25.6	6	5/5	48.8	14.6	3.6	0.0	0:00:01
10	25.6	8	5/5	17.0	5.4	1.0	0.0	0:00:01
10	25.6	10	5/5	3.2	1.0	0.0	0.0	0:00:01
20	74.4	3	5/4	2763.8	17192.2	8.0	1.1	2:15:54
20	68.7	4	5/5	6454.4	24128.2	8.4	0.0	2:29:58
20	68.7	5	5/5	582.8	477.4	4.4	0.0	0:02:31
20	68.7	6	5/5	355.0	169.8	3.7	0.0	0:00:58
20	68.7	8	5/5	322.8	169.4	3.4	0.0	0:00:42
20	68.7	10	5/5	228.8	37.8	2.1	0.0	0:00:19
20	68.7	16	5/5	68.6	7.0	0.7	0.0	0:00:01
30	179.4	3	5/2	4845.6	26101.4	10.8	3.9	7:54:33
30	179.4	4	5/1	15326.2	26165.4	12.2	6.3	8:46:06
30	179.4	5	5/1	15310.0	21576.6	11.1	5.3	8:03:46
30	179.4	6	5/4	7763.2	13739.8	6.4	1.0	5:03:45
30	179.4	8	5/5	2471.2	2261.4	4.1	0.0	0:55:43
30	179.4	10	5/5	2652.8	1824.6	3.1	0.0	0:43:14
30	179.4	16	5/5	2264.0	705.4	2.7	0.0	0:16:27
40	231.6	3	5/0	2237.4	8677.0	10.7	5.8	3:00:00
40	231.6	4	5/0	3790.6	7089.0	10.9	7.0	3:00:00
40	231.6	5	5/0	4203.8	6433.8	10.3	5.8	3:00:00
40	231.6	6	5/0	4863.0	7096.2	8.6	4.3	3:00:00
40	231.6	8	5/2	4336.4	4258.2	5.6	1.4	2:07:27
40	231.6	10	5/2	4573.8	3095.0	5.7	1.6	2:00:23
40	231.6	16	5/4	3850.8	1889.8	3.1	0.4	1:14:11
50	337.2	3	5/0	2089.4	4400.6	12.8	9.3	3:00:00
50	337.2	4	5/0	3123.6	3736.2	15.9	12.2	3:00:00
50	337.2	5	5/0	3680.6	3312.2	13.3	9.7	3:00:00

50	337.2	6	5/0	3871.6	3417.4	12.3	8.9	3:00:00
50	337.2	8	5/0	4227.4	2871.4	8.1	4.9	3:00:00
50	337.2	10	5/0	4315.8	2744.2	5.8	2.8	3:00:00
50	337.2	16	5/1	4448.4	2290.2	3.5	1.3	2:28:00

Conclusion

In this paper, we studied the particular case of the 2CNBR problem in which each edge has a unit length. This problem is practically harder to solve than its weighted counterpart. We presented here new structural properties of this problem, leading to new facet-defining inequalities. Numerical results obtained with a branch-and-cut algorithm using these new inequalities as well as inequalities introduced in Fortz et al. [8] and Fortz and Labbé [7] were reported. From these results, we can conclude that the Branch-and-Cut algorithm is able to solve to optimality instances of small size (up to 30 nodes). However, problems with a small value of the bound (3 to 5) remain difficult to deal with and should probably need specific methods.

References

1. F. Barahona, "Separating from the dominant of the spanning tree polytope," Op. Research Letters, vol. 12, pp. 201–203, 1992.
2. C. Berge, Graphs and Hypergraphs. North-Holland: Amsterdam, 1973.
3. S.C. Boyd and T. Hao, "An integer polytope related to the design of survivable communication networks," SIAM J. Discrete Math. vol. 6, no. 4, pp. 612–630, 1993.
4. W.H. Cunningham, "Optimal attack and reinforcement ofa network," Journal of ACM, vol. 32, pp. 549–561, 1985.
5. B. Fortz, Design of Survivable Networks with Bounded Rings, vol. 2 of Network Theory and Applications. Kluwer Academic Publishers, 2000.
6. B. Fortz and M. Labbe´, "Facets for the polyhedron of two-connected networks with bounded rings," Technical Report IS-MG 99/17, Universite´ Libre de Bruxelles, CP 210/01, B-1050 Bruxelles, Belgique, 1999. Available at http://smg.ulb.ac.be/Preprints/Fortz99 17.html.
7. B. Fortz and M. Labbe´, "Polyhedral results for two-connected networks with bounded rings," Mathematical Programming, vol. 93, no. 1, pp. 27–54, 2002.
8. B. Fortz, M. Labbe´, and F. Maffioli, "Solving the two-connected network with bounded meshes problem," Operations Research, vol. 48, no. 6, pp. 866–877, 2000.
9. B. Fortz, P. Soriano, and C. Wynants, "A tabu search algorithm for self-healing ring network design," Technical Report IS-MG 2000/15, Universite´ Libre de Bruxelles, 2000. To appear in European Journal of Operational Research.
10. R.E. Gomory and T.C. Hu, "Multi-terminal network flows," SIAM J. Appl. Math., vol. 9, pp. 551–570, 1961.
11. M. Grötschel and C.L. Monma, "Integer polyhedra arising from certain design problems with connectivity constraints," SIAM J. Discrete Math., vol. 3, pp. 502–523, 1990.
12. M. Gro¨ tschel, C.L. Monma, and M. Stoer, "Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints," Operations Research, vol. 40, no. 2, pp. 309–330, 1992.
13. M. Gro¨ tschel, C.L. Monma, and M. Stoer, Design of Survivable Networks, vol. 7 on Network models of Handbooks in OR/MS, chap. 10, pp. 617–672. North-Holland, 1995.
14. C.L. Monma and D.F. Shallcross, "Methods for designing communications networks with certain two-connected survivability constraints," Operations Research, vol. 37, no. 4, pp. 531–541, 1989.
15. G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1988.

16. K. Steiglitz, P. Weiner, and D.J. Kleitman, "The design of minimum-cost survivable networks," IEEE Transactions on Circuit Theory, vol. CT-16, pp. 455–460, 1969.
17. S. Thienel, ABACUS-A Branch-And-Cut System. PhD thesis, Universität zu Köln, 1995.
18. S. Thienel, ABACUS-A Branch-And-Cut System, Version 2.0, User's Guide and Reference Manual. Universität zu Köln, 1997.