The k-edge connected subgraph problem: Valid inequalities and Branch-and-Cut

F. Bendali*, I. Diarrassouba*, M. Didi Biha†, A. R. Mahjoub* and J. Mailfert‡
*Université Blaise Pascal, Clermont-Ferrand
Complexe Scientifique des Cézeaux, 63177 Aubière cedex France
Email: (bendali,diarrass)@isima.fr, ridha.mahjoub@math.univ-bpclermont.fr
†Université d'Avignon, 339, chemin des Meinajariès, 84911 Avignon cedex 9 France
E-mail: mohamed.didi-biha@univ-avignon.fr
‡Université d'Auvergne, 49, Boulevard François Mitterrand, 63001 Clermont-Ferrand cedex France
Email: mailfert@isima.fr

***Abstract*— In this paper we describe a Branch-and-Cut algorithm for the k-edge connected subgraph problem. This problem has applications in the design of survivable telecommunication networks. We introduce a new family of valid inequalities for the associated polytope. We give sufficient conditions for these inequalities to be facet defining and devise separation heuristics to separate them. We present further classes of valid inequalities and also discuss some reduction operations that can be used in a preprocessing phase for the separation. Using these results, we develop a Branch-and-Cut algorithm and present some computational results.**

I. Introduction

One of the main concerns when designing telecommunication networks is to compute network topologies that provide a sufficient degree of survivability. Survivable networks must satisfy some connectivity requirements that is, networks that are still functional after the failure of certain links. As pointed out in [15] (see also [13]), the topology that seems to be very efficient (and needed in practice) is the uniform topology, that is to say that corresponding to networks that survive after the failure of $k-1$ or fewer edges, for some $k \geq 2$. The 2-connected topology ($k = 2$) provides an adequate level of survivability since most failure usually can be repaired relatively quickly. However, for many applications, it may be necessary to provide a higher level of connectivity. In this paper, we consider this variant of the survivable network design problem.

A graph $G = (V, E)$ is called k-edge connected (where k is a positive integer) if for every pair of nodes i, $j \in V$, there are at least k edge-disjoint paths between i and j. Given a graph $G = (V, E)$ and a weight function w on E that associates with an edge $e \in E$ the weight $w(e) \in \mathbb{R}$, the *k-edge connected subgraph problem* (kECSP for short) is to find a k-edge connected spanning subgraph $H = (V, F)$ of G such that $\sum_{e \in F} w(e)$ is minimum.

The kECSP is NP-hard for $k \geq 2$ ([9]). When $k = 1$, the kECSP is nothing but the minimum spanning tree problem, and can be solved in polynomial time. The kECSP has been extensively studied when $k = 2$ ([8], [13], [14], [16], [17]). It has, however, received little attention when $k \geq 3$.

In [3], Bienstock et al. describe structural properties of the optimal solution of kECSP when the weight function satisfies the triangle inequalities (*i.e.* $w(e_1) \leq w(e_2) + w(e_3)$ for every three edges e_1, e_2, e_3 defining a triangle). In particular, they show that every node of a minimum weight k-edge connected subgraph has degree k or $k+1$. This generalizes results given by Monma et al. [16] for the case when $k = 2$.

We will denote the graph by $G = (V, E)$ where V is the *node set* and E is the *edge set*. Given a graph $G = (V, E)$ and an edge subset $F \subseteq E$, the 0-1 vector $x^F \in \mathbb{R}^E$ such that $x^F(e) = 1$ if $e \in F$ and $x^F(e) = 0$ if $e \in E \setminus F$ is called the *incidence vector* of F. The convex hull of the incidence vectors of the edge sets of the k-edge connected subgraphs of G, denoted by kECSP(G), is called the *k-edge connected subgraph polytope* of G.

Given $w : E \to \mathbb{R}$ and F a subset of E, $w(F)$ will denote $\sum_{e \in F} w(e)$. For $W \subseteq V$, we let $\overline{W} = V \setminus W$. If $W \subset V$ is a node subset of G, then the set of edges that have only one node in W is called a *cut* and denoted by $\delta(W)$. We will write $\delta(v)$ for $\delta(\{v\})$. If x^F is the incidence vector of the edge set F of a k-edge connected spanning subgraph of G, then x^F satisfies the following inequalities:

$$x(e) \geq 0 \qquad \text{for all } e \in E, \tag{1}$$

$$x(e) \leq 1 \qquad \text{for all } e \in E, \tag{2}$$

$$x(\delta(W)) \geq k \quad \text{for all } W \subset V, W \neq V, W \neq \emptyset. \tag{3}$$

Conversely, any integer solution of the system defined by inequalities (1)-(3) is the incidence vector of the edge set of a k-edge connected subgraph of G. Constraints (1) and (2) are called *trivial inequalities* and constraints (3) are called *cut inequalities*. We will denote by $P(G,k)$ the polytope given by inequalities (1)-(3).

In what follows, we give some definitions and notations. For definitions and notations about polyhedra see [19]. The graphs we consider are finite, undirected, loopless and connected. Given a graph $G=(V,E)$, if $e \in E$ is an edge with endnodes u and v, we also write uv to denote e. Given W and W' two disjoint subsets of V, $[W,W']$ will denote the set of edges of G having one endnode in W and the other one in W'. If $\pi=(V_1,...,V_p)$, $p \geq 2$, is a partition of V, then we denote by $\delta(\pi)$ the set of edges having their endnodes in different sets. We will also write $\delta(V_1,...,V_p)$. For all $F \subseteq E$, $V(F)$ will denote the set of nodes of the edges of F. For $W \subset V$, we denote by $E(W)$ the set of edges of G having both endnodes in W and $G[W]$ the subgraph induced by W. Given a node set $W \subset V$, contracting a node subset W consists in identifying all the nodes of W and preserving the adjacencies. Given a partition $\pi=(V_1,...,V_p)$, $p \geq 2$, we will denote by G_π the subgraph induced by π that is the graph obtained from G by contracting the sets V_i, for $i=1,...,p$. Note that the edge set of G_π is the set $\delta(V_1,...,V_p)$.

II. Facets of kECSP(G)

In this section we present a new class of valid inequalities for kECSP(G). We describe some conditions for these inequalities to be facet defining. Separation procedures for these inequalities will be described in Section IV. We also present in this section other classes of inequalities that are valid for kECSP(G).

A. Odd path inequalities

Let $G=(V,E)$ be a $(k+1)$-edge connected graph and $\pi=(W_1,W_2,V_1,...,V_{2p})$ a partition of V with $p \geq 2$. Let $I_1=\{4r,4r+1, r=1,...,\lceil \frac{p}{2} \rceil -1\}$ and $I_2=\{2,...,2p-1\} \setminus I_1$. We say that π induces an *odd path configuration* if

1. $|[V_i,W_j]|=k-1$ for $(i,j) \in (I_1 \times \{1\}) \cup (I_2 \times \{2\})$,
2. $|[W_1,W_2]| \leq k-1$,
3. $\delta(V_i)=[V_i,W_1] \cup [V_{i-1},V_i] \cup [V_i,V_{i+1}]$ (resp. $\delta(V_i)=[V_i,W_2] \cup [V_{i-1},V_i] \cup [V_i,V_{i+1}]$) if $i \in I_1$ (resp. $i \in I_2$), and $[V_i,V_{i+1}] \neq \emptyset$, for all $i \in \{1,...,2p-1\}$,
4. $\delta(V_1)=[W_1,V_1] \cup [V_1,V_2]$ and $\delta(V_{2p})=[W_1,V_{2p}] \cup [V_{2p-1},V_{2p}]$ (resp. $\delta(V_{2p})=[W_2,V_{2p}] \cup [V_{2p-1},V_{2p}]$) if p is even (resp. odd) (see Figure 1 for $k=3$ and p even).

Let $C=\bigcup_{i=1}^{2p-1}[V_i,V_{i+1}]$. As $[V_l,V_t]=\emptyset$, if $|l-t|>1$, for $l,t \in \{1,...,2p\}$, C can be seen as an odd path of extremities V_1 and V_{2p} in the graph G_π. With an odd path configuration we associate the inequality

$$x(C) \geq p. \tag{4}$$

Inequalities of type (4) will be called *odd path inequalities*.

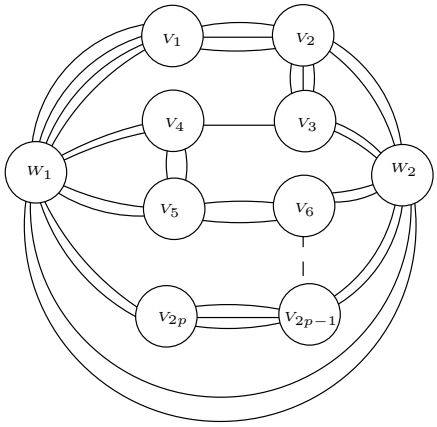

Fig. 1. An odd path configuration with $k=3$ and p even.

We have the following.

Theorem 2.1: Inequalitiy (4) is valid for kECSP(G).

In what follows, we give sufficient conditions for inequality (4) to be facet defining. For the proof see [2].

Theorem 2.2: Inequality (4) defines a facet for kECSP(G) if the following hold.

1. The subgraphs $G[W_1]$, $G[W_2]$ and $G[V_i]$, for $i=1,...,2p$, are $(k+1)$-edge connected;
2. $|[W_1,W_2]|=k-1$, $|[V_1,W_1]|=k$ and $|[V_{2p},W_1]|=k$ (resp. $|[V_{2p},W_2]|=k$) if p is even (resp. odd).

B. Lifting procedure for odd path inequalities

In what follows we are going to describe a lifting procedure for the odd path inequalities. This will permit to extend these inequalities to a more general class of valid inequalities.

A general lifting procedure for inequalities (4) can be described as follows (see [18]). Consider a graph $G=(V,E)$ and a valid inequality $ax \geq \alpha$ for

kECSP(G). Let $G' = (V, E')$ be a graph obtained from G by adding an edge e, that is $E' = E \cup \{e\}$. Then the inequality

$$ax + a(e)x(e) \geq \alpha \tag{5}$$

is valid for kECSP(G') where $a(e) = \alpha - \gamma$ with $\gamma = \min\{ax \mid x \in k\text{ECSP}(G') \text{ and } x(e) = 1\}$. Moreover, if $ax \geq \alpha$ is facet defining for kECSP(G), then inequality (5) is also facet defining for kECSP(G'). Note that if more than one edge have to be added to G, then their lifting coefficients will depend on the order in which they will be added to G. Moreover, if edges $e_1, ..., e_{k-1}, e_k, ..., e_t$ are added to G in this order and $a(e_k)$ is the lifting coefficient of e_k with respect to this order, then $a(e_k) \leq a'(e_k)$ where $a'(e_k)$ is the lifting coefficient of e_k in any order $e_{i_1}, ..., e_{i_{k-1}}, ..., e_{i_t}$ such that $i_l = l$ for $l = 1, ..., k-1$ and $i_s = k$ for some $s > k$.

Theorem 2.3: Let $G = (V, E)$ be a graph and $\pi = (W_1, W_2, V_1, ..., V_{2p})$, $p \geq 2$, a partition of V which induces an odd path configuration. Let C, I_1 and I_2 be defined as in Section II-A. Let $U_1 = \bigcup_{i \in I_1} V_i$, $U_2 = \bigcup_{i \in I_2} V_i$ and $W = U_2 \cup V_{2p} \cup W_2$ (resp. $W = U_2 \cup W_2$) if p is odd (resp. even). If $G' = (V, E \cup L)$ is a graph obtained from G by adding an edge set L, then the following inequality is valid for kECSP(G')

$$x(C) + \sum_{e \in L} a(e)x(e) \geq p, \tag{6}$$

with

$$a(e) = \begin{cases} 1 \text{ if } e \in (\bigcup_{j=1,2}[W_j, U_1 \cup U_2]) \cup \\ [W_1, W_2] \cup (\bigcup_{j=1,2p}[V_j, U_1 \cup U_2]) \cup \\ ([V_1, V_{2p} \cup W_2] \cap \delta(W)) \cup \\ ([V_{2p}, W_1 \cup W_2] \cap \delta(W)); \\ \\ 2 \text{ if } e \in [V_i, V_j], \\ \text{for } i, j \in \{2, ..., 2p-1\} \text{ with } j > i+1 \\ \text{and } i \text{ even, } j \text{ odd}; \\ \\ \lambda \text{ if } e \in [V_i, V_j], \\ \text{for } i, j \in \{2, ..., 2p-1\},\ j > i+1 \text{ and} \\ i \text{ odd or } i \text{ and } j \text{ have same parity}; \\ \\ 0 \text{ otherwise}, \end{cases}$$

where λ is the lifting coefficient obtained using the lifting procedure above. Moreover, we have $1 \leq \lambda \leq 2$.

Observe that the lifting coefficients of the edges other than those between two subsets V_i and V_j such that $i, j \in \{2, ..., 2p-1\}$, $j > i+1$, i is odd or i and j have the same parity do not depend on the order of their addition in G. Inequalities (6) will be called *lifted odd path inequalities*. As it will turn out, these inequalities are very useful for our Branch-and-Cut algorithm.

C. Other facets of kECSP(G)

1) F-partition inequalities: In [17], Mahjoub introduced a class of valid inequalities for 2ECSP(G) as follows. Let $(V_0, V_1, ..., V_p)$, $p \geq 2$, be a partition of V and $F \subseteq \delta(V_0)$ with $|F|$ odd. The following inequality is valid for the 2ECSP(G)

$$x(\Delta) \geq p - \frac{|F| - 1}{2}. \tag{7}$$

Inequalities (7) are called *F-partition inequalities*. These inequalities can be forwardly extended for all $k \geq 2$. In fact one can show that given a partition $(V_0, V_1, ..., V_p)$, $p \geq 2$, of V and $F \subseteq \delta(V_0)$, $F \neq \emptyset$, the inequality

$$x(\delta(V_0, V_1, ..., V_p) \setminus F) \geq \left\lceil \frac{kp - |F|}{2} \right\rceil, \tag{8}$$

is valid for kECSP(G). Note here that $|F|$ can be either odd or even. Also note that if kp and $|F|$ have the same parity, then the corresponding inequality (8) is implied by the cut and the trivial inequalities.

2) SP-partition inequalities: In [4], Chopra introduces a class of valid inequalities for the kECSP when the graph $G = (V, E)$ is outerplanar (a graph is outerplanar if it can be drawn in the plane on a cycle with none crossing chords), $k \geq 1$ is odd, and each edge can be used more than once. He showed that if $\pi = (V_1, ..., V_p)$, $p \geq 2$, is a partition of V, then the inequality

$$x(\delta(V_1, ..., V_p)) \geq \left\lceil \frac{k}{2} \right\rceil p - 1 \tag{9}$$

is valid for kECSP(G).

Didi Biha and Mahjoub [7] extended this result for a general graph and when each edge can be used at most once. They showed that if G is a graph and $\pi = (V_1, ..., V_p)$, $p \geq 2$, is a partition of V such that G_π is series-parallel (a graph is series-parallel if it can be obtained from a single edge by iterative application of the two operations: (i) addition of a parallel edge, and (ii) subdivision of an edge), then inequality (9) is valid for kECSP(G). They called inequalities (9) *SP-partition inequalities*. They also described necessary conditions for inequality (9) to be facet defining and showed that if G is series-parallel and k is odd then kECSP(G) is defined by the trivial, cut and SP-partition inequalities.

Chopra [4] described a lifting procedure for inequalities (9). Let $G = (V, E)$ be a graph and $k \geq 3$ an odd integer. Let $G' = (V, E \cup L)$ be a

graph obtained from G by adding an edge set L. Let $\pi = (V_1, ..., V_p)$ be a partition of V such that G_π is series-parallel. Then the following inequality is valid for kECSP(G')

$$x(\delta_G(V_1, ..., V_p)) + \sum_{e \in L} a(e)x(e) \geq \left\lceil \frac{k}{2} \right\rceil p - 1, \tag{10}$$

where $a(e)$ is the length (in terms of edges) of a shortest path in G_π between the endnodes of e, if $e \in L \cap \delta_{G'}(V_1, ..., V_p)$ and $a(e) = 0$ if not. We will call inequality (10) *lifted SP-partition inequality*. Chopra [4] also showed, when G is outerplanar, that inequality (10) defines a facet of kECSP(G') if G is maximal outerplanar, that is adding one edge e in G lets the resulting graph not outerplanar. This procedure can be easily extended to the case when each edge can be used at most once.

3) Partition inequalities: Grötschel et al. [12] introduced a class of valid inequalities for kECSP(G) called partition inequalities that generalizes the cut inequalities. Let $\pi = (V_1, ..., V_p)$, $p \geq 3$, be a partition of V. The *partition inequality* induced by π is given by

$$x(\delta(V_1, ..., V_p)) \geq \left\lceil \frac{kp}{2} \right\rceil. \tag{11}$$

Clearly, if kp is even, then inequality (11) is redundant with respect to the cut inequalities. Grötschel et al. [12] gave sufficient conditions for the partition inequalities (11) to be facet defining.

III. Reduction operations

In this section, we are going to describe some graph reduction operations. These are based on the concept of critical extreme points of $P(G, k)$ introduced by Fonlupt and Mahjoub [8] for $k = 2$ and extended by Didi Biha and Mahjoub [6] for $k \geq 3$. They [6] introduced the following reduction operation with respect to a solution $\overline{x}$ of $P(G, k)$.

- θ_1: delete an edge $e \in E$ such that $\overline{x}(e) = 0$;
- θ_2: contract a node subset $W \subseteq V$ such that $G[W]$ is k-edge connected and $\overline{x}(e) = 1$ for all $e \in E(W)$;
- θ_3: contract a node subset $W \subseteq V$ such that $|W| \geq 2$, $|\overline{W}| \geq 2$, $|\delta(W)| = k$ and $E(\overline{W})$ contains at least one edge with fractional value;
- θ_4: contract a node subset $W \subseteq V$ such that $|W| \geq 2$, $|\overline{W}| \geq 2$, $G[W]$ is $\lceil \frac{k}{2} \rceil$-edge connected, $|\delta(W)| = k + 1$ and $\overline{x}(e) = 1$ for all $e \in E(W)$.

Let $G' = (V', E')$ and $\overline{x}'$ be obtained by repeated applications of operations $\theta_1, \theta_2, \theta_3, \theta_4$ with respect to $\overline{x}$. Didi Biha and Mahjoub [6] showed that $\overline{x}'$ is an extreme point of $P(G', k)$ if and only if $\overline{x}$ is an extreme point of $P(G, k)$.

These reduction operations can be used in a Branch-and-Cut algorithm for the kECSP. As it will turn out, they can be used in a preprocessing phase for the separation and may be very effective in solving the problem.

IV. A Branch-and-Cut algorithm for the kECSP

In this section, we describe a Branch-and-Cut algorithm for the kECSP. Our aim is to address the algorithmic applications of the theoritical results presented in the previous sections and describe some strategic choices made in order to solve that problem. So, let us assume that we are given a graph $G = (V, E)$ and a weight vector $w \in \mathbb{R}^E$ associated with the edges of G. Let $k \geq 3$ be the connectivity requirement for each node of V.

Given a fractional solution $\overline{x}$ of $P(G, k)$, we let $G' = (V', E')$ and $\overline{x}'$ be obtained by repeated applications of operations $\theta_1, \theta_2, \theta_3, \theta_4$ with respect to $\overline{x}$. As pointed out above, $\overline{x}'$ is an extreme point of $P(G', k)$ if and only if $\overline{x}$ is an extreme point of $P(G, k)$. Moreover, we have the following lemmas which can be easily seen.

Lemma 4.1: Let $a'x \geq \alpha$ be an F-partition inequality (resp. partition inequality) valid for kECSP(G') induced by a partition $\pi' = (V'_0, V'_1 ..., V'_p)$, $p \geq 2$, (resp. $\pi' = (V'_1, ..., V'_p)$, $p \geq 3$) of V'. Let $\pi = (V_0, V_1, ..., V_p)$, $p \geq 2$, (resp. $\pi = (V_1, ..., V_p)$, $p \geq 3$) be the partition of V obtained by expanding the subsets V'_i of π'. Let $ax \geq \alpha$ be an inequality such that

$$a(e) = \begin{cases} a'(e) & \text{for all } e \in E', \\ 1 & \text{for all } e \in (E \setminus E') \cap \delta_G(\pi), \\ 0 & \text{otherwise.} \end{cases}$$

Then $ax \geq \alpha$ is valid for kECSP(G). Moreover, if $a'x \geq \alpha$ is violated by $\overline{x}'$, then $ax \geq \alpha$ is violated by $\overline{x}$.

Lemma 4.2: Let $a'x \geq \alpha$ be an odd path inequality (resp. SP-partition inequality) valid for kECSP(G') induced by a partition $\pi' = (W'_1, W'_2, V'_1, ..., V'_{2p})$, $p \geq 2$ (resp. $\pi' = (V'_1, ..., V'_p)$, $p \geq 3$). Let $\pi = (W_1, W_2, V_1, ..., V_{2p})$, $p \geq 2$ (resp. $\pi = (V_1, ..., V_p)$, $p \geq 3$), be the partition of V obtained by expanding the elements

of π'. Let $ax \geq \alpha$ be the corresponding lifted odd path inequality (resp. lifted SP-partition inequality) obtained from $a'x \geq \alpha$ by application of the lifting procedure described in Section II-B (resp. Section II-C.2) for the edges of $E \setminus E'$. Then $ax \geq \alpha$ is violated by $\overline{x}$ if $a'x \geq \alpha$ is violated by $\overline{x}'$.

Lemmas 4.1 and 4.2 show that looking for an odd path, F-partition, SP-partition or a partition inequality violated by $\overline{x}$ reduces to looking for such inequality violated by $\overline{x}'$ on G'. Note that this procedure can be applied for any solution of $P(G,k)$ and, in consequence, may permit to separate fractional solutions which are not necessarily extreme points of $P(G,k)$. In consequence, for more efficiency, our separation procedures will be performed on the reduced graph G'. The violated inequalities generated in G' with respect to $\overline{x}'$ are lifted to violated inequalities in G with respect to $\overline{x}$ using Lemmas 4.1 and 4.2.

We now describe the framework of our algorithm. To start the optimization we consider the following linear program given by the degree cuts associated with the vertices of the graph G together with the trivial inequalities, that is

$$\begin{aligned} &\text{Min} \sum_{e \in E} w(e)x(e) \\ &x(\delta(u)) \geq k \qquad \text{for all } u \in V, \\ &0 \leq x(e) \leq 1 \qquad \text{for all } e \in E. \end{aligned}$$

The optimal solution $\overline{y} \in \mathbb{R}^E$ of this relaxation of the kECSP is feasible for the problem if $\overline{y}$ is an integer vector that satisfies all the cut inequalities. Usually, the solution $\overline{y}$ is not feasible for the kECSP, and thus, in each iteration of the Branch-and-Cut algorithm, it is necessary to generate further inequalities that are valid for the kECSP but violated by the current solution $\overline{y}$. For this, one has to solve the so-called *separation problem*. This consists, given a class of inequalities, in deciding whether the current solution $\overline{y}$ statisfies all the inequalities of this class, and if not, in finding an inequality that is violated by $\overline{y}$. An algorithm solving this problem is called a *separation algorithm*. The Branch-and-Cut algorithm uses the inequalities previously described and their separations are performed in the following order

1. cut inequalities;
2. SP-partition inequalities;
3. odd path inequalities;
4. F-partition inequalities;
5. partition inequalities.

We remark that all inequalities are global (*i.e.* valid for all the Branch-and-Cut tree) and several inequalities may be added at each iteration. Moreover, we go to the next class of inequalities only if we haven't found any violated inequalities. Our strategy is to try to detect violated inequalities at each node of the Branch-and-Cut tree in order to obtain the best possible lower bound and thus limit the generated nodes. Generated inequalities are added by sets of 200 or less inequalities at a time.

Now we describe the separation procedures used in our Branch-and-Cut algorithm. All these procedures are applied on G' with weights $(\overline{y}'(e), e \in E')$ associated with its edges where $\overline{y}'$ is the restriction on E' to the current LP-solution $\overline{y}$ (G' and $\overline{y}'$ are obtained by repeated applications of operations θ_1, θ_2, θ_3, θ_4).

The separation of the cut inequalities (3) can be performed in polynomial time using the Gomory-Hu algorithm [11] on G'. This algorithm produces the so-called Gomory-Hu tree which gives, for every pair of nodes $(s,t) \in V' \times V'$, the minimum weight st-cut in G'. This separation procedure is exact and requires $|V'| - 1$ maximum flow computation.

In what follows, we consider the separation procedure of the odd path inequalities (4). For this, we need the following lemma.

Lemma 4.3: Let $x \in \mathbb{R}^E$ be a fractional solution of $P(G,k)$ and $\pi = (W_1, W_2, V_1, ..., V_{2p})$, $p \geq 2$, a partition of V, which induces an odd path configuration. If each edge set $[V_i, V_{i+1}]$, $i = 1, ..., 2p-1$, contains an edge with fractional value and

$$x([V_{i-1}, V_i]) + x([V_i, V_{i+1}]) \leq 1, \text{ for } i = 2, ..., 2p-1,$$

then the odd path inequality induced by π is violated by x.

Using Lemma 4.3 we can devise a heuristic to separate inequalities (4). The idea is to find a partition $\pi = (W'_1, W'_2, V'_1, ..., V'_{2p})$, $p \geq 2$, which induces an odd path configuration that satisfies the conditions of Lemma 4.3. The procedure works as follows. We first look, using a greedy method, for a path $\Gamma = \{e_1, ..., e_{2p-1}\}$, $p \geq 2$, in G' such that the edges $e_1, ..., e_{2p-1}$ have fractional values and $\overline{y}'(e_{i-1}) + \overline{y}'(e_i) \leq 1$, for $i = 2, ..., 2p-1$. If $v'_1, ..., v'_{2p}$ are the nodes of Γ taken in this order when going through Γ, we let $V'_i = \{v'_i\}$, $i = 1, ...2p$, and $T_1 = (\bigcup_{i \in I_1} V'_i) \cup V'_1$ (resp. $T_1 = (\bigcup_{i \in I_1} V'_i) \cup V'_1 \cup V'_{2p}$) if p is odd (resp. even) and

$T_2 = (\bigcup_{i \in I_2} V'_i) \cup V'_{2p}$ (resp. $T_2 = (\bigcup_{i \in I_2} V'_i)$) if p is odd (resp. even) where I_1 and I_2 are as defined in Section II-A. In order to determine W'_1 and W'_2, we compute a minimum cut separating T_1 and T_2. If $\delta(W)$ is such a cut with $T_1 \subseteq W$, we let $W'_1 = W \setminus T_1$ and $W'_2 = V' \setminus (W \cup T_2)$. If the partition $\pi = (W'_1, W'_2, V'_1, ..., V'_{2p})$ thus obtained induces an odd path configuration, then, by Lemma 4.3, the corresponding odd path inequality is violated by $\overline{y}'$. If not, we apply again that procedure by looking for an other path. In order to avoid the detection of the same path, we label the edges of the previous ones, so that they won't be considered in the search of a new path. This procedure is iterated until either a violated odd path inequality is found or all the edges having fractional values are labeled. The routine that permits to look for an odd path runs in $O(m'n')$ time. To compute the minimum cut separating T_1 and T_2, we use Goldberg and Tarjan maximum flow algorithm [10]. Hence, our separation procedure is polynomial.

In the lifting procedure for inequalities (4) given in Section II-B we have to compute a coefficient λ for some edges $e \in E \setminus E'$. Since the computation of this coefficient is itself a hard problem, and $\lambda \leq 2$, we consider 2 as lifting coefficient rather than λ for those edges.

To the best of our knowledge, there does not exist polynomial-time algorithm for the separation of F-partition, SP-partition and partition inequalities. For our Branch-and-Cut algorithm, we used the heuristics developed in [2].

To store the generated inequalities, we create a pool whose size increases dynamically. 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. We first separate inequalities from the pool. If all the inequalities in the pool are satisfied by the current LP-solution, we separate the classes of inequalities in the order given above.

V. Computational results

The Branch-and-Cut algorithm described in the previous section has been implemented in C++, using ABACUS 2.4 alpha ([1], [20]) to manage the Branch-and-Cut tree, and CPLEX 9.0 [5] as LP-solver. It was tested on a Pentium IV 3.4 Ghz with 1 Gb of RAM, running under Linux. We fixed the maximum CPU time to 5 hours. The test problems were obtained by taking TSP test problems from the TSPLIB library [21]. The test set consists in complete graphs whose edge weights are the rounded euclidian distance between the edge's vertices. The tests were performed for $k = 3, 4, 5$. In all our experiments, we have used the reduction operations described in the previous sections, unless otherwise specified. Each instance is given by its name followed by an extension representing the number of nodes of the graph. The other entries of the various tables are:

NCut	:	number of generated cut inequalities;
NSP	:	number of generated SP-partition inequalities;
NOP	:	number of generated odd path inequalities;
NFP	:	number of generated F-partition inequalities;
NP	:	number of generated partition inequalities;
COpt	:	weight of the optimal solution obtained;
Gap	:	the relative error between the best upper bound (the optimal solution if the problem has been solved to optimality) and the lower bound obtained at the root node of the Branch-and-Cut tree;
NSub	:	number of subproblems in the Branch-and-Cut tree;
TT	:	total CPU time in hours:min:sec.

The instances indicated with "*" are those whose CPU time exceeded 5 hours. For these instances, the gap is indicated in italic.

Our first series of experiments concerns the kECSP for $k = 3$. The instances we have considered have graphs with 99 to 318 nodes. The results are summarized in Table I.

It appears from Table I that all the instances have been solved to optimality within the time limit except the last five instances. For most of the other instances, the gap is less than 1%. We also observe that our separation procedures detect a large enough number of F-partition and SP-partition inequalities and seem to be quite efficient.

Our next series of experiments concerns the kECSP with $k = 4, 5$. The results are given in Table II for $k = 4$ and Table III for $k = 5$. The instances considered have graphs with 195 to 561 nodes. Note that for $k = 4$, the SP-partition and partition inequalities are redundant with respect to the cut inequalities (3). Thus, they are not included in the resolution process for $k = 4$ and therefore do not appear in Table II.

First observe that for $k = 4$, the CPU time for all the instances is relatively small and most of the instances have been solved in less than 1 minute. We can also observe that 9 instances over 12 are solved in the cutting plane phase. Moreover, a few number of odd path inequalities are generated. However a large enough number of F-partition inequalities is detected. Thus, these later inequalities seem to be very effective for solving the kECSP when k is even. This also shows that the kECSP is easier to

Instance	NCut	NSP	NOP	NFP	NP	COpt	Gap	NSub	TT
rat99	41	26	13	341	23	2029	0.38	41	0:00:47
rd100	101	74	11	418	18	13284	0.43	171	0:03:37
gr120	50	45	6	588	17	11442	0.19	99	0:11:15
bier127	46	59	4	276	13	198184	0.15	11	0:01:55
ch130	121	132	30	1355	40	10400	0.55	1693	1:05:05
ch150	92	93	19	588	22	11027	0.41	193	0:20:31
rat195	24	19	3	514	1	3934	0.06	7	0:08:21
d198	171	105	23	617	59	25624	0.21	159	1:04:19
gr202	77	69	14	558	22	65729	0.11	69	0:13:16
*pr226	364	248	35	162	41	-	*9.02*	261	5:00:00
*gr229	179	245	23	1568	94	-	*1.00*	1219	5:00:00
*pr264	275	181	145	668	62	-	*12.29*	69	5:00:00
*a280	142	84	56	2539	59	-	*2.69*	459	5:00:00
*lin318	189	147	15	610	58	-	*4.94*	25	5:00:00

TABLE I

RESULTS FOR $k = 3$.

Instance	NCut	NOP	NFP	COpt	Gap	NSub	TT
rat195	0	0	37	5750	0.00	1	0:00:13
d198	43	0	71	35404	0.01	3	0:00:16
gr202	13	3	220	94841	0.02	3	0:01:28
gr229	24	2	15	318565	0.00	1	0:00:03
pr264	59	1	7	122941	0.00	1	0:00:06
a280	3	0	180	6317	0.00	1	0:01:00
pr299	30	0	427	117559	0.00	1	0:00:20
lin318	28	0	2	105000	0.00	1	0:00:06
rd400	21	2	232	36676	0.00	1	0:07:39
pr439	78	3	61	264975	0.02	19	0:02:52
si535	0	0	4	53604	0.00	1	0:00:39
pa561	10	1	306	6724	0.00	1	0:08:37

TABLE II

RESULTS FOR $k = 4$.

Instance	NCut	NSP	NOP	NFP	NP	COpt	Gap	NSub	TT
rat195	1	0	0	508	0	7773	0.00	1	0:20:54
d198	56	9	6	1093	32	47614	0.15	337	1:50:40
gr202	0	0	0	64	0	128990	0.00	1	0:00:31
gr229	18	1	11	679	9	434422	0.06	43	0:31:58
*pr264	105	12	38	1327	28	-	*1.78*	43	5:00:00
a280	2	0	2	302	0	8643	0.02	7	0:05:05
pr299	11	3	2	637	1	161576	0.00	1	0:05:12
lin318	24	3	11	1548	11	144341	0.02	7	4:34:39
rd400	11	1	15	691	6	49893	0.01	17	1:29:09
*pr439	46	2	8	746	0	-	*3.46*	1	5:00:00
si535	0	0	0	0	0	79115	0.00	1	0:00:19
pa561	1	0	2	286	1	9161	0.00	1	3:26:58

TABLE III

RESULTS FOR $k = 5$.

solve when k is even, what is also confirmed by the results of Table III for $k = 5$. In fact, the instance pr264 has been solved for $k = 4$ in 1 second, whereas it could not be solved to optimality for $k = 5$ after 5 hours. The same observation can be done for pr439. Also, we can remark that the CPU time for all the instances when $k = 5$ is higher than that when $k = 4$. For instance, the test problem d198 has been solved in 1h 50mn when $k = 5$, whereas only 16 seconds were needed to solve it when $k = 4$.

Compared to Table I, Tables II and III also show that, for the same parity of k, the kECSP becomes easier to solve when k increases. In fact, with $k = 3$, we could not solve to optimality instances with more than 202 nodes, whereas for $k = 5$, we could solve larger instances.

The results for $k = 3, 4, 5$ can also be compared to those obtained by Kerivin et al. [14] for the 2ECSP. It turns out that for the same instances, the problem has been easier to solve for $k = 2$

than for $k = 3$. However, for $k = 4$ the problem appeared to be easier to solve than for $k = 2$. This shows again that the case when k is odd is harder to solve than that when k is even and that the problem becomes easier when k increases with the same parity.

In order to evaluate the impact of the reduction operations θ_1, θ_2, θ_3, θ_4 on the separation procedures, we tried to solve the kECSP, for $k = 3$, without using them.

These experiments show that the CPU time increased for the majority of the instances when the reduction operations are not used. In particular, for the instance pr107 we did not reach the optimal solution after 5 hours, whereas it has been solved to optimality after 1h 26mn. Almost, the CPU time for the instances ch130 and d198, for example, increased from 1 hour to more than 4 hours. Moreover, we remark that when using the reduction operations, we generate more SP-partition, F-partition and partition inequalities and fewer nodes in the Branch-and-Cut tree. This implies that our separation heuristics are less efficient without the reduction operations. Thus, it seems that the reduction operations play an important role in the resolution of the problem. They permit to strengthen much more the linear relaxation of the problem and accelerate its resolution.

We also tried to measure the effect of the different non-basic classes of inequalities (*i.e.* inequalities other than cut and trivial inequalities). For this, we have first considered a Branch-and-Cut algorithm for the kECSP with $k = 3$ using only the cut constraints. In this case, we could not solve any of the instances having more than 52 nodes. Even more, after less than 10 minutes of CPU time, the Branch-and-Cut tree gets a very big size and the resolution process stops. To illustrate this, we take for example the instance brazil58. For this instance, the Branch-and-Cut tree contained 11769 nodes after 10 minutes when the Branch-and-Cut algorithm used only the cut inequalities, whereas it has been solved without branching when using the other classes of inequalities.

Finally, we tried to evaluate separately the efficiency of each class of the non-basic inequalities. For this, we also considered the case when $k = 3$. We have seen that all the classes of inequalities have a big effect on the resolution of the problem. In particular, the SP-partition inequalities seem to play a central role. This can be seen by considering the instance d198. This instance has been solved in 1h 04mn using all the constraints. However, without the SP-partition inequalities, we could not reach the optimal solution after 5 hours. We also remarked that the gap increased when one of these classes of inequalities is not used in the Branch-and-Cut algorithm.

REFERENCES

[1] ABACUS - A Branch-And CUt System, "http://www.informatik.uni-koeln.de/abacus".

[2] F. Bendali, I. Diarrassouba, M. Didi Biha, A. R. Mahjoub and J. Mailfert, "The k-edge connected subgraph problem: Valid Inequalities and Branch-and-Cut", *LIMOS Research Report: LIMOS/RR-07-01*, 2007.

[3] D. Bienstock, E. F. Brickel and C. L. Monma, "On the structure of minimum weight k-connected networks", *SIAM Journal on Discrete Mathematics 3*, 1990, pp. 320-329.

[4] S. Chopra, "The k-edge connected spanning subgraph polyhedron", *SIAM Journal on Discrete Matematics 7*, 1994, pp. 245-259.

[5] Cplex, "http://www.ilog.com".

[6] M. Didi Biha and A. R. Mahjoub, "The k-edge connected subgraph problem I: Polytopes and critical extreme points", *Linear Algebra and Its applications 381*, 2004, pp. 117-139.

[7] M. Didi Biha and A. R. Mahjoub, "k-edge connected polyhedra on series-parallel graphs", *Operations Research Letters 19*, 1996, pp. 71-78.

[8] J. Fonlupt and A. R. Mahjoub, "Critical extreme points of the 2-edge connected spanning subgraph polytope", *Mathematical Programming 105*, 2006, pp. 289-310.

[9] M. R. Garey and D. J. Johnson, "Computer and Intractability: A Guide to the Theory of NP-completness", *Freeman, San Francisco*, 1979.

[10] A. Goldberg and R. E. Tarjan, "A New Approach to the Maximum Flow Problem", *Journal of the Association for Computing Machinery 35*, 1988, pp. 921-940.

[11] R. E. Gomory and T. C. Hu, "Multi-Terminal Network Flows", *Journal of the Society for Industrial and Applied Mathematics 9*, 1961, pp. 551-570.

[12] M. Grötschel, C. L. Monma and M. Stoer, "Polyhedral approches to network survivability", In F. Roberts, F. Hwang and C. L. Monma, eds, *Reliability of computer and Communication newtorks, Vol 5, Series Discrete Mathematics and Computer Science, AMS/ACM*, 1991, pp. 121-141.

[13] H. Kerivin and A. R. Mahjoub, "Design of Survivable Networks: A Survey", *Networks 46*, 2005, pp. 1-21.

[14] H. Kerivin, A. R. Mahjoub and C. Nocq, "(1,2)-survivable networks: Facets and Branch-and-Cut", *The Sharpest Cut, MPS-SIAM Series in Optimization*, M. Grötschel (Editor), 2004, pp. 121-152.

[15] C.-W. Ko and C. L. Monma, "Heuristics for designing highly survivable communication networks", *Technical Report, Bellcore, Morristown, New Jersey*, 1989.

[16] C. L. Monma, B. S. Munson and W. R. Pulleyblank, "Minimum weight two-connected spanning networks", *Operations Research 37*, 1989, pp. 153-171.

[17] A. R. Mahjoub, "Two-edge Connected spanning subgraphs and polyhedra", *Mathematical Programming 64*, 1994, pp. 199-208.

[18] G. L. Nemhauser and L. A. Wolsey, "Integer and Combinatorial Optimization", Wiley Editions, 1988, pp. 259-295.

[19] W. R. Pulleyblank, "Polyhedral combinatorics", G. L. Nemhauser et al. editors, Handbook in OR-MS, volume 1n North-Holland, Amsterdam, pp. 371-446.

[20] S. Thienel, "ABACUS - A Branch-And-CUt System", PhD thesis, Universität zu Köln, 1995.

[21] TSPLIB,"http://www.iwr.uni-heidjhelberg.de/groups/comopt/software/TSPLIB95/".