Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Discrete Mathematics 41 (2013) 551–558

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

Two Node-Disjoint 3-Hop-Constrained Survivable Network Design and Polyhedra

Ibrahima Diarrassouba [1]

Laboratoire LMAH
Le Havre University
Le Havre, France

Hakan Kutucu [2]

Department of Computer Engineering
Karabuk University
Karabuk, Turkey

A. Ridha Mahjoub [3]

Laboratoire LAMSADE
Université Paris Dauphine
Paris, France

Abstract

Given a weighted undirected graph G with a set of pairs of terminals $\{s_i, t_i\}, i = 1, ..., d$, and an integer $L \geq 2$, the two node-disjoint hop-constrained survivable network design problem (TNHNDP) is to find a minimum weight subgraph of G such that between every s_i and t_i there exist at least two node-disjoint paths of length at most L. This problem has applications to the design of survivable telecommunications networks with QoS-constraints. We discuss this problem from a polyhedral point of view. We present several classes of valid inequalities along with necessary and/or sufficient conditions for these inequalities to be facet defining. We also discuss separation routines for these classes of inequalities. Using this, we propose a Branch-and-Cut algorithm for the problem when $L = 3$, and present some computational results.

Keywords: Survivable network, node-disjoint paths, hop constraint, polyhedron, facet, branch-and-cut

http://dx.doi.org/10.1016/j.endm.2013.05.137

1 Introduction

Given a weighted undirected graph $G = (N, E)$, an integer $L \geq 2$, and a set of demands $D \subseteq N \times N$, the two node-disjoint hop-constrained survivable network design problem (TNHNDP) consists in finding a minimum weight subgraph of G containing at least two node-disjoint paths of at most L hops between each pair of nodes $\{s, t\}$ in D.

The edge version of the problem (TEHNDP) has been already investigated by several authors when $L = 2, 3$. In particular, [5] give a complete and minimal linear description of the corresponding polytope when $L = 2, 3$ and $|D| = 1$. [3] and [1] have studied the problem when $|D| \geq 2$ and when two and k edge-disjoint paths, respectively, are required. They devise Branch-and-Cut algorithms for the problem when $L = 2, 3$ and give some computational results. Also, Huygens and Mahjoub [4] have studied the two versions of the problem (TNHNDP and TEHNDP) when $L = 4$. They give an integer programming formulation for the problem in the two cases.

Given 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 $x^F(e) = 0$ otherwise, is called the *incidence vector* of F. The convex hull of the incidence vectors of the solutions to the TNHNDP on G, denoted by $P(G, L)$, will be called the TNHNDP polytope. If $W \subset N$ 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 $\delta(\{v\})$. A cut $\delta(W)$ such that $s \in W$ and $t \in N \setminus W$ will be called an st-cut. Given a node $z \in N$, the graph $G - z$ is the subgraph obtained from G by deleting the node z and all its incident edges (but not their other end nodes). Let $V_0, V_1, \ldots, V_{L+1}$ be a partition of N such that $s \in V_0$, $t \in V_{L+1}$, and $V_i \neq 0$ for all $i = 1, \ldots, L$. Let T be the set of edges $e = uv$, where $u \in V_i$, $v \in V_j$, and $|i - j| > 1$. The set T is called an *L-path-cut*. The nodes in N that do not belong to any demand of D will be called *Steiner* nodes. $\delta(V_0, V_1, \ldots, V_{L+1})$ is the set of edges between any two subsets of the partition of N.

The following linear system along with the integrality constraints formulates the TNHNDP as an integer program when $L = 2, 3, 4$ (see [4]).

[1] Email: diarrasi@univ-lehavre.fr
[2] Email: hakankutucu@karabuk.edu.tr
[3] Email: mahjoub@lamsade.dauphine.fr

$$x(\delta(W)) \geq 2 \quad \text{for all } st\text{-cuts } \delta(W), \tag{1}$$
$$x(\delta_{G-z}(W)) \geq 1 \quad \text{for all } st\text{-cuts } \delta_{G-z}(W), \text{ for all } z \in N \setminus \{s,t\}, \tag{2}$$
$$x(T) \geq 2 \quad \text{for all } L\text{-path-cuts } T, \tag{3}$$
$$x(T_{G-z}) \geq 1 \quad \text{for all } L\text{-path-cuts } T_{G-z}, \text{ for all } z \in N \setminus \{s,t\}, \tag{4}$$
$$x(e) \leq 1 \quad \text{for all } e \in E, \tag{5}$$
$$x(e) \geq 0 \quad \text{for all } e \in E. \tag{6}$$

Inequalities (1),(2),(3) and (4) are called *st-cut inequalities*, *st-node-cut inequalities*, *L-path-cut inequalities* and *L-path node-cut inequalities*, respectively. Inequalities (5) and (6) are called *trivial inequalities*.

2 Further Valid Inequalities

In this section, we describe further classes of inequalities that are valid for $P(G,L)$. As it will turn out, these inequalities define facets in some cases and reinforce the linear relaxation of the integer programming formulation presented in the previous section.

In the following two theorems, we present two classes of inequalities that are valid for the TEHNDP polytope. As we will mention below, they are also valid for $P(G,L)$ when $L = 3$.

Theorem 2.1 *[3]*
Let $L = 3$ and $T = \{t_1, \ldots, t_p\}$ be a subset of p destination nodes w.r.t. to a node s. Let $\pi = (V_0, V_1, \ldots, V_p)$ be a partition of N such that $s \in V_0$, and $t_i \in V_i$, $i = 1, \ldots, p$. Then, the inequality

$$x(\delta(V_0, \ldots, V_p)) \geq \lceil 4p/3 \rceil \tag{7}$$

is valid for the two-edge connected hop-constrained network design problem.

Theorem 2.2 *[1,5]*
Let $L = 3$ and $\Pi = \{V_0^1, V_0^2, V_1, V_2, V_3, V_4\}$ be a partition of V such that $\pi = (V_0^1, V_0^2 \cup V_1, V_2, V_3, V_4)$ induces a 3-st-path-cut, and V_1 induces a valid st-cut in G. If $F \subseteq [V_0^2 \cup V_1 \cup V_4, V_2]$ is chosen such that $|F|$ is odd, then the inequality

$$x([V_0^1, V_1 \cup V_2 \cup V_3 \cup V_4]) + x([V_0^2, V_1 \cup V_3 \cup V_4]) + x([V_1, V_3 \cup V_4]) + x([V_0^2 \cup V_1 \cup V_4, V_2]) \geq \lceil 3 - |F|/2 \rceil \tag{8}$$

is valid for the two-edge connected hop-constrained network design problem.

Inequalities of type (7) are called *Rooted Partition inequalities* and inequalities of (8) are called *Double Cut inequalities*. Since every solution of the TNHNDP is also solution of the TEHNDP, Inequalities (7) and (8) are also valid for TNHNDP. As we will see in the next section, rooted partition inequalities define facets in some cases.

Now, in the following theorem, we introduce a new class of inequalities which allows to describe the optimal solutions of the TNHNDP in the case where the demands are rooted and when the edge weights satisfy the triangle inequalities.

Theorem 2.3 *Suppose that $L \geq 2$ and the weight of the edges of G satisfy the triangle inequalities. Let $u \in V$ be a Steiner node and $F \subseteq E$ an optimal solution of the rooted TNHNDP w.r.t. to these edges weights. If s is the root of the demands and $F \cap [s,u] = \emptyset$, then $\delta(u) \cap F = \emptyset$.*

Proof. See [2]. □

Corollary 2.4 *Consider the rooted TNHNDP and let s be the root of the demand set. When $L = 3$ and when the weight of the edges satisfy the triangle inequalities, then the incidence vector of any optimal solution of the rooted TNHNDP satisfies*

$$x(su) \geq \sum_{uv \in \delta(u) \setminus \{su\}} x(uv), \text{ for every Steiner node } u \in N. \tag{9}$$

Inequalities (9) are called *optimality constraints*. They are in polynomial number and, as we will see in Section 5, they are effective in solving the rooted TNHNDP.

3 Facets of the TNHNDP Polytope

In this section, we give necessary and sufficient conditions for inequalities (1)-(6). We also give sufficient conditions for inequalities (7) to define facets of $P(G,L)$. These conditions will be used later to derive efficient separation procedures.

First, we discuss the dimension of $P(G,L)$ when $L = 3$. An edge e is said to be *essential* if it belongs to an st-cut or a 3-st-path-cut of cardinality 2, or to an st-node-cut or a 3-st-path-node-cut of cardinality 1. We denote by E^* the set of essential edges of G. We have the following theorem.

Theorem 3.1 *$dim(P(G,3)) = |E| - |E^*|$.*

If $G = (N, E)$ is complete and $|N| \geq 4$, then $E^* = \emptyset$. In the remainder of this paper, G is a complete and simple graph with $|N| \geq 4$. Thus, $P(G,3)$ is full dimensional. If G is not complete, one can make it complete and obtain an equivalent problem by adding the missing edges with sufficiently high weights.

Theorem 3.2 ***(i)*** *Inequality $x(e) \leq 1$ defines a facet of $P(G,3)$ for all $e \in E$.*
(ii) *Inequality $x(e) \geq 0$ defines a facet of $P(G,3)$ if and only if either*
(a) $|N| \geq 5$ or
(b) $|N| = 4$ and $e = uv$ with $\{u, v\} \in N \setminus \{s, t\}$.

Theorem 3.3 *Every st-cut inequality and every st-node-cut inequality other than those induced by $\{s\}$ or $\{t\}$ defines a facet of $P(G,3)$.*

Theorem 3.4 *Every L-path-cut and L-path-node-cut inequality defines a facet of $P(G,3)$ if and only if $|V_0| = 1$ and $|V_{L+1}| = 1$.*

Theorem 3.5 *The rooted partition inequality (7) induced by a partition $\pi = (V_0, V_1, ..., V_p)$ defines a facet of $P(G,3)$ if $|V_i| = 1$ for all $i = 1, \ldots, p$ and p is not a multiple of 3.*

4 Separation Procedures

In order to develop our Branch-and-Cut algorithm, we devise separation algorithms for Inequalities (1)-(4) and Inequalities (7) and (8).

As showed in [1], Inequalities (1)-(4) can be separated in polynomial time by computing a maximum flow in a special directed graph. For double cut inequalities, we use the separation heuristic developed by [1].

In the following, we describe a separation heuristic for the rooted partition inequalities when $L = 3$ and $|V_i| = 1$, for all $i \in \{1, ..., p\}$, and p is not multiple of 3. By Theorem 3.5, they define facets in this case. Our separation algorithm in this case is based on the reduction of the separation problem to that of finding a maximum flow with lower bounds in a special graph.

First, we can easily see that Inequality (7) is equivalent to

$$x(\delta(V_0)) + \sum_{u \in N \setminus V_0} x(\delta(u)) \geq \frac{8}{3}p + \alpha \tag{10}$$

with

$$\alpha = \begin{cases} \frac{4}{3} & \text{if } p = 3q+1, \text{ for some } q \in \mathbb{N} \\ \frac{2}{3} & \text{if } p = 3q+2, \text{ for some } q \in \mathbb{N}. \end{cases}$$

Since $p = |V \setminus V_0|$ and $x(\delta(V_0)) = x(\delta(N \setminus V_0))$, Inequality (10) is equivalent to

$$x(\delta(N \setminus V_0)) \geq \sum_{u \in N \setminus V_0} y_u + \alpha \tag{11}$$

where $y_u = \frac{8}{3} - x(\delta(u))$, for all $u \in N \setminus V_0$.

Therefore, if $\overline{x}$ is a solution of $\mathbb{R}^E$, then there exists a violated rooted partition inequality induced by a partition $\pi = (V_0, V_1, ..., V_p)$ with $|V_i| = 1$ for all $i = 1, ..., p$, if and only if the inequality (11) induced by π is violated by $\overline{x}$.

The separation problem of the rooted partition inequalities in this case then reduces to finding a set of terminals W for which the corresponding inequality (11) is violated by $\overline{x}$. This can be done by computing a feasible flow with lower bounds in a special directed graph obtained from G and $\overline{x}$ and can be implemented in polynomial time. For more details, the reader can refeer to [2].

5 Branch-and-Cut and Computational Results

Based on the results described in previous sections, we have developed a Branch-and-Cut algorithm to solve the TNHNDP when $L = 3$. The algorithm has been implemented in C++, using ABACUS 3.2 to manage the Branch-and-Cut tree and CPLEX 12.2 as linear solver. It was tested on a Xeon Quad-Core E5507 machine at 2.27 GHz with 8GB RAM, running under Linux. The maximum CPU time has been fixed to 5 hours.

The test problems are composed of complete graphs from TSPLIB (with euclidean edge weights). The demands are randomly generated. Each set of demand is either rooted at the same node s or arbitrary having multiple sources and destinations.

We use in our Branch-and-Cut algorithm all the different inequalities we have presented above, except in the non-rooted case where we do not use optimality constraints (since they are not valid in this case). The separations of the different constraints are performed in the following order: (i) st-cut inequalities, (ii) 3-st-path-cut inequalities, (iii) st-node-cut inequalities, (iv)

3-*st*-path-node-cut inequalities, (v) rooted-partition inequalities, (vi) double cut inequalities.

Table 1 below presents the results obtained for instances with graphs having up to 76 nodes. For each instance, we give the type of the demand set ("a" for arbitrary, "r" for rooted), the number of nodes $|N|$ and the number of demands $|D|$, the number of generated constraints (NC and NNC for *st*-cut and *st*-node-cut inequalities, LPC and LPNC for 3-*st*-path-cut and 3-*st*-path-node-cut inequalities, RP and DC for rooted partition and double cut inequalities), the relative error, in percentage, between the best upper bound and the best lower bound obtained at the root node of the Branch-and-Cut tree (resp., the best lower bound obtained over all the Branch-and-Cut tree) Gap1 (resp. Gt1). We also give Gap2 and Gt2 which are the gaps (as defined before) achieved when the optimality constraints, the rooted partition and double cut inequalities are not used in the Branch-and-Cut algorithm. We finally give the number of nodes in the Branch-and-Cut tree and the total CPU time in hours:min.sec. Remark that a value of 0 for Gt1 and Gt2 indicates that the upper bound obtained is optimal.

Table 1
Results for real instances when $L = 3$

$\|N\|$	$\|D\|$	NC	NNC	LPC	LPNC	RP	DC	Gap1	Gap2	Gt1	Gt2	Tree1	Tree2	CPU1	CPU2
a 14	10	45	13	5978	517	0	23	14.7	14.7	0	0	563	575	0:01:19	0:01:29
a 14	7	4	0	191	0	0	0	2.14	2.14	0	0	9	9	0:00:01	0:00:01
r 14	10	37	15	257	15	4	0	3.58	9.5	0	0	47	171	0:00:01	0:00:05
r 14	7	17	7	56	6	1	0	3.44	10.38	0	0	21	53	0:00:01	0:00:01
a 17	45	475	72	43186	138487	0	43	24.5	30.65	5.46	14.97	1639	795	5:00:00	5:00:00
a 17	8	32	6	17330	694	0	80	14.18	14.18	0	0	1801	2259	0:14:01	0:20:03
r 17	16	74	33	9522	1659	52	0	8.22	16.21	0	4.63	1915	949	0:07:36	5:00:00
a 30	10	78	2	3108	35	0	0	6.19	6.19	0	0	97	97	0:00:52	0:00:57
a 30	15	19	2	52744	13	0	0	43.97	43.97	42.91	42.91	101	101	5:00:00	5:00:00
r 30	10	318	751	509	72	8	0	6.38	9	0	2.98	443	481	0:01:17	5:00:00
r 30	15	778	1435	14098	1726	92	0	11.53	29.28	0	19.24	7411	3007	1:30:53	5:00:00
a 48	10	111	18	76852	18	0	0	42.14	42.14	40.01	40.01	13	13	5:00:00	5:00:00
a 48	15	68	0	28895	0	0	0	58.9	58.9	58.79	58.79	7	7	5:00:00	5:00:00
r 48	10	2425	1411	641	139	18	0	8.1	29.59	0	27.81	387	15	1:05:13	5:00:00
r 48	15	631	897	4001	601	55	0	9.62	45.07	0	40.96	1649	45	0:43:42	5:00:00
a 52	10	202	4	41755	11	0	0	19.15	19.15	15.87	15.87	35	35	5:00:00	5:00:00
a 52	20	57	2	73270	0	0	0	56.35	56.35	56.11	56.11	35	35	5:00:00	5:00:00
r 52	10	1188	2583	562	81	8	0	6.38	25.68	0	23.34	451	35	1:36:51	5:00:00
r 58	20	183	215	27780	3125	91	0	22.27	∞	16.5	∞	2339	1	5:00:00	5:00:00
r 58	30	69	31	21317	284	38	0	41.42	71.87	38.9	71.87	823	3	5:00:00	5:00:00
r 58	40	244	9	23881	127	23	0	71.88	74.87	70.76	74.75	215	7	5:00:00	5:00:00
r 76	20	26	46	14097	2288	37	0	27.72	45.03	23.68	43.33	1635	17	5:00:00	5:00:00
r 76	40	14	1	12171	116	10	0	53.88	∞	53.01	∞	305	1	5:00:00	5:00:00

Table 1 shows that for the test problems used in these experiments, 12 instances over 23 have been solved to optimality. For the instances solved to optimality, the CPU1 time varies from 1 sec to 1h36min. We notice that a small number of double cut inequalities are generated and for almost all the instances, the rooted partition inequalities are generated. Also, we notice that for the instances which are not solved to optimality, the gaps (Gap1 and Gap2)

are relatively high while the number of nodes in the Branch-and-Cut tree is relatively small. Also, we notice that a large number of basic inequalities are generated. This let us suppose that the algorithm spends a lot of time in the separation of the different inequalities and does not have enough time to explore more solutions in the Branch-and-Cut tree.

We have also checked the efficiency of the different constraints presented in this paper, especially for rooted partition inequalities and optimality constraints. For this, we have tried to solve the problem without rooted partition and double cut inequalities and optimality constraints. In this case, we remark that the efficiency of the algorithm is significantly decreased. For some instances, we do not have the optimal solution after 5 hours when these constraints are removed while the algorithm is able to obtain the optimal solution, in short time, when they are used (see for example (r17,16) and (r30,10)). Moreover, for some instances (see (r58,20) and (r76,40)), the algorithm spends all the time at the root node when the optimality constraints are removed. This shows the efficiency of the additional inequalities, especially the rooted partition inequalities and the optimality constraints, in solving the problem.

Acknowledgments

The research of the second author has been supported by a TUBITAK-BIDEB fellowship.

References

[1] Diarrassouba I., Survivable Network Design Problems with High Connectivity Requirements, PhD Thesis, Université Blaise Pascal, France, 2009.

[2] Diarrassouba I., Kutucu H. and Mahjoub A.R., Two Node-Disjoint Hop-Constrained Survivable Network Design and Polyhedra, Technical Report, Cahier de Recherche LAMSADE 332, France, 2013.

[3] Huygens D., Labbé M., Mahjoub A.R., Pesneau P., The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. Networks 49(1) (2007), 116-133.

[4] Huygens D. and Mahjoub A.R., Integer programming formulations for the two 4-hop-constarined paths problem, Networks 49 (2007), 135-144

[5] Huygens D., Mahjoub A.R. and Pesneau P., Two edge-disjoint hop-constrained paths and polyhedra, SIAM J Disc Math 18 (2004), 287-312.